

# StarPU Internal Handbook

---

for StarPU 1.3.10

---

<b>1 Introduction</b>	<b>3</b>
1.1 Motivation	3
<b>2 StarPU Core</b>	<b>5</b>
2.1 StarPU Core Entities	5
2.1.1 Overview	5
2.1.2 Workers	5
2.1.3 Scheduling Contexts	8
2.1.4 Workers and Scheduling Contexts	8
2.1.5 Drivers	9
2.1.6 Tasks and Jobs	9
2.1.7 Data	9
<b>3 Module Index</b>	<b>11</b>
3.1 Modules	11
<b>4 Module Documentation</b>	<b>13</b>
4.1 Workers	13
<b>5 File Index</b>	<b>15</b>
5.1 File List	15
<b>6 StarPU File Documentation</b>	<b>19</b>
6.1 barrier.h File Reference	19
6.1.1 Data Structure Documentation	19
6.2 barrier_counter.h File Reference	19
6.2.1 Data Structure Documentation	20
6.3 bound.h File Reference	20
6.3.1 Function Documentation	20
6.3.2 Variable Documentation	21
6.4 cg.h File Reference	21
6.4.1 Data Structure Documentation	22
6.4.2 Macro Definition Documentation	23
6.5 coherency.h File Reference	23
6.5.1 Data Structure Documentation	24
6.5.2 Function Documentation	29
6.6 combined_workers.h File Reference	30
6.7 config.h File Reference	30
6.8 copy_driver.h File Reference	36
6.8.1 Data Structure Documentation	36
6.8.2 Enumeration Type Documentation	38
6.9 data_concurrency.h File Reference	38
6.10 data_interface.h File Reference	38
6.10.1 Data Structure Documentation	39

6.10.2 Variable Documentation . . . . .	40
6.11 data_request.h File Reference . . . . .	40
6.11.1 Macro Definition Documentation . . . . .	40
6.12 datastats.h File Reference . . . . .	40
6.13 datawizard.h File Reference . . . . .	41
6.14 debug.h File Reference . . . . .	41
6.14.1 Function Documentation . . . . .	42
6.14.2 Variable Documentation . . . . .	42
6.15 detect_combined_workers.h File Reference . . . . .	42
6.15.1 Function Documentation . . . . .	43
6.16 disk.h File Reference . . . . .	43
6.16.1 Function Documentation . . . . .	43
6.17 disk_unistd_global.h File Reference . . . . .	45
6.17.1 Data Structure Documentation . . . . .	45
6.18 driver_common.h File Reference . . . . .	46
6.19 driver_cpu.h File Reference . . . . .	46
6.20 driver_cuda.h File Reference . . . . .	46
6.21 driver_disk.h File Reference . . . . .	48
6.22 driver_mic_common.h File Reference . . . . .	48
6.22.1 Data Structure Documentation . . . . .	49
6.23 driver_mic_sink.h File Reference . . . . .	49
6.24 driver_mic_source.h File Reference . . . . .	50
6.24.1 Variable Documentation . . . . .	51
6.25 driver_mpi_common.h File Reference . . . . .	51
6.26 driver_mpi_sink.h File Reference . . . . .	51
6.27 driver_mpi_source.h File Reference . . . . .	51
6.28 driver_opencl.h File Reference . . . . .	51
6.29 driver_opencl_utils.h File Reference . . . . .	52
6.30 drivers.h File Reference . . . . .	53
6.31 errorcheck.h File Reference . . . . .	53
6.31.1 Enumeration Type Documentation . . . . .	53
6.31.2 Function Documentation . . . . .	53
6.32 fifo_queues.h File Reference . . . . .	54
6.32.1 Data Structure Documentation . . . . .	54
6.33 filters.h File Reference . . . . .	55
6.33.1 Function Documentation . . . . .	55
6.34 footprint.h File Reference . . . . .	55
6.34.1 Function Documentation . . . . .	55
6.35 fxt.h File Reference . . . . .	56
6.36 graph.h File Reference . . . . .	61
6.36.1 Data Structure Documentation . . . . .	61
6.36.2 Function Documentation . . . . .	62

6.37 helper_mct.h File Reference . . . . .	63
6.37.1 Data Structure Documentation . . . . .	63
6.38 idle_hook.h File Reference . . . . .	64
6.39 implicit_data_deps.h File Reference . . . . .	64
6.39.1 Function Documentation . . . . .	64
6.40 jobs.h File Reference . . . . .	64
6.40.1 Data Structure Documentation . . . . .	66
6.40.2 Typedef Documentation . . . . .	66
6.40.3 Function Documentation . . . . .	66
6.41 malloc.h File Reference . . . . .	68
6.41.1 Function Documentation . . . . .	68
6.42 memalloc.h File Reference . . . . .	68
6.43 memory_manager.h File Reference . . . . .	68
6.43.1 Function Documentation . . . . .	69
6.44 memory_nodes.h File Reference . . . . .	69
6.44.1 Data Structure Documentation . . . . .	70
6.44.2 Function Documentation . . . . .	71
6.45 memstats.h File Reference . . . . .	71
6.46 mp_common.h File Reference . . . . .	71
6.47 multiple_regression.h File Reference . . . . .	72
6.48 node_ops.h File Reference . . . . .	72
6.49 openmp_runtime_support.h File Reference . . . . .	72
6.49.1 Data Structure Documentation . . . . .	73
6.49.2 Macro Definition Documentation . . . . .	76
6.49.3 Enumeration Type Documentation . . . . .	76
6.50 perfmodel.h File Reference . . . . .	76
6.50.1 Data Structure Documentation . . . . .	77
6.50.2 Macro Definition Documentation . . . . .	78
6.51 prio_deque.h File Reference . . . . .	78
6.51.1 Data Structure Documentation . . . . .	78
6.51.2 Function Documentation . . . . .	79
6.52 prio_list.h File Reference . . . . .	79
6.53 profiling.h File Reference . . . . .	79
6.53.1 Function Documentation . . . . .	80
6.54 progress_hook.h File Reference . . . . .	81
6.55 rbtree.h File Reference . . . . .	81
6.55.1 Macro Definition Documentation . . . . .	82
6.55.2 Function Documentation . . . . .	84
6.56 rbtree_i.h File Reference . . . . .	85
6.56.1 Data Structure Documentation . . . . .	85
6.56.2 Macro Definition Documentation . . . . .	86
6.56.3 Function Documentation . . . . .	86

6.57 regression.h File Reference . . . . .	88
6.58 rwlock.h File Reference . . . . .	88
6.58.1 Data Structure Documentation . . . . .	88
6.58.2 Function Documentation . . . . .	89
6.59 sched_component.h File Reference . . . . .	89
6.59.1 Function Documentation . . . . .	89
6.60 sched_ctx.h File Reference . . . . .	90
6.60.1 Data Structure Documentation . . . . .	91
6.60.2 Function Documentation . . . . .	91
6.61 sched_ctx_list.h File Reference . . . . .	94
6.61.1 Data Structure Documentation . . . . .	94
6.61.2 Function Documentation . . . . .	95
6.62 sched_policy.h File Reference . . . . .	96
6.62.1 Function Documentation . . . . .	97
6.63 simgrid.h File Reference . . . . .	98
6.63.1 Macro Definition Documentation . . . . .	99
6.63.2 Function Documentation . . . . .	99
6.64 sink_common.h File Reference . . . . .	99
6.65 sort_data_handles.h File Reference . . . . .	99
6.65.1 Function Documentation . . . . .	99
6.66 source_common.h File Reference . . . . .	100
6.67 starpu_clusters_create.h File Reference . . . . .	100
6.68 starpu_data_cpy.h File Reference . . . . .	100
6.69 starpu_debug_helpers.h File Reference . . . . .	100
6.69.1 Function Documentation . . . . .	100
6.70 starpu_fxt.h File Reference . . . . .	101
6.71 starpu_parameters.h File Reference . . . . .	101
6.72 starpu_spinlock.h File Reference . . . . .	101
6.72.1 Data Structure Documentation . . . . .	102
6.73 starpu_task_insert_utils.h File Reference . . . . .	102
6.74 tags.h File Reference . . . . .	102
6.74.1 Data Structure Documentation . . . . .	103
6.74.2 Enumeration Type Documentation . . . . .	103
6.74.3 Function Documentation . . . . .	103
6.75 task.h File Reference . . . . .	103
6.75.1 Function Documentation . . . . .	104
6.76 task_bundle.h File Reference . . . . .	105
6.76.1 Data Structure Documentation . . . . .	105
6.76.2 Function Documentation . . . . .	106
6.77 thread.h File Reference . . . . .	107
6.78 timing.h File Reference . . . . .	107
6.78.1 Function Documentation . . . . .	107

6.79 topology.h File Reference . . . . .	108
6.79.1 Function Documentation . . . . .	108
6.80 utils.h File Reference . . . . .	109
6.80.1 Function Documentation . . . . .	111
6.81 uthash.h File Reference . . . . .	111
6.81.1 Data Structure Documentation . . . . .	113
6.82 write_back.h File Reference . . . . .	113
6.82.1 Function Documentation . . . . .	114
<b>7 StarPU MPI File Documentation</b>	<b>115</b>
7.1 starpu_mpi_cache.h File Reference . . . . .	115
7.2 starpu_mpi_driver.h File Reference . . . . .	115
7.3 starpu_mpi_init.h File Reference . . . . .	115
7.4 starpu_mpi_nmad_backend.h File Reference . . . . .	116
7.4.1 Data Structure Documentation . . . . .	116
7.5 starpu_mpi_stats.h File Reference . . . . .	116
7.6 starpu_mpi_cache_stats.h File Reference . . . . .	117
7.7 starpu_mpi_early_data.h File Reference . . . . .	117
7.7.1 Data Structure Documentation . . . . .	117
7.8 starpu_mpi_sync_data.h File Reference . . . . .	118
7.9 starpu_mpi_comm.h File Reference . . . . .	118
7.10 starpu_mpi_early_request.h File Reference . . . . .	119
7.10.1 Data Structure Documentation . . . . .	119
7.11 starpu_mpi_mpi_backend.h File Reference . . . . .	119
7.11.1 Data Structure Documentation . . . . .	120
7.12 starpu_mpi_private.h File Reference . . . . .	120
7.12.1 Data Structure Documentation . . . . .	122
7.13 starpu_mpi_tag.h File Reference . . . . .	123
7.14 starpu_mpi_datatype.h File Reference . . . . .	123
7.15 starpu_mpi_fxt.h File Reference . . . . .	124
7.16 starpu_mpi_select_node.h File Reference . . . . .	125
7.17 starpu_mpi_task_insert.h File Reference . . . . .	125
7.18 load_balancer_policy.h File Reference . . . . .	125
7.19 load_data_interface.h File Reference . . . . .	126
7.19.1 Data Structure Documentation . . . . .	126
7.20 data_movements_interface.h File Reference . . . . .	127
7.20.1 Data Structure Documentation . . . . .	127
<b>8 StarPU Resource Manager File Documentation</b>	<b>129</b>
8.1 starpurm_private.h File Reference . . . . .	129
8.1.1 Data Structure Documentation . . . . .	129

This manual documents the internal usage of StarPU version 1.3.10. Its contents was last updated on 15 November 2022.

Copyright © 2009–2020 Université de Bordeaux, CNRS (LaBRI UMR 5800), Inria

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

# Chapter 1

## Introduction

### 1.1 Motivation





## Chapter 2

# StarPU Core

## 2.1 StarPU Core Entities

TODO

### 2.1.1 Overview

Execution entities:

- **worker**: A worker (see [Workers](#), [Workers and Scheduling Contexts](#)) entity is a CPU thread created by StarPU to manage one computing unit. The computing unit can be a local CPU core, an accelerator or GPU device, or — on the master side when running in master-slave distributed mode — a remote slave computing node. It is responsible for querying scheduling policies for tasks to execute.
- **sched\_context**: A scheduling context (see [Scheduling Contexts](#), [Workers and Scheduling Contexts](#)) is a logical set of workers governed by an instance of a scheduling policy. It defines the computing units to which the scheduling policy instance may assign work entities.
- **driver**: A driver is the set of hardware-dependent routines used by a worker to initialize its associated computing unit, execute work entities on it, and finalize the computing unit usage at the end of the session.

Work entities:

- **task**: A task is a high level work request submitted to StarPU by the application, or internally by StarPU itself.
- **job**: A job is a low level view of a work request. It is not exposed to the application. A job structure may be shared among several task structures in the case of a parallel task.

Data entities:

- **data handle**: A data handle is a high-level, application opaque object designating a piece of data currently registered to the StarPU data management layer. Internally, it is a `_starpu_data_state` structure.
- **data replicate**: A data replicate is a low-level object designating one copy of a piece of data registered to StarPU as a data handle, residing in one memory node managed by StarPU. It is not exposed to the application.

### 2.1.2 Workers

A **worker** is a CPU thread created by StarPU. Its role is to manage one computing unit. This computing unit can be a local CPU core, in which case, the worker thread manages the actual CPU core to which it is assigned; or it can be a computing device such as a GPU or an accelerator (or even a remote computing node when StarPU is running in distributed master-slave mode.) When a worker manages a computing device, the CPU core to which the worker's thread is by default exclusively assigned to the device management work and does not participate to computation.

### 2.1.2.1 States

#### Scheduling operations related state

While a worker is conducting a scheduling operations, e.g. the worker is in the process of selecting a new task to execute, flag `state_sched_op_pending` is set to `!0`, otherwise it is set to `0`.

While `state_sched_op_pending` is `!0`, the following exhaustive list of operations on that workers are restricted in the stated way:

- adding the worker to a context is not allowed;
- removing the worker from a context is not allowed;
- adding the worker to a parallel task team is not allowed;
- removing the worker from a parallel task team is not allowed;
- querying state information about the worker is only allowed while `state_relax_refcnt > 0`;
  - in particular, querying whether the worker is blocked on a parallel team entry is only allowed while `state_relax_refcnt > 0`.

Entering and leaving the `state_sched_op_pending` state is done through calls to `_starpw_worker_enter_sched_op()` and `_starpw_worker_leave_sched_op()` respectively (see these functions in use in functions `_starpw_get_worker_task()` and `_starpw_get_multi_worker_task()`). These calls ensure that any pending conflicting operation deferred while the worker was in the `state_sched_op_pending` state is performed in an orderly manner.

#### Scheduling contexts related states

Flag `state_changing_ctx_notice` is set to `!0` when a thread is about to add the worker to a scheduling context or remove it from a scheduling context, and is currently waiting for a safe window to do so, until the targeted worker is not in a scheduling operation or parallel task operation anymore. This flag set to `!0` will also prevent the targeted worker to attempt a fresh scheduling operation or parallel task operation to avoid starving conditions. However, a scheduling operation that was already in progress before the notice is allowed to complete.

Flag `state_changing_ctx_waiting` is set to `!0` when a scheduling context worker addition or removal involving the targeted worker is about to occur and the worker is currently performing a scheduling operation to tell the targeted worker that the initiator thread is waiting for the scheduling operation to complete and should be woken up upon completion.

#### Relaxed synchronization related states

Any StarPU worker may participate to scheduling operations, and in this process, may be forced to observe state information from other workers. A StarPU worker thread may therefore be observed by any thread, even other StarPU workers. Since workers may observe each other in any order, it is not possible to rely exclusively on the `sched_mutex` of each worker to protect the observation of worker state flags by other workers, because worker A observing worker B would involve locking workers in (A B) sequence, while worker B observing worker A would involve locking workers in (B A) sequence, leading to lock inversion deadlocks.

In consequence, no thread must hold more than one worker's `sched_mutex` at any time. Instead, workers implement a relaxed locking scheme based on the `state_relax_refcnt` counter, itself protected by the worker's `sched_mutex`. When `state_relax_refcnt > 0`, the targeted worker state flags may be observed, otherwise the thread attempting the observation must repeatedly wait on the targeted worker's `sched_cond` condition until `state_relax_refcnt > 0`.

The relaxed mode, while on, can actually be seen as a transactional consistency model, where concurrent accesses are authorized and potential conflicts are resolved after the fact. When the relaxed mode is off, the consistency model becomes a mutual exclusion model, where the `sched_mutex` of the worker must be held in order to access or change the worker state.

#### Parallel tasks related states

When a worker is scheduled to participate to the execution of a parallel task, it must wait for the whole team of workers participating to the execution of this task to be ready. While the worker waits for its teammates, it is not available to run other tasks or perform other operations. Such a waiting operation can therefore not start while conflicting operations such as scheduling operations and scheduling context resizing involving the worker are on-going. Conversely these operations and other may query whether the worker is blocked on a parallel task entry with `starpw_worker_is_blocked_in_parallel()`.

The `starpu_worker_is_blocked_in_parallel()` function is allowed to proceed while and only while `state_relax_refcnt > 0`. Due to the relaxed worker locking scheme, the `state_blocked_in_parallel` flag of the targeted worker may change after it has been observed by an observer thread. In consequence, flag `state_blocked_in_parallel_observed` of the targeted worker is set to 1 by the observer immediately after the observation to "taint" the targeted worker. The targeted worker will clear the `state_blocked_in_parallel_observed` flag tainting and defer the processing of parallel task related requests until a full scheduling operation shot completes without the `state_blocked_in_parallel_observed` flag being tainted again. The purpose of this tainting flag is to prevent parallel task operations to be started immediately after the observation of a transient scheduling state.

Worker's management of parallel tasks is governed by the following set of state flags and counters:

- `state_blocked_in_parallel`: set to !0 while the worker is currently blocked on a parallel task;
- `state_blocked_in_parallel_observed`: set to !0 to taint the worker when a thread has observed the `state_blocked_in_parallel` flag of this worker while its `state_relax_refcnt` state counter was >0. Any pending request to add or remove the worker from a parallel task team will be deferred until a whole scheduling operation shot completes without being tainted again.
- `state_block_in_parallel_req`: set to !0 when a thread is waiting on a request for the worker to be added to a parallel task team. Must be protected by the worker's `sched_mutex`.
- `state_block_in_parallel_ack`: set to !0 by the worker when acknowledging a request for being added to a parallel task team. Must be protected by the worker's `sched_mutex`.
- `state_unblock_in_parallel_req`: set to !0 when a thread is waiting on a request for the worker to be removed from a parallel task team. Must be protected by the worker's `sched_mutex`.
- `state_unblock_in_parallel_ack`: set to !0 by the worker when acknowledging a request for being removed from a parallel task team. Must be protected by the worker's `sched_mutex`.
- `block_in_parallel_ref_count`: counts the number of consecutive pending requests to enter parallel task teams. Only the first of a train of requests for entering parallel task teams triggers the transition of the `state_block_in_parallel_req` flag from 0 to 1. Only the last of a train of requests to leave a parallel task team triggers the transition of flag `state_unblock_in_parallel_req` from 0 to 1. Must be protected by the worker's `sched_mutex`.

### 2.1.2.2 Operations

#### Entry point

All the operations of a worker are handled in an iterative fashion, either by the application code on a thread launched by the application, or automatically by StarPU on a device-dependent CPU thread launched by StarPU. Whether a worker's operation cycle is managed automatically or not is controlled per session by the field `not_launched_drivers` of the `starpu_conf` struct, and is decided in `_starpu_launch_drivers()` function.

When managed automatically, cycles of operations for a worker are handled by the corresponding driver specific `_starpu_<DRV>_worker()` function, where DRV is a driver name such as `cpu` (`_starpu_cpu_worker`) or `cuda` (`_starpu_cuda_worker`), for instance. Otherwise, the application must supply a thread which will repeatedly call `starpu_driver_run_once()` for the corresponding worker.

In both cases, control is then transferred to `_starpu_cpu_driver_run_once()` (or the corresponding driver specific func). The cycle of operations typically includes, at least, the following operations:

- **task scheduling**
- **parallel task team build-up**
- **task input processing**
- **data transfer processing**
- **task execution**

When the worker cycles are handled by StarPU automatically, the iterative operation processing ends when the `running` field of `_starpu_config` becomes false. This field should not be read directly, instead it should be read through the `_starpu_machine_is_running()` function.

### Task scheduling

If the worker does not yet have a queued task, it calls `_starpu_get_worker_task()` to try and obtain a task. This may involve scheduling operations such as stealing a queued but not yet executed task from another worker. The operation may not necessarily succeed if no tasks are ready and/or suitable to run on the worker's computing unit.

### Parallel task team build-up

If the worker has a task ready to run and the corresponding job has a size  $> 1$ , then the task is a parallel job and the worker must synchronize with the other workers participating to the parallel execution of the job to assign a unique rank for each worker. The synchronization is done through the job's `sync_mutex` mutex.

### Task input processing

Before the task can be executed, its input data must be made available on a memory node reachable by the worker's computing unit. To do so, the worker calls `_starpu_fetch_task_input()`

### Data transfer processing

The worker makes pending data transfers (involving memory node(s) that it is driving) progress, with a call to `__starpu_datawizard_progress()`,

### Task execution

Once the worker has a pending task assigned and the input data for that task are available in the memory node reachable by the worker's computing unit, the worker calls `_starpu_cpu_driver_execute_task()` (or the corresponding driver specific function) to proceed to the execution of the task.

## 2.1.3 Scheduling Contexts

A scheduling context is a logical set of workers governed by an instance of a scheduling policy. Tasks submitted to a given scheduling context are confined to the computing units governed by the workers belonging to this scheduling context at the time they get scheduled.

A scheduling context is identified by an unsigned integer identifier between 0 and `STARPU_NMAX_SCHED_CTXS - 1`. The `STARPU_NMAX_SCHED_CTXS` identifier value is reserved to indicated an unallocated, invalid or deleted scheduling context.

Accesses to the scheduling context structure are governed by a multiple-readers/single-writer lock (`rwlock` field). Changes to the structure contents, additions or removals of workers, statistics updates, all must be done with proper exclusive write access.

## 2.1.4 Workers and Scheduling Contexts

A worker can be assigned to one or more **scheduling contexts**. It exclusively receives tasks submitted to the scheduling context(s) it is currently assigned at the time such tasks are scheduled. A worker may add itself to or remove itself from a scheduling context.

### Locking and synchronization rules between workers and scheduling contexts

A thread currently holding a worker `sched_mutex` must not attempt to acquire a scheduling context `rwlock`, neither for writing nor for reading. Such an attempt constitutes a lock inversion and may result in a deadlock.

A worker currently in a scheduling operation must enter the relaxed state before attempting to acquire a scheduling context `rwlock`, either for reading or for writing.

When the set of workers assigned to a scheduling context is about to be modified, all the workers in the union between the workers belonging to the scheduling context before the change and the workers expected to belong to the scheduling context after the change must be notified using the `notify_workers_about_changing_ctx_pending()` function prior to the update. After the update, all the workers in that same union must be notified for the update completion with a call to `notify_workers_about_changing_ctx_done()`.

The function `notify_workers_about_changing_ctx_pending()` places every worker passed in argument in a state compatible with changing the scheduling context assignment of that worker, possibly blocking until that worker leaves incompatible states such as a pending scheduling operation. If the caller of `notify_workers_about_changing_ctx_pending()` is itself a worker included in the set of workers passed in argument, it does not notify itself, with the assumption that the worker is already calling `notify_workers_about_changing_ctx_pending()` from a state compatible with a scheduling context assignment update. Once a worker has

been notified about a scheduling context change pending, it cannot proceed with incompatible operations such as a scheduling operation until it receives a notification that the context update operation is complete.

### 2.1.5 Drivers

Each driver defines a set of routines depending on some specific hardware. These routines include hardware discovery/initialization, task execution, device memory management and data transfers.

While most hardware dependent routines are in source files located in the `/src/drivers` subdirectory of the StarPU tree, some can be found elsewhere in the tree such as `src/datawizard/malloc.c` for memory allocation routines or the subdirectories of `src/datawizard/interfaces/` for data transfer routines.

The driver ABI defined in the `_starpu_driver_ops` structure includes the following operations:

- `.init`: initialize a driver instance for the calling worker managing a hardware computing unit compatible with this driver.
- `.run_once`: perform a single driver progress cycle for the calling worker (see [Operations](#)).
- `.deinit`: deinitialize the driver instance for the calling worker
- `.run`: executes the following sequence automatically: call `.init`, repeatedly call `.run_once` until the function `_starpu_machine_is_running()` returns false, call `.deinit`.

The source code common to all drivers is shared in `src/drivers/driver_common/driver_↔common.[ch]`. This file includes services such as grabbing a new task to execute on a worker, managing statistics accounting on job startup and completion and updating the worker status

#### 2.1.5.1 Master/Slave Drivers

A subset of the drivers corresponds to drivers managing computing units in master/slave mode, that is, drivers involving a local master instance managing one or more remote slave instances on the targeted device(s). This includes devices such as discrete manycore accelerators (e.g. Intel's Knight Corners board, for instance), or pseudo devices such as a cluster of cpu nodes driver through StarPU's MPI master/slave mode. A driver instance on the master side is named the **source**, while a driver instances on the slave side is named the **sink**.

A significant part of the work realized on the source and sink sides of master/slave drivers is identical among all master/slave drivers, due to the similarities in the software pattern. Therefore, many routines are shared among all these drivers in the `src/drivers/mp_common` subdirectory. In particular, a set of default commands to be used between sources and sinks is defined, assuming the availability of some communication channel between them (see `enum _starpu_mp_command`)

TODO

### 2.1.6 Tasks and Jobs

TODO

### 2.1.7 Data

TODO



## Chapter 3

# Module Index

### 3.1 Modules

Here is a list of all modules:

Workers . . . . .	13
-------------------	----





## Chapter 4

# Module Documentation

### 4.1 Workers



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">barrier.h</a>	19
<a href="#">barrier_counter.h</a>	19
<a href="#">bound.h</a>	20
<a href="#">cg.h</a>	21
<a href="#">coherency.h</a>	23
<a href="#">combined_workers.h</a>	30
<a href="#">config.h</a>	30
<a href="#">copy_driver.h</a>	36
<a href="#">data_concurrency.h</a>	38
<a href="#">data_interface.h</a>	38
<a href="#">data_movements_interface.h</a>	127
<a href="#">data_request.h</a>	40
<a href="#">datastats.h</a>	40
<a href="#">datawizard.h</a>	41
<a href="#">debug.h</a>	41
<a href="#">detect_combined_workers.h</a>	42
<a href="#">disk.h</a>	43
<a href="#">disk_unistd_global.h</a>	45
<a href="#">driver_common.h</a>	46
<a href="#">driver_cpu.h</a>	46
<a href="#">driver_cuda.h</a>	46
<a href="#">driver_disk.h</a>	48
<a href="#">driver_mic_common.h</a>	48
<a href="#">driver_mic_sink.h</a>	49
<a href="#">driver_mic_source.h</a>	50
<a href="#">driver_mpi_common.h</a>	51
<a href="#">driver_mpi_sink.h</a>	51
<a href="#">driver_mpi_source.h</a>	51
<a href="#">driver_opengl.h</a>	51
<a href="#">driver_opengl_utils.h</a>	52
<a href="#">drivers.h</a>	53
<a href="#">errorcheck.h</a>	53
<a href="#">fifo_queues.h</a>	54
<a href="#">filters.h</a>	55
<a href="#">footprint.h</a>	55
<a href="#">fxt.h</a>	56
<a href="#">graph.h</a>	61
<a href="#">helper_mct.h</a>	63
<a href="#">idle_hook.h</a>	64
<a href="#">implicit_data_deps.h</a>	64

jobs.h	64
list.h	??
load_balancer_policy.h	125
load_data_interface.h	126
malloc.h	68
memalloc.h	68
memory_manager.h	68
memory_nodes.h	69
memstats.h	71
mp_common.h	71
multiple_regression.h	72
node_ops.h	72
openmp_runtime_support.h	72
perfmodel.h	76
prio_deque.h	78
prio_list.h	79
profiling.h	79
progress_hook.h	81
rbtree.h	81
rbtree_i.h	85
regression.h	88
rwlock.h	88
sched_component.h	89
sched_ctx.h	90
sched_ctx_list.h	94
sched_policy.h	96
simgrid.h	98
sink_common.h	99
sort_data_handles.h	99
source_common.h	100
starpu_clusters_create.h	100
starpu_data_cpy.h	100
starpu_debug_helpers.h	100
starpu_fxt.h	101
starpu_mpi_cache.h	115
starpu_mpi_cache_stats.h	117
starpu_mpi_comm.h	118
starpu_mpi_datatype.h	123
starpu_mpi_driver.h	115
starpu_mpi_early_data.h	117
starpu_mpi_early_request.h	119
starpu_mpi_fxt.h	124
starpu_mpi_init.h	115
starpu_mpi_mpi_backend.h	119
starpu_mpi_nmad_backend.h	116
starpu_mpi_private.h	120
starpu_mpi_select_node.h	125
starpu_mpi_stats.h	116
starpu_mpi_sync_data.h	118
starpu_mpi_tag.h	123
starpu_mpi_task_insert.h	125
starpu_parameters.h	101
starpu_spinlock.h	101
starpu_task_insert_utils.h	102
starpurm_private.h	129
tags.h	102
task.h	103
task_bundle.h	105

<a href="#">thread.h</a>	107
<a href="#">timing.h</a>	107
<a href="#">topology.h</a>	108
<a href="#">uthash.h</a>	111
<a href="#">utils.h</a>	109
<b>workers.h</b>	<b>??</b>
<a href="#">write_back.h</a>	113



## Chapter 6

# StarPU File Documentation

### 6.1 barrier.h File Reference

```
#include <starpu_thread.h>
```

#### Data Structures

- [struct \\_starpu\\_barrier](#)

#### Functions

- `int _starpu_barrier_init` ([struct \\_starpu\\_barrier](#) \*barrier, int count)
- `int _starpu_barrier_destroy` ([struct \\_starpu\\_barrier](#) \*barrier)
- `int _starpu_barrier_wait` ([struct \\_starpu\\_barrier](#) \*barrier)

#### 6.1.1 Data Structure Documentation

##### 6.1.1.1 struct \_starpu\_barrier

###### Data Fields

unsigned	count	
unsigned	reached_start	
unsigned	reached_exit	
double	reached_flops	
starpu_pthread_mutex_t	mutex	
starpu_pthread_mutex_t	mutex_exit	
starpu_pthread_cond_t	cond	

### 6.2 barrier\_counter.h File Reference

```
#include <common/utils.h>  
#include <common/barrier.h>
```

#### Data Structures

- [struct \\_starpu\\_barrier\\_counter](#)



## Functions

- `int _starpu_barrier_counter_init (struct _starpu_barrier_counter *barrier_c, unsigned count)`
- `int _starpu_barrier_counter_destroy (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_wait_for_empty_counter (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_wait_until_counter_reaches_down_to_n (struct _starpu_barrier_counter *barrier_c, unsigned n)`
- `int _starpu_barrier_counter_wait_until_counter_reaches_up_to_n (struct _starpu_barrier_counter *barrier_c, unsigned n)`
- `int _starpu_barrier_counter_wait_for_full_counter (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_decrement_until_empty_counter (struct _starpu_barrier_counter *barrier_c, double flops)`
- `int _starpu_barrier_counter_increment_until_full_counter (struct _starpu_barrier_counter *barrier_c, double flops)`
- `int _starpu_barrier_counter_increment (struct _starpu_barrier_counter *barrier_c, double flops)`
- `int _starpu_barrier_counter_check (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_get_reached_start (struct _starpu_barrier_counter *barrier_c)`
- `double _starpu_barrier_counter_get_reached_flops (struct _starpu_barrier_counter *barrier_c)`

## 6.2.1 Data Structure Documentation

### 6.2.1.1 struct \_starpu\_barrier\_counter

#### Data Fields

<code>struct _starpu_barrier</code>	barrier	
unsigned	min_threshold	
unsigned	max_threshold	
<code>starpu_pthread_cond_t</code>	cond2	

## 6.3 bound.h File Reference

```
#include <starpu.h>
#include <starpu_bound.h>
#include <core/jobs.h>
```

## Functions

- `void _starpu_bound_record (struct _starpu_job *j)`
- `void _starpu_bound_tag_dep (starpu_tag_t id, starpu_tag_t dep_id)`
- `void _starpu_bound_task_dep (struct _starpu_job *j, struct _starpu_job *dep_j)`
- `void _starpu_bound_job_id_dep (starpu_data_handle_t handle, struct _starpu_job *dep_j, unsigned long job_id)`
- `void starpu_bound_clear (void)`

## Variables

- `int _starpu_bound_recording`

### 6.3.1 Function Documentation

**6.3.1.1 \_starpu\_bound\_record()**

```
void _starpu_bound_record (
    struct _starpu_job * j )
```

Record task for bound computation

**6.3.1.2 \_starpu\_bound\_tag\_dep()**

```
void _starpu_bound_tag_dep (
    starpu_tag_t id,
    starpu_tag_t dep_id )
```

Record tag dependency: id depends on dep\_id

**6.3.1.3 \_starpu\_bound\_task\_dep()**

```
void _starpu_bound_task_dep (
    struct _starpu_job * j,
    struct _starpu_job * dep_j )
```

Record task dependency: j depends on dep\_j

**6.3.1.4 \_starpu\_bound\_job\_id\_dep()**

```
void _starpu_bound_job_id_dep (
    starpu_data_handle_t handle,
    struct _starpu_job * dep_j,
    unsigned long job_id )
```

Record job id dependency: j depends on job\_id

**6.3.1.5 starpu\_bound\_clear()**

```
void starpu_bound_clear (
    void )
```

Clear recording

**6.3.2 Variable Documentation****6.3.2.1 \_starpu\_bound\_recording**

```
int _starpu_bound_recording [extern]
```

Are we recording?

**6.4 cg.h File Reference**

```
#include <starpu.h>
#include <common/config.h>
```

**Data Structures**

- struct [\\_starpu\\_cg\\_list](#)
- struct [\\_starpu\\_cg](#)
- union [\\_starpu\\_cg.succ](#)
- struct [\\_starpu\\_cg.succ.succ\\_apps](#)

**Macros**

- #define [STARPU\\_DYNAMIC\\_DEPS\\_SIZE](#)

## Typedefs

- typedef [struct](#) \_starpu\_notify\_job\_start\_data **\_starpu\_notify\_job\_start\_data**

## Enumerations

- enum **\_starpu\_cg\_type** { STARPU\_CG\_APPS , STARPU\_CG\_TAG , STARPU\_CG\_TASK }

## Functions

- void **\_starpu\_notify\_dependencies** ([struct](#) \_starpu\_job \*j)
- void **\_starpu\_job\_notify\_start** ([struct](#) \_starpu\_job \*j, [struct](#) starpu\_perfmodel\_arch \*perf\_arch)
- void **\_starpu\_job\_notify\_ready\_soon** ([struct](#) \_starpu\_job \*j, \_starpu\_notify\_job\_start\_data \*data)
- void **\_starpu\_cg\_list\_init** ([struct](#) \_starpu\_cg\_list \*list)
- void **\_starpu\_cg\_list\_deinit** ([struct](#) \_starpu\_cg\_list \*list)
- int **\_starpu\_add\_successor\_to\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, [struct](#) \_starpu\_cg \*cg)
- int **\_starpu\_list\_task\_successors\_in\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, unsigned ndeps, [struct](#) starpu\_task \*task\_array[])
- int **\_starpu\_list\_task\_scheduled\_successors\_in\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, unsigned ndeps, [struct](#) starpu\_task \*task\_array[])
- int **\_starpu\_list\_tag\_successors\_in\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, unsigned ndeps, starpu\_tag\_t tag\_array[])
- void **\_starpu\_notify\_cg** (void \*pred, [struct](#) \_starpu\_cg \*cg)
- void **\_starpu\_notify\_cg\_list** (void \*pred, [struct](#) \_starpu\_cg\_list \*successors)
- void **\_starpu\_notify\_job\_start\_cg\_list** (void \*pred, [struct](#) \_starpu\_cg\_list \*successors, \_starpu\_notify\_job\_start\_data \*data)
- void **\_starpu\_notify\_task\_dependencies** ([struct](#) \_starpu\_job \*j)
- void **\_starpu\_notify\_job\_start\_tasks** ([struct](#) \_starpu\_job \*j, \_starpu\_notify\_job\_start\_data \*data)

## 6.4.1 Data Structure Documentation

### 6.4.1.1 [struct](#) \_starpu\_cg\_list

Completion Group list, records both the number of expected notifications before the completion can start, and the list of successors when the completion is finished.

#### Data Fields

<a href="#">struct</a> _starpu_spinlock	lock	Protects atomicity of the list and the terminated flag
unsigned	ndeps	Number of notifications to be waited for
unsigned	ndeps_completed	
unsigned	terminated	Whether the completion is finished. For restartable/restarted tasks, only the first iteration is taken into account here.
unsigned	nsuccs	List of successors
unsigned	succ_list_size	How many allocated items in succ
<a href="#">struct</a> _starpu_cg **	succ	

### 6.4.1.2 [struct](#) \_starpu\_cg

Completion Group

#### Data Fields

unsigned	ntags	number of tags depended on
unsigned	remaining	number of remaining tags
enum _starpu_cg_type	cg_type	

## Data Fields

union <a href="#">_starpu_cg</a>	succ	
----------------------------------	------	--

6.4.1.3 union [\\_starpu\\_cg.succ](#)

## Data Fields

<a href="#">struct _starpu_tag *</a>	tag	STARPU_CG_TAG
<a href="#">struct _starpu_job *</a>	job	STARPU_CG_TASK
<a href="#">succ</a>	succ_apps	STARPU_CG_APPS in case this completion group is related to an application, we have to explicitly wake the waiting thread instead of reschedule the corresponding task

6.4.1.4 struct [\\_starpu\\_cg.succ.succ\\_apps](#)

STARPU\_CG\_APPS in case this completion group is related to an application, we have to explicitly wake the waiting thread instead of reschedule the corresponding task

## Data Fields

unsigned	completed	
<a href="#">starpu_pthread_mutex_t</a>	cg_mutex	
<a href="#">starpu_pthread_cond_t</a>	cg_cond	

## 6.4.2 Macro Definition Documentation

## 6.4.2.1 STARPU\_DYNAMIC\_DEPS\_SIZE

```
#define STARPU_DYNAMIC_DEPS_SIZE
```

we do not necessarily want to allocate room for 256 dependencies, but we want to handle the few situation where there are a lot of dependencies as well

## 6.5 coherency.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <common/rwlock.h>
#include <common/timing.h>
#include <common/fxt.h>
#include <common/list.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/datastats.h>
#include <datawizard/memstats.h>
#include <datawizard/data_request.h>
```

## Data Structures

- [struct \\_starpu\\_data\\_replicate](#)

- [struct \\_starpu\\_jobid\\_list](#)
- [struct \\_starpu\\_task\\_wrapper\\_list](#)
- [struct \\_starpu\\_task\\_wrapper\\_dlist](#)
- [struct \\_starpu\\_data\\_state](#)

## Typedefs

- `typedef void(*_starpu_data_handle_unregister_hook) (starpu_data_handle_t)`

## Enumerations

- `enum _starpu_cache_state { STARPU_OWNER , STARPU_SHARED , STARPU_INVALID }`

## Functions

- `int _starpu_fetch_data_on_node` (starpu\_data\_handle\_t handle, int node, [struct \\_starpu\\_data\\_replicate](#) \*replicate, enum starpu\_data\_access\_mode mode, unsigned detached, enum [\\_starpu\\_is\\_prefetch](#) is\_↵ prefetch, unsigned async, void(\*callback\_func)(void \*), void \*callback\_arg, int prio, const char \*origin)
- `void _starpu_release_data_on_node` ([struct \\_starpu\\_data\\_state](#) \*state, uint32\_t default\_wt\_mask, [struct \\_starpu\\_data\\_replicate](#) \*replicate)
- `void _starpu_update_data_state` (starpu\_data\_handle\_t handle, [struct \\_starpu\\_data\\_replicate](#) \*requesting\_↵ \_replicate, enum starpu\_data\_access\_mode mode)
- `uint32_t _starpu_get_data_refcnt` ([struct \\_starpu\\_data\\_state](#) \*state, unsigned node)
- `size_t _starpu_data_get_size` (starpu\_data\_handle\_t handle)
- `size_t _starpu_data_get_alloc_size` (starpu\_data\_handle\_t handle)
- `uint32_t _starpu_data_get_footprint` (starpu\_data\_handle\_t handle)
- `void __starpu_push_task_output` ([struct \\_starpu\\_job](#) \*j)
- `void _starpu_push_task_output` ([struct \\_starpu\\_job](#) \*j)
- `void _starpu_release_nowhere_task_output` ([struct \\_starpu\\_job](#) \*j)
- `STARPU_ATTRIBUTE_WARN_UNUSED_RESULT int _starpu_fetch_task_input` ([struct](#) starpu\_task \*task, [struct \\_starpu\\_job](#) \*j, int async)
- `void _starpu_fetch_task_input_tail` ([struct](#) starpu\_task \*task, [struct \\_starpu\\_job](#) \*j, [struct \\_starpu\\_worker](#) \*worker)
- `void _starpu_fetch_nowhere_task_input` ([struct \\_starpu\\_job](#) \*j)
- `int _starpu_select_src_node` ([struct \\_starpu\\_data\\_state](#) \*state, unsigned destination)
- `int _starpu_determine_request_path` (starpu\_data\_handle\_t handle, int src\_node, int dst\_node, enum starpu\_data\_access\_mode mode, int max\_len, unsigned \*src\_nodes, unsigned \*dst\_nodes, unsigned \*handling\_nodes, unsigned write\_invalidation)
- `struct _starpu_data_request * _starpu_create_request_to_fetch_data` (starpu\_data\_handle\_t handle, [struct \\_starpu\\_data\\_replicate](#) \*dst\_replicate, enum starpu\_data\_access\_mode mode, enum [\\_starpu\\_is\\_prefetch](#) is\_↵ prefetch, unsigned async, void(\*callback\_func)(void \*), void \*callback\_arg, int prio, const char \*origin)
- `void _starpu_redux_init_data_replicate` (starpu\_data\_handle\_t handle, [struct \\_starpu\\_data\\_replicate](#) \*replicate, int workerid)
- `void _starpu_data_start_reduction_mode` (starpu\_data\_handle\_t handle)
- `void _starpu_data_end_reduction_mode` (starpu\_data\_handle\_t handle)
- `void _starpu_data_end_reduction_mode_terminate` (starpu\_data\_handle\_t handle)
- `void _starpu_data_set_unregister_hook` (starpu\_data\_handle\_t handle, \_starpu\_data\_handle\_↵ unregister\_hook func)

## Variables

- `int _starpu_has_not_important_data`

### 6.5.1 Data Structure Documentation

#### 6.5.1.1 struct \_starpu\_data\_replicate

this should contain the information relative to a given data replicate

## Data Fields

starpu_data_handle_t	handle	
void *	data_interface	describe the actual data layout, as manipulated by data interfaces in *_interface.c
int	refcnt	How many requests or tasks are currently working with this replicate
char	memory_node	
enum _starpu_cache_state	state: 2	describes the state of the local data in term of coherency
unsigned	relaxed_coherency:2	A buffer that is used for SCRATCH or reduction cannot be used with filters.
unsigned	initialized:1	We may need to initialize the replicate with some value before using it.
unsigned	allocated:1	is the data locally allocated ?
unsigned	automatically_allocated:1	was it automatically allocated ? (else it's the application-provided buffer, don't ever try to free it!) perhaps the allocation was perform higher in the hierarchy for now this is just translated into !automatically_allocated
struct _starpu_data_request *	request[STARPU_MAXNODES]	To help the scheduling policies to make some decision, we may keep a track of the tasks that are likely to request this data on the current node. It is the responsibility of the scheduling _policy_ to set that flag when it assigns a task to a queue, policies which do not use this hint can simply ignore it.
struct _starpu_mem_chunk *	mc	Pointer to memchunk for LRU strategy

## 6.5.1.2 struct \_starpu\_jobid\_list

## Data Fields

unsigned long	id	
struct _starpu_jobid_list *	next	

## 6.5.1.3 struct \_starpu\_task\_wrapper\_list

This structure describes a simply-linked list of task

## Data Fields

struct starpu_task *	task	
struct _starpu_task_wrapper_list *	next	

## 6.5.1.4 struct \_starpu\_task\_wrapper\_dlist

This structure describes a doubly-linked list of task

## Data Fields

<code>struct starpu_task *</code>	<code>task</code>	
<code>struct _starpu_task_wrapper_dlist *</code>	<code>next</code>	
<code>struct _starpu_task_wrapper_dlist *</code>	<code>prev</code>	

6.5.1.5 `struct _starpu_data_state`

This is initialized in both `_starpu_register_new_data` and `_starpu_data_partition`

## Data Fields

<code>int</code>	<code>magic</code>	
<code>struct _starpu_data_requester_prio_list</code>	<code>req_list</code>	
<code>unsigned</code>	<code>refcnt</code>	the number of requests currently in the scheduling engine (not in the <code>req_list</code> anymore), i.e. the number of holders of the current_mode rwlock
<code>unsigned</code>	<code>unlocking_reqs</code>	whether we are already unlocking data requests
<code>enum starpu_data_access_mode</code>	<code>current_mode</code>	Current access mode. Is always either <code>STARPU_R</code> , <code>STARPU_W</code> , <code>STARPU_SCRATCH</code> or <code>STARPU_REDUX</code> , but never a combination such as <code>STARPU_RW</code> .
<code>struct _starpu_spinlock</code>	<code>header_lock</code>	protect meta data
<code>unsigned</code>	<code>busy_count</code>	Condition to make application wait for all transfers before freeing handle <code>busy_count</code> is the number of <code>handle-&gt;refcnt</code> , <code>handle-&gt;per_node[*]-&gt;refcnt</code> , number of <code>starpu_data_requesters</code> , and number of tasks that have released it but are still registered on the implicit data dependency lists. Core code which releases <code>busy_count</code> has to call <code>_starpu_data_check_not_busy</code> to let <code>starpu_data_unregister</code> proceed
<code>unsigned</code>	<code>busy_waiting</code>	Is <code>starpu_data_unregister</code> waiting for <code>busy_count</code> ?
<code>starpu_pthread_mutex_t</code>	<code>busy_mutex</code>	
<code>starpu_pthread_cond_t</code>	<code>busy_cond</code>	
<code>struct _starpu_data_state *</code>	<code>root_handle</code>	In case we use filters, the handle may describe a sub-data
<code>struct _starpu_data_state *</code>	<code>father_handle</code>	root of the tree
<code>starpu_data_handle_t *</code>	<code>active_children</code>	father of the node, NULL if the current node is the root
<code>starpu_data_handle_t **</code>	<code>active_readonly_children</code>	The currently active set of read-write children

## Data Fields

unsigned	nactive_readonly_children	The currently active set of read-only children
unsigned	nsiblings	Size of active_readonly_children array Our siblings in the father partitioning
starpu_data_handle_t *	siblings	How many siblings
unsigned	sibling_index	
unsigned	depth	indicate which child this node is from the father's perspective (if any)
starpu_data_handle_t	children	what's the depth of the tree ? Synchronous partitioning
unsigned	nchildren	
unsigned	nplans	How many partition plans this handle has
struct starpu_codelet *	switch_cl	Switch codelet for asynchronous partitioning
unsigned	switch_cl_nparts	size of dyn_nodes recorded in switch_cl
unsigned	partitioned	Whether a partition plan is currently submitted and the corresponding unpartition has not been yet Or the number of partition plans currently submitted in readonly mode.
unsigned	readonly:1	Whether a partition plan is currently submitted in readonly mode
unsigned	active:1	Whether our father is currently partitioned into ourself
unsigned	active_ro:1	
struct_starpu_data_replicate	per_node[STARPU_MAXNODES]	describe the state of the data in term of coherency
struct_starpu_data_replicate *	per_worker	
struct_starpu_data_interface_ops *	ops	
uint32_t	footprint	Footprint which identifies data layout
int	home_node	where is the data home, i.e. which node it was registered from ? -1 if none yet
uint32_t	wt_mask	what is the default write-through mask for that data ?
unsigned	is_not_important	in some case, the application may explicitly tell StarPU that a piece of data is not likely to be used soon again
unsigned	sequential_consistency	Does StarPU have to enforce some implicit data-dependencies ?
unsigned	initialized	Is the data initialized, or a task is already submitted to initialize it



## Data Fields

unsigned	ooc	Can the data be pushed to the disk?
starpu_pthread_mutex_t	sequential_consistency_mutex	This lock should protect any operation to enforce sequential_consistency
enum starpu_data_access_mode	last_submitted_mode	The last submitted task (or application data request) that declared it would modify the piece of data ? Any task accessing the data in a read-only mode should depend on that task implicitly if the sequential_consistency flag is enabled.
struct starpu_task *	last_sync_task	
struct_starpu_task_wrapper_dlist	last_submitted_accessors	
unsigned	last_submitted_ghost_sync_id_is_valid	If FxT is enabled, we keep track of "ghost dependencies": that is to say the dependencies that are not needed anymore, but that should appear in the post-mortem DAG. For instance if we have the sequence f(Aw) g(Aw), and that g is submitted after the termination of f, we want to have f->g appear in the DAG even if StarPU does not need to enforce this dependency anymore.
unsigned long	last_submitted_ghost_sync_id	
struct_starpu_jobid_list *	last_submitted_ghost_accessors_id	
struct_starpu_task_wrapper_list *	post_sync_tasks	protected by sequential_consistency_mutex
unsigned	post_sync_tasks_cnt	
struct_starpu_codelet *	redu_x_cl	During reduction we need some specific methods: redu_x_func performs the reduction of an interface into another one (eg. "+="), and init_func initializes the data interface to a default value that is stable by reduction (eg. 0 for "+=").
struct_starpu_codelet *	init_cl	
unsigned	reduction_refcnt	Are we currently performing a reduction on that handle ? If so the reduction_refcnt should be non null until there are pending tasks that are performing the reduction.
struct_starpu_data_requester_prio_list	reduction_req_list	List of requesters that are specific to the pending reduction. This list is used when the requests in the req_list list are frozen until the end of the reduction.
starpu_data_handle_t *	reduction_tmp_handles	
struct_starpu_data_request *	write_invalidation_req	Final request for write invalidation

## Data Fields

unsigned	lazy_unregister	
unsigned	removed_from_context_hash	
void *	mpi_data	Used for MPI
_starpu_memory_stats_t	memory_stats	
unsigned int	mf_node	
_starpu_data_handle_unregister_hook	unregister_hook	hook to be called when unregistering the data
struct starpu_arbiter *	arbiter	
struct _starpu_data_requester_prio_list	arbitered_req_list	This is protected by the arbiter mutex
int	last_locality	Data maintained by schedulers themselves Last worker that took this data in locality mode, or -1 if nobody took it yet
unsigned	dimensions	Application-provided coordinates. The maximum dimension (5) is relatively arbitrary.
int	coordinates[5]	
void *	user_data	A generic pointer to data in the user land (could be anything and this is not manage by StarPU)

## 6.5.2 Function Documentation

## 6.5.2.1 \_starpu\_fetch\_data\_on\_node()

```
int _starpu_fetch_data_on_node (
    starpu_data_handle_t handle,
    int node,
    struct _starpu_data_replicate * replicate,
    enum starpu_data_access_mode mode,
    unsigned detached,
    enum _starpu_is_prefetch is_prefetch,
    unsigned async,
    void(*) (void *) callback_func,
    void * callback_arg,
    int prio,
    const char * origin )
```

This does not take a reference on the handle, the caller has to do it, e.g. through `_starpu_attempt_to_submit_data_request_from_apps()` detached means that the core is allowed to drop the request. The caller should thus *\*not\** take a reference since it can not know whether the request will complete async means that `_starpu_fetch_data_on_node` will wait for completion of the request

## 6.5.2.2 \_starpu\_release\_data\_on\_node()

```
void _starpu_release_data_on_node (
    struct _starpu_data_state * state,
    uint32_t default_wt_mask,
    struct _starpu_data_replicate * replicate )
```

This releases a reference on the handle

### 6.5.2.3 `_starpu_push_task_output()`

```
void _starpu_push_task_output (
    struct _starpu_job * j )
```

Version with driver trace

### 6.5.2.4 `_starpu_create_request_to_fetch_data()`

```
struct _starpu_data_request* _starpu_create_request_to_fetch_data (
    starpu_data_handle_t handle,
    struct _starpu_data_replicate * dst_replicate,
    enum starpu_data_access_mode mode,
    enum _starpu_is_prefetch is_prefetch,
    unsigned async,
    void(*) (void *) callback_func,
    void * callback_arg,
    int prio,
    const char * origin )
```

`is_prefetch` is whether the DSM may drop the request (when there is not enough memory for instance `async` is whether the caller wants a reference on the last request, to be able to wait for it (which will release that reference).

## 6.6 `combined_workers.h` File Reference

```
#include <starpu.h>
#include <common/config.h>
```

## 6.7 `config.h` File Reference

### Macros

- `#define CONFIG_FUT`
- `#define HAVE_AIO_H`
- `#define HAVE_AYUDAME_H`
- `#define HAVE_CBLAS_SGEMV`
- `#define HAVE_CLENQUEUEMARKERWITHWAITLIST`
- `#define HAVE_CLGETEXTENSIONFUNCTIONADDRESSFORPLATFORM`
- `#define HAVE_CLOCK_GETTIME`
- `#define HAVE_CL_CL_EXT_H`
- `#define HAVE_COPY_FILE_RANGE`
- `#define HAVE_CUDA_GL_INTEROP_H`
- `#define HAVE_CXX11`
- `#define HAVE_DECL_CUSPARSESETSTREAM`
- `#define HAVE_DECL_ENABLE_FUT_FLUSH`
- `#define HAVE_DECL_FUT_SETUP_FLUSH_CALLBACK`
- `#define HAVE_DECL_FUT_SET_FILENAME`
- `#define HAVE_DECL_HWLOC_CUDA_GET_DEVICE_OSDEV_BY_INDEX`
- `#define HAVE_DECL_NVMLDEVICEGETTOTAENERGYCONSUMPTION`
- `#define HAVE_DECL_SMPI_PROCESS_SET_USER_DATA`
- `#define HAVE_DLB_H`
- `#define HAVE_DLFCN_H`
- `#define HAVE_ENABLE_FUT_FLUSH`
- `#define HAVE_FUT_SETUP_FLUSH_CALLBACK`
- `#define HAVE_FUT_SET_FILENAME`
- `#define HAVE_FXT_CLOSE`
- `#define HAVE_GETRLIMIT`

- `#define HAVE_GLPK_H`
- `#define HAVE_HDF5_H`
- `#define HAVE_HWLOC_CPUKINDS_GET_NR`
- `#define HAVE_HWLOC_GLIBC_SCHED_H`
- `#define HAVE_HWLOC_TOPOLOGY_DUP`
- `#define HAVE_HWLOC_TOPOLOGY_SET_COMPONENTS`
- `#define HAVE_INTTYPES_H`
- `#define HAVE_LEVELDB_DB_H`
- `#define HAVE_LIBATLAS`
- `#define HAVE_LIBBLAS_OPENBLAS`
- `#define HAVE_LIBCBLAS`
- `#define HAVE_LIBCUSPARSE`
- `#define HAVE_LIBDLB`
- `#define HAVE_LIBGFORTTRAN`
- `#define HAVE_LIBGL`
- `#define HAVE_LIBGLPK`
- `#define HAVE_LIBGLU`
- `#define HAVE_LIBGLUT`
- `#define HAVE_LIBGOTO`
- `#define HAVE_LIBGOTO2`
- `#define HAVE_LIBHDF5`
- `#define HAVE_LIBIFCORE`
- `#define HAVE_LIBLEVELDB`
- `#define HAVE_LIBOPENBLAS`
- `#define HAVE_LIBRT`
- `#define HAVE_LIBSIMGRID`
- `#define HAVE_LIBWS2_32`
- `#define HAVE_MALLOC_H`
- `#define HAVE_MEMALIGN`
- `#define HAVE_MKDTEMP`
- `#define HAVE_MKOSTEMP`
- `#define HAVE_MPI_COMM_F2C`
- `#define HAVE_MSG_ENVIRONMENT_GET_ROUTING_ROOT`
- `#define HAVE_MSG_GET_AS_BY_NAME`
- `#define HAVE_MSG_HOST_GET_SPEED`
- `#define HAVE_MSG_MSG_H`
- `#define HAVE_MSG_PROCESS_ATTACH`
- `#define HAVE_MSG_PROCESS_SELF_NAME`
- `#define HAVE_MSG_PROCESS_USERDATA_INIT`
- `#define HAVE_MSG_ZONE_GET_BY_NAME`
- `#define HAVE_MSG_ZONE_GET_HOSTS`
- `#define HAVE_NVMLDEVICEGETTOTAENERGYCONSUMPTION`
- `#define HAVE_PIOM_LTASK_SET_BOUND_THREAD_OS_INDEXES`
- `#define HAVE_POSIX_MEMALIGN`
- `#define HAVE_POTI_INIT_CUSTOM`
- `#define HAVE_POTI_USER_NEWEVENT`
- `#define HAVE_PREAD`
- `#define HAVE_PTHREAD_SETAFFINITY_NP`
- `#define HAVE_PTHREAD_SPIN_LOCK`
- `#define HAVE_PWRITE`
- `#define HAVE_SCANDIR`
- `#define HAVE_SG_ACTOR_ATTACH`
- `#define HAVE_SG_ACTOR_DATA`
- `#define HAVE_SG_ACTOR_EXECUTE`
- `#define HAVE_SG_ACTOR_GET_DATA`

- #define HAVE\_SG\_ACTOR\_INIT
- #define HAVE\_SG\_ACTOR\_ON\_EXIT
- #define HAVE\_SG\_ACTOR\_REF
- #define HAVE\_SG\_ACTOR\_SELF
- #define HAVE\_SG\_ACTOR\_SELF\_EXECUTE
- #define HAVE\_SG\_ACTOR\_SET\_DATA
- #define HAVE\_SG\_ACTOR\_SLEEP\_FOR
- #define HAVE\_SG\_CFG\_SET\_INT
- #define HAVE\_SG\_CONFIG\_CONTINUE\_AFTER\_HELP
- #define HAVE\_SG\_HOST\_GET\_PROPERTIES
- #define HAVE\_SG\_HOST\_GET\_ROUTE
- #define HAVE\_SG\_HOST\_GET\_SPEED
- #define HAVE\_SG\_HOST\_LIST
- #define HAVE\_SG\_HOST\_ROUTE
- #define HAVE\_SG\_HOST\_SELF
- #define HAVE\_SG\_HOST\_SENDTO
- #define HAVE\_SG\_HOST\_SEND\_TO
- #define HAVE\_SG\_HOST\_SPEED
- #define HAVE\_SG\_LINK\_BANDWIDTH\_SET
- #define HAVE\_SG\_LINK\_GET\_NAME
- #define HAVE\_SG\_LINK\_NAME
- #define HAVE\_SG\_LINK\_SET\_BANDWIDTH
- #define HAVE\_SG\_ZONE\_GET\_BY\_NAME
- #define HAVE\_SG\_ZONE\_GET\_HOSTS
- #define HAVE\_SIMCALL\_PROCESS\_CREATE
- #define HAVE\_SIMGRID\_ACTOR\_H
- #define HAVE\_SIMGRID\_BARRIER\_H
- #define HAVE\_SIMGRID\_COND\_H
- #define HAVE\_SIMGRID\_ENGINE\_H
- #define HAVE\_SIMGRID\_GET\_CLOCK
- #define HAVE\_SIMGRID\_HOST\_H
- #define HAVE\_SIMGRID\_INIT
- #define HAVE\_SIMGRID\_LINK\_H
- #define HAVE\_SIMGRID\_MSG\_H
- #define HAVE\_SIMGRID\_MUTEX\_H
- #define HAVE\_SIMGRID\_SEMAPHORE\_H
- #define HAVE\_SIMGRID\_SET\_MAESTRO
- #define HAVE\_SIMGRID\_SIMDAG\_H
- #define HAVE\_SIMGRID\_VERSION\_H
- #define HAVE\_SIMGRID\_ZONE\_H
- #define HAVE\_SMPI\_PROCESS\_SET\_USER\_DATA
- #define HAVE\_SMPI\_THREAD\_CREATE
- #define HAVE\_SMX\_ACTOR\_T
- #define HAVE\_STDINT\_H
- #define HAVE\_STDIO\_H
- #define HAVE\_STDLIB\_H
- #define HAVE\_STRINGS\_H
- #define HAVE\_STRING\_H
- #define HAVE\_SYSCONF
- #define HAVE\_SYS\_STAT\_H
- #define HAVE\_SYS\_TYPES\_H
- #define HAVE\_UNISTD\_H
- #define HAVE\_VALGRIND\_HELGRIND\_H
- #define HAVE\_VALGRIND\_MEMCHECK\_H
- #define HAVE\_VALGRIND\_VALGRIND\_H

- #define HAVE\_XBT\_BARRIER\_INIT
- #define HAVE\_XBT\_BASE\_H
- #define HAVE\_XBT\_CONFIG\_H
- #define HAVE\_XBT\_MUTEX\_TRY\_ACQUIRE
- #define HAVE\_XBT\_SYNCHRO\_H
- #define LT\_OBJDIR
- #define PACKAGE
- #define PACKAGE\_BUGREPORT
- #define PACKAGE\_NAME
- #define PACKAGE\_STRING
- #define PACKAGE\_TARNAME
- #define PACKAGE\_URL
- #define PACKAGE\_VERSION
- #define SIZEOF\_VOID\_P
- #define STARPURM\_DLB\_VERBOSE
- #define STARPURM\_HAVE\_DLB
- #define STARPURM\_HAVE\_DLB\_CALLBACK\_ARG
- #define STARPURM\_STARPU\_HAVE\_WORKER\_CALLBACKS
- #define STARPURM\_VERBOSE
- #define STARPU\_ARMPL
- #define STARPU\_ATLAS
- #define STARPU\_BUILD\_DIR
- #define STARPU\_BUILT\_IN\_MIN\_DGELS
- #define STARPU\_CLUSTER
- #define STARPU\_DEBUG
- #define STARPU\_DEVEL
- #define STARPU\_DISABLE\_ASYNCHRONOUS\_COPY
- #define STARPU\_DISABLE\_ASYNCHRONOUS\_CUDA\_COPY
- #define STARPU\_DISABLE\_ASYNCHRONOUS\_MIC\_COPY
- #define STARPU\_DISABLE\_ASYNCHRONOUS\_MPI\_MS\_COPY
- #define STARPU\_DISABLE\_ASYNCHRONOUS\_OPENCL\_COPY
- #define STARPU\_EXTRA\_VERBOSE
- #define STARPU\_FXT\_LOCK\_TRACES
- #define STARPU\_GDB\_PATH
- #define STARPU\_GOTO
- #define STARPU\_HAVE\_ATOMIC\_COMPARE\_EXCHANGE\_N
- #define STARPU\_HAVE\_ATOMIC\_EXCHANGE\_N
- #define STARPU\_HAVE\_ATOMIC\_FETCH\_ADD
- #define STARPU\_HAVE\_ATOMIC\_FETCH\_OR
- #define STARPU\_HAVE\_ATOMIC\_TEST\_AND\_SET
- #define STARPU\_HAVE\_BLAS
- #define STARPU\_HAVE\_BUSID
- #define STARPU\_HAVE\_CBLAS\_H
- #define STARPU\_HAVE\_CUDA\_MEMCPY\_PEER
- #define STARPU\_HAVE\_CUFFTDOUBLECOMPLEX
- #define STARPU\_HAVE\_CURAND
- #define STARPU\_HAVE\_CXX11
- #define STARPU\_HAVE\_DARWIN
- #define STARPU\_HAVE\_DOMAINID
- #define STARPU\_HAVE\_F77\_H
- #define STARPU\_HAVE\_FC
- #define STARPU\_HAVE\_FFTW
- #define STARPU\_HAVE\_FFTWF
- #define STARPU\_HAVE\_FFTWL
- #define STARPU\_HAVE\_GLPK\_H

- `#define STARPU_HAVE_HDF5`
- `#define STARPU_HAVE_HELGRIND_H`
- `#define STARPU_HAVE_HWLOC`
- `#define STARPU_HAVE_ICC`
- `#define STARPU_HAVE_LEVELDB`
- `#define STARPU_HAVE_LIBNUMA`
- `#define STARPU_HAVE_LIBNVIDIA_ML`
- `#define STARPU_HAVE_MAGMA`
- `#define STARPU_HAVE_MALLOC_H`
- `#define STARPU_HAVE_MEMALIGN`
- `#define STARPU_HAVE_MEMCHECK_H`
- `#define STARPU_HAVE_MSG_MSG_H`
- `#define STARPU_HAVE_NEARBYINTF`
- `#define STARPU_HAVE_POSIX_MEMALIGN`
- `#define STARPU_HAVE_POTI`
- `#define STARPU_HAVE_PTHREAD_BARRIER`
- `#define STARPU_HAVE_PTHREAD_SETNAME_NP`
- `#define STARPU_HAVE_PTHREAD_SPIN_LOCK`
- `#define STARPU_HAVE_RINTF`
- `#define STARPU_HAVE_SCHED_YIELD`
- `#define STARPU_HAVE_SETENV`
- `#define STARPU_HAVE_SIMGRID_ACTOR_H`
- `#define STARPU_HAVE_SIMGRID_BARRIER_H`
- `#define STARPU_HAVE_SIMGRID_COND_H`
- `#define STARPU_HAVE_SIMGRID_ENGINE_H`
- `#define STARPU_HAVE_SIMGRID_HOST_H`
- `#define STARPU_HAVE_SIMGRID_LINK_H`
- `#define STARPU_HAVE_SIMGRID_MSG_H`
- `#define STARPU_HAVE_SIMGRID_MUTEX_H`
- `#define STARPU_HAVE_SIMGRID_SEMAPHORE_H`
- `#define STARPU_HAVE_SIMGRID_SIMDAG_H`
- `#define STARPU_HAVE_SIMGRID_VERSION_H`
- `#define STARPU_HAVE_SIMGRID_ZONE_H`
- `#define STARPU_HAVE_SMX_ACTOR_T`
- `#define STARPU_HAVE_STATEMENT_EXPRESSIONS`
- `#define STARPU_HAVE_STRERROR_R`
- `#define STARPU_HAVE_STRUCT_TIMESPEC`
- `#define STARPU_HAVE_SYNC_BOOL_COMPARE_AND_SWAP`
- `#define STARPU_HAVE_SYNC_FETCH_AND_ADD`
- `#define STARPU_HAVE_SYNC_FETCH_AND_OR`
- `#define STARPU_HAVE_SYNC_LOCK_TEST_AND_SET`
- `#define STARPU_HAVE_SYNC_SYNCHRONIZE`
- `#define STARPU_HAVE_SYNC_VAL_COMPARE_AND_SWAP`
- `#define STARPU_HAVE_UNISTD_H`
- `#define STARPU_HAVE_UNSETENV`
- `#define STARPU_HAVE_VALGRIND_H`
- `#define STARPU_HAVE_WINDOWS`
- `#define STARPU_HAVE_X11`
- `#define STARPU_HAVE_XBT_BASE_H`
- `#define STARPU_HAVE_XBT_CONFIG_H`
- `#define STARPU_HAVE_XBT_SYNCHRO_H`
- `#define STARPU_HISTORYMAXERROR`
- `#define STARPU_LINUX_SYS`
- `#define STARPU_LONG_CHECK`
- `#define STARPU_MAJOR_VERSION`

- `#define STARPU_MAXCPUS`
- `#define STARPU_MAXCUDADEVs`
- `#define STARPU_MAXIMPLEMENTATIONS`
- `#define STARPU_MAXMICCORES`
- `#define STARPU_MAXMICDEVs`
- `#define STARPU_MAXMPIDEVs`
- `#define STARPU_MAXMPKernels`
- `#define STARPU_MAXNODES`
- `#define STARPU_MAXNUMANODES`
- `#define STARPU_MAXOPENCLDEVs`
- `#define STARPU_MEMORY_STATS`
- `#define STARPU_MIC_USE_RMA`
- `#define STARPU_MINOR_VERSION`
- `#define STARPU_MKL`
- `#define STARPU_MLR_MODEL`
- `#define STARPU_MODEL_DEBUG`
- `#define STARPU_MPI_EXTRA_VERBOSE`
- `#define STARPU_MPI_MASTER_SLAVE_MULTIPLE_THREAD`
- `#define STARPU_MPI_PEDANTIC_ISEND`
- `#define STARPU_MPI_VERBOSE`
- `#define STARPU_NATIVE_WINTHREADS`
- `#define STARPU_NEW_CHECK`
- `#define STARPU_NMAXBUFS`
- `#define STARPU_NMAXWORKERS`
- `#define STARPU_NMAX_COMBINEDWORKERS`
- `#define STARPU_NMAX_SCHED_CTXs`
- `#define STARPU_NON_BLOCKING_DRIVERS`
- `#define STARPU_NO_ASSERT`
- `#define STARPU_OPENBLAS`
- `#define STARPU_OPENBSD_SYS`
- `#define STARPU_OPENCL_SIMULATOR`
- `#define STARPU_OPENGL_RENDER`
- `#define STARPU_OPENMP`
- `#define STARPU_PERF_DEBUG`
- `#define STARPU_PERF_MODEL_DIR`
- `#define STARPU_QUICK_CHECK`
- `#define STARPU_RELEASE_VERSION`
- `#define STARPU_SC_HYPERVISOR_DEBUG`
- `#define STARPU_SIMGRID`
- `#define STARPU_SIMGRID_HAVE_SIMGRID_INIT`
- `#define STARPU_SIMGRID_HAVE_XBT_BARRIER_INIT`
- `#define STARPU_SIMGRID_MC`
- `#define STARPU_SPINLOCK_CHECK`
- `#define STARPU_SRC_DIR`
- `#define STARPU_STATIC_ONLY`
- `#define STARPU_SYSTEM_BLAS`
- `#define STARPU_USE_ALLOCATION_CACHE`
- `#define STARPU_USE_AYUDAME1`
- `#define STARPU_USE_AYUDAME2`
- `#define STARPU_USE_CPU`
- `#define STARPU_USE_CUDA`
- `#define STARPU_USE_DRAND48`
- `#define STARPU_USE_ERAND48_R`
- `#define STARPU_USE_FXT`
- `#define STARPU_USE_MIC`



- `#define STARPU_USE_MP`
- `#define STARPU_USE_MPI`
- `#define STARPU_USE_MPI_MASTER_SLAVE`
- `#define STARPU_USE_MPI_MPI`
- `#define STARPU_USE_MPI_NMAD`
- `#define STARPU_USE_OPENCL`
- `#define STARPU_USE_SC_HYPERVISOR`
- `#define STARPU_VALGRIND_FULL`
- `#define STARPU_VERBOSE`
- `#define STARPU_WORKER_CALLBACKS`
- `#define STDC_HEADERS`
- `#define VERSION`
- `#define X_DISPLAY_MISSING`
- `#define restrict`

## 6.8 copy\_driver.h File Reference

```
#include <common/config.h>
#include <common/list.h>
#include <cuda.h>
#include <cuda_runtime.h>
#include <starpu_openccl.h>
```

### Data Structures

- struct [\\_starpu\\_mic\\_async\\_event](#)
- struct [\\_starpu\\_disk\\_backend\\_event](#)
- struct [\\_starpu\\_disk\\_async\\_event](#)
- union [\\_starpu\\_async\\_channel\\_event](#)
- struct [\\_starpu\\_async\\_channel](#)
- struct [\\_starpu\\_async\\_channel\\_event.\\_\\_\\_\\_unnamed1\\_\\_](#)

### Enumerations

- enum [\\_starpu\\_is\\_prefetch](#) { [STARPU\\_FETCH](#) , [STARPU\\_PREFETCH](#) , [STARPU\\_IDLEFETCH](#) , [STARPU\\_NFETCH](#) }

### Functions

- void [\\_starpu\\_wake\\_all\\_blocked\\_workers\\_on\\_node](#) (unsigned nodeid)
- int [\\_starpu\\_driver\\_copy\\_data\\_1\\_to\\_1](#) (starpu\_data\_handle\_t handle, [struct \\_starpu\\_data\\_replicate](#) \*src↔\_replicate, [struct \\_starpu\\_data\\_replicate](#) \*dst\_replicate, unsigned donotread, [struct \\_starpu\\_data\\_request](#) \*req, unsigned may\_alloc, enum [\\_starpu\\_is\\_prefetch](#) prefetch)
- unsigned [\\_starpu\\_driver\\_test\\_request\\_completion](#) ([struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- void [\\_starpu\\_driver\\_wait\\_request\\_completion](#) ([struct \\_starpu\\_async\\_channel](#) \*async\_channel)

## 6.8.1 Data Structure Documentation

### 6.8.1.1 struct \_starpu\_mic\_async\_event

MIC needs memory\_node to know which MIC is concerned. mark is used to wait asynchronous request. signal is used to test asynchronous request.

#### Data Fields

unsigned	memory_node	
----------	-------------	--

## Data Fields

int	mark	
uint64_t *	signal	

## 6.8.1.2 struct \_starpu\_disk\_backend\_event

## Data Fields

void *	backend_event	
--------	---------------	--

## 6.8.1.3 struct \_starpu\_disk\_async\_event

## Data Fields

unsigned	memory_node	
struct _starpu_disk_backend_event_list *	requests	
void *	ptr	
unsigned	node	
size_t	size	
starpu_data_handle_t	handle	

## 6.8.1.4 union \_starpu\_async\_channel\_event

this is a structure that can be queried to see whether an asynchronous transfer has terminated or not

## Data Fields

struct _starpu_async_channel_event	__unnamed__	
cudaEvent_t	cuda_event	
cl_event	openccl_event	
struct _starpu_mic_async_event	mic_event	
struct _starpu_disk_async_event	disk_event	

## 6.8.1.5 struct \_starpu\_async\_channel

## Data Fields

union _starpu_async_channel_event	event	
struct _starpu_node_ops *	node_ops	
struct _starpu_mp_node *	polling_node_sender	Which node to polling when needing ACK msg
struct _starpu_mp_node *	polling_node_receiver	
volatile int	starpu_mp_common_finished_sender	Used to know if the acknowledgment msg is arrived from sinks
volatile int	starpu_mp_common_finished_receiver	

### 6.8.1.6 struct\_starpu\_async\_channel\_event.\_\_unnamed1\_\_

#### Data Fields

unsigned	finished	
starpu_pthread_queue_t *	queue	

## 6.8.2 Enumeration Type Documentation

### 6.8.2.1 \_starpu\_is\_prefetch

enum [\\_starpu\\_is\\_prefetch](#)

#### Enumerator

STARPU_FETCH	A task really needs it now!
STARPU_PREFETCH	It is a good idea to have it asap
STARPU_IDLEFETCH	Get this here when you have time to

## 6.9 data\_concurrency.h File Reference

```
#include <core/jobs.h>
```

### Functions

- void **\_starpu\_job\_set\_ordered\_buffers** ([struct\\_starpu\\_job](#) \*)
- unsigned **\_starpu\_concurrent\_data\_access** ([struct\\_starpu\\_job](#) \*)
- void **\_starpu\_submit\_job\_enforce\_arbitered\_deps** ([struct\\_starpu\\_job](#) \*, unsigned buf, unsigned nbufs)
- void **\_starpu\_enforce\_data\_deps\_notify\_job\_ready\_soon** ([struct\\_starpu\\_job](#) \*, [\\_starpu\\_notify\\_job\\_](#)↔ start\_data \*data)
- int **\_starpu\_notify\_data\_dependencies** (starpu\_data\_handle\_t handle)
- void **\_starpu\_notify\_arbitered\_dependencies** (starpu\_data\_handle\_t handle)
- unsigned **\_starpu\_attempt\_to\_submit\_data\_request\_from\_apps** (starpu\_data\_handle\_t handle, enum starpu\_data\_access\_mode mode, void(\*callback)(void \*), void \*argcb)
- unsigned **\_starpu\_attempt\_to\_submit\_arbitered\_data\_request** (unsigned request\_from\_codelet, starpu\_data\_handle\_t handle, enum starpu\_data\_access\_mode mode, void(\*callback)(void \*), void \*argcb, [struct\\_starpu\\_job](#) \*, unsigned buffer\_index)

## 6.10 data\_interface.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/uthash.h>
#include <util/openmp_runtime_support.h>
```

### Data Structures

- union [\\_starpu\\_interface](#)

## Macros

- `#define _starpu_data_check_not_busy(handle)`
- `#define _starpu_data_is_multiformat_handle(handle)`

## Functions

- `void _starpu_data_free_interfaces (starpu_data_handle_t handle) STARPU_ATTRIBUTE_INTERNAL`
- `int _starpu_data_handle_init (starpu_data_handle_t handle, struct starpu_data_interface_ops *interface_ops, unsigned int mf_node)`
- `void _starpu_data_initialize_per_worker (starpu_data_handle_t handle)`
- `void _starpu_data_interface_init (void) STARPU_ATTRIBUTE_INTERNAL`
- `int __starpu_data_check_not_busy (starpu_data_handle_t handle) STARPU_ATTRIBUTE_INTERNAL STARPU_ATTRIBUTE_WARN_UNUSED_RESULT`
- `void _starpu_data_interface_shutdown (void) STARPU_ATTRIBUTE_INTERNAL`
- `void _starpu_omp_unregister_region_handles (struct starpu_omp_region *region)`
- `void _starpu_omp_unregister_task_handles (struct starpu_omp_task *task)`
- `struct starpu_data_interface_ops * _starpu_data_interface_get_ops (unsigned interface_id)`
- `void _starpu_data_register_ram_pointer (starpu_data_handle_t handle, void *ptr) STARPU_ATTRIBUTE_INTERNAL`
- `void _starpu_data_unregister_ram_pointer (starpu_data_handle_t handle, unsigned node) STARPU_ATTRIBUTE_INTERNAL`
- `void _starpu_data_invalidate_submit_noplan (starpu_data_handle_t handle)`

## Variables

- `struct starpu_data_interface_ops starpu_interface_matrix_ops`
- `struct starpu_data_interface_ops starpu_interface_block_ops`
- `struct starpu_data_interface_ops starpu_interface_vector_ops`
- `struct starpu_data_interface_ops starpu_interface_csr_ops`
- `struct starpu_data_interface_ops starpu_interface_bcsr_ops`
- `struct starpu_data_interface_ops starpu_interface_variable_ops`
- `struct starpu_data_interface_ops starpu_interface_void_ops`
- `struct starpu_data_interface_ops starpu_interface_multiformat_ops`
- `struct starpu_arbiter * _starpu_global_arbiter`

### 6.10.1 Data Structure Documentation

#### 6.10.1.1 union \_starpu\_interface

Generic type representing an interface, for now it's only used before execution on message-passing devices but it can be useful in other cases.

##### Data Fields

<code>struct starpu_variable_interface</code>	variable	
<code>struct starpu_vector_interface</code>	vector	
<code>struct starpu_matrix_interface</code>	matrix	
<code>struct starpu_block_interface</code>	block	
<code>struct starpu_csr_interface</code>	csr	
<code>struct starpu_bcsr_interface</code>	bcsr	
<code>struct starpu_coo_interface</code>	coo	

## 6.10.2 Variable Documentation

### 6.10.2.1 starpu\_interface\_matrix\_ops

```
struct starpu_data_interface_ops starpu_interface_matrix_ops [extern]
```

Some data interfaces or filters use this interface internally

## 6.11 data\_request.h File Reference

```
#include <datawizard/coherency.h>
#include <semaphore.h>
#include <datawizard/copy_driver.h>
#include <common/list.h>
#include <common/prio_list.h>
#include <common/starpu_spinlock.h>
```

### Data Structures

- struct [\\_starpu\\_callback\\_list](#)

### Macros

- #define **MAX\_PENDING\_REQUESTS\_PER\_NODE**
- #define **MAX\_PENDING\_PREFETCH\_REQUESTS\_PER\_NODE**
- #define **MAX\_PENDING\_IDLE\_REQUESTS\_PER\_NODE**
- #define [MAX\\_PUSH\\_TIME](#)

### 6.11.1 Macro Definition Documentation

#### 6.11.1.1 MAX\_PUSH\_TIME

```
#define MAX_PUSH_TIME
```

Maximum time in us that we can afford pushing requests before going back to the driver loop, e.g. for checking GPU task termination

## 6.12 datastats.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <stdint.h>
#include <stdlib.h>
```

### Macros

- #define **\_starpu\_msi\_cache\_hit**(node)
- #define **\_starpu\_msi\_cache\_miss**(node)
- #define **\_starpu\_allocation\_cache\_hit**(node)
- #define **\_starpu\_data\_allocation\_inc\_stats**(node)

## Functions

- void **\_\_starpu\_datastats\_init** ()
- static int **starpu\_enable\_stats** (void)
- void **\_\_starpu\_msi\_cache\_hit** (unsigned node)
- void **\_\_starpu\_msi\_cache\_miss** (unsigned node)
- void **\_\_starpu\_display\_msi\_stats** (FILE \*stream)
- void **\_\_starpu\_allocation\_cache\_hit** (unsigned node STARPU\_ATTRIBUTE\_UNUSED)
- void **\_\_starpu\_data\_allocation\_inc\_stats** (unsigned node STARPU\_ATTRIBUTE\_UNUSED)
- void **\_\_starpu\_display\_alloc\_cache\_stats** (FILE \*stream)

## Variables

- int **\_\_starpu\_enable\_stats**

## 6.13 datawizard.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/utils.h>
#include <datawizard/coherency.h>
#include <datawizard/filters.h>
#include <datawizard/copy_driver.h>
#include <datawizard/footprint.h>
#include <datawizard/data_request.h>
#include <datawizard/interfaces/data_interface.h>
#include <core/dependencies/implicit_data_deps.h>
```

## Functions

- int **\_\_starpu\_datawizard\_progress** (unsigned memory\_node, unsigned may\_alloc, unsigned push\_requests)
- int **\_\_starpu\_datawizard\_progress** (unsigned may\_alloc, unsigned push\_requests)
- void **\_\_starpu\_datawizard\_progress** (unsigned may\_alloc)

## 6.14 debug.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <common/config.h>
#include <core/workers.h>
```

## Macros

- #define **STARPU\_AYU\_EVENT**
- #define **STARPU\_AYU\_PREINIT**()
- #define **STARPU\_AYU\_INIT**()
- #define **STARPU\_AYU\_FINISH**()
- #define **STARPU\_AYU\_ADDDEPENDENCY**(previous, handle, next\_job)
- #define **STARPU\_AYU\_REMOVETASK**(job\_id)
- #define **STARPU\_AYU\_ADDTASK**(job\_id, task)
- #define **STARPU\_AYU\_PRERUNTASK**(job\_id, workerid)
- #define **STARPU\_AYU\_RUNTASK**(job\_id)

- `#define STARPU_AYU_POSTRUNTASK(job_id)`
- `#define STARPU_AYU_ADDTOTASKQUEUE(job_id, worker_id)`
- `#define STARPU_AYU_BARRIER()`

## Functions

- `void _starpu_open_debug_logfile (void)`
- `void _starpu_close_debug_logfile (void)`
- `void _starpu_print_to_logfile (const char *format,...) STARPU_ATTRIBUTE_FORMAT(printf`
- `void _starpu_watchdog_init (void)`
- `void _starpu_watchdog_shutdown (void)`

## Variables

- `void int _starpu_use_fxt`

### 6.14.1 Function Documentation

#### 6.14.1.1 `_starpu_open_debug_logfile()`

```
void _starpu_open_debug_logfile (
    void )
```

Create a file that will contain StarPU's log

#### 6.14.1.2 `_starpu_close_debug_logfile()`

```
void _starpu_close_debug_logfile (
    void )
```

Close StarPU's log file

#### 6.14.1.3 `_starpu_print_to_logfile()`

```
void _starpu_print_to_logfile (
    const char * format,
    ... )
```

Write into StarPU's log file

### 6.14.2 Variable Documentation

#### 6.14.2.1 `_starpu_use_fxt`

```
void int _starpu_use_fxt [extern]
```

Tell gdb whether FXT is compiled in or not

## 6.15 `detect_combined_workers.h` File Reference

```
#include <starpu.h>
```

## Functions

- `void _starpu_sched_find_worker_combinations (int *workerids, int nworkers)`

## Variables

- `int _starpu_initialized_combined_workers`

### 6.15.1 Function Documentation

#### 6.15.1.1 `_starpu_sched_find_worker_combinations()`

```
void _starpu_sched_find_worker_combinations (
    int * workerids,
    int nworkers )
```

Initialize combined workers

## 6.16 disk.h File Reference

```
#include <datawizard/copy_driver.h>
#include <datawizard/malloc.h>
```

## Macros

- `#define STARPU_DISK_ALL`
- `#define STARPU_DISK_NO_RECLAIM`

## Functions

- `void * _starpu_disk_alloc` (unsigned node, size\_t size) STARPU\_ATTRIBUTE\_MALLOC
- `void _starpu_disk_free` (unsigned node, void \*obj, size\_t size)
- `int _starpu_disk_read` (unsigned src\_node, unsigned dst\_node, void \*obj, void \*buf, off\_t offset, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_write` (unsigned src\_node, unsigned dst\_node, void \*obj, void \*buf, off\_t offset, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_full_read` (unsigned src\_node, unsigned dst\_node, void \*obj, void \*\*ptr, size\_t \*size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_full_write` (unsigned src\_node, unsigned dst\_node, void \*obj, void \*ptr, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_copy` (unsigned node\_src, void \*obj\_src, off\_t offset\_src, unsigned node\_dst, void \*obj\_dst, off\_t offset\_dst, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `void starpu_disk_wait_request` ([struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `int starpu_disk_test_request` ([struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `void starpu_disk_free_request` ([struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_can_copy` (unsigned node1, unsigned node2)
- `void _starpu_set_disk_flag` (unsigned node, int flag)
- `int _starpu_get_disk_flag` (unsigned node)
- `void _starpu_disk_unregister` (void)
- `void _starpu_swap_init` (void)

### 6.16.1 Function Documentation



**6.16.1.1 \_starpu\_disk\_alloc()**

```
void* _starpu_disk_alloc (
    unsigned node,
    size_t size )
```

interface to manipulate memory disk

**6.16.1.2 \_starpu\_disk\_read()**

```
int _starpu_disk_read (
    unsigned src_node,
    unsigned dst_node,
    void * obj,
    void * buf,
    off_t offset,
    size_t size,
    struct _starpu_async_channel * async_channel )
```

src\_node is a disk node, dst\_node is for the moment the STARPU\_MAIN\_RAM

**6.16.1.3 \_starpu\_disk\_write()**

```
int _starpu_disk_write (
    unsigned src_node,
    unsigned dst_node,
    void * obj,
    void * buf,
    off_t offset,
    size_t size,
    struct _starpu_async_channel * async_channel )
```

src\_node is for the moment the STARPU\_MAIN\_RAM, dst\_node is a disk node

**6.16.1.4 starpu\_disk\_wait\_request()**

```
void starpu_disk_wait_request (
    struct _starpu_async_channel * async_channel )
```

force the request to compute

**6.16.1.5 starpu\_disk\_test\_request()**

```
int starpu_disk_test_request (
    struct _starpu_async_channel * async_channel )
```

return 1 if the request is finished, 0 if not finished

**6.16.1.6 \_starpu\_disk\_can\_copy()**

```
int _starpu_disk_can_copy (
    unsigned node1,
    unsigned node2 )
```

interface to compare memory disk

**6.16.1.7 \_starpu\_set\_disk\_flag()**

```
void _starpu_set_disk_flag (
    unsigned node,
    int flag )
```

change disk flag

### 6.16.1.8 \_\_starpu\_disk\_unregister()

```
void __starpu_disk_unregister (
    void )
unregister disk
```

## 6.17 disk\_unistd\_global.h File Reference

```
#include <fcntl.h>
```

### Data Structures

- struct [starpu\\_unistd\\_global\\_obj](#)

### Macros

- #define **O\_BINARY**
- #define **STARPU\_UNISTD\_USE\_COPY**

### Typedefs

- typedef off\_t **starpu\_loff\_t**

### Functions

- void \* **starpu\_unistd\_global\_alloc** (struct [starpu\\_unistd\\_global\\_obj](#) \*obj, void \*base, size\_t size)
- void **starpu\_unistd\_global\_free** (void \*base, void \*obj, size\_t size)
- void \* **starpu\_unistd\_global\_open** (struct [starpu\\_unistd\\_global\\_obj](#) \*obj, void \*base, void \*pos, size\_t size)
- void **starpu\_unistd\_global\_close** (void \*base, void \*obj, size\_t size)
- int **starpu\_unistd\_global\_read** (void \*base, void \*obj, void \*buf, off\_t offset, size\_t size)
- int **starpu\_unistd\_global\_write** (void \*base, void \*obj, const void \*buf, off\_t offset, size\_t size)
- void \* **starpu\_unistd\_global\_plug** (void \*parameter, starpu\_ssize\_t size)
- void **starpu\_unistd\_global\_unplug** (void \*base)
- int **\_\_starpu\_get\_unistd\_global\_bandwidth\_between\_disk\_and\_main\_ram** (unsigned node, void \*base)
- void \* **starpu\_unistd\_global\_async\_read** (void \*base, void \*obj, void \*buf, off\_t offset, size\_t size)
- void \* **starpu\_unistd\_global\_async\_write** (void \*base, void \*obj, void \*buf, off\_t offset, size\_t size)
- void \* **starpu\_unistd\_global\_async\_full\_write** (void \*base, void \*obj, void \*ptr, size\_t size)
- void \* **starpu\_unistd\_global\_async\_full\_read** (void \*base, void \*obj, void \*\*ptr, size\_t \*size, unsigned dst\_node)
- void **starpu\_unistd\_global\_wait\_request** (void \*async\_channel)
- int **starpu\_unistd\_global\_test\_request** (void \*async\_channel)
- void **starpu\_unistd\_global\_free\_request** (void \*async\_channel)
- int **starpu\_unistd\_global\_full\_read** (void \*base, void \*obj, void \*\*ptr, size\_t \*size, unsigned dst\_node)
- int **starpu\_unistd\_global\_full\_write** (void \*base, void \*obj, void \*ptr, size\_t size)

### 6.17.1 Data Structure Documentation

#### 6.17.1.1 struct starpu\_unistd\_global\_obj

##### Data Fields

int	descriptor	
char *	path	
size_t	size	

## Data Fields

	int	flags	
starpu_pthread_mutex_t		mutex	

## 6.18 driver\_common.h File Reference

```
#include <starpu.h>
#include <starpu_util.h>
#include <core/jobs.h>
#include <common/utils.h>
```

### Functions

- void **\_starpu\_driver\_start\_job** (struct \_starpu\_worker \*args, struct \_starpu\_job \*j, struct starpu\_perfm↵  
model\_arch \*perf\_arch, int rank, int profiling)
- void **\_starpu\_driver\_end\_job** (struct \_starpu\_worker \*args, struct \_starpu\_job \*j, struct starpu\_perfm↵  
\_arch \*perf\_arch, int rank, int profiling)
- void **\_starpu\_driver\_update\_job\_feedback** (struct \_starpu\_job \*j, struct \_starpu\_worker \*worker\_args,  
struct starpu\_perfm↵\_arch \*perf\_arch, int profiling)
- struct starpu\_task \* **\_starpu\_get\_worker\_task** (struct \_starpu\_worker \*args, int workerid, unsigned  
memnode)
- int **\_starpu\_get\_multi\_worker\_task** (struct \_starpu\_worker \*workers, struct starpu\_task \*\*tasks, int  
nworker, unsigned memnode)

## 6.19 driver\_cpu.h File Reference

```
#include <common/config.h>
#include <datawizard/node_ops.h>
```

### Functions

- void \* **\_starpu\_cpu\_worker** (void \*)
- int **\_starpu\_cpu\_copy\_interface** (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node,  
void \*dst\_interface, unsigned dst\_node, struct \_starpu\_data\_request \*req)
- int **\_starpu\_cpu\_copy\_data** (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t↵  
\_t dst\_offset, unsigned dst\_node, size\_t ssize, struct \_starpu\_async\_channel \*async\_channel)
- int **\_starpu\_cpu\_is\_direct\_access\_supported** (unsigned node, unsigned handling\_node)
- uintptr\_t **\_starpu\_cpu\_malloc\_on\_node** (unsigned dst\_node, size\_t size, int flags)
- void **\_starpu\_cpu\_free\_on\_node** (unsigned dst\_node, uintptr\_t addr, size\_t size, int flags)

### Variables

- struct \_starpu\_driver\_ops **\_starpu\_driver\_cpu\_ops**
- struct \_starpu\_node\_ops **\_starpu\_driver\_cpu\_node\_ops**

## 6.20 driver\_cuda.h File Reference

```
#include <common/config.h>
#include <cuda.h>
#include <cuda_runtime_api.h>
#include <cublas.h>
#include <starpu.h>
```

```
#include <core/workers.h>
#include <datawizard/node_ops.h>
```

## Functions

- void **\_starpu\_cuda\_init** (void)
- unsigned **\_starpu\_get\_cuda\_device\_count** (void)
- void **\_starpu\_cuda\_discover\_devices** (struct [\\_starpu\\_machine\\_config](#) \*)
- void **\_starpu\_init\_cuda** (void)
- void \* **\_starpu\_cuda\_worker** (void \*)
- cudaStream\_t **starpu\_cuda\_get\_local\_in\_transfer\_stream** (void)
- cudaStream\_t **starpu\_cuda\_get\_in\_transfer\_stream** (unsigned dst\_node)
- cudaStream\_t **starpu\_cuda\_get\_local\_out\_transfer\_stream** (void)
- cudaStream\_t **starpu\_cuda\_get\_out\_transfer\_stream** (unsigned src\_node)
- cudaStream\_t **starpu\_cuda\_get\_peer\_transfer\_stream** (unsigned src\_node, unsigned dst\_node)
- unsigned **\_starpu\_cuda\_test\_request\_completion** (struct [\\_starpu\\_async\\_channel](#) \*async\_channel)
- void **\_starpu\_cuda\_wait\_request\_completion** (struct [\\_starpu\\_async\\_channel](#) \*async\_channel)
- int **\_starpu\_cuda\_copy\_interface\_from\_cpu\_to\_cuda** (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, struct [\\_starpu\\_data\\_request](#) \*req)
- int **\_starpu\_cuda\_copy\_interface\_from\_cuda\_to\_cuda** (starpu\_data\_handle\_t handle, void \*src\_↵ interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, struct [\\_starpu\\_data\\_request](#) \*req)
- int **\_starpu\_cuda\_copy\_interface\_from\_cuda\_to\_cpu** (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, struct [\\_starpu\\_data\\_request](#) \*req)
- int **\_starpu\_cuda\_copy\_data\_from\_cuda\_to\_cuda** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, struct [\\_starpu\\_async\\_channel](#) \*async\_↵ channel)
- int **\_starpu\_cuda\_copy\_data\_from\_cuda\_to\_cpu** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, struct [\\_starpu\\_async\\_channel](#) \*async\_↵ channel)
- int **\_starpu\_cuda\_copy\_data\_from\_cpu\_to\_cuda** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, struct [\\_starpu\\_async\\_channel](#) \*async\_↵ channel)
- int **\_starpu\_cuda\_copy2d\_data\_from\_cuda\_to\_cuda** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks, size\_t ld\_src, size\_t ld\_dst, struct [\\_starpu\\_async\\_channel](#) \*async\_channel)
- int **\_starpu\_cuda\_copy2d\_data\_from\_cuda\_to\_cpu** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks, size\_t ld\_src, size\_t ld\_dst, struct [\\_starpu\\_async\\_channel](#) \*async\_channel)
- int **\_starpu\_cuda\_copy2d\_data\_from\_cpu\_to\_cuda** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks, size\_t ld\_src, size\_t ld\_dst, struct [\\_starpu\\_async\\_channel](#) \*async\_channel)
- int **\_starpu\_cuda\_copy3d\_data\_from\_cuda\_to\_cuda** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks\_1, size\_t ld1\_src, size\_t ld1\_dst, size\_t numblocks\_2, size\_t ld2\_src, size\_t ld2\_dst, struct [\\_starpu\\_async\\_channel](#) \*async\_↵ channel)
- int **\_starpu\_cuda\_copy3d\_data\_from\_cuda\_to\_cpu** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks\_1, size\_t ld1\_src, size\_t ld1\_dst, size\_t numblocks\_2, size\_t ld2\_src, size\_t ld2\_dst, struct [\\_starpu\\_async\\_channel](#) \*async\_↵ channel)
- int **\_starpu\_cuda\_copy3d\_data\_from\_cpu\_to\_cuda** (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks\_1, size\_t ld1\_src, size\_t ld1\_dst, size\_t numblocks\_2, size\_t ld2\_src, size\_t ld2\_dst, struct [\\_starpu\\_async\\_channel](#) \*async\_↵ channel)
- int **\_starpu\_cuda\_is\_direct\_access\_supported** (unsigned node, unsigned handling\_node)
- uintptr\_t **\_starpu\_cuda\_malloc\_on\_node** (unsigned dst\_node, size\_t size, int flags)
- void **\_starpu\_cuda\_free\_on\_node** (unsigned dst\_node, uintptr\_t addr, size\_t size, int flags)

## Variables

- [struct \\_starpu\\_driver\\_ops](#) `_starpu_driver_cuda_ops`
- [struct \\_starpu\\_node\\_ops](#) `_starpu_driver_cuda_node_ops`
- `int _starpu_cuda_bus_ids` `[STARPU_MAXCUDADEV+STARPU_MAXNUMANODES][STARPU_MAXCUDADEV+STARPU_MAXNUMANODES]`

## 6.21 driver\_disk.h File Reference

```
#include <datawizard/node_ops.h>
```

## Functions

- `int _starpu_disk_copy_src_to_disk` (void \*src, unsigned src\_node, void \*dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, void \*async\_channel)
- `int _starpu_disk_copy_disk_to_src` (void \*src, size\_t src\_offset, unsigned src\_node, void \*dst, unsigned dst\_node, size\_t size, void \*async\_channel)
- `int _starpu_disk_copy_disk_to_disk` (void \*src, size\_t src\_offset, unsigned src\_node, void \*dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, void \*async\_channel)
- `unsigned _starpu_disk_test_request_completion` ([struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `void _starpu_disk_wait_request_completion` ([struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_copy_interface_from_disk_to_cpu` (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, [struct \\_starpu\\_data\\_request](#) \*req)
- `int _starpu_disk_copy_interface_from_disk_to_disk` (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, [struct \\_starpu\\_data\\_request](#) \*req)
- `int _starpu_disk_copy_interface_from_cpu_to_disk` (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, [struct \\_starpu\\_data\\_request](#) \*req)
- `int _starpu_disk_copy_data_from_disk_to_cpu` (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_copy_data_from_disk_to_disk` (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_copy_data_from_cpu_to_disk` (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `int _starpu_disk_is_direct_access_supported` (unsigned node, unsigned handling\_node)
- `uintptr_t _starpu_disk_malloc_on_node` (unsigned dst\_node, size\_t size, int flags)
- `void _starpu_disk_free_on_node` (unsigned dst\_node, uintptr\_t addr, size\_t size, int flags)

## Variables

- [struct \\_starpu\\_node\\_ops](#) `_starpu_driver_disk_node_ops`

## 6.22 driver\_mic\_common.h File Reference

```
#include <common/config.h>
#include <source/COIPProcess_source.h>
```

## Data Structures

- [struct \\_starpu\\_mic\\_free\\_command](#)

## Macros

- `#define STARPU_TO_MIC_ID(id)`
- `#define STARPU_MIC_PORTS_BEGIN`
- `#define STARPU_MIC_SOURCE_PORT_NUMBER`
- `#define STARPU_MIC_SINK_PORT_NUMBER(id)`
- `#define STARPU_MIC_SOURCE_DT_PORT_NUMBER`
- `#define STARPU_MIC_SINK_DT_PORT_NUMBER(id)`
- `#define STARPU_MIC_SINK_SINK_DT_PORT_NUMBER(me, peer_id)`
- `#define STARPU_MIC_PAGE_SIZE`
- `#define STARPU_MIC_GET_PAGE_SIZE_MULTIPLE(size)`
- `#define STARPU_MIC_COMMON_REPORT_SCIF_ERROR(status)`

## Functions

- `void _starpu_mic_common_report_scif_error (const char *func, const char *file, int line, const int status)`
- `int _starpu_mic_common_recv_is_ready (const struct _starpu_mp_node *mp_node)`
- `void _starpu_mic_common_send (const struct _starpu_mp_node *node, void *msg, int len)`
- `void _starpu_mic_common_recv (const struct _starpu_mp_node *node, void *msg, int len)`
- `void _starpu_mic_common_dt_send (const struct _starpu_mp_node *node, void *msg, int len, void *event)`
- `void _starpu_mic_common_dt_recv (const struct _starpu_mp_node *node, void *msg, int len, void *event)`
- `void _starpu_mic_common_connect (scif_epd_t *endpoint, uint16_t remote_node, COIPROCESS process, uint16_t local_port_number, uint16_t remote_port_number)`
- `void _starpu_mic_common_accept (scif_epd_t *endpoint, uint16_t port_number)`

### 6.22.1 Data Structure Documentation

#### 6.22.1.1 `struct _starpu_mic_free_command`

##### Data Fields

<code>void *</code>	<code>addr</code>	
<code>size_t</code>	<code>size</code>	

## 6.23 driver\_mic\_sink.h File Reference

```
#include <common/config.h>
#include <scif.h>
#include <drivers/mp_common/mp_common.h>
#include <drivers/mp_common/sink_common.h>
```

## Macros

- `#define STARPU_MIC_SINK_REPORT_ERROR(status)`

## Functions

- `void _starpu_mic_sink_report_error (const char *func, const char *file, const int line, const int status)`
- `void _starpu_mic_sink_init (struct _starpu_mp_node *node)`
- `void _starpu_mic_sink_launch_workers (struct _starpu_mp_node *node)`
- `void _starpu_mic_sink_deinit (struct _starpu_mp_node *node)`
- `void _starpu_mic_sink_allocate (const struct _starpu_mp_node *mp_node, void *arg, int arg_size)`
- `void _starpu_mic_sink_free (const struct _starpu_mp_node *mp_node STARPU_ATTRIBUTE_UNUSED, void *arg, int arg_size)`

- void **\_starpu\_mic\_sink\_bind\_thread** (const [struct](#) \_starpu\_mp\_node \*mp\_node STARPU\_ATTRIBUTE\_UNUSED, int coreid, int \*core\_table, int nb\_core)

## Variables

- void(\*) (void) **\_starpu\_mic\_sink\_lookup** (const [struct](#) \_starpu\_mp\_node \*node STARPU\_ATTRIBUTE\_UNUSED, char \*func\_name)

## 6.24 driver\_mic\_source.h File Reference

```
#include <starpu_mic.h>
#include <common/config.h>
#include <source/COIProcess_source.h>
#include <source/COIEngine_source.h>
#include <core/workers.h>
#include <drivers/mp_common/mp_common.h>
#include <datawizard/node_ops.h>
```

## Macros

- #define **STARPU\_MIC\_REQUEST\_COMPLETE**
- #define **STARPU\_MIC\_SRC\_REPORT\_COI\_ERROR**(status)
- #define **STARPU\_MIC\_SRC\_REPORT\_SCIF\_ERROR**(status)

## Functions

- [struct](#) \_starpu\_mp\_node \* **\_starpu\_mic\_src\_get\_actual\_thread\_mp\_node** ()
- [struct](#) \_starpu\_mp\_node \* **\_starpu\_mic\_src\_get\_mp\_node\_from\_memory\_node** (int memory\_node)
- int **\_starpu\_mic\_src\_register\_kernel** (starpu\_mic\_func\_symbol\_t \*symbol, const char \*func\_name)
- starpu\_mic\_kernel\_t **\_starpu\_mic\_src\_get\_kernel** (starpu\_mic\_func\_symbol\_t symbol)
- void **\_starpu\_mic\_src\_report\_coi\_error** (const char \*func, const char \*file, int line, const COIRESET status)
- void **\_starpu\_mic\_src\_report\_scif\_error** (const char \*func, const char \*file, int line, const int status)
- unsigned **\_starpu\_mic\_src\_get\_device\_count** (void)
- starpu\_mic\_kernel\_t **\_starpu\_mic\_src\_get\_kernel\_from\_codelet** ([struct](#) starpu\_codelet \*cl, unsigned nimpl)
- void **\_starpu\_mic\_src\_init** ([struct](#) \_starpu\_mp\_node \*node)
- void **\_starpu\_mic\_clear\_kernels** (void)
- void **\_starpu\_mic\_src\_deinit** ([struct](#) \_starpu\_mp\_node \*node)
- size\_t **\_starpu\_mic\_get\_global\_mem\_size** (int devid)
- size\_t **\_starpu\_mic\_get\_free\_mem\_size** (int devid)
- int **\_starpu\_mic\_allocate\_memory** (void \*\*addr, size\_t size, unsigned memory\_node)
- void **\_starpu\_mic\_free\_memory** (void \*addr, size\_t size, unsigned memory\_node)
- int **\_starpu\_mic\_copy\_ram\_to\_mic** (void \*src, unsigned src\_node STARPU\_ATTRIBUTE\_UNUSED, void \*dst, unsigned dst\_node, size\_t size)
- int **\_starpu\_mic\_copy\_mic\_to\_ram** (void \*src, unsigned src\_node, void \*dst, unsigned dst\_node STARPU\_ATTRIBUTE\_UNUSED, size\_t size)
- int **\_starpu\_mic\_copy\_ram\_to\_mic\_async** (void \*src, unsigned src\_node STARPU\_ATTRIBUTE\_UNUSED, void \*dst, unsigned dst\_node, size\_t size)
- int **\_starpu\_mic\_copy\_mic\_to\_ram\_async** (void \*src, unsigned src\_node, void \*dst, unsigned dst\_node STARPU\_ATTRIBUTE\_UNUSED, size\_t size)
- int **\_starpu\_mic\_init\_event** ([struct](#) \_starpu\_mic\_async\_event \*event, unsigned memory\_node)
- void \* **\_starpu\_mic\_src\_worker** (void \*arg)
- unsigned **\_starpu\_mic\_test\_request\_completion** ([struct](#) \_starpu\_async\_channel \*async\_channel)
- void **\_starpu\_mic\_wait\_request\_completion** ([struct](#) \_starpu\_async\_channel \*async\_channel)

- `int _starpu_mic_copy_data_from_mic_to_cpu` (`starpu_data_handle_t` handle, `void *src_interface`, unsigned `src_node`, `void *dst_interface`, unsigned `dst_node`, `struct _starpu_data_request` \*req)
- `int _starpu_mic_copy_data_from_cpu_to_mic` (`starpu_data_handle_t` handle, `void *src_interface`, unsigned `src_node`, `void *dst_interface`, unsigned `dst_node`, `struct _starpu_data_request` \*req)
- `int _starpu_mic_copy_interface_from_mic_to_cpu` (`uintptr_t` src, `size_t` src\_offset, unsigned `src_node`, `uintptr_t` dst, `size_t` dst\_offset, unsigned `dst_node`, `size_t` size, `struct _starpu_async_channel` \*async\_↵ channel)
- `int _starpu_mic_copy_interface_from_cpu_to_mic` (`uintptr_t` src, `size_t` src\_offset, unsigned `src_node`, `uintptr_t` dst, `size_t` dst\_offset, unsigned `dst_node`, `size_t` size, `struct _starpu_async_channel` \*async\_↵ channel)
- `int _starpu_mic_is_direct_access_supported` (unsigned `node`, unsigned `handling_node`)
- `uintptr_t _starpu_mic_malloc_on_node` (unsigned `dst_node`, `size_t` size, int flags)
- `void _starpu_mic_free_on_node` (unsigned `dst_node`, `uintptr_t` addr, `size_t` size, int flags)

## Variables

- `struct _starpu_node_ops _starpu_driver_mic_node_ops`
- `struct _starpu_mp_node * _starpu_mic_nodes` [STARPU\_MAXMICDEVS]
- `struct _starpu_mic_async_event * event`
- `void (*)(void) _starpu_mic_src_get_kernel_from_job` (const `struct _starpu_mp_node` \*node STARPU\_↵ ATTRIBUTE\_UNUSED, `struct _starpu_job` \*j)

### 6.24.1 Variable Documentation

#### 6.24.1.1 \_starpu\_mic\_nodes

`struct _starpu_mp_node* _starpu_mic_nodes`[STARPU\_MAXMICDEVS] [extern]

Array of structures containing all the informations useful to send and receive informations with devices

## 6.25 driver\_mpi\_common.h File Reference

```
#include <drivers/mp_common/mp_common.h>
#include <drivers/mpi/driver_mpi_source.h>
```

## 6.26 driver\_mpi\_sink.h File Reference

```
#include <drivers/mp_common/sink_common.h>
```

## 6.27 driver\_mpi\_source.h File Reference

```
#include <drivers/mp_common/mp_common.h>
#include <starpu_mpi_ms.h>
#include <datawizard/node_ops.h>
```

## 6.28 driver\_opencl.h File Reference

```
#include <CL/cl.h>
#include <core/workers.h>
#include <datawizard/node_ops.h>
```



## Macros

- `#define _GNU_SOURCE`
- `#define CL_TARGET_OPENCL_VERSION`

## Functions

- `void _starpu_opencil_discover_devices (struct _starpu_machine_config *config)`
- `unsigned _starpu_opencil_get_device_count (void)`
- `void _starpu_opencil_init (void)`
- `void * _starpu_opencil_worker (void *)`
- `int _starpu_run_opencil (struct _starpu_worker *)`
- `int _starpu_opencil_driver_init (struct _starpu_worker *)`
- `int _starpu_opencil_driver_run_once (struct _starpu_worker *)`
- `int _starpu_opencil_driver_deinit (struct _starpu_worker *)`
- `int _starpu_opencil_init_context (int devid)`
- `int _starpu_opencil_deinit_context (int devid)`
- `cl_device_type _starpu_opencil_get_device_type (int devid)`
- `unsigned _starpu_opencil_test_request_completion (struct _starpu_async_channel *async_channel)`
- `void _starpu_opencil_wait_request_completion (struct _starpu_async_channel *async_channel)`
- `int _starpu_opencil_copy_interface_from_opencil_to_opencil (starpu_data_handle_t handle, void *src_↵_interface, unsigned src_node, void *dst_interface, unsigned dst_node, struct _starpu_data_request *req)`
- `int _starpu_opencil_copy_interface_from_opencil_to_cpu (starpu_data_handle_t handle, void *src_↵_interface, unsigned src_node, void *dst_interface, unsigned dst_node, struct _starpu_data_request *req)`
- `int _starpu_opencil_copy_interface_from_cpu_to_opencil (starpu_data_handle_t handle, void *src_↵_interface, unsigned src_node, void *dst_interface, unsigned dst_node, struct _starpu_data_request *req)`
- `int _starpu_opencil_copy_data_from_opencil_to_cpu (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel *async_↵_channel)`
- `int _starpu_opencil_copy_data_from_opencil_to_opencil (uintptr_t src, size_t src_offset, unsigned src_↵_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel *async_↵_channel)`
- `int _starpu_opencil_copy_data_from_cpu_to_opencil (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel *async_↵_channel)`
- `int _starpu_opencil_is_direct_access_supported (unsigned node, unsigned handling_node)`
- `uintptr_t _starpu_opencil_malloc_on_node (unsigned dst_node, size_t size, int flags)`
- `void _starpu_opencil_free_on_node (unsigned dst_node, uintptr_t addr, size_t size, int flags)`

## Variables

- `struct _starpu_node_ops _starpu_driver_opencil_node_ops`
- `struct _starpu_driver_ops _starpu_driver_opencil_ops`
- `char * _starpu_opencil_program_dir`

## 6.29 driver\_opencil\_utils.h File Reference

### Macros

- `#define _STARPU_OPENCL_PLATFORM_MAX`

### Functions

- `char * _starpu_opencil_get_device_type_as_string (int id)`

## 6.30 drivers.h File Reference

### Data Structures

- struct [\\_starpu\\_driver\\_ops](#)

## 6.31 errorcheck.h File Reference

```
#include <starpu.h>
```

### Enumerations

- enum [\\_starpu\\_worker\\_status](#) {  
[STATUS\\_INVALID](#) , [STATUS\\_UNKNOWN](#) , [STATUS\\_INITIALIZING](#) , [STATUS\\_EXECUTING](#) ,  
[STATUS\\_CALLBACK](#) , [STATUS\\_SCHEDULING](#) , [STATUS\\_WAITING](#) , [STATUS\\_SLEEPING\\_SCHEDULING](#)  
, [STATUS\\_SLEEPING](#) }

### Functions

- void [\\_starpu\\_set\\_worker\\_status](#) (struct [\\_starpu\\_worker](#) \*worker, enum [\\_starpu\\_worker\\_status](#) st)
- void [\\_starpu\\_set\\_local\\_worker\\_status](#) (enum [\\_starpu\\_worker\\_status](#) st)
- enum [\\_starpu\\_worker\\_status](#) [\\_starpu\\_get\\_local\\_worker\\_status](#) (void)
- unsigned [\\_starpu\\_worker\\_may\\_perform\\_blocking\\_calls](#) (void)

### 6.31.1 Enumeration Type Documentation

#### 6.31.1.1 [\\_starpu\\_worker\\_status](#)

enum [\\_starpu\\_worker\\_status](#)

This type describes in which state a worker may be.

#### Enumerator

<a href="#">STATUS_INVALID</a>	invalid status (for instance if we request the status of some thread that is not controlled by StarPU)
<a href="#">STATUS_UNKNOWN</a>	everything that does not fit the other status
<a href="#">STATUS_INITIALIZING</a>	during the initialization
<a href="#">STATUS_EXECUTING</a>	during the execution of a codelet
<a href="#">STATUS_CALLBACK</a>	during the execution of the callback
<a href="#">STATUS_SCHEDULING</a>	while executing the scheduler code
<a href="#">STATUS_WAITING</a>	while waiting for a data transfer
<a href="#">STATUS_SLEEPING_SCHEDULING</a>	while sleeping because there is nothing to do, but looking for tasks to do
<a href="#">STATUS_SLEEPING</a>	while sleeping because there is nothing to do, and not even scheduling

### 6.31.2 Function Documentation

### 6.31.2.1 `_starpu_set_worker_status()`

```
void _starpu_set_worker_status (
    struct _starpu_worker * worker,
    enum _starpu_worker_status st )
```

Specify what the local worker is currently doing (eg. executing a callback). This permits to detect if this is legal to do a blocking call for instance.

### 6.31.2.2 `_starpu_get_local_worker_status()`

```
enum _starpu_worker_status _starpu_get_local_worker_status (
    void )
```

Indicate what type of operation the worker is currently doing.

### 6.31.2.3 `_starpu_worker_may_perform_blocking_calls()`

```
unsigned _starpu_worker_may_perform_blocking_calls (
    void )
```

It is forbidden to do blocking calls during some operations such as callback or during the execution of a task. This function indicates whether it is legal to call a blocking operation in the current context.

## 6.32 `fifo_queues.h` File Reference

```
#include <starpu.h>
#include <core/task.h>
```

### Data Structures

- [struct `\_starpu\_fifo\_taskq`](#)

### Functions

- [struct `\_starpu\_fifo\_taskq`](#) \* `_starpu_create_fifo` (void) STARPU\_ATTRIBUTE\_MALLOC
- void `_starpu_destroy_fifo` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo)
- int `_starpu_fifo_empty` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo)
- double `_starpu_fifo_get_exp_len_prev_task_list` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo\_queue, [struct](#) starpu\_task \*task, int workerid, int nimpl, int \*fifo\_ntasks)
- int `_starpu_fifo_push_sorted_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo\_queue, [struct](#) starpu\_task \*task)
- int `_starpu_fifo_push_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo, [struct](#) starpu\_task \*task)
- int `_starpu_fifo_push_back_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo\_queue, [struct](#) starpu\_task \*task)
- int `_starpu_fifo_pop_this_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo\_queue, int workerid, [struct](#) starpu\_task \*task)
- [struct](#) starpu\_task \* `_starpu_fifo_pop_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo, int workerid)
- [struct](#) starpu\_task \* `_starpu_fifo_pop_local_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo)
- [struct](#) starpu\_task \* `_starpu_fifo_pop_every_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo, int workerid)
- int `_starpu_normalize_prio` (int priority, int num\_priorities, unsigned sched\_ctx\_id)
- int `_starpu_count_non_ready_buffers` ([struct](#) starpu\_task \*task, unsigned worker)
- size\_t `_starpu_size_non_ready_buffers` ([struct](#) starpu\_task \*task, unsigned worker)
- [struct](#) starpu\_task \* `_starpu_fifo_pop_first_ready_task` ([struct `\_starpu\_fifo\_taskq`](#) \*fifo\_queue, unsigned workerid, int num\_priorities)

### 6.32.1 Data Structure Documentation

#### 6.32.1.1 `struct _starpu_fifo_taskq`

## Data Fields

<a href="#">struct</a> starpu_task_list	taskq	the actual list
unsigned	ntasks	the number of tasks currently in the queue
unsigned *	ntasks_per_priority	the number of tasks currently in the queue corresponding to each priority
unsigned	nprocessed	the number of tasks that were processed
double	exp_start	only meaningful if the queue is only used by a single worker
double	exp_end	Expected start date of next item to do in the queue (i.e. not started yet). This is thus updated when we start it.
double	exp_len	Expected end date of last task in the queue
double *	exp_len_per_priority	Expected duration of the set of tasks in the queue
double	pipeline_len	Expected duration of the set of tasks in the queue corresponding to each priority

## 6.33 filters.h File Reference

```
#include <stdarg.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
#include <starpu.h>
#include <common/config.h>
```

### Functions

- void [\\_starpu\\_data\\_partition\\_access\\_submit](#) (starpu\_data\_handle\_t target, int write)

#### 6.33.1 Function Documentation

##### 6.33.1.1 \_starpu\_data\_partition\_access\_submit()

```
void _starpu_data_partition_access_submit (
    starpu_data_handle_t target,
    int write )
```

submit asynchronous unpartitioning / partitioning to make target active read-only or read-write

## 6.34 footprint.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <core/jobs.h>
```

### Functions

- uint32\_t [\\_starpu\\_compute\\_buffers\\_footprint](#) (struct starpu\_perfmodel \*model, [struct](#) starpu\_perfmodel\_arch \*arch, unsigned nimpl, [struct](#) \_starpu\_job \*)
- uint32\_t [\\_starpu\\_compute\\_data\\_footprint](#) (starpu\_data\_handle\_t handle)
- uint32\_t [\\_starpu\\_compute\\_data\\_alloc\\_footprint](#) (starpu\_data\_handle\_t handle)

#### 6.34.1 Function Documentation

#### 6.34.1.1 `_starpu_compute_buffers_footprint()`

```
uint32_t _starpu_compute_buffers_footprint (
    struct starpu_perfmodel * model,
    struct starpu_perfmodel_arch * arch,
    unsigned nimpl,
    struct _starpu_job * j )
```

Compute the footprint that characterizes the job and cache it into the job structure.

#### 6.34.1.2 `_starpu_compute_data_footprint()`

```
uint32_t _starpu_compute_data_footprint (
    starpu_data_handle_t handle )
```

Compute the footprint that characterizes the layout of the data handle.

#### 6.34.1.3 `_starpu_compute_data_alloc_footprint()`

```
uint32_t _starpu_compute_data_alloc_footprint (
    starpu_data_handle_t handle )
```

Compute the footprint that characterizes the allocation of the data handle.

## 6.35 `fxt.h` File Reference

```
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <common/config.h>
#include <common/utils.h>
#include <starpu.h>
```

### Macros

- `#define _GNU_SOURCE`
- `#define _STARPU_FUT_APPS_KEY`
- `#define _STARPU_FUT_CPU_KEY`
- `#define _STARPU_FUT_CUDA_KEY`
- `#define _STARPU_FUT_OPENCL_KEY`
- `#define _STARPU_FUT_MIC_KEY`
- `#define _STARPU_FUT_MPI_KEY`
- `#define _STARPU_FUT_WORKER_INIT_START`
- `#define _STARPU_FUT_WORKER_INIT_END`
- `#define _STARPU_FUT_START_CODELET_BODY`
- `#define _STARPU_FUT_END_CODELET_BODY`
- `#define _STARPU_FUT_JOB_PUSH`
- `#define _STARPU_FUT_JOB_POP`
- `#define _STARPU_FUT_UPDATE_TASK_CNT`
- `#define _STARPU_FUT_START_FETCH_INPUT_ON_TID`
- `#define _STARPU_FUT_END_FETCH_INPUT_ON_TID`
- `#define _STARPU_FUT_START_PUSH_OUTPUT_ON_TID`
- `#define _STARPU_FUT_END_PUSH_OUTPUT_ON_TID`
- `#define _STARPU_FUT_TAG`
- `#define _STARPU_FUT_TAG_DEPS`
- `#define _STARPU_FUT_TASK_DEPS`
- `#define _STARPU_FUT_DATA_COPY`
- `#define _STARPU_FUT_WORK_STEALING`
- `#define _STARPU_FUT_WORKER_DEINIT_START`

- `#define _STARPU_FUT_WORKER_DEINIT_END`
- `#define _STARPU_FUT_WORKER_SLEEP_START`
- `#define _STARPU_FUT_WORKER_SLEEP_END`
- `#define _STARPU_FUT_TASK_SUBMIT`
- `#define _STARPU_FUT_CODELET_DATA_HANDLE`
- `#define _STARPU_FUT_MODEL_NAME`
- `#define _STARPU_FUT_DATA_NAME`
- `#define _STARPU_FUT_DATA_COORDINATES`
- `#define _STARPU_FUT_HANDLE_DATA_UNREGISTER`
- `#define _STARPU_FUT_USER_DEFINED_START`
- `#define _STARPU_FUT_USER_DEFINED_END`
- `#define _STARPU_FUT_NEW_MEM_NODE`
- `#define _STARPU_FUT_START_CALLBACK`
- `#define _STARPU_FUT_END_CALLBACK`
- `#define _STARPU_FUT_TASK_DONE`
- `#define _STARPU_FUT_TAG_DONE`
- `#define _STARPU_FUT_START_ALLOC`
- `#define _STARPU_FUT_END_ALLOC`
- `#define _STARPU_FUT_START_ALLOC_REUSE`
- `#define _STARPU_FUT_END_ALLOC_REUSE`
- `#define _STARPU_FUT_USED_MEM`
- `#define _STARPU_FUT_TASK_NAME`
- `#define _STARPU_FUT_DATA_WONT_USE`
- `#define _STARPU_FUT_TASK_COLOR`
- `#define _STARPU_FUT_DATA_DOING_WONT_USE`
- `#define _STARPU_FUT_START_MEMRECLAIM`
- `#define _STARPU_FUT_END_MEMRECLAIM`
- `#define _STARPU_FUT_START_DRIVER_COPY`
- `#define _STARPU_FUT_END_DRIVER_COPY`
- `#define _STARPU_FUT_START_DRIVER_COPY_ASYNC`
- `#define _STARPU_FUT_END_DRIVER_COPY_ASYNC`
- `#define _STARPU_FUT_START_PROGRESS_ON_TID`
- `#define _STARPU_FUT_END_PROGRESS_ON_TID`
- `#define _STARPU_FUT_USER_EVENT`
- `#define _STARPU_FUT_SET_PROFILING`
- `#define _STARPU_FUT_TASK_WAIT_FOR_ALL`
- `#define _STARPU_FUT_EVENT`
- `#define _STARPU_FUT_THREAD_EVENT`
- `#define _STARPU_FUT_CODELET_DETAILS`
- `#define _STARPU_FUT_CODELET_DATA`
- `#define _STARPU_FUT_LOCKING_MUTEX`
- `#define _STARPU_FUT_MUTEX_LOCKED`
- `#define _STARPU_FUT_UNLOCKING_MUTEX`
- `#define _STARPU_FUT_MUTEX_UNLOCKED`
- `#define _STARPU_FUT_TRYLOCK_MUTEX`
- `#define _STARPU_FUT_RDLOCKING_RWLOCK`
- `#define _STARPU_FUT_RWLOCK_RDLOCKED`
- `#define _STARPU_FUT_WRLOCKING_RWLOCK`
- `#define _STARPU_FUT_RWLOCK_WRLOCKED`
- `#define _STARPU_FUT_UNLOCKING_RWLOCK`
- `#define _STARPU_FUT_RWLOCK_UNLOCKED`
- `#define _STARPU_FUT_LOCKING_SPINLOCK`
- `#define _STARPU_FUT_SPINLOCK_LOCKED`
- `#define _STARPU_FUT_UNLOCKING_SPINLOCK`
- `#define _STARPU_FUT_SPINLOCK_UNLOCKED`

- #define \_STARPU\_FUT\_TRYLOCK\_SPINLOCK
- #define \_STARPU\_FUT\_COND\_WAIT\_BEGIN
- #define \_STARPU\_FUT\_COND\_WAIT\_END
- #define \_STARPU\_FUT\_MEMORY\_FULL
- #define \_STARPU\_FUT\_DATA\_LOAD
- #define \_STARPU\_FUT\_START\_UNPARTITION\_ON\_TID
- #define \_STARPU\_FUT\_END\_UNPARTITION\_ON\_TID
- #define \_STARPU\_FUT\_START\_FREE
- #define \_STARPU\_FUT\_END\_FREE
- #define \_STARPU\_FUT\_START\_WRITEBACK
- #define \_STARPU\_FUT\_END\_WRITEBACK
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_PUSH\_PRIO
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_POP\_PRIO
- #define \_STARPU\_FUT\_START\_WRITEBACK\_ASYNC
- #define \_STARPU\_FUT\_END\_WRITEBACK\_ASYNC
- #define \_STARPU\_FUT\_HYPERVISOR\_BEGIN
- #define \_STARPU\_FUT\_HYPERVISOR\_END
- #define \_STARPU\_FUT\_BARRIER\_WAIT\_BEGIN
- #define \_STARPU\_FUT\_BARRIER\_WAIT\_END
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_START
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_END
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_PUSH
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_POP
- #define \_STARPU\_FUT\_START\_EXECUTING
- #define \_STARPU\_FUT\_END\_EXECUTING
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_NEW
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_CONNECT
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_PUSH
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_PULL
- #define \_STARPU\_FUT\_TASK\_SUBMIT\_START
- #define \_STARPU\_FUT\_TASK\_SUBMIT\_END
- #define \_STARPU\_FUT\_TASK\_BUILD\_START
- #define \_STARPU\_FUT\_TASK\_BUILD\_END
- #define \_STARPU\_FUT\_TASK\_MPI\_DECODE\_START
- #define \_STARPU\_FUT\_TASK\_MPI\_DECODE\_END
- #define \_STARPU\_FUT\_TASK\_MPI\_PRE\_START
- #define \_STARPU\_FUT\_TASK\_MPI\_PRE\_END
- #define \_STARPU\_FUT\_TASK\_MPI\_POST\_START
- #define \_STARPU\_FUT\_TASK\_MPI\_POST\_END
- #define \_STARPU\_FUT\_TASK\_WAIT\_START
- #define \_STARPU\_FUT\_TASK\_WAIT\_END
- #define \_STARPU\_FUT\_TASK\_WAIT\_FOR\_ALL\_START
- #define \_STARPU\_FUT\_TASK\_WAIT\_FOR\_ALL\_END
- #define \_STARPU\_FUT\_HANDLE\_DATA\_REGISTER
- #define \_STARPU\_FUT\_START\_FETCH\_INPUT
- #define \_STARPU\_FUT\_END\_FETCH\_INPUT
- #define \_STARPU\_FUT\_TASK\_THROTTLE\_START
- #define \_STARPU\_FUT\_TASK\_THROTTLE\_END
- #define \_STARPU\_FUT\_DATA\_STATE\_INVALID
- #define \_STARPU\_FUT\_DATA\_STATE\_OWNER
- #define \_STARPU\_FUT\_DATA\_STATE\_SHARED
- #define \_STARPU\_FUT\_DATA\_REQUEST\_CREATED
- #define \_STARPU\_FUT\_TASK\_EXCLUDE\_FROM\_DAG
- #define \_STARPU\_TRACE\_NEW\_MEM\_NODE(nodeid)
- #define \_STARPU\_TRACE\_WORKER\_INIT\_START(a, b, c, d, e, f)

- #define `_STARPU_TRACE_WORKER_INIT_END`(workerid)
- #define `_STARPU_TRACE_START_CODELET_BODY`(job, nimpl, perf\_arch, workerid)
- #define `_STARPU_TRACE_END_CODELET_BODY`(job, nimpl, perf\_arch, workerid)
- #define `_STARPU_TRACE_START_EXECUTING`()
- #define `_STARPU_TRACE_END_EXECUTING`()
- #define `_STARPU_TRACE_START_CALLBACK`(job)
- #define `_STARPU_TRACE_END_CALLBACK`(job)
- #define `_STARPU_TRACE_JOB_PUSH`(task, prio)
- #define `_STARPU_TRACE_JOB_POP`(task, prio)
- #define `_STARPU_TRACE_UPDATE_TASK_CNT`(counter)
- #define `_STARPU_TRACE_START_FETCH_INPUT`(job)
- #define `_STARPU_TRACE_END_FETCH_INPUT`(job)
- #define `_STARPU_TRACE_START_PUSH_OUTPUT`(job)
- #define `_STARPU_TRACE_END_PUSH_OUTPUT`(job)
- #define `_STARPU_TRACE_TAG`(tag, job)
- #define `_STARPU_TRACE_TAG_DEPS`(a, b)
- #define `_STARPU_TRACE_TASK_DEPS`(a, b)
- #define `_STARPU_TRACE_GHOST_TASK_DEPS`(a, b)
- #define `_STARPU_TRACE_TASK_EXCLUDE_FROM_DAG`(a)
- #define `_STARPU_TRACE_TASK_NAME`(a)
- #define `_STARPU_TRACE_TASK_COLOR`(a)
- #define `_STARPU_TRACE_TASK_DONE`(a)
- #define `_STARPU_TRACE_TAG_DONE`(a)
- #define `_STARPU_TRACE_DATA_NAME`(a, b)
- #define `_STARPU_TRACE_DATA_COORDINATES`(a, b, c)
- #define `_STARPU_TRACE_DATA_COPY`(a, b, c)
- #define `_STARPU_TRACE_DATA_WONT_USE`(a)
- #define `_STARPU_TRACE_DATA_DOING_WONT_USE`(a)
- #define `_STARPU_TRACE_START_DRIVER_COPY`(a, b, c, d, e, f)
- #define `_STARPU_TRACE_END_DRIVER_COPY`(a, b, c, d, e)
- #define `_STARPU_TRACE_START_DRIVER_COPY_ASYNC`(a, b)
- #define `_STARPU_TRACE_END_DRIVER_COPY_ASYNC`(a, b)
- #define `_STARPU_TRACE_WORK_STEALING`(a, b)
- #define `_STARPU_TRACE_WORKER_DEINIT_START`
- #define `_STARPU_TRACE_WORKER_DEINIT_END`(a)
- #define `_STARPU_TRACE_WORKER_SCHEDULING_START`
- #define `_STARPU_TRACE_WORKER_SCHEDULING_END`
- #define `_STARPU_TRACE_WORKER_SCHEDULING_PUSH`
- #define `_STARPU_TRACE_WORKER_SCHEDULING_POP`
- #define `_STARPU_TRACE_WORKER_SLEEP_START`
- #define `_STARPU_TRACE_WORKER_SLEEP_END`
- #define `_STARPU_TRACE_TASK_SUBMIT`(job, a, b)
- #define `_STARPU_TRACE_TASK_SUBMIT_START`()
- #define `_STARPU_TRACE_TASK_SUBMIT_END`()
- #define `_STARPU_TRACE_TASK_THROTTLE_START`()
- #define `_STARPU_TRACE_TASK_THROTTLE_END`()
- #define `_STARPU_TRACE_TASK_BUILD_START`()
- #define `_STARPU_TRACE_TASK_BUILD_END`()
- #define `_STARPU_TRACE_TASK_MPI_DECODE_START`()
- #define `_STARPU_TRACE_TASK_MPI_DECODE_END`()
- #define `_STARPU_TRACE_TASK_MPI_PRE_START`()
- #define `_STARPU_TRACE_TASK_MPI_PRE_END`()
- #define `_STARPU_TRACE_TASK_MPI_POST_START`()
- #define `_STARPU_TRACE_TASK_MPI_POST_END`()
- #define `_STARPU_TRACE_TASK_WAIT_START`(job)



- #define \_STARPU\_TRACE\_TASK\_WAIT\_END()
- #define \_STARPU\_TRACE\_TASK\_WAIT\_FOR\_ALL\_START()
- #define \_STARPU\_TRACE\_TASK\_WAIT\_FOR\_ALL\_END()
- #define \_STARPU\_TRACE\_USER\_DEFINED\_START()
- #define \_STARPU\_TRACE\_USER\_DEFINED\_END()
- #define \_STARPU\_TRACE\_START\_ALLOC(memnode, size, handle, is\_prefetch)
- #define \_STARPU\_TRACE\_END\_ALLOC(memnode, handle, r)
- #define \_STARPU\_TRACE\_START\_ALLOC\_REUSE(a, size, handle, is\_prefetch)
- #define \_STARPU\_TRACE\_END\_ALLOC\_REUSE(a, handle, r)
- #define \_STARPU\_TRACE\_START\_FREE(memnode, size, handle)
- #define \_STARPU\_TRACE\_END\_FREE(memnode, handle)
- #define \_STARPU\_TRACE\_START\_WRITEBACK(memnode, handle)
- #define \_STARPU\_TRACE\_END\_WRITEBACK(memnode, handle)
- #define \_STARPU\_TRACE\_USED\_MEM(memnode, used)
- #define \_STARPU\_TRACE\_START\_MEMRECLAIM(memnode, is\_prefetch)
- #define \_STARPU\_TRACE\_END\_MEMRECLAIM(memnode, is\_prefetch)
- #define \_STARPU\_TRACE\_START\_WRITEBACK\_ASYNC(memnode)
- #define \_STARPU\_TRACE\_END\_WRITEBACK\_ASYNC(memnode)
- #define \_STARPU\_TRACE\_START\_PROGRESS(memnode)
- #define \_STARPU\_TRACE\_END\_PROGRESS(memnode)
- #define \_STARPU\_TRACE\_USER\_EVENT(code)
- #define \_STARPU\_TRACE\_SET\_PROFILING(status)
- #define \_STARPU\_TRACE\_TASK\_WAIT\_FOR\_ALL()
- #define \_STARPU\_TRACE\_EVENT(S)
- #define \_STARPU\_TRACE\_THREAD\_EVENT(S)
- #define \_STARPU\_TRACE\_LOCKING\_MUTEX()
- #define \_STARPU\_TRACE\_MUTEX\_LOCKED()
- #define \_STARPU\_TRACE\_UNLOCKING\_MUTEX()
- #define \_STARPU\_TRACE\_MUTEX\_UNLOCKED()
- #define \_STARPU\_TRACE\_TRYLOCK\_MUTEX()
- #define \_STARPU\_TRACE\_RDLOCKING\_RWLOCK()
- #define \_STARPU\_TRACE\_RWLOCK\_RDLOCKED()
- #define \_STARPU\_TRACE\_WRLOCKING\_RWLOCK()
- #define \_STARPU\_TRACE\_RWLOCK\_WRLOCKED()
- #define \_STARPU\_TRACE\_UNLOCKING\_RWLOCK()
- #define \_STARPU\_TRACE\_RWLOCK\_UNLOCKED()
- #define \_STARPU\_TRACE\_LOCKING\_SPINLOCK(file, line)
- #define \_STARPU\_TRACE\_SPINLOCK\_LOCKED(file, line)
- #define \_STARPU\_TRACE\_UNLOCKING\_SPINLOCK(file, line)
- #define \_STARPU\_TRACE\_SPINLOCK\_UNLOCKED(file, line)
- #define \_STARPU\_TRACE\_TRYLOCK\_SPINLOCK(file, line)
- #define \_STARPU\_TRACE\_COND\_WAIT\_BEGIN()
- #define \_STARPU\_TRACE\_COND\_WAIT\_END()
- #define \_STARPU\_TRACE\_BARRIER\_WAIT\_BEGIN()
- #define \_STARPU\_TRACE\_BARRIER\_WAIT\_END()
- #define \_STARPU\_TRACE\_MEMORY\_FULL(size)
- #define \_STARPU\_TRACE\_DATA\_LOAD(workerid, size)
- #define \_STARPU\_TRACE\_START\_UNPARTITION(handle, memnode)
- #define \_STARPU\_TRACE\_END\_UNPARTITION(handle, memnode)
- #define \_STARPU\_TRACE\_SCHED\_COMPONENT\_PUSH\_PRIO(workerid, ntasks, exp\_len)
- #define \_STARPU\_TRACE\_SCHED\_COMPONENT\_POP\_PRIO(workerid, ntasks, exp\_len)
- #define \_STARPU\_TRACE\_HYPERVISOR\_BEGIN()
- #define \_STARPU\_TRACE\_HYPERVISOR\_END()
- #define \_STARPU\_TRACE\_SCHED\_COMPONENT\_NEW(component)
- #define \_STARPU\_TRACE\_SCHED\_COMPONENT\_CONNECT(parent, child)

- `#define _STARPU_TRACE_SCHED_COMPONENT_PUSH(from, to, task, prio)`
- `#define _STARPU_TRACE_SCHED_COMPONENT_PULL(from, to, task)`
- `#define _STARPU_TRACE_HANDLE_DATA_REGISTER(handle)`
- `#define _STARPU_TRACE_HANDLE_DATA_UNREGISTER(handle)`
- `#define _STARPU_TRACE_WORKER_START_FETCH_INPUT(job, id)`
- `#define _STARPU_TRACE_WORKER_END_FETCH_INPUT(job, id)`
- `#define _STARPU_TRACE_DATA_STATE_INVALID(handle, node)`
- `#define _STARPU_TRACE_DATA_STATE_OWNER(handle, node)`
- `#define _STARPU_TRACE_DATA_STATE_SHARED(handle, node)`
- `#define _STARPU_TRACE_DATA_REQUEST_CREATED(handle, orig, dest, prio, is_pre)`

## Functions

- static unsigned long `_starpu_fxt_get_job_id` (void)

## Variables

- unsigned long `_starpu_job_cnt`

## 6.36 graph.h File Reference

```
#include <common/list.h>
```

## Data Structures

- struct [\\_starpu\\_graph\\_node](#)

## Functions

- void `_starpu_graph_init` (void)
- void `_starpu_graph_wrlock` (void)
- void `_starpu_graph_rdlock` (void)
- void `_starpu_graph_wrunlock` (void)
- void `_starpu_graph_rdunlock` (void)
- void `_starpu_graph_add_job` (struct [\\_starpu\\_job](#) \*job)
- void `_starpu_graph_add_job_dep` (struct [\\_starpu\\_job](#) \*job, struct [\\_starpu\\_job](#) \*prev\_job)
- void `_starpu_graph_drop_job` (struct [\\_starpu\\_job](#) \*job)
- void `_starpu_graph_drop_dropped_nodes` (void)
- void `_starpu_graph_compute_depths` (void)
- void `_starpu_graph_compute_descendants` (void)
- void `_starpu_graph_foreach` (void(\*func)(void \*data, struct [\\_starpu\\_graph\\_node](#) \*node), void \*data)

## Variables

- int `_starpu_graph_record`

### 6.36.1 Data Structure Documentation

#### 6.36.1.1 struct [\\_starpu\\_graph\\_node](#)

##### Data Fields

<code>starpu_thread_mutex_t</code>	mutex	protects access to the job
<code>struct <a href="#">_starpu_job</a> *</code>	job	pointer to the job, if it is still alive, NULL otherwise

## Data Fields

<code>struct _starpu_graph_node_multilist_top</code>	top	Fields for graph analysis for scheduling heuristics Member of list of all jobs without incoming dependency
<code>struct _starpu_graph_node_multilist_bottom</code>	bottom	Member of list of all jobs without outgoing dependency
<code>struct _starpu_graph_node_multilist_all</code>	all	Member of list of all jobs
<code>struct _starpu_graph_node_multilist_dropped</code>	dropped	Member of list of dropped jobs
<code>struct _starpu_graph_node **</code>	incoming	set of incoming dependencies May contain NULLs for terminated jobs
<code>unsigned *</code>	incoming_slot	Index within corresponding outgoing array
<code>unsigned</code>	n_incoming	Number of slots used
<code>unsigned</code>	alloc_incoming	Size of incoming
<code>struct _starpu_graph_node **</code>	outgoing	set of outgoing dependencies
<code>unsigned *</code>	outgoing_slot	Index within corresponding incoming array
<code>unsigned</code>	n_outgoing	Number of slots used
<code>unsigned</code>	alloc_outgoing	Size of outgoing
<code>unsigned</code>	depth	Rank from bottom, in number of jobs Only available if <code>_starpu_graph_compute_depths</code> was called
<code>unsigned</code>	descendants	Number of children, grand-children, etc. Only available if <code>_starpu_graph_compute_descendants</code> was called
<code>int</code>	graph_n	Variable available for graph flow

## 6.36.2 Function Documentation

### 6.36.2.1 `_starpu_graph_add_job()`

```
void _starpu_graph_add_job (
    struct _starpu_job * job )
```

Add a job to the graph, called before any `_starpu_graph_add_job_dep` call

### 6.36.2.2 `_starpu_graph_add_job_dep()`

```
void _starpu_graph_add_job_dep (
    struct _starpu_job * job,
    struct _starpu_job * prev_job )
```

Add a dependency between jobs

### 6.36.2.3 `_starpu_graph_drop_job()`

```
void _starpu_graph_drop_job (
    struct _starpu_job * job )
```

Remove a job from the graph

### 6.36.2.4 `_starpu_graph_drop_dropped_nodes()`

```
void _starpu_graph_drop_dropped_nodes (
    void )
```

Really drop the nodes from the graph now

**6.36.2.5 \_\_starpup\_graph\_compute\_depths()**

```
void __starpup_graph_compute_depths (
    void )
```

This make StarPU compute for each task the depth, i.e. the length of the longest path to a task without outgoing dependencies. This does not take job duration into account, just the number

**6.36.2.6 \_\_starpup\_graph\_compute\_descendants()**

```
void __starpup_graph_compute_descendants (
    void )
```

Compute the descendants of jobs in the graph

**6.36.2.7 \_\_starpup\_graph\_foreach()**

```
void __starpup_graph_foreach (
    void(*) (void *data, struct __starpup_graph_node *node) func,
    void * data )
```

This calls *func* for each node of the task graph, passing also *data* as it Apply func on each job of the graph

**6.37 helper\_mct.h File Reference****Data Structures**

- struct [\\_\\_starpup\\_mct\\_data](#)

**Functions**

- struct [\\_\\_starpup\\_mct\\_data](#) \* **starpup\_mct\_init\_parameters** (struct starpup\_sched\_component\_mct\_data \*params)
- unsigned **starpup\_mct\_compute\_execution\_times** (struct starpup\_sched\_component \*component, struct starpup\_task \*task, double \*estimated\_lengths, double \*estimated\_transfer\_length, unsigned \*suitable\_components)
- void **starpup\_mct\_compute\_expected\_times** (struct starpup\_sched\_component \*component, struct starpup\_task \*task, double \*estimated\_lengths, double \*estimated\_transfer\_length, double \*estimated\_ends\_with\_task, double \*min\_exp\_end\_with\_task, double \*max\_exp\_end\_with\_task, unsigned \*suitable\_components, unsigned nsuitable\_components)
- double **starpup\_mct\_compute\_fitness** (struct [\\_\\_starpup\\_mct\\_data](#) \*d, double exp\_end, double min\_exp\_end, double max\_exp\_end, double transfer\_len, double local\_energy)
- int **starpup\_mct\_get\_best\_component** (struct [\\_\\_starpup\\_mct\\_data](#) \*d, struct starpup\_task \*task, double \*estimated\_lengths, double \*estimated\_transfer\_length, double \*estimated\_ends\_with\_task, double min\_exp\_end\_with\_task, double max\_exp\_end\_with\_task, unsigned \*suitable\_components, unsigned nsuitable\_components)

**6.37.1 Data Structure Documentation****6.37.1.1 struct \_\_starpup\_mct\_data****Data Fields**

double	alpha	
double	beta	
double	_gamma	
double	idle_power	
starpup_pthread_mutex_t	scheduling_mutex	

## 6.38 idle\_hook.h File Reference

### Functions

- void **\_starpu\_init\_idle\_hooks** (void)
- unsigned **\_starpu\_execute\_registered\_idle\_hooks** (void)

## 6.39 implicit\_data\_deps.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

### Functions

- **struct** starpu\_task \* **\_starpu\_detect\_implicit\_data\_deps\_with\_handle** (**struct** starpu\_task \*pre\_sync\_task, **struct** starpu\_task \*post\_sync\_task, **struct** [\\_starpu\\_task\\_wrapper\\_dlist](#) \*post\_sync\_task\_dependency\_slot, starpu\_data\_handle\_t handle, enum starpu\_data\_access\_mode mode, unsigned task\_handle\_sequential\_consistency)
- int **\_starpu\_test\_implicit\_data\_deps\_with\_handle** (starpu\_data\_handle\_t handle, enum starpu\_data\_access\_mode mode)
- void **\_starpu\_detect\_implicit\_data\_deps** (**struct** starpu\_task \*task)
- void **\_starpu\_release\_data\_enforce\_sequential\_consistency** (**struct** starpu\_task \*task, **struct** [\\_starpu\\_task\\_wrapper\\_dlist](#) \*task\_dependency\_slot, starpu\_data\_handle\_t handle)
- void **\_starpu\_release\_task\_enforce\_sequential\_consistency** (**struct** [\\_starpu\\_job](#) \*j)
- void **\_starpu\_add\_post\_sync\_tasks** (**struct** starpu\_task \*post\_sync\_task, starpu\_data\_handle\_t handle)
- void **\_starpu\_unlock\_post\_sync\_tasks** (starpu\_data\_handle\_t handle)
- void **\_starpu\_implicit\_data\_deps\_write\_hook** (void(\*func)(starpu\_data\_handle\_t))
- int **\_starpu\_data\_wait\_until\_available** (starpu\_data\_handle\_t handle, enum starpu\_data\_access\_mode mode, const char \*sync\_name)
- void **\_starpu\_data\_clear\_implicit** (starpu\_data\_handle\_t handle)

### 6.39.1 Function Documentation

#### 6.39.1.1 \_starpu\_implicit\_data\_deps\_write\_hook()

```
void _starpu_implicit_data_deps_write_hook (
    void(*) (starpu_data_handle_t) func )
```

Register a hook to be called when a write is submitted

#### 6.39.1.2 \_starpu\_data\_wait\_until\_available()

```
int _starpu_data_wait_until_available (
    starpu_data_handle_t handle,
    enum starpu_data_access_mode mode,
    const char * sync_name )
```

This function blocks until the handle is available in the requested mode

## 6.40 jobs.h File Reference

```
#include <starpu.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

```

#include <string.h>
#include <stdarg.h>
#include <common/config.h>
#include <common/timing.h>
#include <common/list.h>
#include <common/fxt.h>
#include <core/dependencies/tags.h>
#include <datawizard/datawizard.h>
#include <core/perfmodel/perfmodel.h>
#include <core/errorcheck.h>
#include <common/barrier.h>
#include <common/utils.h>
#include <cuda.h>

```

## Data Structures

- struct [\\_starpu\\_data\\_descr](#)
- struct [\\_starpu\\_job](#)

## Macros

- `#define _STARPU_CPU_MAY_PERFORM(j)`
- `#define _STARPU_CUDA_MAY_PERFORM(j)`
- `#define _STARPU_OPENCL_MAY_PERFORM(j)`
- `#define _STARPU_MIC_MAY_PERFORM(j)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_INDEX(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_HANDLE(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_MODE(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_NODE(job, i)`
- `#define _STARPU_JOB_SET_ORDERED_BUFFER_INDEX(job, __index, i)`
- `#define _STARPU_JOB_SET_ORDERED_BUFFER_HANDLE(job, __handle, i)`
- `#define _STARPU_JOB_SET_ORDERED_BUFFER_MODE(job, __mode, i)`
- `#define _STARPU_JOB_SET_ORDERED_BUFFER_NODE(job, __node, i)`
- `#define _STARPU_JOB_SET_ORDERED_BUFFER(job, buffer, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFERS(job)`
- `#define _STARPU_JOB_GET_DEP_SLOTS(job)`

## Typedefs

- typedef void(\* [\\_starpu\\_cl\\_func\\_t](#)) (void \*\*, void \*)

## Functions

- void [\\_starpu\\_job\\_init](#) (void)
- void [\\_starpu\\_job\\_fini](#) (void)
- struct [\\_starpu\\_job](#) \* [\\_starpu\\_job\\_create](#) (struct [starpu\\_task](#) \*task) STARPU\_ATTRIBUTE\_MALLOC
- void [\\_starpu\\_job\\_destroy](#) (struct [\\_starpu\\_job](#) \*j)
- int [\\_starpu\\_job\\_finished](#) (struct [\\_starpu\\_job](#) \*j)
- void [\\_starpu\\_wait\\_job](#) (struct [\\_starpu\\_job](#) \*j)
- int [\\_starpu\\_test\\_job\\_termination](#) (struct [\\_starpu\\_job](#) \*j)
- void [\\_starpu\\_job\\_prepare\\_for\\_continuation\\_ext](#) (struct [\\_starpu\\_job](#) \*j, unsigned continuation\_resubmit, void(\*continuation\_callback\_on\_sleep)(void \*arg), void \*continuation\_callback\_on\_sleep\_arg)
- void [\\_starpu\\_job\\_prepare\\_for\\_continuation](#) (struct [\\_starpu\\_job](#) \*j)
- void [\\_starpu\\_job\\_set\\_omp\\_cleanup\\_callback](#) (struct [\\_starpu\\_job](#) \*j, void(\*omp\_cleanup\_callback)(void \*arg), void \*omp\_cleanup\_callback\_arg)

- void `_starpu_exclude_task_from_dag` (struct starpu\_task \*task)
- unsigned `_starpu_enforce_deps_and_schedule` (struct \_starpu\_job \*j)
- unsigned `_starpu_enforce_deps_starting_from_task` (struct \_starpu\_job \*j)
- unsigned `_starpu_reenforce_task_deps_and_schedule` (struct \_starpu\_job \*j)
- void `_starpu_enforce_deps_notify_job_ready_soon` (struct \_starpu\_job \*j, \_starpu\_notify\_job\_start\_data \*data, int tag)
- void `_starpu_handle_job_submission` (struct \_starpu\_job \*j)
- void `_starpu_handle_job_termination` (struct \_starpu\_job \*j)
- size\_t `_starpu_job_get_data_size` (struct starpu\_perfmodel \*model, struct starpu\_perfmodel\_arch \*arch, unsigned nimpl, struct \_starpu\_job \*j)
- struct starpu\_task \* `_starpu_pop_local_task` (struct \_starpu\_worker \*worker)
- int `_starpu_push_local_task` (struct \_starpu\_worker \*worker, struct starpu\_task \*task, int prio)

## 6.40.1 Data Structure Documentation

### 6.40.1.1 struct \_starpu\_data\_descr

#### Data Fields

starpu_data_handle_t	handle	
enum starpu_data_access_mode	mode	
int	node	
int	index	This is the value actually chosen, only set by <code>_starpu_fetch_task_input</code> for coherency with <code>_starpu_push_task_output</code>
int	orderedindex	

## 6.40.2 Typedef Documentation

### 6.40.2.1 \_starpu\_cl\_func\_t

```
typedef void(* _starpu_cl_func_t) (void **, void *)
codelet function
```

## 6.40.3 Function Documentation

### 6.40.3.1 \_starpu\_job\_create()

```
struct _starpu_job* _starpu_job_create (
    struct starpu_task * task )
```

Create an internal struct `_starpu_job` \*structure to encapsulate the task.

### 6.40.3.2 \_starpu\_job\_destroy()

```
void _starpu_job_destroy (
    struct _starpu_job * j )
```

Destroy the data structure associated to the job structure

### 6.40.3.3 \_starpu\_job\_finished()

```
int _starpu_job_finished (
    struct _starpu_job * j )
```

Test for the termination of the job

**6.40.3.4 \_\_starpu\_wait\_job()**

```
void __starpu_wait_job (
    struct __starpu_job * j )
```

Wait for the termination of the job

**6.40.3.5 \_\_starpu\_test\_job\_termination()**

```
int __starpu_test_job_termination (
    struct __starpu_job * j )
```

Test for the termination of the job

**6.40.3.6 \_\_starpu\_job\_prepare\_for\_continuation\_ext()**

```
void __starpu_job_prepare_for_continuation_ext (
    struct __starpu_job * j,
    unsigned continuation_resubmit,
    void(*) (void *arg) continuation_callback_on_sleep,
    void * continuation_callback_on_sleep_arg )
```

Prepare the job for accepting new dependencies before becoming a continuation.

**6.40.3.7 \_\_starpu\_exclude\_task\_from\_dag()**

```
void __starpu_exclude_task_from_dag (
    struct starpu_task * task )
```

Specify that the task should not appear in the DAG generated by debug tools.

**6.40.3.8 \_\_starpu\_enforce\_deps\_and\_schedule()**

```
unsigned __starpu_enforce_deps_and_schedule (
    struct __starpu_job * j )
```

try to submit job j, enqueue it if it's not schedulable yet. The job's sync mutex is supposed to be held already

**6.40.3.9 \_\_starpu\_reenforce\_task\_deps\_and\_schedule()**

```
unsigned __starpu_reenforce_task_deps_and_schedule (
    struct __starpu_job * j )
```

When waking up a continuation, we only enforce new task dependencies

**6.40.3.10 \_\_starpu\_handle\_job\_submission()**

```
void __starpu_handle_job_submission (
    struct __starpu_job * j )
```

Called at the submission of the job

**6.40.3.11 \_\_starpu\_handle\_job\_termination()**

```
void __starpu_handle_job_termination (
    struct __starpu_job * j )
```

This function must be called after the execution of a job, this triggers all job's dependencies and perform the callback function if any.

**6.40.3.12 \_\_starpu\_job\_get\_data\_size()**

```
size_t __starpu_job_get_data_size (
    struct starpu_perfmodel * model,
    struct starpu_perfmodel_arch * arch,
    unsigned nimpl,
    struct __starpu_job * j )
```

Get the sum of the size of the data accessed by the job.



### 6.40.3.13 `_starpupoplocaltask()`

```
struct starpu_task* _starpupoplocaltask (
    struct _starpupworker * worker )
```

Get a task from the local pool of tasks that were explicitly attributed to that worker.

### 6.40.3.14 `_starpupushlocaltask()`

```
int _starpupushlocaltask (
    struct _starpupworker * worker,
    struct starpu_task * task,
    int prio )
```

Put a task into the pool of tasks that are explicitly attributed to the specified worker. If "back" is set, the task is put at the back of the list. Considering the tasks are popped from the back, this value should be 0 to enforce a FIFO ordering.

## 6.41 malloc.h File Reference

### Functions

- void `_starpumallocinit` (unsigned dst\_node)
- void `_starpumallocshutdown` (unsigned dst\_node)
- void `_starpufreeonnode` (unsigned dst\_node, uintptr\_t addr, size\_t size)
- int `_starpumallocflagsonnode` (unsigned dst\_node, void \*\*A, size\_t dim, int flags)
- int `_starpufreeflagsonnode` (unsigned dst\_node, void \*A, size\_t dim, int flags)
- int `_starpumallocwillpinonnode` (unsigned dst\_node)

### 6.41.1 Function Documentation

#### 6.41.1.1 `_starpumallocwillpinonnode()`

```
int _starpumallocwillpinonnode (
    unsigned dst_node )
```

Returns whether when allocating data on `dst_node`, we will do pinning, i.e. the allocation will be very expensive, and should thus be moved out from the critical path

## 6.42 memalloc.h File Reference

```
#include <starpup.h>
#include <common/config.h>
#include <common/list.h>
#include <datawizzard/interfaces/data_interface.h>
#include <datawizzard/coherency.h>
#include <datawizzard/copy_driver.h>
#include <datawizzard/data_request.h>
```

## 6.43 memory\_manager.h File Reference

```
#include <starpup.h>
```

## Functions

- [int \\_starpup\\_memory\\_manager\\_init\(\)](#)
- [void \\_starpup\\_memory\\_manager\\_set\\_global\\_memory\\_size](#) (unsigned node, size\_t size)
- [size\\_t \\_starpup\\_memory\\_manager\\_get\\_global\\_memory\\_size](#) (unsigned node)
- [int \\_starpup\\_memory\\_manager\\_test\\_allocate\\_size](#) (unsigned node, size\_t size)

### 6.43.1 Function Documentation

#### 6.43.1.1 \_starpup\_memory\_manager\_init()

`int _starpup_memory_manager_init ( )`  
Initialises the memory manager

#### 6.43.1.2 \_starpup\_memory\_manager\_set\_global\_memory\_size()

`void _starpup_memory_manager_set_global_memory_size (`  
    unsigned *node*,  
    size\_t *size* )

Initialises the global memory size for the given node

#### 6.43.1.3 \_starpup\_memory\_manager\_get\_global\_memory\_size()

`size_t _starpup_memory_manager_get_global_memory_size (`  
    unsigned *node* )

Gets the global memory size for the given node

## 6.44 memory\_nodes.h File Reference

```
#include <starpup.h>
#include <common/config.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
#include <datawizard/node_ops.h>
#include <common/utills.h>
#include <core/workers.h>
#include <core/simgrid.h>
```

## Data Structures

- [struct \\_starpup\\_cond\\_and\\_worker](#)
- [struct \\_starpup\\_memory\\_node\\_descr](#)

## Macros

- `#define starpup_node_get_kind`
- `#define starpup_memory_nodes_get_count`
- `#define starpup_worker_get_memory_node`
- `#define starpup_worker_get_local_memory_node`

## Functions

- void `_starpu_memory_nodes_init` (void)
- void `_starpu_memory_nodes_deinit` (void)
- static void `_starpu_memory_node_add_nworkers` (unsigned node)
- void `_starpu_worker_drives_memory_node` (struct `_starpu_worker` \*worker, unsigned memnode)
- static struct `_starpu_node_ops` \* `_starpu_memory_node_get_node_ops` (unsigned node)
- static unsigned `_starpu_memory_node_get_nworkers` (unsigned node)
- static void `_starpu_simgrid_memory_node_set_host` (unsigned node, starpu\_sg\_host\_t host)
- static starpu\_sg\_host\_t `_starpu_simgrid_memory_node_get_host` (unsigned node)
- unsigned `_starpu_memory_node_register` (enum starpu\_node\_kind kind, int devid, struct `_starpu_node_ops` \*node\_ops)
- void `_starpu_memory_node_register_condition` (struct `_starpu_worker` \*worker, starpu\_pthread\_cond\_t \*cond, unsigned nodeid)
- static struct `_starpu_memory_node_descr` \* `_starpu_memory_node_get_description` (void)
- static enum starpu\_node\_kind `_starpu_node_get_kind` (unsigned node)
- static unsigned `_starpu_memory_nodes_get_count` (void)
- static unsigned `_starpu_worker_get_memory_node` (unsigned workerid)
- static unsigned `_starpu_worker_get_local_memory_node` (void)

## Variables

- char `_starpu_worker_drives_memory` [STARPU\_NMAXWORKERS][STARPU\_MAXNODES]
- struct `_starpu_memory_node_descr` `_starpu_descr`

### 6.44.1 Data Structure Documentation

#### 6.44.1.1 struct `_starpu_cond_and_worker`

##### Data Fields

starpu_pthread_cond_t *	cond	
struct <code>_starpu_worker</code> *	worker	

#### 6.44.1.2 struct `_starpu_memory_node_descr`

##### Data Fields

unsigned	nnodes	
enum starpu_node_kind	nodes[STARPU_MAXNODES]	
struct <code>_starpu_node_ops</code> *	node_ops[STARPU_MAXNODES]	
int	devid[STARPU_MAXNODES]	Get the device id associated to this node, or -1 if not applicable
unsigned	nworkers[STARPU_MAXNODES]	
starpu_sg_host_t	host[STARPU_MAXNODES]	
starpu_pthread_rwlock_t	conditions_rwlock	Every worker is associated to a condition variable on which the worker waits when there is task available. It is possible that multiple worker share the same condition variable, so we maintain a list of all these condition variables so that we can wake up all worker attached to a memory node that are waiting on a task.

## Data Fields

<a href="#">struct _starpu_cond_and_worker</a>	conditions_attached_to_node[STARPU_MAXNODES][STARPU_NMAXWORKERS]	
<a href="#">struct _starpu_cond_and_worker</a>	conditions_all[STARPU_MAXNODES * STARPU_NMAXWORKERS]	
unsigned	total_condition_count	the number of queues attached to each node
unsigned	condition_count[STARPU_MAXNODES]	

## 6.44.2 Function Documentation

## 6.44.2.1 \_starpu\_worker\_drives\_memory\_node()

```
void _starpu_worker_drives_memory_node (
    struct _starpu_worker * worker,
    unsigned memnode )
same utility as _starpu_memory_node_add_nworkers
```

## 6.44.2.2 \_starpu\_worker\_get\_memory\_node()

```
static unsigned _starpu_worker_get_memory_node (
    unsigned workerid ) [inline], [static]
This workerid may either be a basic worker or a combined worker
We have a combined worker
```

## 6.45 memstats.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

## Typedefs

- typedef void \* **\_starpu\_memory\_stats\_t**

## Functions

- void **\_starpu\_memory\_stats\_init** (starpu\_data\_handle\_t handle)
- void **\_starpu\_memory\_stats\_init\_per\_node** (starpu\_data\_handle\_t handle, unsigned node)
- void **\_starpu\_memory\_stats\_free** (starpu\_data\_handle\_t handle)
- void **\_starpu\_memory\_display\_handle\_stats** (FILE \*stream, starpu\_data\_handle\_t handle)
- void **\_starpu\_memory\_handle\_stats\_cache\_hit** (starpu\_data\_handle\_t handle, unsigned node)
- void **\_starpu\_memory\_handle\_stats\_loaded\_shared** (starpu\_data\_handle\_t handle, unsigned node)
- void **\_starpu\_memory\_handle\_stats\_loaded\_owner** (starpu\_data\_handle\_t handle, unsigned node)
- void **\_starpu\_memory\_handle\_stats\_shared\_to\_owner** (starpu\_data\_handle\_t handle, unsigned node)
- void **\_starpu\_memory\_handle\_stats\_invalidated** (starpu\_data\_handle\_t handle, unsigned node)

## 6.46 mp\_common.h File Reference

```
#include <semaphore.h>
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <common/barrier.h>
```

```
#include <common/thread.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/copy_driver.h>
```

## 6.47 multiple\_regression.h File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <core/perfmodel/perfmodel.h>
#include <starpu.h>
```

### Functions

- `int _starpu_multiple_regression` ([struct](#) starpu\_perfmodel\_history\_list \*ptr, double \*coeff, unsigned ncoeff, unsigned nparameters, const char \*\*parameters\_names, unsigned \*\*combinations, const char \*codelet\_name)

## 6.48 node\_ops.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <datawizard/copy_driver.h>
```

### Data Structures

- [struct \\_starpu\\_node\\_ops](#)

### Typedefs

- `typedef int(* copy_interface_func_t)` (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, [struct \\_starpu\\_data\\_request](#) \*req)
- `typedef int(* copy_data_t)` (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t dst\_offset, unsigned dst\_node, size\_t ssize, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `typedef int(* copy2d_data_t)` (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks, size\_t ld\_src, size\_t ld\_dst, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- `typedef int(* copy3d_data_t)` (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks\_1, size\_t ld1\_src, size\_t ld1\_dst, size\_t numblocks\_2, size\_t ld2\_src, size\_t ld2\_dst, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)

### Functions

- `const char * _starpu_node_get_prefix` (enum starpu\_node\_kind kind)

## 6.49 openmp\_runtime\_support.h File Reference

```
#include <starpu.h>
#include <common/list.h>
#include <common/starpu_spinlock.h>
#include <common/uthash.h>
#include <ucontext.h>
```

## Data Structures

- struct [starpu\\_omp\\_numeric\\_place](#)
- struct [starpu\\_omp\\_place](#)
- struct [starpu\\_omp\\_data\\_environment\\_icvs](#)
- struct [starpu\\_omp\\_device\\_icvs](#)
- struct [starpu\\_omp\\_implicit\\_task\\_icvs](#)
- struct [starpu\\_omp\\_global\\_icvs](#)
- struct [starpu\\_omp\\_initial\\_icv\\_values](#)
- struct [starpu\\_omp\\_task\\_group](#)
- struct [starpu\\_omp\\_task\\_link](#)
- struct [starpu\\_omp\\_condition](#)
- struct [starpu\\_omp\\_critical](#)

## Macros

- `#define` [\\_XOPEN\\_SOURCE](#)
- `#define` [STARPU\\_OMP\\_MAX\\_ACTIVE\\_LEVELS](#)

## Enumerations

- enum [starpu\\_omp\\_place\\_name](#) {  
  [starpu\\_omp\\_place\\_undefined](#) , [starpu\\_omp\\_place\\_threads](#) , [starpu\\_omp\\_place\\_cores](#) , [starpu\\_omp\\_place\\_sockets](#) ,  
  [starpu\\_omp\\_place\\_numerical](#) }
- enum [starpu\\_omp\\_task\\_state](#) {  
  [starpu\\_omp\\_task\\_state\\_clear](#) , [starpu\\_omp\\_task\\_state\\_preempted](#) , [starpu\\_omp\\_task\\_state\\_terminated](#) , [starpu\\_omp\\_task\\_state\\_zombie](#) ,  
  [starpu\\_omp\\_task\\_state\\_target](#) }
- enum [starpu\\_omp\\_task\\_wait\\_on](#) {  
  [starpu\\_omp\\_task\\_wait\\_on\\_task\\_childs](#) , [starpu\\_omp\\_task\\_wait\\_on\\_region\\_tasks](#) , [starpu\\_omp\\_task\\_wait\\_on\\_barrier](#) , [starpu\\_omp\\_task\\_wait\\_on\\_group](#) ,  
  [starpu\\_omp\\_task\\_wait\\_on\\_critical](#) , [starpu\\_omp\\_task\\_wait\\_on\\_ordered](#) , [starpu\\_omp\\_task\\_wait\\_on\\_lock](#) , [starpu\\_omp\\_task\\_wait\\_on\\_nest\\_lock](#) }
- enum [starpu\\_omp\\_task\\_flags](#) { [STARPU\\_OMP\\_TASK\\_FLAGS\\_IMPLICIT](#) , [STARPU\\_OMP\\_TASK\\_FLAGS\\_UNDEFERRED](#) , [STARPU\\_OMP\\_TASK\\_FLAGS\\_FINAL](#) , [STARPU\\_OMP\\_TASK\\_FLAGS\\_UNTIED](#) }

## Variables

- [starpu\\_pthread\\_key\\_t omp\\_thread\\_key](#)
- [starpu\\_pthread\\_key\\_t omp\\_task\\_key](#)

### 6.49.1 Data Structure Documentation

#### 6.49.1.1 struct [starpu\\_omp\\_numeric\\_place](#)

##### Data Fields

int	<a href="#">excluded_place</a>	
int *	<a href="#">included_numeric_items</a>	
int	<a href="#">nb_included_numeric_items</a>	
int *	<a href="#">excluded_numeric_items</a>	
int	<a href="#">nb_excluded_numeric_items</a>	

### 6.49.1.2 struct starpu\_omp\_place

OpenMP place for thread affinity, defined by the OpenMP spec

#### Data Fields

	int	abstract_name	
	int	abstract_excluded	
	int	abstract_length	
	<a href="#">struct starpu_omp_numeric_place *</a>	numeric_places	
	int	nb_numeric_places	

### 6.49.1.3 struct starpu\_omp\_data\_environment\_icvs

Internal Control Variables (ICVs) declared following OpenMP 4.0.0 spec section 2.3.1

#### Data Fields

	int	dyn_var	parallel region icvs
	int	nest_var	
	int *	nthreads_var	
	int	thread_limit_var	nthreads_var ICV is a list
	int	active_levels_var	
	int	levels_var	
	int *	bind_var	
	int	run_sched_var	bind_var ICV is a list loop region icvs
	unsigned long long	run_sched_chunk_var	
	int	default_device_var	program execution icvs
	int	max_task_priority_var	

### 6.49.1.4 struct starpu\_omp\_device\_icvs

#### Data Fields

	int	max_active_levels_var	parallel region icvs
	int	def_sched_var	loop region icvs
	unsigned long long	def_sched_chunk_var	
	int	stacksize_var	program execution icvs
	int	wait_policy_var	

### 6.49.1.5 struct starpu\_omp\_implicit\_task\_icvs

#### Data Fields

int	place_partition_var	parallel region icvs
-----	---------------------	----------------------

### 6.49.1.6 struct starpu\_omp\_global\_icvs

#### Data Fields

int	cancel_var	program execution icvs
-----	------------	------------------------

**6.49.1.7 struct starpu\_omp\_initial\_icv\_values**

## Data Fields

int	dyn_var	
int	nest_var	
int *	nthreads_var	
int	run_sched_var	
unsigned long long	run_sched_chunk_var	
int	def_sched_var	
unsigned long long	def_sched_chunk_var	
int *	bind_var	
int	stacksize_var	
int	wait_policy_var	
int	thread_limit_var	
int	max_active_levels_var	
int	active_levels_var	
int	levels_var	
int	place_partition_var	
int	cancel_var	
int	default_device_var	
int	max_task_priority_var	
<a href="#">struct starpu_omp_place</a>	places	not a real ICV, but needed to store the contents of OMP_PLACES

**6.49.1.8 struct starpu\_omp\_task\_group**

## Data Fields

int	descendent_task_count	
<a href="#">struct starpu_omp_task *</a>	leader_task	
<a href="#">struct starpu_omp_task_group *</a>	p_previous_task_group	

**6.49.1.9 struct starpu\_omp\_task\_link**

## Data Fields

<a href="#">struct starpu_omp_task *</a>	task	
<a href="#">struct starpu_omp_task_link *</a>	next	

**6.49.1.10 struct starpu\_omp\_condition**

## Data Fields

<a href="#">struct starpu_omp_task_link *</a>	contention_list_head	
---	----------------------	--

**6.49.1.11 struct starpu\_omp\_critical**

## Data Fields

<a href="#">UT_hash_handle</a>	hh	
--------------------------------	----	--



## Data Fields

<a href="#">struct _starpu_spinlock</a>	lock	
unsigned	state	
<a href="#">struct starpu_omp_task_link *</a>	contention_list_head	
const char *	name	

## 6.49.2 Macro Definition Documentation

### 6.49.2.1 \_XOPEN\_SOURCE

```
#define _XOPEN_SOURCE
```

ucontexts have been deprecated as of POSIX 1-2004 `_XOPEN_SOURCE` required at least on OS/X  
 TODO: add detection in `configure.ac`

### 6.49.2.2 STARPU\_OMP\_MAX\_ACTIVE\_LEVELS

```
#define STARPU_OMP_MAX_ACTIVE_LEVELS
```

Arbitrary limit on the number of nested parallel sections

## 6.49.3 Enumeration Type Documentation

### 6.49.3.1 starpu\_omp\_place\_name

```
enum starpu_omp_place_name
```

Possible abstract names for OpenMP places

### 6.49.3.2 starpu\_omp\_task\_state

```
enum starpu_omp_task_state
```

## Enumerator

<code>starpu_omp_task_state_target</code>	target tasks are non-preemptible tasks, without dedicated stack and OpenMP Runtime Support context
---	--

## 6.50 perfmodel.h File Reference

```
#include <common/config.h>
#include <starpu.h>
#include <core/task_bundle.h>
#include <stdio.h>
```

## Data Structures

- [struct \\_starpu\\_perfmodel\\_state](#)

## Macros

- `#define _STARPU_PERFMODEL_VERSION`

## Functions

- char \* [\\_starpu\\_get\\_perf\\_model\\_dir\\_codelet](#) ()
- char \* [\\_starpu\\_get\\_perf\\_model\\_dir\\_bus](#) ()
- char \* [\\_starpu\\_get\\_perf\\_model\\_dir\\_debug](#) ()
- double [\\_starpu\\_history\\_based\\_job\\_expected\\_perf](#) ([struct](#) starpu\_perfmodel \*model, [struct](#) starpu\_perfm←  
model\_arch \*arch, [struct](#) [starpu\\_job](#) \*j, unsigned nimpl)
- void [\\_starpu\\_load\\_history\\_based\\_model](#) ([struct](#) starpu\_perfmodel \*model, unsigned scan\_history)
- void [\\_starpu\\_init\\_and\\_load\\_perfmodel](#) ([struct](#) starpu\_perfmodel \*model)
- void [\\_starpu\\_initialize\\_registered\\_performance\\_models](#) (void)
- void [\\_starpu\\_deinitialize\\_registered\\_performance\\_models](#) (void)
- void [\\_starpu\\_deinitialize\\_performance\\_model](#) ([struct](#) starpu\_perfmodel \*model)
- double [\\_starpu\\_regression\\_based\\_job\\_expected\\_perf](#) ([struct](#) starpu\_perfmodel \*model, [struct](#) starpu\_per←  
fm\_model\_arch \*arch, [struct](#) [starpu\\_job](#) \*j, unsigned nimpl)
- double [\\_starpu\\_non\\_linear\\_regression\\_based\\_job\\_expected\\_perf](#) ([struct](#) starpu\_perfmodel \*model, [struct](#) starpu\_perfm←  
model\_arch \*arch, [struct](#) [starpu\\_job](#) \*j, unsigned nimpl)
- double [\\_starpu\\_multiple\\_regression\\_based\\_job\\_expected\\_perf](#) ([struct](#) starpu\_perfmodel \*model, [struct](#) starpu\_perfm←  
model\_arch \*arch, [struct](#) [starpu\\_job](#) \*j, unsigned nimpl)
- void [\\_starpu\\_update\\_perfmodel\\_history](#) ([struct](#) [starpu\\_job](#) \*j, [struct](#) starpu\_perfmodel \*model, [struct](#) starpu\_perfm←  
model\_arch \*arch, unsigned cpuid, double measured, unsigned nimpl)
- int [\\_starpu\\_perfmodel\\_create\\_comb\\_if\\_needed](#) ([struct](#) starpu\_perfmodel\_arch \*arch)
- void [\\_starpu\\_create\\_sampling\\_directory\\_if\\_needed](#) (void)
- void [\\_starpu\\_load\\_bus\\_performance\\_files](#) (void)
- void [\\_starpu\\_set\\_calibrate\\_flag](#) (unsigned val)
- unsigned [\\_starpu\\_get\\_calibrate\\_flag](#) (void)
- unsigned \* [\\_starpu\\_get\\_cuda\\_affinity\\_vector](#) (unsigned gpuid)
- unsigned \* [\\_starpu\\_get\\_opencil\\_affinity\\_vector](#) (unsigned gpuid)
- void [\\_starpu\\_save\\_bandwidth\\_and\\_latency\\_disk](#) (double bandwidth\_write, double bandwidth\_read, dou-  
ble latency\_write, double latency\_read, unsigned node, const char \*name)
- void [\\_starpu\\_write\\_double](#) (FILE \*f, const char \*format, double val)
- int [\\_starpu\\_read\\_double](#) (FILE \*f, char \*format, double \*val)
- void [\\_starpu\\_simgrid\\_get\\_platform\\_path](#) (int version, char \*path, size\_t maxlen)
- void [\\_starpu\\_perfmodel\\_realloc](#) ([struct](#) starpu\_perfmodel \*model, int nb)
- void [\\_starpu\\_free\\_arch\\_combs](#) (void)
- hwloc\_topology\_t [\\_starpu\\_perfmodel\\_get\\_hwtopology](#) ()

## Variables

- unsigned [\\_starpu\\_calibration\\_minimum](#)

### 6.50.1 Data Structure Documentation

#### 6.50.1.1 [struct](#) [starpu\\_perfmodel\\_state](#)

##### Data Fields

<a href="#">struct</a> starpu_perfmodel_per_arch **	per_arch	
int **	per_arch_is_set	
starpu_pthread_rwlock_t	model_rwlock	
int *	nimpls	
int *	nimpls_set	
int	ncombs	The number of combinations currently used by the model
int	ncombs_set	The number of combinations allocated in the array nimpls and ncombs
int *	combs	

## 6.50.2 Macro Definition Documentation

### 6.50.2.1 `_STARPU_PERFMODEL_VERSION`

```
#define _STARPU_PERFMODEL_VERSION
```

Performance models files are stored in a directory whose name include the version of the performance model format. The version number is also written in the file itself. When updating the format, the variable `_STARPU_PERFMODEL_VERSION` should be updated. It is then possible to switch easily between different versions of StarPU having different performance model formats.

## 6.51 `prio_deque.h` File Reference

```
#include <starpu.h>
#include <starpu_scheduler.h>
#include <core/task.h>
```

### Data Structures

- [struct `\_starpu\_prio\_deque`](#)

### Functions

- static void `_starpu_prio_deque_init` ([struct `\_starpu\_prio\_deque`](#) \*pdeque)
- static void `_starpu_prio_deque_destroy` ([struct `\_starpu\_prio\_deque`](#) \*pdeque)
- static int `_starpu_prio_deque_is_empty` ([struct `\_starpu\_prio\_deque`](#) \*pdeque)
- static void `_starpu_prio_deque_erase` ([struct `\_starpu\_prio\_deque`](#) \*pdeque, [struct `starpu\_task`](#) \*task)
- static int `_starpu_prio_deque_push_front_task` ([struct `\_starpu\_prio\_deque`](#) \*pdeque, [struct `starpu\_task`](#) \*task)
- static int `_starpu_prio_deque_push_back_task` ([struct `\_starpu\_prio\_deque`](#) \*pdeque, [struct `starpu\_task`](#) \*task)
- static [struct `starpu\_task`](#) \* `_starpu_prio_deque_highest_task` ([struct `\_starpu\_prio\_deque`](#) \*pdeque)
- static [struct `starpu\_task`](#) \* `_starpu_prio_deque_pop_task` ([struct `\_starpu\_prio\_deque`](#) \*pdeque)
- static [struct `starpu\_task`](#) \* `_starpu_prio_deque_pop_back_task` ([struct `\_starpu\_prio\_deque`](#) \*pdeque)
- static int `_starpu_prio_deque_pop_this_task` ([struct `\_starpu\_prio\_deque`](#) \*pdeque, int workerid, [struct `starpu\_task`](#) \*task)
- [struct `starpu\_task`](#) \* `_starpu_prio_deque_pop_task_for_worker` ([struct `\_starpu\_prio\_deque`](#) \*, int workerid, int \*skipped)
- [struct `starpu\_task`](#) \* `_starpu_prio_deque_dequeue_task_for_worker` ([struct `\_starpu\_prio\_deque`](#) \*, int workerid, int \*skipped)
- [struct `starpu\_task`](#) \* `_starpu_prio_deque_dequeue_first_ready_task` ([struct `\_starpu\_prio\_deque`](#) \*, unsigned workerid)

### 6.51.1 Data Structure Documentation

#### 6.51.1.1 `struct _starpu_prio_deque`

##### Data Fields

<a href="#">struct <code>starpu_task_prio_list</code></a>	list	
unsigned	ntasks	
unsigned	nprocessed	
double	exp_start	
double	exp_end	
double	exp_len	

## 6.51.2 Function Documentation

### 6.51.2.1 \_\_starpu\_prio\_deque\_is\_empty()

```
static int __starpu_prio_deque_is_empty (
    struct __starpu_prio_deque * pdeque ) [inline], [static]
```

return 0 iff the struct `__starpu_prio_deque` is not empty

### 6.51.2.2 \_\_starpu\_prio\_deque\_push\_front\_task()

```
static int __starpu_prio_deque_push_front_task (
    struct __starpu_prio_deque * pdeque,
    struct starpu_task * task ) [inline], [static]
```

push a task in  $O(\lg(\text{nb priorities}))$

### 6.51.2.3 \_\_starpu\_prio\_deque\_pop\_task()

```
static struct starpu_task* __starpu_prio_deque_pop_task (
    struct __starpu_prio_deque * pdeque ) [inline], [static]
```

all `__starpu_prio_deque_pop/dequeue_task` function return a task or a NULL pointer if none are available in  $O(\lg(\text{nb priorities}))$

### 6.51.2.4 \_\_starpu\_prio\_deque\_pop\_task\_for\_worker()

```
struct starpu_task* __starpu_prio_deque_pop_task_for_worker (
    struct __starpu_prio_deque * ,
    int workerid,
    int * skipped )
```

return a task that can be executed by workerid

### 6.51.2.5 \_\_starpu\_prio\_deque\_dequeue\_task\_for\_worker()

```
struct starpu_task* __starpu_prio_dequeue_dequeue_task_for_worker (
    struct __starpu_prio_deque * ,
    int workerid,
    int * skipped )
```

return a task that can be executed by workerid

## 6.52 prio\_list.h File Reference

```
#include <common/rbtree.h>
```

### Macros

- `#define PRIO_LIST_INLINE`
- `#define PRIO_struct`
- `#define PRIO_LIST_CREATE_TYPE(ENAME, PRIOFIELD)`

## 6.53 profiling.h File Reference

```
#include <starpu.h>
#include <starpu_profiling.h>
#include <starpu_util.h>
#include <common/config.h>
```

## Functions

- `struct starpu_profiling_task_info * _starpu_allocate_profiling_info_if_needed (struct starpu_task *task)`
- `void _starpu_worker_update_profiling_info_executing (int workerid, struct timespec *executing_time, int executed_tasks, uint64_t used_cycles, uint64_t stall_cycles, double consumed_energy, double flops)`
- `void _starpu_worker_restart_sleeping (int workerid)`
- `void _starpu_worker_stop_sleeping (int workerid)`
- `void _starpu_worker_register_executing_start_date (int workerid, struct timespec *executing_start)`
- `void _starpu_worker_register_executing_end (int workerid)`
- `void _starpu_initialize_busid_matrix (void)`
- `int _starpu_register_bus (int src_node, int dst_node)`
- `void _starpu_bus_update_profiling_info (int src_node, int dst_node, size_t size)`
- `void _starpu_profiling_set_task_push_start_time (struct starpu_task *task)`
- `void _starpu_profiling_set_task_push_end_time (struct starpu_task *task)`
- `void _starpu_profiling_init (void)`
- `void _starpu_profiling_start (void)`
- `void _starpu_profiling_terminate (void)`

### 6.53.1 Function Documentation

#### 6.53.1.1 \_starpu\_allocate\_profiling\_info\_if\_needed()

```
struct starpu_profiling_task_info* _starpu_allocate_profiling_info_if_needed (
    struct starpu_task * task )
```

Create a task profiling info structure (with the proper time stamps) in case profiling is enabled.

#### 6.53.1.2 \_starpu\_worker\_update\_profiling\_info\_executing()

```
void _starpu_worker_update_profiling_info_executing (
    int workerid,
    struct timespec * executing_time,
    int executed_tasks,
    uint64_t used_cycles,
    uint64_t stall_cycles,
    double consumed_energy,
    double flops )
```

Update the per-worker profiling info after a task (or more) was executed. This tells StarPU how much time was spent doing computation.

#### 6.53.1.3 \_starpu\_worker\_restart\_sleeping()

```
void _starpu_worker_restart_sleeping (
    int workerid )
```

Record the date when the worker started to sleep. This permits to measure how much time was spent sleeping.

#### 6.53.1.4 \_starpu\_worker\_stop\_sleeping()

```
void _starpu_worker_stop_sleeping (
    int workerid )
```

Record the date when the worker stopped sleeping. This permits to measure how much time was spent sleeping.

#### 6.53.1.5 \_starpu\_worker\_register\_executing\_start\_date()

```
void _starpu_worker_register_executing_start_date (
    int workerid,
    struct timespec * executing_start )
```

Record the date when the worker started to execute a piece of code. This permits to measure how much time was really spent doing computation at the end of the codelet.

**6.53.1.6 \_\_starpu\_worker\_register\_executing\_end()**

```
void __starpu_worker_register_executing_end (
    int workerid )
```

Record that the worker is not executing any more.

**6.53.1.7 \_\_starpu\_initialize\_busid\_matrix()**

```
void __starpu_initialize_busid_matrix (
    void )
```

When StarPU is initialized, a matrix describing all the bus between memory nodes is created: it indicates whether there is a physical link between two memory nodes or not. This matrix should contain the identifier of the bus between two nodes or -1 in case there is no link.

**6.53.1.8 \_\_starpu\_register\_bus()**

```
int __starpu_register_bus (
    int src_node,
    int dst_node )
```

Tell StarPU that there exists a link between the two memory nodes. This function returns the identifier associated to the bus which can be used to retrieve profiling information about the bus activity later on.

**6.53.1.9 \_\_starpu\_bus\_update\_profiling\_info()**

```
void __starpu_bus_update_profiling_info (
    int src_node,
    int dst_node,
    size_t size )
```

Tell StarPU that "size" bytes were transferred between the two specified memory nodes.

**6.53.1.10 \_\_starpu\_profiling\_init()**

```
void __starpu_profiling_init (
    void )
```

This function needs to be called before other starpu\_profile\_\* functions

**6.53.1.11 \_\_starpu\_profiling\_start()**

```
void __starpu_profiling_start (
    void )
```

This function starts profiling if the STARPU\_PROFILING environment variable was set

**6.54 progress\_hook.h File Reference****Functions**

- void **\_\_starpu\_init\_progression\_hooks** (void)
- unsigned **\_\_starpu\_execute\_registered\_progression\_hooks** (void)

**6.55 rbtree.h File Reference**

```
#include <stddef.h>
#include <assert.h>
#include <stdint.h>
#include <sys/types.h>
#include "rbtree_i.h"
```

## Macros

- `#define MACRO_BEGIN`
- `#define MACRO_END`
- `#define STARPU_RBTREE_LEFT`
- `#define STARPU_RBTREE_RIGHT`
- `#define STARPU_RBTREE_INITIALIZER`
- `#define starpu_rbtree_entry(node, type, member)`
- `#define starpu_rbtree_lookup(tree, key, cmp_fn)`
- `#define starpu_rbtree_lookup_nearest(tree, key, cmp_fn, dir)`
- `#define starpu_rbtree_insert(tree, node, cmp_fn)`
- `#define starpu_rbtree_lookup_slot(tree, key, cmp_fn, slot)`
- `#define starpu_rbtree_first(tree)`
- `#define starpu_rbtree_last(tree)`
- `#define starpu_rbtree_prev(node)`
- `#define starpu_rbtree_next(node)`
- `#define starpu_rbtree_for_each_remove(tree, node, tmp)`

## Functions

- static void `starpu_rbtree_init` (`struct starpu_rbtree *tree`)
- static void `starpu_rbtree_node_init` (`struct starpu_rbtree_node *node`)
- static int `starpu_rbtree_node_unlinked` (`const struct starpu_rbtree_node *node`)
- static int `starpu_rbtree_empty` (`const struct starpu_rbtree *tree`)
- static void `starpu_rbtree_insert_slot` (`struct starpu_rbtree *tree`, `uintptr_t slot`, `struct starpu_rbtree_node *node`)
- void `starpu_rbtree_remove` (`struct starpu_rbtree *tree`, `struct starpu_rbtree_node *node`)

## 6.55.1 Macro Definition Documentation

### 6.55.1.1 STARPU\_RBTREE\_INITIALIZER

```
#define STARPU_RBTREE_INITIALIZER
```

Static tree initializer.

### 6.55.1.2 starpu\_rbtree\_entry

```
#define starpu_rbtree_entry(  
    node,  
    type,  
    member )
```

Macro that evaluates to the address of the structure containing the given node based on the given type and member.

### 6.55.1.3 starpu\_rbtree\_lookup

```
#define starpu_rbtree_lookup(  
    tree,  
    key,  
    cmp_fn )
```

Look up a node in a tree.

Note that implementing the lookup algorithm as a macro gives two benefits: First, it avoids the overhead of a callback function. Next, the type of the `cmp_fn` parameter isn't rigid. The only guarantee offered by this implementation is that the key parameter is the first parameter given to `cmp_fn`. This way, users can pass only the value they need for comparison instead of e.g. allocating a full structure on the stack.

See `starpu_rbtree_insert()`.

#### 6.55.1.4 `starpu_rbtrees_lookup_nearest`

```
#define starpu_rbtrees_lookup_nearest(  
    tree,  
    key,  
    cmp_fn,  
    dir )
```

Look up a node or one of its nearest nodes in a tree.

This macro essentially acts as [starpu\\_rbtrees\\_lookup\(\)](#) but if no entry matched the key, an additional step is performed to obtain the next or previous node, depending on the direction (left or right).

The constraints that apply to the key parameter are the same as for [starpu\\_rbtrees\\_lookup\(\)](#).

#### 6.55.1.5 `starpu_rbtrees_insert`

```
#define starpu_rbtrees_insert(  
    tree,  
    node,  
    cmp_fn )
```

Insert a node in a tree.

This macro performs a standard lookup to obtain the insertion point of the given node in the tree (it is assumed that the inserted node never compares equal to any other entry in the tree) and links the node. It then checks red-black rules violations, and rebalances the tree if necessary.

Unlike [starpu\\_rbtrees\\_lookup\(\)](#), the `cmp_fn` parameter must compare two complete entries, so it is suggested to use two different comparison inline functions, such as [myobj\\_cmp\\_lookup\(\)](#) and [myobj\\_cmp\\_insert\(\)](#). There is no guarantee about the order of the nodes given to the comparison function.

See [starpu\\_rbtrees\\_lookup\(\)](#).

#### 6.55.1.6 `starpu_rbtrees_lookup_slot`

```
#define starpu_rbtrees_lookup_slot(  
    tree,  
    key,  
    cmp_fn,  
    slot )
```

Look up a node/slot pair in a tree.

This macro essentially acts as [starpu\\_rbtrees\\_lookup\(\)](#) but in addition to a node, it also returns a slot, which identifies an insertion point in the tree. If the returned node is null, the slot can be used by [starpu\\_rbtrees\\_insert\\_slot\(\)](#) to insert without the overhead of an additional lookup. The slot is a simple `uintptr_t` integer.

The constraints that apply to the key parameter are the same as for [starpu\\_rbtrees\\_lookup\(\)](#).

#### 6.55.1.7 `starpu_rbtrees_first`

```
#define starpu_rbtrees_first(  
    tree )
```

Return the first node of a tree.

#### 6.55.1.8 `starpu_rbtrees_last`

```
#define starpu_rbtrees_last(  
    tree )
```

Return the last node of a tree.

#### 6.55.1.9 `starpu_rbtrees_prev`

```
#define starpu_rbtrees_prev(  
    node )
```

Return the node previous to the given node.



### 6.55.1.10 `starpu_rbtrees_next`

```
#define starpu_rbtrees_next (
    node )
```

Return the node next to the given node.

### 6.55.1.11 `starpu_rbtrees_for_each_remove`

```
#define starpu_rbtrees_for_each_remove (
    tree,
    node,
    tmp )
```

Forge a loop to process all nodes of a tree, removing them when visited.

This macro can only be used to destroy a tree, so that the resources used by the entries can be released by the user. It basically removes all nodes without doing any color checking.

After completion, all nodes and the tree root member are stale.

## 6.55.2 Function Documentation

### 6.55.2.1 `starpu_rbtrees_init()`

```
static void starpu_rbtrees_init (
    struct starpu_rbtrees * tree ) [inline], [static]
```

Initialize a tree.

### 6.55.2.2 `starpu_rbtrees_node_init()`

```
static void starpu_rbtrees_node_init (
    struct starpu_rbtrees_node * node ) [inline], [static]
```

Initialize a node.

A node is in no tree when its parent points to itself.

### 6.55.2.3 `starpu_rbtrees_empty()`

```
static int starpu_rbtrees_empty (
    const struct starpu_rbtrees * tree ) [inline], [static]
```

Return true if tree is empty.

### 6.55.2.4 `starpu_rbtrees_insert_slot()`

```
static void starpu_rbtrees_insert_slot (
    struct starpu_rbtrees * tree,
    uintptr_t slot,
    struct starpu_rbtrees_node * node ) [inline], [static]
```

Insert a node at an insertion point in a tree.

This macro essentially acts as `starpu_rbtrees_insert()` except that it doesn't obtain the insertion point with a standard lookup. The insertion point is obtained by calling `starpu_rbtrees_lookup_slot()`. In addition, the new node must not compare equal to an existing node in the tree (i.e. the slot must denote a null node).

### 6.55.2.5 `starpu_rbtrees_remove()`

```
void starpu_rbtrees_remove (
    struct starpu_rbtrees * tree,
    struct starpu_rbtrees_node * node )
```

Remove a node from a tree.

After completion, the node is stale.

## 6.56 rbtree\_i.h File Reference

```
#include <assert.h>
```

### Data Structures

- struct [starpu\\_rbtree\\_node](#)
- struct [starpu\\_rbtree](#)

### Macros

- #define [STARPU\\_RBTREE\\_COLOR\\_MASK](#)
- #define [STARPU\\_RBTREE\\_PARENT\\_MASK](#)
- #define [STARPU\\_RBTREE\\_COLOR\\_RED](#)
- #define [STARPU\\_RBTREE\\_COLOR\\_BLACK](#)
- #define [STARPU\\_RBTREE\\_SLOT\\_INDEX\\_MASK](#)
- #define [STARPU\\_RBTREE\\_SLOT\\_PARENT\\_MASK](#)

### Functions

- static int [starpu\\_rbtree\\_check\\_alignment](#) (const struct [starpu\\_rbtree\\_node](#) \*node)
- static int [starpu\\_rbtree\\_check\\_index](#) (int index)
- static int [starpu\\_rbtree\\_d2i](#) (int diff)
- static struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_parent](#) (const struct [starpu\\_rbtree\\_node](#) \*node)
- static uintptr\_t [starpu\\_rbtree\\_slot](#) (struct [starpu\\_rbtree\\_node](#) \*parent, int index)
- static struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_slot\\_parent](#) (uintptr\_t slot)
- static int [starpu\\_rbtree\\_slot\\_index](#) (uintptr\_t slot)
- void [starpu\\_rbtree\\_insert\\_rebalance](#) (struct [starpu\\_rbtree](#) \*tree, struct [starpu\\_rbtree\\_node](#) \*parent, int index, struct [starpu\\_rbtree\\_node](#) \*node)
- struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_nearest](#) (struct [starpu\\_rbtree\\_node](#) \*parent, int index, int direction)
- struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_firstlast](#) (const struct [starpu\\_rbtree](#) \*tree, int direction)
- struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_walk](#) (struct [starpu\\_rbtree\\_node](#) \*node, int direction)
- struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_postwalk\\_deepest](#) (const struct [starpu\\_rbtree](#) \*tree)
- struct [starpu\\_rbtree\\_node](#) \* [starpu\\_rbtree\\_postwalk\\_unlink](#) (struct [starpu\\_rbtree\\_node](#) \*node)

### 6.56.1 Data Structure Documentation

#### 6.56.1.1 struct starpu\_rbtree\_node

Red-black node structure.

To reduce the number of branches and the instruction cache footprint, the left and right child pointers are stored in an array, and the symmetry of most tree operations is exploited by using left/right variables when referring to children.

In addition, this implementation assumes that all nodes are 4-byte aligned, so that the least significant bit of the parent member can be used to store the color of the node. This is true for all modern 32 and 64 bits architectures, as long as the nodes aren't embedded in structures with special alignment constraints such as member packing.

#### Data Fields

uintptr_t	parent	
struct <a href="#">starpu_rbtree_node</a> *	children[2]	

### 6.56.1.2 struct starpu\_rbtrees

Red-black tree structure.

Data Fields

<code>struct starpu_rbtrees_node *</code>	<code>root</code>	
---	-------------------	--

## 6.56.2 Macro Definition Documentation

### 6.56.2.1 STARPU\_RBTREE\_COLOR\_MASK

```
#define STARPU_RBTREE_COLOR_MASK
```

Masks applied on the parent member of a node to obtain either the color or the parent address.

### 6.56.2.2 STARPU\_RBTREE\_COLOR\_RED

```
#define STARPU_RBTREE_COLOR_RED
```

Node colors.

### 6.56.2.3 STARPU\_RBTREE\_SLOT\_INDEX\_MASK

```
#define STARPU_RBTREE_SLOT_INDEX_MASK
```

Masks applied on slots to obtain either the child index or the parent address.

## 6.56.3 Function Documentation

### 6.56.3.1 starpu\_rbtrees\_check\_alignment()

```
static int starpu_rbtrees_check_alignment (
    const struct starpu_rbtrees_node * node ) [inline], [static]
```

Return true if the given pointer is suitably aligned.

### 6.56.3.2 starpu\_rbtrees\_check\_index()

```
static int starpu_rbtrees_check_index (
    int index ) [inline], [static]
```

Return true if the given index is a valid child index.

### 6.56.3.3 starpu\_rbtrees\_d2i()

```
static int starpu_rbtrees_d2i (
    int diff ) [inline], [static]
```

Convert the result of a comparison into an index in the children array (0 or 1).

This function is mostly used when looking up a node.

### 6.56.3.4 starpu\_rbtrees\_parent()

```
static struct starpu_rbtrees_node* starpu_rbtrees_parent (
    const struct starpu_rbtrees_node * node ) [inline], [static]
```

Return the parent of a node.

**6.56.3.5 starpu\_rbtree\_slot()**

```
static uintptr_t starpu_rbtree_slot (
    struct starpu_rbtree_node * parent,
    int index ) [inline], [static]
```

Translate an insertion point into a slot.

**6.56.3.6 starpu\_rbtree\_slot\_parent()**

```
static struct starpu_rbtree_node* starpu_rbtree_slot_parent (
    uintptr_t slot ) [inline], [static]
```

Extract the parent address from a slot.

**6.56.3.7 starpu\_rbtree\_slot\_index()**

```
static int starpu_rbtree_slot_index (
    uintptr_t slot ) [inline], [static]
```

Extract the index from a slot.

**6.56.3.8 starpu\_rbtree\_insert\_rebalance()**

```
void starpu_rbtree_insert_rebalance (
    struct starpu_rbtree * tree,
    struct starpu_rbtree_node * parent,
    int index,
    struct starpu_rbtree_node * node )
```

Insert a node in a tree, rebalancing it if necessary.

The index parameter is the index in the children array of the parent where the new node is to be inserted. It is ignored if the parent is null.

This function is intended to be used by the [starpu\\_rbtree\\_insert\(\)](#) macro only.

**6.56.3.9 starpu\_rbtree\_nearest()**

```
struct starpu_rbtree_node* starpu_rbtree_nearest (
    struct starpu_rbtree_node * parent,
    int index,
    int direction )
```

Return the previous or next node relative to a location in a tree.

The parent and index parameters define the location, which can be empty. The direction parameter is either STARPU\_RBTREE\_LEFT (to obtain the previous node) or STARPU\_RBTREE\_RIGHT (to obtain the next one).

**6.56.3.10 starpu\_rbtree\_firstlast()**

```
struct starpu_rbtree_node* starpu_rbtree_firstlast (
    const struct starpu_rbtree * tree,
    int direction )
```

Return the first or last node of a tree.

The direction parameter is either STARPU\_RBTREE\_LEFT (to obtain the first node) or STARPU\_RBTREE\_RIGHT (to obtain the last one).

**6.56.3.11 starpu\_rbtree\_walk()**

```
struct starpu_rbtree_node* starpu_rbtree_walk (
    struct starpu_rbtree_node * node,
    int direction )
```

Return the node next to, or previous to the given node.

The direction parameter is either STARPU\_RBTREE\_LEFT (to obtain the previous node) or STARPU\_RBTREE\_RIGHT (to obtain the next one).

### 6.56.3.12 `starpu_rbtrees_postwalk_deepest()`

```
struct starpu_rbtrees_node* starpu_rbtrees_postwalk_deepest (
    const struct starpu_rbtrees * tree )
```

Return the left-most deepest node of a tree, which is the starting point of the postorder traversal performed by `starpu_rbtrees_for_each_remove()`.

### 6.56.3.13 `starpu_rbtrees_postwalk_unlink()`

```
struct starpu_rbtrees_node* starpu_rbtrees_postwalk_unlink (
    struct starpu_rbtrees_node * node )
```

Unlink a node from its tree and return the next (right) node in postorder.

## 6.57 `regression.h` File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <core/perfmodel/perfmodel.h>
#include <starpu.h>
```

### Functions

- `int _starpu_regression_non_linear_power` (`struct starpu_perfmodel_history_list *ptr`, `double *a`, `double *b`, `double *c`)

## 6.58 `rwlock.h` File Reference

```
#include <stdint.h>
#include <starpu.h>
```

### Data Structures

- `struct _starpu_rw_lock`

### Functions

- `void _starpu_init_rw_lock` (`struct _starpu_rw_lock *lock`)
- `void _starpu_take_rw_lock_write` (`struct _starpu_rw_lock *lock`)
- `void _starpu_take_rw_lock_read` (`struct _starpu_rw_lock *lock`)
- `int _starpu_take_rw_lock_write_try` (`struct _starpu_rw_lock *lock`)
- `int _starpu_take_rw_lock_read_try` (`struct _starpu_rw_lock *lock`)
- `void _starpu_release_rw_lock` (`struct _starpu_rw_lock *lock`)

### 6.58.1 Data Structure Documentation

#### 6.58.1.1 `struct _starpu_rw_lock`

Dummy implementation of a RW-lock using a spinlock.

#### Data Fields

<code>uint32_t</code>	<code>busy</code>	
<code>uint8_t</code>	<code>writer</code>	
<code>uint16_t</code>	<code>readercnt</code>	

## 6.58.2 Function Documentation

### 6.58.2.1 \_\_starpu\_init\_rw\_lock()

```
void __starpu_init_rw_lock (
    struct __starpu_rw_lock * lock )
```

Initialize the RW-lock

### 6.58.2.2 \_\_starpu\_take\_rw\_lock\_write()

```
void __starpu_take_rw_lock_write (
    struct __starpu_rw_lock * lock )
```

Grab the RW-lock in a write mode

### 6.58.2.3 \_\_starpu\_take\_rw\_lock\_read()

```
void __starpu_take_rw_lock_read (
    struct __starpu_rw_lock * lock )
```

Grab the RW-lock in a read mode

### 6.58.2.4 \_\_starpu\_take\_rw\_lock\_write\_try()

```
int __starpu_take_rw_lock_write_try (
    struct __starpu_rw_lock * lock )
```

Try to grab the RW-lock in a write mode. Returns 0 in case of success, -1 otherwise.

### 6.58.2.5 \_\_starpu\_take\_rw\_lock\_read\_try()

```
int __starpu_take_rw_lock_read_try (
    struct __starpu_rw_lock * lock )
```

Try to grab the RW-lock in a read mode. Returns 0 in case of success, -1 otherwise.

### 6.58.2.6 \_\_starpu\_release\_rw\_lock()

```
void __starpu_release_rw_lock (
    struct __starpu_rw_lock * lock )
```

Unlock the RW-lock.

## 6.59 sched\_component.h File Reference

```
#include <starpu_sched_component.h>
```

### Functions

- void [\\_\\_starpu\\_sched\\_component\\_lock\\_all\\_workers](#) (void)
- void [\\_\\_starpu\\_sched\\_component\\_unlock\\_all\\_workers](#) (void)
- void [\\_\\_starpu\\_sched\\_component\\_workers\\_destroy](#) (void)
- [struct \\_\\_starpu\\_worker](#) \* [\\_\\_starpu\\_sched\\_component\\_worker\\_get\\_worker](#) ([struct](#) starpu\_sched\_↵ component \*)
- [struct](#) starpu\_bitmap \* [\\_\\_starpu\\_get\\_worker\\_mask](#) (unsigned sched\_ctx\_id)

### 6.59.1 Function Documentation

### 6.59.1.1 `_starpu_sched_component_lock_all_workers()`

```
void _starpu_sched_component_lock_all_workers (
    void )
```

lock and unlock drivers for modifying schedulers

## 6.60 `sched_ctx.h` File Reference

```
#include <starpu.h>
#include <starpu_sched_ctx.h>
#include <starpu_sched_ctx_hypervisor.h>
#include <starpu_scheduler.h>
#include <common/config.h>
#include <common/barrier_counter.h>
#include <common/utils.h>
#include <profiling/profiling.h>
#include <semaphore.h>
#include <core/task.h>
#include "sched_ctx_list.h"
#include <hwloc.h>
```

### Data Structures

- [struct `\_starpu\_sched\_ctx`](#)
- [struct `\_starpu\_ctx\_change`](#)

### Macros

- `#define NO_RESIZE`
- `#define REQ_RESIZE`
- `#define DO_RESIZE`
- `#define STARPU_GLOBAL_SCHED_CTX`
- `#define STARPU_NMAXSMS`
- `#define _starpu_sched_ctx_get_sched_ctx_for_worker_and_job(w, j)`
- `#define STARPU_SCHED_CTX_CHECK_LOCK(sched_ctx_id)`

### Functions

- [void `\_starpu\_init\_all\_sched\_ctxs`](#) ([struct `\_starpu\_machine\_config`](#) \*config)
- [struct `\_starpu\_sched\_ctx` \\* `\_starpu\_create\_sched\_ctx`](#) ([struct `starpu\_sched\_policy`](#) \*policy, int \*workerid, int nworkerids, unsigned is\_init\_sched, const char \*sched\_name, int min\_prio\_set, int min\_prio, int max\_prio\_set, int max\_prio, unsigned awake\_workers, void(\*sched\_policy\_init)(unsigned), void \*user\_data, int nsub\_ctxs, int \*sub\_ctxs, int nsms)
- [void `\_starpu\_delete\_all\_sched\_ctxs`](#) ()
- [int `\_starpu\_wait\_for\_all\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id)
- [int `\_starpu\_wait\_for\_n\_submitted\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id, unsigned n)
- [void `\_starpu\_decrement\_nsubmitted\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id)
- [void `\_starpu\_increment\_nsubmitted\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id)
- [int `\_starpu\_get\_nsubmitted\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id)
- [int `\_starpu\_check\_nsubmitted\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id)
- [void `\_starpu\_decrement\_nready\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id, double ready\_flops)
- [unsigned `\_starpu\_increment\_nready\_tasks\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id, double ready\_flops, [struct `starpu\_task`](#) \*task)
- [int `\_starpu\_wait\_for\_no\_ready\_of\_sched\_ctx`](#) (unsigned sched\_ctx\_id)
- [int `\_starpu\_get\_index\_in\_ctx\_of\_workerid`](#) (unsigned sched\_ctx, unsigned workerid)

- `starpu_pthread_mutex_t * _starpu_get_sched_mutex (struct _starpu_sched_ctx *sched_ctx, int worker)`
- `int _starpu_get_workers_of_sched_ctx (unsigned sched_ctx_id, int *pus, enum starpu_worker_archtype arch)`
- `void _starpu_worker_gets_out_of_ctx (unsigned sched_ctx_id, struct _starpu_worker *worker)`
- `unsigned _starpu_worker_belongs_to_a_sched_ctx (int workerid, unsigned sched_ctx_id)`
- `unsigned _starpu_sched_ctx_last_worker_awake (struct _starpu_worker *worker)`
- `unsigned _starpu_sched_ctx_get_current_context ()`
- `int _starpu_workers_able_to_execute_task (struct starpu_task *task, struct _starpu_sched_ctx *sched_ctx)`
- `void _starpu_fetch_tasks_from_empty_ctx_list (struct _starpu_sched_ctx *sched_ctx)`
- `unsigned _starpu_sched_ctx_allow_hypervisor (unsigned sched_ctx_id)`
- `struct starpu_perfmodel_arch * _starpu_sched_ctx_get_perf_archtype (unsigned sched_ctx)`
- `void _starpu_sched_ctx_post_exec_task_cb (int workerid, struct starpu_task *task, size_t data_size, uint32_t footprint)`
- `void starpu_sched_ctx_add_combined_workers (int *combined_workers_to_add, unsigned n_combined_workers_to_add, unsigned sched_ctx_id)`
- `struct _starpu_sched_ctx * __starpu_sched_ctx_get_sched_ctx_for_worker_and_job (struct _starpu_worker *worker, struct _starpu_job *j)`
- `static struct _starpu_sched_ctx * _starpu_get_sched_ctx_struct (unsigned id)`
- `static int _starpu_sched_ctx_check_write_locked (unsigned sched_ctx_id)`
- `static void _starpu_sched_ctx_lock_write (unsigned sched_ctx_id)`
- `static void _starpu_sched_ctx_unlock_write (unsigned sched_ctx_id)`
- `static void _starpu_sched_ctx_lock_read (unsigned sched_ctx_id)`
- `static void _starpu_sched_ctx_unlock_read (unsigned sched_ctx_id)`
- `static unsigned _starpu_sched_ctx_worker_is_master_for_child_ctx (unsigned sched_ctx_id, unsigned workerid, struct starpu_task *task)`
- `void _starpu_worker_apply_deferred_ctx_changes (void)`

## 6.60.1 Data Structure Documentation

### 6.60.1.1 struct \_starpu\_ctx\_change

per-worker list of deferred ctx\_change ops

#### Data Fields

int	sched_ctx_id	
int	op	
int	nworkers_to_notify	
int *	workerids_to_notify	
int	nworkers_to_change	
int *	workerids_to_change	

## 6.60.2 Function Documentation

### 6.60.2.1 \_starpu\_init\_all\_sched\_ctxs()

```
void _starpu_init_all_sched_ctxs (
    struct _starpu_machine_config * config )
```

init sched\_ctx\_id of all contextes

### 6.60.2.2 \_starpu\_create\_sched\_ctx()

```
struct _starpu_sched_ctx* _starpu_create_sched_ctx (
```



```

    struct starpu_sched_policy * policy,
    int * workerid,
    int nworkerids,
    unsigned is_init_sched,
    const char * sched_name,
    int min_prio_set,
    int min_prio,
    int max_prio_set,
    int max_prio,
    unsigned awake_workers,
    void(*) (unsigned) sched_policy_init,
    void * user_data,
    int nsub_ctxs,
    int * sub_ctxs,
    int nsms )

```

allocate all structures belonging to a context

### 6.60.2.3 `_starpu_delete_all_sched_ctxs()`

```
void _starpu_delete_all_sched_ctxs ( )
```

delete all sched\_ctx

### 6.60.2.4 `_starpu_wait_for_all_tasks_of_sched_ctx()`

```
int _starpu_wait_for_all_tasks_of_sched_ctx (
    unsigned sched_ctx_id )
```

This function waits until all the tasks that were already submitted to a specific context have been executed.

### 6.60.2.5 `_starpu_wait_for_n_submitted_tasks_of_sched_ctx()`

```
int _starpu_wait_for_n_submitted_tasks_of_sched_ctx (
    unsigned sched_ctx_id,
    unsigned n )
```

This function waits until at most n tasks are still submitted.

### 6.60.2.6 `_starpu_decrement_nsubmitted_tasks_of_sched_ctx()`

```
void _starpu_decrement_nsubmitted_tasks_of_sched_ctx (
    unsigned sched_ctx_id )
```

In order to implement `starpu_wait_for_all_tasks_of_ctx`, we keep track of the number of task currently submitted to the context

### 6.60.2.7 `_starpu_get_index_in_ctx_of_workerid()`

```
int _starpu_get_index_in_ctx_of_workerid (
    unsigned sched_ctx,
    unsigned workerid )
```

Return the corresponding index of the workerid in the ctx table

### 6.60.2.8 `_starpu_get_sched_mutex()`

```
starpu_pthread_mutex_t* _starpu_get_sched_mutex (
    struct _starpu_sched_ctx * sched_ctx,
    int worker )
```

Get the mutex corresponding to the global workerid

### 6.60.2.9 `_starpu_get_workers_of_sched_ctx()`

```
int _starpu_get_workers_of_sched_ctx (
    unsigned sched_ctx_id,
```

```
int * pus,
enum starpu_worker_archtype arch )
```

Get workers belonging to a certain context, it returns the number of workers take care: no mutex taken, the list of workers might not be updated

#### 6.60.2.10 `_starpu_worker_gets_out_of_ctx()`

```
void _starpu_worker_gets_out_of_ctx (
    unsigned sched_ctx_id,
    struct _starpu_worker * worker )
```

Let the worker know it does not belong to the context and that it should stop popping from it

#### 6.60.2.11 `_starpu_worker_belongs_to_a_sched_ctx()`

```
unsigned _starpu_worker_belongs_to_a_sched_ctx (
    int workerid,
    unsigned sched_ctx_id )
```

Check if the worker belongs to another sched\_ctx

#### 6.60.2.12 `_starpu_sched_ctx_last_worker_awake()`

```
unsigned _starpu_sched_ctx_last_worker_awake (
    struct _starpu_worker * worker )
```

indicates wheather this worker should go to sleep or not (if it is the last one awake in a context he should better keep awake)

#### 6.60.2.13 `_starpu_sched_ctx_get_current_context()`

```
unsigned _starpu_sched_ctx_get_current_context ( )
```

If `starpu_sched_ctx_set_context()` has been called, returns the context id set by its last call, or the id of the initial context

#### 6.60.2.14 `_starpu_workers_able_to_execute_task()`

```
int _starpu_workers_able_to_execute_task (
    struct starpu_task * task,
    struct _starpu_sched_ctx * sched_ctx )
```

verify that some worker can execute a certain task

#### 6.60.2.15 `_starpu_sched_ctx_post_exec_task_cb()`

```
void _starpu_sched_ctx_post_exec_task_cb (
    int workerid,
    struct starpu_task * task,
    size_t data_size,
    uint32_t footprint )
```

Notifies the hypervisor that a tasks was popped from the workers' list

#### 6.60.2.16 `__starpu_sched_ctx_get_sched_ctx_for_worker_and_job()`

```
struct _starpu_sched_ctx* __starpu_sched_ctx_get_sched_ctx_for_worker_and_job (
    struct _starpu_worker * worker,
    struct _starpu_job * j )
```

if the worker is the master of a parallel context, and the job is meant to be executed on this parallel context, return a pointer to the context

### 6.60.2.17 `_starpu_worker_apply_deferred_ctx_changes()`

```
void _starpu_worker_apply_deferred_ctx_changes (
    void )
```

Go through the list of deferred ctx changes of the current worker and apply any ctx change operation found until the list is empty

## 6.61 `sched_ctx_list.h` File Reference

### Data Structures

- [struct `\_starpu\_sched\_ctx\_list`](#)
- [struct `\_starpu\_sched\_ctx\_elt`](#)
- [struct `\_starpu\_sched\_ctx\_list\_iterator`](#)

### Functions

- [struct `\_starpu\_sched\_ctx\_elt` \\* `\_starpu\_sched\_ctx\_elt\_find` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [void `\_starpu\_sched\_ctx\_elt\_ensure\_consistency` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [void `\_starpu\_sched\_ctx\_elt\_init` \(struct `\_starpu\_sched\_ctx\_elt` \\*elt, unsigned sched\\_ctx\)](#)
- [struct `\_starpu\_sched\_ctx\_elt` \\* `\_starpu\_sched\_ctx\_elt\_add\_after` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [struct `\_starpu\_sched\_ctx\_elt` \\* `\_starpu\_sched\_ctx\_elt\_add\_before` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [struct `\_starpu\_sched\_ctx\_elt` \\* `\_starpu\_sched\_ctx\_elt\_add` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [void `\_starpu\_sched\_ctx\_elt\_remove` \(struct `\_starpu\_sched\_ctx\_list` \\*list, struct `\_starpu\_sched\_ctx\_elt` \\*elt\)](#)
- [int `\_starpu\_sched\_ctx\_elt\_exists` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [int `\_starpu\_sched\_ctx\_elt\_get\_priority` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [struct `\_starpu\_sched\_ctx\_list` \\* `\_starpu\_sched\_ctx\_list\_find` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned prio\)](#)
- [struct `\_starpu\_sched\_ctx\_elt` \\* `\_starpu\_sched\_ctx\_list\_add\_prio` \(struct `\_starpu\_sched\_ctx\_list` \\*\\*list, unsigned prio, unsigned sched\\_ctx\)](#)
- [int `\_starpu\_sched\_ctx\_list\_add` \(struct `\_starpu\_sched\_ctx\_list` \\*\\*list, unsigned sched\\_ctx\)](#)
- [void `\_starpu\_sched\_ctx\_list\_remove\_elt` \(struct `\_starpu\_sched\_ctx\_list` \\*\\*list, struct `\_starpu\_sched\_ctx\_elt` \\*rm\)](#)
- [int `\_starpu\_sched\_ctx\_list\_remove` \(struct `\_starpu\_sched\_ctx\_list` \\*\\*list, unsigned sched\\_ctx\)](#)
- [int `\_starpu\_sched\_ctx\_list\_move` \(struct `\_starpu\_sched\_ctx\_list` \\*\\*list, unsigned sched\\_ctx, unsigned prio\\_to\)](#)
- [int `\_starpu\_sched\_ctx\_list\_exists` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned prio\)](#)
- [void `\_starpu\_sched\_ctx\_list\_remove\_all` \(struct `\_starpu\_sched\_ctx\_list` \\*list\)](#)
- [void `\_starpu\_sched\_ctx\_list\_delete` \(struct `\_starpu\_sched\_ctx\_list` \\*\\*list\)](#)
- [int `\_starpu\_sched\_ctx\_list\_push\_event` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [int `\_starpu\_sched\_ctx\_list\_pop\_event` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [int `\_starpu\_sched\_ctx\_list\_pop\_all\_event` \(struct `\_starpu\_sched\_ctx\_list` \\*list, unsigned sched\\_ctx\)](#)
- [int `\_starpu\_sched\_ctx\_list\_iterator\_init` \(struct `\_starpu\_sched\_ctx\_list` \\*list, struct `\_starpu\_sched\_ctx\_list\_iterator` \\*it\)](#)
- [int `\_starpu\_sched\_ctx\_list\_iterator\_has\_next` \(struct `\_starpu\_sched\_ctx\_list\_iterator` \\*it\)](#)
- [struct `\_starpu\_sched\_ctx\_elt` \\* `\_starpu\_sched\_ctx\_list\_iterator\_get\_next` \(struct `\_starpu\_sched\_ctx\_list\_iterator` \\*it\)](#)

### 6.61.1 Data Structure Documentation

#### 6.61.1.1 struct `_starpu_sched_ctx_list`

## Data Fields

<a href="#">struct _starpusched_ctx_list</a> *	prev	
<a href="#">struct _starpusched_ctx_list</a> *	next	
<a href="#">struct _starpusched_ctx_elt</a> *	head	
unsigned	priority	

## 6.61.1.2 struct \_starpusched\_ctx\_elt

Represents a circular list of sched context.

## Data Fields

<a href="#">struct _starpusched_ctx_elt</a> *	prev	
<a href="#">struct _starpusched_ctx_elt</a> *	next	
<a href="#">struct _starpusched_ctx_list</a> *	parent	
unsigned	sched_ctx	
long	task_number	
unsigned	last_poped	

## 6.61.1.3 struct \_starpusched\_ctx\_list\_iterator

## Data Fields

<a href="#">struct _starpusched_ctx_list</a> *	list_head	
<a href="#">struct _starpusched_ctx_elt</a> *	cursor	

## 6.61.2 Function Documentation

## 6.61.2.1 \_starpusched\_ctx\_elt\_find()

```
struct _starpusched_ctx_elt* _starpusched_ctx_elt_find (
    struct _starpusched_ctx_list * list,
    unsigned sched_ctx )
```

Element (sched\_ctx) level operations

## 6.61.2.2 \_starpusched\_ctx\_list\_find()

```
struct _starpusched_ctx_list* _starpusched_ctx_list_find (
    struct _starpusched_ctx_list * list,
    unsigned prio )
```

List (priority) level operations

## 6.61.2.3 \_starpusched\_ctx\_list\_push\_event()

```
int _starpusched_ctx_list_push_event (
    struct _starpusched_ctx_list * list,
    unsigned sched_ctx )
```

Task number management

## 6.61.2.4 \_starpusched\_ctx\_list\_iterator\_init()

```
int _starpusched_ctx_list_iterator_init (
```

```

    struct _starpu_sched_ctx_list * list,
    struct _starpu_sched_ctx_list_iterator * it )

```

Iterator operations

## 6.62 sched\_policy.h File Reference

```

#include <starpu.h>
#include <signal.h>
#include <core/workers.h>
#include <core/sched_ctx.h>
#include <starpu_scheduler.h>
#include <core/simgrid.h>

```

### Macros

- `#define _STARPU_SCHED_BEGIN`
- `#define _STARPU_SCHED_END`
- `#define _STARPU_TASK_BREAK_ON(task, what)`

### Functions

- `void _starpu_sched_init (void)`
- `struct starpu_sched_policy * _starpu_get_sched_policy (struct _starpu_sched_ctx *sched_ctx)`
- `void _starpu_init_sched_policy (struct _starpu_machine_config *config, struct _starpu_sched_ctx *sched_ctx, struct starpu_sched_policy *policy)`
- `void _starpu_deinit_sched_policy (struct _starpu_sched_ctx *sched_ctx)`
- `struct starpu_sched_policy * _starpu_select_sched_policy (struct _starpu_machine_config *config, const char *required_policy)`
- `void _starpu_sched_task_submit (struct starpu_task *task)`
- `void _starpu_sched_do_schedule (unsigned sched_ctx_id)`
- `int _starpu_push_task (struct _starpu_job *task)`
- `int _starpu_repush_task (struct _starpu_job *task)`
- `int _starpu_push_task_to_workers (struct starpu_task *task)`
- `struct starpu_task * _starpu_pop_task (struct _starpu_worker *worker)`
- `struct starpu_task * _starpu_pop_every_task (struct _starpu_sched_ctx *sched_ctx)`
- `void _starpu_sched_post_exec_hook (struct starpu_task *task)`
- `int _starpu_pop_task_end (struct starpu_task *task)`
- `void _starpu_wait_on_sched_event (void)`
- `struct starpu_task * _starpu_create_conversion_task (starpu_data_handle_t handle, unsigned int node) STARPU_ATTRIBUTE_MALLOC`
- `struct starpu_task * _starpu_create_conversion_task_for_arch (starpu_data_handle_t handle, enum starpu_node_kind node_kind) STARPU_ATTRIBUTE_MALLOC`
- `void _starpu_sched_pre_exec_hook (struct starpu_task *task)`
- `void _starpu_print_idle_time ()`

### Variables

- `struct starpu_sched_policy _starpu_sched_lws_policy`
- `struct starpu_sched_policy _starpu_sched_ws_policy`
- `struct starpu_sched_policy _starpu_sched_prio_policy`
- `struct starpu_sched_policy _starpu_sched_random_policy`
- `struct starpu_sched_policy _starpu_sched_dm_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_prio_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_ready_policy`

- `struct` `starpu_sched_policy _starpu_sched_dmda_sorted_policy`
- `struct` `starpu_sched_policy _starpu_sched_dmda_sorted_decision_policy`
- `struct` `starpu_sched_policy _starpu_sched_eager_policy`
- `struct` `starpu_sched_policy _starpu_sched_parallel_heft_policy`
- `struct` `starpu_sched_policy _starpu_sched_peager_policy`
- `struct` `starpu_sched_policy _starpu_sched_heteroprio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_eager_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_eager_prefetching_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_eager_prio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_gemm_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_prio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_prio_prefetching_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_random_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_random_prio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_random_prefetching_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_random_prio_prefetching_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_parallel_random_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_parallel_random_prio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_ws_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_heft_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_heft_prio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_heft2_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_heteroprio_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_heteroprio_heft_policy`
- `struct` `starpu_sched_policy _starpu_sched_modular_parallel_heft_policy`
- `struct` `starpu_sched_policy _starpu_sched_graph_test_policy`
- `struct` `starpu_sched_policy _starpu_sched_tree_heft_hierarchical_policy`
- `long` `_starpu_task_break_on_push`
- `long` `_starpu_task_break_on_sched`
- `long` `_starpu_task_break_on_pop`
- `long` `_starpu_task_break_on_exec`

## 6.62.1 Function Documentation

### 6.62.1.1 `_starpu_push_task_to_workers()`

```
int _starpu_push_task_to_workers (
    struct starpu_task * task )
```

actually pushes the tasks to the specific worker or to the scheduler

### 6.62.1.2 `_starpu_pop_task()`

```
struct starpu_task* _starpu_pop_task (
    struct _starpu_worker * worker )
```

pop a task that can be executed on the worker

### 6.62.1.3 `_starpu_pop_every_task()`

```
struct starpu_task* _starpu_pop_every_task (
    struct _starpu_sched_ctx * sched_ctx )
```

pop every task that can be executed on the worker

## 6.63 simgrid.h File Reference

```
#include <xbt/xbt_os_time.h>
```

### Data Structures

- [struct \\_starpu\\_pthread\\_args](#)

### Macros

- `#define MAX_TSD`
- `#define STARPU_MPI_AS_PREFIX`
- `#define _starpu_simgrid_running_smpi()`
- `#define _starpu_simgrid_cuda_malloc_cost()`
- `#define _starpu_simgrid_cuda_queue_cost()`
- `#define _starpu_simgrid_task_submit_cost()`
- `#define _starpu_simgrid_fetching_input_cost()`
- `#define _starpu_simgrid_sched_cost()`
- `#define _SIMGRID_TIMER_BEGIN(cond)`
- `#define _SIMGRID_TIMER_END`
- `#define _starpu_simgrid_data_new(size)`
- `#define _starpu_simgrid_data_increase(size)`
- `#define _starpu_simgrid_data_alloc(size)`
- `#define _starpu_simgrid_data_free(size)`
- `#define _starpu_simgrid_data_transfer(size, src_node, dst_node)`

### Typedefs

- `typedef SD_link_t starpu_sg_link_t`

### Functions

- `void _starpu_start_simgrid (int *argc, char **argv)`
- `void _starpu_simgrid_init_early (int *argc, char ***argv)`
- `void _starpu_simgrid_init (void)`
- `void _starpu_simgrid_deinit (void)`
- `void _starpu_simgrid_deinit_late (void)`
- `void _starpu_simgrid_actor_setup (void)`
- `void _starpu_simgrid_wait_tasks (int workerid)`
- `void _starpu_simgrid_submit_job (int workerid, struct \_starpu\_job *job, struct starpu_perfmodel_arch *perf_arch, double length, unsigned *finished)`
- `int _starpu_simgrid_transfer (size_t size, unsigned src_node, unsigned dst_node, struct _starpu_data_↔ request *req)`
- `int _starpu_simgrid_wait_transfer_event (union \_starpu\_async\_channel\_event *event)`
- `int _starpu_simgrid_test_transfer_event (union \_starpu\_async\_channel\_event *event)`
- `void _starpu_simgrid_sync_gpus (void)`
- `int \_starpu\_simgrid\_get\_nbhosts (const char *prefix)`
- `unsigned long long \_starpu\_simgrid\_get\_memsizes (const char *prefix, unsigned devid)`
- `starpu_sg_host_t \_starpu\_simgrid\_get\_host\_by\_name (const char *name)`
- `starpu_sg_host_t \_starpu\_simgrid\_get\_memnode\_host (unsigned node)`
- `starpu_sg_host_t \_starpu\_simgrid\_get\_host\_by\_worker (struct _starpu_worker *worker)`
- `void \_starpu\_simgrid\_get\_platform\_path (int version, char *path, size_t maxlen)`
- `msg_as_t \_starpu\_simgrid\_get\_as\_by\_name (const char *name)`
- `int starpu\_mpi\_world\_rank (void)`
- `int \_starpu\_mpi\_simgrid\_init (int argc, char *argv[])`
- `void \_starpu\_simgrid\_count\_ngpus (void)`
- `void \_starpu\_simgrid\_xbt\_thread\_create (const char *name, void_f_pvoid_t code, void *param)`

## Variables

- `starpu_pthread_queue_t_starpu_simgrid_transfer_queue` [STARPU\_MAXNODES]
- `starpu_pthread_queue_t_starpu_simgrid_task_queue` [STARPU\_NMAXWORKERS]

### 6.63.1 Macro Definition Documentation

#### 6.63.1.1 `_starpu_simgrid_data_new`

```
#define _starpu_simgrid_data_new(  
    size )
```

Experimental functions for OOC stochastic analysis

### 6.63.2 Function Documentation

#### 6.63.2.1 `_starpu_simgrid_get_nbhosts()`

```
int _starpu_simgrid_get_nbhosts (  
    const char * prefix )
```

Return the number of hosts prefixed by PREFIX

#### 6.63.2.2 `_starpu_simgrid_count_ngpus()`

```
void _starpu_simgrid_count_ngpus (  
    void )
```

Called at initialization to count how many GPUs are interfering with each bus

## 6.64 sink\_common.h File Reference

```
#include <common/config.h>
```

## 6.65 sort\_data\_handles.h File Reference

```
#include <starpu.h>  
#include <common/config.h>  
#include <stdlib.h>  
#include <stdarg.h>  
#include <core/jobs.h>  
#include <datawizard/coherency.h>  
#include <datawizard/memalloc.h>
```

## Functions

- void `_starpu_sort_task_handles` (struct `_starpu_data_descr` descr[], unsigned nbufers)

### 6.65.1 Function Documentation



### 6.65.1.1 `_starpu_sort_task_handles()`

```
void _starpu_sort_task_handles (
    struct _starpu_data_descr descr[],
    unsigned nbuffers )
```

To avoid deadlocks, we reorder the different buffers accessed to by the task so that we always grab the rw-lock associated to the handles in the same order.

## 6.66 `source_common.h` File Reference

## 6.67 `starpu_clusters_create.h` File Reference

```
#include <starpu.h>
#include <core/workers.h>
#include <common/list.h>
#include <string.h>
#include <omp.h>
```

## 6.68 `starpu_data_cpy.h` File Reference

```
#include <starpu.h>
```

### Functions

- `int _starpu_data_cpy` (`starpu_data_handle_t` dst\_handle, `starpu_data_handle_t` src\_handle, `int` asynchronous, `void(*callback_func)(void *)`, `void *callback_arg`, `int` reduction, `struct starpu_task` \*reduction\_task, `dep_task`)

## 6.69 `starpu_debug_helpers.h` File Reference

```
#include <starpu.h>
#include <starpu_config.h>
#include <starpu_util.h>
```

### Functions

- `void _starpu_benchmark_ping_pong` (`starpu_data_handle_t` handle, `unsigned` node0, `unsigned` node1, `unsigned` niter)
- `void _starpu_debug_display_structures_size` (`FILE` \*stream)

### 6.69.1 Function Documentation

#### 6.69.1.1 `_starpu_benchmark_ping_pong()`

```
void _starpu_benchmark_ping_pong (
    starpu_data_handle_t handle,
    unsigned node0,
    unsigned node1,
    unsigned niter )
```

Perform a ping pong between the two memory nodes

### 6.69.1.2 \_starpu\_debug\_display\_structures\_size()

```
void _starpu_debug_display_structures_size (
    FILE * stream )
```

Display the size of different data structures

## 6.70 starpu\_fxt.h File Reference

```
#include <starpu.h>
#include <starpu_config.h>
#include <common/config.h>
```

## 6.71 starpu\_parameters.h File Reference

### Macros

- `#define _STARPU_CPU_ALPHA`
- `#define _STARPU_CUDA_ALPHA`
- `#define _STARPU_OPENCL_ALPHA`
- `#define _STARPU_MIC_ALPHA`
- `#define _STARPU_MPI_MS_ALPHA`

## 6.72 starpu\_spinlock.h File Reference

```
#include <errno.h>
#include <stdint.h>
#include <common/config.h>
#include <common/fxt.h>
#include <common/thread.h>
#include <starpu.h>
```

### Data Structures

- [struct \\_starpu\\_spinlock](#)

### Macros

- `#define _starpu_spin_destroy(_lock)`
- `#define _starpu_spin_checklocked(_lock)`
- `#define _starpu_spin_lock(lock)`
- `#define _starpu_spin_trylock(lock)`
- `#define _starpu_spin_unlock(lock)`
- `#define STARPU_SPIN_MAXTRY`

### Functions

- static int `_starpu_spin_init` ([struct \\_starpu\\_spinlock](#) \*lock)
- static int `__starpu_spin_lock` ([struct \\_starpu\\_spinlock](#) \*lock, const char \*file STARPU\_ATTRIBUTE\_UNUSED, int line STARPU\_ATTRIBUTE\_UNUSED, const char \*func STARPU\_ATTRIBUTE\_UNUSED)
- static int `__starpu_spin_trylock` ([struct \\_starpu\\_spinlock](#) \*lock, const char \*file STARPU\_ATTRIBUTE\_UNUSED, int line STARPU\_ATTRIBUTE\_UNUSED, const char \*func STARPU\_ATTRIBUTE\_UNUSED)
- static int `__starpu_spin_unlock` ([struct \\_starpu\\_spinlock](#) \*lock, const char \*file STARPU\_ATTRIBUTE\_UNUSED, int line STARPU\_ATTRIBUTE\_UNUSED, const char \*func STARPU\_ATTRIBUTE\_UNUSED)

## 6.72.1 Data Structure Documentation

### 6.72.1.1 struct \_starpu\_spinlock

#### Data Fields

starpu_pthread_spinlock_t	lock	
---------------------------	------	--

## 6.73 starpu\_task\_insert\_utils.h File Reference

```
#include <stdlib.h>
#include <stdarg.h>
#include <starpu.h>
```

### Typedefs

- typedef void(\* \_starpu\_callback\_func\_t) (void \*)

### Functions

- int \_starpu\_codelet\_pack\_args (void \*\*arg\_buffer, size\_t \*arg\_buffer\_size, va\_list varg\_list)
- int \_starpu\_task\_insert\_create (struct starpu\_codelet \*cl, struct starpu\_task \*task, va\_list varg\_list)
- int \_fstarpu\_task\_insert\_create (struct starpu\_codelet \*cl, struct starpu\_task \*task, void \*\*arglist)

## 6.74 tags.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <core/dependencies/cg.h>
```

### Data Structures

- struct \_starpu\_tag

### Macros

- #define \_STARPU\_TAG\_SIZE

### Enumerations

- enum \_starpu\_tag\_state {  
STARPU\_INVALID\_STATE , STARPU\_ASSOCIATED , STARPU\_BLOCKED , STARPU\_READY ,  
STARPU\_DONE }

### Functions

- void \_starpu\_init\_tags (void)
- void \_starpu\_notify\_tag\_dependencies (struct \_starpu\_tag \*tag)
- void \_starpu\_notify\_job\_start\_tag\_dependencies (struct \_starpu\_tag \*tag, \_starpu\_notify\_job\_start\_data \*data)
- void \_starpu\_tag\_declare (starpu\_tag\_t id, struct \_starpu\_job \*job)
- void \_starpu\_tag\_set\_ready (struct \_starpu\_tag \*tag)
- unsigned \_starpu\_submit\_job\_enforce\_task\_deps (struct \_starpu\_job \*j)
- void \_starpu\_tag\_clear (void)

## 6.74.1 Data Structure Documentation

### 6.74.1.1 struct \_starpu\_tag

#### Data Fields

<a href="#">struct _starpu_spinlock</a>	lock	Lock for this structure. Locking order is in dependency order: a tag must not be locked before locking a tag it depends on
starpu_tag_t	id	an identifier for the task
<a href="#">enum _starpu_tag_state</a>	state	
<a href="#">struct _starpu_cg_list</a>	tag_successors	
<a href="#">struct _starpu_job *</a>	job	which job is associated to the tag if any ?
unsigned	is_assigned	
unsigned	is_submitted	

## 6.74.2 Enumeration Type Documentation

### 6.74.2.1 \_starpu\_tag\_state

[enum \\_starpu\\_tag\\_state](#)

#### Enumerator

STARPU_INVALID_STATE	this tag is not declared by any task
STARPU_ASSOCIATED	_starpu_tag_declare was called to associate the tag to a task
STARPU_BLOCKED	some task dependencies are not fulfilled yet
STARPU_READY	the task can be (or has been) submitted to the scheduler (all deps fulfilled)
STARPU_DONE	the task has been performed

## 6.74.3 Function Documentation

### 6.74.3.1 \_starpu\_tag\_set\_ready()

```
void _starpu_tag_set_ready (
    struct \_starpu\_tag * tag )
```

lock should be taken, and this releases it

## 6.75 task.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <core/jobs.h>
```

### Macros

- `#define _STARPU_JOB_UNSET`
- `#define _STARPU_JOB_SETTING`
- `#define _STARPU_TASK_SET_INTERFACE(task, interface, i)`
- `#define _STARPU_TASK_GET_INTERFACES(task)`

## Functions

- void `_starpu_task_destroy` (`struct starpu_task *task`)
- int `_starpu_task_test_termination` (`struct starpu_task *task`)
- void `_starpu_task_init` (void)
- void `_starpu_task_deinit` (void)
- void `_starpu_set_current_task` (`struct starpu_task *task`)
- int `_starpu_submit_job` (`struct _starpu_job *j`)
- int `_starpu_task_submit_nodeps` (`struct starpu_task *task`)
- void `_starpu_task_declare_deps_array` (`struct starpu_task *task`, unsigned ndeps, `struct starpu_task *task_array[]`, int check)
- `struct starpu_job *` `_starpu_get_job_associated_to_task_slow` (`struct starpu_task *task`, `struct starpu_job *job`)
- static `struct starpu_job *` `_starpu_get_job_associated_to_task` (`struct starpu_task *task`)
- int `_starpu_task_submit_internally` (`struct starpu_task *task`)
- int `_starpu_handle_needs_conversion_task` (`starpu_data_handle_t handle`, unsigned int node)
- int `_starpu_handle_needs_conversion_task_for_arch` (`starpu_data_handle_t handle`, enum `starpu_↵ node_kind node_kind`)
- void `_starpu_task_prepare_for_continuation_ext` (unsigned continuation\_resubmit, void(\*continuation\_↵ callback\_on\_sleep)(void \*arg), void \*continuation\_callback\_on\_sleep\_arg)
- void `_starpu_task_prepare_for_continuation` (void)
- void `_starpu_task_set_omp_cleanup_callback` (`struct starpu_task *task`, void(\*omp\_cleanup\_↵ callback)(void \*arg), void \*omp\_cleanup\_callback\_arg)
- int `_starpu_task_uses_multiformat_handles` (`struct starpu_task *task`)
- int `_starpu_task_submit_conversion_task` (`struct starpu_task *task`, unsigned int workerid)
- void `_starpu_task_check_deprecated_fields` (`struct starpu_task *task`)
- void `_starpu_codelet_check_deprecated_fields` (`struct starpu_codelet *cl`)
- static `starpu_cpu_func_t` `_starpu_task_get_cpu_nth_implementation` (`struct starpu_codelet *cl`, unsigned nimpl)
- static `starpu_cuda_func_t` `_starpu_task_get_cuda_nth_implementation` (`struct starpu_codelet *cl`, unsigned nimpl)
- static `starpu_opencl_func_t` `_starpu_task_get_opencl_nth_implementation` (`struct starpu_codelet *cl`, unsigned nimpl)
- static `starpu_mic_func_t` `_starpu_task_get_mic_nth_implementation` (`struct starpu_codelet *cl`, unsigned nimpl)
- static `starpu_mpi_ms_func_t` `_starpu_task_get_mpi_ms_nth_implementation` (`struct starpu_codelet *cl`, unsigned nimpl)
- static const char \* `_starpu_task_get_cpu_name_nth_implementation` (`struct starpu_codelet *cl`, unsigned nimpl)
- void `_starpu_watchdog_init` (void)
- void `_starpu_watchdog_shutdown` (void)
- int `_starpu_task_wait_for_all_and_return_nb_waited_tasks` (void)
- int `_starpu_task_wait_for_all_in_ctx_and_return_nb_waited_tasks` (unsigned sched\_ctx)

### 6.75.1 Function Documentation

#### 6.75.1.1 `_starpu_task_destroy()`

```
void _starpu_task_destroy (
    struct starpu_task * task )
```

Internal version of `starpu_task_destroy`: don't check task->destroy flag

**6.75.1.2 \_\_starpu\_task\_test\_termination()**

```
int __starpu_task_test_termination (
    struct starpu_task * task )
```

Test for the termination of the task. Call `starpu_task_destroy` if required and the task is terminated.

**6.75.1.3 \_\_starpu\_task\_init()**

```
void __starpu_task_init (
    void )
```

A pthread key is used to store the task currently executed on the thread. `__starpu_task_init` initializes this pthread key and `__starpu_set_current_task` updates its current value.

**6.75.1.4 \_\_starpu\_get\_job\_associated\_to\_task\_slow()**

```
struct __starpu_job* __starpu_get_job_associated_to_task_slow (
    struct starpu_task * task,
    struct __starpu_job * job )
```

Returns the job structure (which is the internal data structure associated to a task).

**6.75.1.5 \_\_starpu\_task\_submit\_internally()**

```
int __starpu_task_submit_internally (
    struct starpu_task * task )
```

Submits starpu internal tasks to the initial context

**6.75.1.6 \_\_starpu\_task\_prepare\_for\_continuation\_ext()**

```
void __starpu_task_prepare_for_continuation_ext (
    unsigned continuation_resubmit,
    void(*) (void *arg) continuation_callback_on_sleep,
    void * continuation_callback_on_sleep_arg )
```

Prepare the current task for accepting new dependencies before becoming a continuation.

**6.76 task\_bundle.h File Reference**

```
#include <starpu_thread.h>
```

**Data Structures**

- struct [\\_\\_starpu\\_task\\_bundle\\_entry](#)
- struct [\\_\\_starpu\\_task\\_bundle](#)
- struct [\\_\\_starpu\\_handle\\_list](#)

**Functions**

- void [\\_\\_starpu\\_task\\_bundle\\_destroy](#) (starpu\_task\_bundle\_t bundle)
- void [\\_\\_starpu\\_insertion\\_handle\\_sorted](#) (struct [\\_\\_starpu\\_handle\\_list](#) \*\*listp, starpu\_data\_handle\_t handle, enum starpu\_data\_access\_mode mode)

**6.76.1 Data Structure Documentation****6.76.1.1 struct \_\_starpu\_task\_bundle\_entry**

struct [\\_\\_starpu\\_task\\_bundle\\_entry](#) ===== Purpose ===== Structure used to describe a linked list containing tasks in [\\_\\_starpu\\_task\\_bundle](#).

Fields ===== task Pointer to the task structure.

next Pointer to the next element in the linked list.

## Data Fields

<a href="#">struct starpu_task *</a>	task	
<a href="#">struct _starpu_task_bundle_entry *</a>	next	

**6.76.1.2 struct \_starpu\_task\_bundle**

[struct \\_starpu\\_task\\_bundle](#) ===== Purpose ===== Structure describing a list of tasks that should be scheduled on the same worker whenever it's possible. It must be considered as a hint given to the scheduler as there is no guarantee that they will be executed on the same worker.

Fields ===== mutex Mutex protecting the structure.

list Array of tasks included in the bundle.

closed Used to know if the user is still willing to add/remove some tasks in the bundle. Especially useful for the runtime to know whether it is safe to destroy a bundle.

## Data Fields

<a href="#">starpu_pthread_mutex_t</a>	mutex	Mutex protecting the bundle
<a href="#">struct _starpu_task_bundle_entry *</a>	list	
int	closed	

**6.76.1.3 struct \_starpu\_handle\_list**

[struct \\_starpu\\_handle\\_list](#) ===== Purpose ===== Structure describing a list of handles sorted by address to speed-up when looking for an element. The list cannot contains duplicate handles.

Fields ===== handle Pointer to the handle structure.

access\_mode Total access mode over the whole bundle.

next Pointer to the next element in the linked list.

## Data Fields

<a href="#">starpu_data_handle_t</a>	handle	
<a href="#">enum starpu_data_access_mode</a>	mode	
<a href="#">struct _starpu_handle_list *</a>	next	

**6.76.2 Function Documentation****6.76.2.1 \_starpu\_task\_bundle\_destroy()**

```
void _starpu_task_bundle_destroy (
    starpu_task_bundle_t bundle )
```

[\\_starpu\\_task\\_bundle\\_destroy](#) ===== Purpose ===== Destroy and deinitialize a bundle, memory previously allocated is freed.

Arguments ===== bundle (input) Bundle to destroy.

**6.76.2.2 \_starpu\_insertion\_handle\_sorted()**

```
void _starpu_insertion_handle_sorted (
    struct _starpu_handle_list ** listp,
    starpu_data_handle_t handle,
    enum starpu_data_access_mode mode )
```

`_starpu_insertion_handle_sorted` ===== Purpose ===== Insert an handle in a `_starpu_handle_list`, elements are sorted in increasing order, considering their physical address. As the list doesn't accept duplicate elements, a handle with the same address as an handle contained in the list is not inserted, but its mode access is merged with the one of the latter.

Arguments ===== `listp` (input, output) Pointer to the first element of the list. In the case of an empty list or an inserted handle with small address, it should have changed when the call returns.

`handle` (input) Handle to insert in the list.

`mode` (input) Access mode of the handle.

## 6.77 thread.h File Reference

```
#include <common/utils.h>
```

### Macros

- `#define starpu_pthread_spin_init`
- `#define starpu_pthread_spin_destroy`
- `#define starpu_pthread_spin_lock`
- `#define starpu_pthread_spin_trylock`
- `#define starpu_pthread_spin_unlock`

### Functions

- `static int _starpu_pthread_spin_init` (`starpu_pthread_spinlock_t *lock`, `int pshared STARPU_ATTRIBUTE_UNUSED`)
- `static int _starpu_pthread_spin_destroy` (`starpu_pthread_spinlock_t *lock STARPU_ATTRIBUTE_UNUSED`)
- `static int _starpu_pthread_spin_lock` (`starpu_pthread_spinlock_t *lock`)
- `static void _starpu_pthread_spin_checklocked` (`starpu_pthread_spinlock_t *lock STARPU_ATTRIBUTE_UNUSED`)
- `static int _starpu_pthread_spin_trylock` (`starpu_pthread_spinlock_t *lock`)
- `static int _starpu_pthread_spin_unlock` (`starpu_pthread_spinlock_t *lock`)

## 6.78 timing.h File Reference

```
#include <stdint.h>
#include <common/config.h>
#include <starpu.h>
#include <starpu_util.h>
```

### Functions

- `void _starpu_timing_init` (`void`)
- `void _starpu_clock_gettime` (`struct timespec *ts`)

### 6.78.1 Function Documentation

#### 6.78.1.1 \_starpu\_timing\_init()

```
void _starpu_timing_init (
    void )
```

`_starpu_timing_init` must be called prior to using any of these timing functions.



## 6.79 topology.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <common/fxt.h>
```

### Macros

- `#define STARPU_NOWORKERID`
- `#define STARPU_ACTIVETHREAD`
- `#define STARPU_NONACTIVETHREAD`

### Functions

- `int _starpu_build_topology (struct _starpu_machine_config *config, int no_mp_config)`
- `void _starpu_destroy_machine_config (struct _starpu_machine_config *config)`
- `void _starpu_destroy_topology (struct _starpu_machine_config *config)`
- `unsigned _starpu_topology_get_nhwcpu (struct _starpu_machine_config *config)`
- `unsigned _starpu_topology_get_nhwpu (struct _starpu_machine_config *config)`
- `unsigned _starpu_topology_get_nnumanodes (struct _starpu_machine_config *config)`
- `unsigned _starpu_get_nhyperthreads ()`
- `void _starpu_topology_filter (hwloc_topology_t topology)`
- `int _starpu_bind_thread_on_cpu (int cpuid, int workerid, const char *name)`
- `void _starpu_bind_thread_on_cpus (struct _starpu_combined_worker *combined_worker)`
- `struct _starpu_worker * _starpu_get_worker_from_driver (struct starpu_driver *d)`
- `int starpu_memory_nodes_get_numa_count (void)`
- `int starpu_memory_nodes_numa_id_to_hwloclogid (unsigned id)`
- `int _starpu_task_data_get_node_on_node (struct starpu_task *task, unsigned index, unsigned target_node)`
- `int _starpu_task_data_get_node_on_worker (struct starpu_task *task, unsigned index, unsigned worker)`

### 6.79.1 Function Documentation

#### 6.79.1.1 `_starpu_build_topology()`

```
int _starpu_build_topology (
    struct _starpu_machine_config * config,
    int no_mp_config )
```

Detect the number of memory nodes and where to bind the different workers.

#### 6.79.1.2 `_starpu_destroy_machine_config()`

```
void _starpu_destroy_machine_config (
    struct _starpu_machine_config * config )
```

Should be called instead of `_starpu_destroy_topology` when `_starpu_build_topology` returns a non zero value.

#### 6.79.1.3 `_starpu_destroy_topology()`

```
void _starpu_destroy_topology (
    struct _starpu_machine_config * config )
```

Destroy all resources used to store the topology of the machine.

**6.79.1.4 \_\_starpu\_topology\_get\_nhwcpu()**

```
unsigned __starpu_topology_get_nhwcpu (
    struct __starpu_machine_config * config )
```

returns the number of physical cpus

**6.79.1.5 \_\_starpu\_topology\_get\_nhwpu()**

```
unsigned __starpu_topology_get_nhwpu (
    struct __starpu_machine_config * config )
```

returns the number of logical cpus

**6.79.1.6 \_\_starpu\_topology\_get\_nnumanodes()**

```
unsigned __starpu_topology_get_nnumanodes (
    struct __starpu_machine_config * config )
```

returns the number of NUMA nodes

**6.79.1.7 \_\_starpu\_get\_nhyperthreads()**

```
unsigned __starpu_get_nhyperthreads ( )
```

returns the number of hyperthreads per core

**6.79.1.8 \_\_starpu\_topology\_filter()**

```
void __starpu_topology_filter (
    hwloc_topology_t topology )
```

Small convenient function to filter hwloc topology depending on HWLOC API version

**6.79.1.9 \_\_starpu\_bind\_thread\_on\_cpu()**

```
int __starpu_bind_thread_on_cpu (
    int cpuid,
    int workerid,
    const char * name )
```

Bind the current thread on the CPU logically identified by "cpuid". The logical ordering of the processors is either that of hwloc (if available), or the ordering exposed by the OS.

**6.79.1.10 \_\_starpu\_bind\_thread\_on\_cpus()**

```
void __starpu_bind_thread_on_cpus (
    struct __starpu_combined_worker * combined_worker )
```

Bind the current thread on the set of CPUs for the given combined worker.

**6.79.1.11 \_\_starpu\_task\_data\_get\_node\_on\_node()**

```
int __starpu_task_data_get_node_on_node (
    struct starpu_task * task,
    unsigned index,
    unsigned target_node )
```

Get the memory node for data number i when task is to be executed on memory node target\_node

**6.80 utils.h File Reference**

```
#include <common/config.h>
#include <starpu.h>
#include <sys/stat.h>
#include <string.h>
#include <stdlib.h>
```

```
#include <math.h>
```

## Macros

- `#define DO_CREQ_v_WW(_creqF, _ty1F, _arg1F, _ty2F, _arg2F)`
- `#define DO_CREQ_v_W(_creqF, _ty1F, _arg1F)`
- `#define ANNOTATE_HAPPENS_BEFORE(obj)`
- `#define ANNOTATE_HAPPENS_BEFORE_FORGET_ALL(obj)`
- `#define ANNOTATE_HAPPENS_AFTER(obj)`
- `#define VALGRIND_HG_DISABLE_CHECKING(start, len)`
- `#define VALGRIND_HG_ENABLE_CHECKING(start, len)`
- `#define VALGRIND_STACK_REGISTER(stackbottom, stacktop)`
- `#define VALGRIND_STACK_DEREGISTER(id)`
- `#define RUNNING_ON_VALGRIND`
- `#define STARPU_RUNNING_ON_VALGRIND`
- `#define STARPU_HG_DISABLE_CHECKING(variable)`
- `#define STARPU_HG_ENABLE_CHECKING(variable)`
- `#define STARPU_DEBUG_PREFIX`
- `#define _STARPU_UYIELD()`
- `#define STARPU_VALGRIND_YIELD()`
- `#define STARPU_UYIELD()`
- `#define _STARPU_DEBUG(fmt, ...)`
- `#define _STARPU_DEBUG_NO_HEADER(fmt, ...)`
- `#define _STARPU_EXTRA_DEBUG(fmt, ...)`
- `#define _STARPU_LOG_IN()`
- `#define _STARPU_LOG_OUT()`
- `#define _STARPU_LOG_OUT_TAG(outtag)`
- `#define _STARPU_MSG(fmt, ...)`
- `#define _STARPU_DISP(fmt, ...)`
- `#define _STARPU_ERROR(fmt, ...)`
- `#define _STARPU_DECLTYPE(x)`
- `#define _STARPU_MALLOC(ptr, size)`
- `#define _STARPU_CALLOC(ptr, nmemb, size)`
- `#define _STARPU_REALLOC(ptr, size)`
- `#define _STARPU_IS_ZERO(a)`

## Functions

- `char * _starpu_mktemp_internal (char *tmpl)`
- `char * _starpu_mktemp (char *tmpl)`
- `int _starpu_mkpath (const char *s, mode_t mode)`
- `void _starpu_mkpath_and_check (const char *s, mode_t mode)`
- `char * _starpu_mktemp (const char *directory, int flags, int *fd)`
- `char * _starpu_mktemp_many (const char *directory, int depth, int flags, int *fd)`
- `void _starpu_rmtmp_many (char *path, int depth)`
- `void _starpu_rmdir_many (char *path, int depth)`
- `int _starpu_fftruncate (FILE *file, size_t length)`
- `int _starpu_ftruncate (int fd, size_t length)`
- `int _starpu_frdlock (FILE *file)`
- `int _starpu_frdunlock (FILE *file)`
- `int _starpu_fwrllock (FILE *file)`
- `int _starpu_fwrunlock (FILE *file)`
- `char * _starpu_get_home_path (void)`
- `void _starpu_gethostname (char *hostname, size_t size)`

- void [\\_starpup\\_drop\\_comments](#) (FILE \*f)
- const char \* [\\_starpup\\_job\\_get\\_model\\_name](#) (struct [\\_starpup\\_job](#) \*j)
- const char \* [\\_starpup\\_job\\_get\\_task\\_name](#) (struct [\\_starpup\\_job](#) \*j)
- const char \* [\\_starpup\\_codelet\\_get\\_model\\_name](#) (struct [starpup\\_codelet](#) \*cl)
- int [\\_starpup\\_check\\_mutex\\_deadlock](#) (starpup\_thread\_mutex\_t \*mutex)
- void [\\_starpup\\_util\\_init](#) (void)

## 6.80.1 Function Documentation

### 6.80.1.1 [\\_starpup\\_mktemp\\_many\(\)](#)

```
char* \_starpup\_mktemp\_many (
    const char * directory,
    int depth,
    int flags,
    int * fd )
```

This version creates a hierarchy of n temporary directories, useful when creating a lot of temporary files to be stored in the same place

### 6.80.1.2 [\\_starpup\\_drop\\_comments\(\)](#)

```
void \_starpup\_drop\_comments (
    FILE * f )
```

If FILE is currently on a comment line, eat it.

### 6.80.1.3 [\\_starpup\\_job\\_get\\_model\\_name\(\)](#)

```
const char* \_starpup\_job\_get\_model\_name (
    struct \_starpup\_job * j )
```

Returns the symbol associated to that job if any.

### 6.80.1.4 [\\_starpup\\_job\\_get\\_task\\_name\(\)](#)

```
const char* \_starpup\_job\_get\_task\_name (
    struct \_starpup\_job * j )
```

Returns the name associated to that job if any.

### 6.80.1.5 [\\_starpup\\_codelet\\_get\\_model\\_name\(\)](#)

```
const char* \_starpup\_codelet\_get\_model\_name (
    struct starpup\_codelet * cl )
```

Returns the symbol associated to that job if any.

## 6.81 uthash.h File Reference

```
#include <string.h>
#include <stddef.h>
#include <inttypes.h>
```

### Data Structures

- struct [UT\\_hash\\_bucket](#)
- struct [UT\\_hash\\_table](#)
- struct [UT\\_hash\\_handle](#)

## Macros

- `#define DECLTYPE(x)`
- `#define DECLTYPE_ASSIGN(dst, src)`
- `#define UTHASH_VERSION`
- `#define uthash_fatal(msg)`
- `#define uthash_malloc(sz)`
- `#define uthash_free(ptr, sz)`
- `#define uthash_noexpand_fyi(tbl)`
- `#define uthash_expand_fyi(tbl)`
- `#define HASH_INITIAL_NUM_BUCKETS`
- `#define HASH_INITIAL_NUM_BUCKETS_LOG2`
- `#define HASH_BKT_CAPACITY_THRESH`
- `#define ELMT_FROM_HH(tbl, hhptr)`
- `#define HASH_FIND(hh, head, keyptr, keylen, out)`
- `#define HASH_BLOOM_MAKE(tbl)`
- `#define HASH_BLOOM_FREE(tbl)`
- `#define HASH_BLOOM_ADD(tbl, hashv)`
- `#define HASH_BLOOM_TEST(tbl, hashv)`
- `#define HASH_MAKE_TABLE(hh, head)`
- `#define HASH_ADD(hh, head, fieldname, keylen_in, add)`
- `#define HASH_CHECK_KEY(hh, head, keyptr, keylen, out)`
- `#define HASH_ADD_KEYPTR(hh, head, keyptr, keylen_in, add)`
- `#define HASH_TO_BKT(hashv, num_bkts, bkt)`
- `#define HASH_DELETE(hh, head, delptr)`
- `#define HASH_FIND_STR(head, findstr, out)`
- `#define HASH_ADD_STR(head, strfield, add)`
- `#define HASH_FIND_INT(head, findint, out)`
- `#define HASH_ADD_INT(head, intfield, add)`
- `#define HASH_FIND_PTR(head, findptr, out)`
- `#define HASH_ADD_PTR(head, ptrfield, add)`
- `#define HASH_DEL(head, delptr)`
- `#define HASH_FSCK(hh, head)`
- `#define HASH_EMIT_KEY(hh, head, keyptr, fieldlen)`
- `#define HASH_FCN`
- `#define HASH_BER(key, keylen, num_bkts, hashv, bkt)`
- `#define HASH_SAX(key, keylen, num_bkts, hashv, bkt)`
- `#define HASH_FNV(key, keylen, num_bkts, hashv, bkt)`
- `#define HASH_OAT(key, keylen, num_bkts, hashv, bkt)`
- `#define HASH_JEN_MIX(a, b, c)`
- `#define HASH_JEN(key, keylen, num_bkts, hashv, bkt)`
- `#define get16bits(d)`
- `#define HASH_SFH(key, keylen, num_bkts, hashv, bkt)`
- `#define HASH_KEYCMP(a, b, len)`
- `#define HASH_FIND_IN_BKT(tbl, hh, head, keyptr, keylen_in, out)`
- `#define HASH_ADD_TO_BKT(head, addhh)`
- `#define HASH_DEL_IN_BKT(hh, head, hh_del)`
- `#define HASH_EXPAND_BUCKETS(tbl)`
- `#define HASH_SORT(head, cmpfcn)`
- `#define HASH_SRT(hh, head, cmpfcn)`
- `#define HASH_SELECT(hh_dst, dst, hh_src, src, cond)`
- `#define HASH_CLEAR(hh, head)`
- `#define HASH_ITER(hh, head, el, tmp)`
- `#define HASH_COUNT(head)`
- `#define HASH_CNT(hh, head)`
- `#define HASH_SIGNATURE`
- `#define HASH_BLOOM_SIGNATURE`

## Typedefs

- typedef [struct UT\\_hash\\_bucket](#) **UT\_hash\_bucket**
- typedef [struct UT\\_hash\\_table](#) **UT\_hash\_table**
- typedef [struct UT\\_hash\\_handle](#) **UT\_hash\_handle**

## 6.81.1 Data Structure Documentation

### 6.81.1.1 struct UT\_hash\_bucket

#### Data Fields

<a href="#">struct UT_hash_handle *</a>	hh_head	
unsigned	count	
unsigned	expand_mult	

### 6.81.1.2 struct UT\_hash\_table

#### Data Fields

<a href="#">UT_hash_bucket *</a>	buckets	
unsigned	num_buckets	
unsigned	log2_num_buckets	
unsigned	num_items	
<a href="#">struct UT_hash_handle *</a>	tail	
ptrdiff_t	hho	
unsigned	ideal_chain_maxlen	
unsigned	nonideal_items	
unsigned	ineff_expands	
unsigned	noexpand	
uint32_t	signature	

### 6.81.1.3 struct UT\_hash\_handle

#### Data Fields

<a href="#">struct UT_hash_table *</a>	tbl	
void *	prev	
void *	next	
<a href="#">struct UT_hash_handle *</a>	hh_prev	
<a href="#">struct UT_hash_handle *</a>	hh_next	
void *	key	
unsigned	keylen	
unsigned	hashv	

## 6.82 write\_back.h File Reference

```
#include <starpu.h>
#include <datawizard/coherency.h>
```

## Functions

- void [\\_starpu\\_write\\_through\\_data](#) (starpu\_data\_handle\_t handle, unsigned requesting\_node, uint32\_t write\_through\_mask)

### 6.82.1 Function Documentation

#### 6.82.1.1 [\\_starpu\\_write\\_through\\_data\(\)](#)

```
void _starpu_write_through_data (
    starpu_data_handle_t handle,
    unsigned requesting_node,
    uint32_t write_through_mask )
```

If a write-through mask is associated to that data handle, this propagates the the current value of the data onto the different memory nodes in the write\_through\_mask.

## Chapter 7

# StarPU MPI File Documentation

### 7.1 `starpu_mpi_cache.h` File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

#### Functions

- `void _starpu_mpi_cache_init (MPI_Comm comm)`
- `void _starpu_mpi_cache_shutdown ()`
- `void _starpu_mpi_cache_data_init (starpu_data_handle_t data_handle)`
- `void _starpu_mpi_cache_data_clear (starpu_data_handle_t data_handle)`
- `void _starpu_mpi_cache_flush (starpu_data_handle_t data_handle)`

#### Variables

- `int _starpu_cache_enabled`

### 7.2 `starpu_mpi_driver.h` File Reference

```
#include <starpu.h>
```

#### Functions

- `void _starpu_mpi_driver_init (struct starpu_conf *conf)`
- `void _starpu_mpi_driver_shutdown ()`

### 7.3 `starpu_mpi_init.h` File Reference

```
#include <starpu.h>
#include <starpu_mpi.h>
```

#### Functions

- `void _starpu_mpi_do_initialize (struct _starpu_mpi_argc_argv *argc_argv)`



## 7.4 starpu\_mpi\_nmad\_backend.h File Reference

```
#include <common/config.h>
#include <nm_sendrecv_interface.h>
#include <nm_session_interface.h>
#include <nm_mpi_nmad.h>
```

### Data Structures

- [struct\\_starpu\\_mpi\\_req\\_backend](#)

### 7.4.1 Data Structure Documentation

#### 7.4.1.1 struct\_starpu\_mpi\_req\_backend

##### Data Fields

MPI_Request	data_request	
starpu_thread_mutex_t	req_mutex	
starpu_thread_cond_t	req_cond	
starpu_thread_cond_t	posted_cond	
<a href="#">struct_starpu_mpi_req *</a>	other_request	In the case of a Wait/Test request, we are going to post a request to test the completion of another request
MPI_Request	size_req	
<a href="#">struct_starpu_mpi_envelope *</a>	envelope	
unsigned	is_internal_req:1	
unsigned	to_destroy:1	
<a href="#">struct_starpu_mpi_req *</a>	internal_req	
<a href="#">struct_starpu_mpi_early_data_handle *</a>	early_data_handle	
<a href="#">UT_hash_handle</a>	hh	
nm_gate_t	gate	
nm_session_t	session	
nm_sr_request_t	data_request	
int	waited	
piom_cond_t	req_cond	
nm_sr_request_t	size_req	

## 7.5 starpu\_mpi\_stats.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
```

### Functions

- void [starpu\\_mpi\\_comm\\_amounts\\_init](#) (MPI\_Comm comm)
- void [starpu\\_mpi\\_comm\\_amounts\\_shutdown](#) ()
- void [starpu\\_mpi\\_comm\\_amounts\\_inc](#) (MPI\_Comm comm, unsigned dst, MPI\_Datatype datatype, int count)
- void [starpu\\_mpi\\_comm\\_amounts\\_display](#) (FILE \*stream, int node)

## 7.6 starpu\_mpi\_cache\_stats.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

### Macros

- `#define _starpu_mpi_cache_stats_inc(dst, data_handle)`
- `#define _starpu_mpi_cache_stats_dec(dst, data_handle)`

### Functions

- `void _starpu_mpi_cache_stats_init ()`
- `void _starpu_mpi_cache_stats_shutdown ()`
- `void _starpu_mpi_cache_stats_update (unsigned dst, starpu_data_handle_t data_handle, int count)`

## 7.7 starpu\_mpi\_early\_data.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
#include <common/uthash.h>
#include <starpu_mpi_private.h>
```

### Data Structures

- `struct _starpu_mpi_early_data_handle`
- `struct _starpu_mpi_early_data_handle_tag_hashlist`

### Functions

- `void _starpu_mpi_early_data_init (void)`
- `void _starpu_mpi_early_data_check_termination (void)`
- `void _starpu_mpi_early_data_shutdown (void)`
- `struct _starpu_mpi_early_data_handle * _starpu_mpi_early_data_create (struct _starpu_mpi_envelope *envelope, int source, MPI_Comm comm) STARPU_ATTRIBUTE_MALLOC`
- `struct _starpu_mpi_early_data_handle * _starpu_mpi_early_data_find (struct _starpu_mpi_node_tag *node_tag)`
- `void _starpu_mpi_early_data_add (struct _starpu_mpi_early_data_handle *early_data_handle)`
- `struct _starpu_mpi_early_data_handle_tag_hashlist * _starpu_mpi_early_data_extract (struct _starpu_mpi_node_tag *node_tag)`

### 7.7.1 Data Structure Documentation

#### 7.7.1.1 struct \_starpu\_mpi\_early\_data\_handle

##### Data Fields

<code>starpu_data_handle_t</code>	<code>handle</code>	
<code>struct _starpu_mpi_req *</code>	<code>req</code>	
<code>void *</code>	<code>buffer</code>	

## Data Fields

size_t	size	
<a href="#">struct_starpu_mpi_node_tag</a>	node_tag	
starpu_pthread_mutex_t	req_mutex	
starpu_pthread_cond_t	req_cond	

## 7.7.1.2 struct\_starpu\_mpi\_early\_data\_handle\_tag\_hashlist

## Data Fields

<a href="#">struct_starpu_mpi_early_data_handle_list</a>	list	
<a href="#">UT_hash_handle</a>	hh	
starpu_mpi_tag_t	data_tag	

## 7.8 starpu\_mpi\_sync\_data.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

## Functions

- void [\\_starpu\\_mpi\\_sync\\_data\\_init](#) (void)
- void [\\_starpu\\_mpi\\_sync\\_data\\_check\\_termination](#) (void)
- void [\\_starpu\\_mpi\\_sync\\_data\\_shutdown](#) (void)
- [struct\\_starpu\\_mpi\\_req](#) \* [\\_starpu\\_mpi\\_sync\\_data\\_find](#) (starpu\_mpi\_tag\_t data\_tag, int source, MPI\_Comm comm)
- void [\\_starpu\\_mpi\\_sync\\_data\\_add](#) ([struct\\_starpu\\_mpi\\_req](#) \*req)
- int [\\_starpu\\_mpi\\_sync\\_data\\_count](#) ()

## 7.9 starpu\_mpi\_comm.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <mpi/starpu_mpi_mpi_backend.h>
```

## Functions

- void [\\_starpu\\_mpi\\_comm\\_init](#) (MPI\_Comm comm)
- void [\\_starpu\\_mpi\\_comm\\_shutdown](#) ()
- void [\\_starpu\\_mpi\\_comm\\_register](#) (MPI\_Comm comm)
- void [\\_starpu\\_mpi\\_comm\\_post\\_recv](#) ()
- int [\\_starpu\\_mpi\\_comm\\_test\\_recv](#) (MPI\_Status \*status, [struct\\_starpu\\_mpi\\_envelope](#) \*\*envelope, MPI\_Comm \*comm)
- void [\\_starpu\\_mpi\\_comm\\_cancel\\_recv](#) ()

## 7.10 starpu\_mpi\_early\_request.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

### Data Structures

- [struct \\_starpu\\_mpi\\_early\\_request\\_tag\\_hashlist](#)

### Functions

- `void _starpu_mpi_early_request_init (void)`
- `void _starpu_mpi_early_request_shutdown (void)`
- `int _starpu_mpi_early_request_count (void)`
- `void _starpu_mpi_early_request_check_termination (void)`
- `void _starpu_mpi_early_request_enqueue (struct \_starpu\_mpi\_req *req)`
- `struct \_starpu\_mpi\_req * _starpu_mpi_early_request_dequeue (starpu_mpi_tag_t data_tag, int source, MPI_Comm comm)`
- `struct \_starpu\_mpi\_early\_request\_tag\_hashlist * _starpu_mpi_early_request_extract (starpu_mpi_tag_t data_tag, int source, MPI_Comm comm)`

#### 7.10.1 Data Structure Documentation

##### 7.10.1.1 [struct \\_starpu\\_mpi\\_early\\_request\\_tag\\_hashlist](#)

###### Data Fields

<a href="#">struct _starpu_mpi_req_list</a>	list	
<a href="#">UT_hash_handle</a>	hh	
starpu_mpi_tag_t	data_tag	

## 7.11 starpu\_mpi\_mpi\_backend.h File Reference

```
#include <common/config.h>
#include <common/uthash.h>
```

### Data Structures

- [struct \\_starpu\\_mpi\\_envelope](#)
- [struct \\_starpu\\_mpi\\_req\\_backend](#)

### Macros

- `#define _STARPU_MPI_TAG_ENVELOPE`
- `#define _STARPU_MPI_TAG_DATA`
- `#define _STARPU_MPI_TAG_SYNC_DATA`

### Enumerations

- `enum _starpu_envelope_mode { _STARPU_MPI_ENVELOPE_DATA , _STARPU_MPI_ENVELOPE_SYNC_READY }`

## Variables

- `int _starpu_mpi_tag`

### 7.11.1 Data Structure Documentation

#### 7.11.1.1 `struct _starpu_mpi_envelope`

##### Data Fields

<code>enum _starpu_envelope_mode</code>	<code>mode</code>	
<code>starpu_ssize_t</code>	<code>size</code>	
<code>starpu_mpi_tag_t</code>	<code>data_tag</code>	
<code>unsigned</code>	<code>sync</code>	

#### 7.11.1.2 `struct _starpu_mpi_req_backend`

##### Data Fields

<code>MPI_Request</code>	<code>data_request</code>	
<code>starpu_pthread_mutex_t</code>	<code>req_mutex</code>	
<code>starpu_pthread_cond_t</code>	<code>req_cond</code>	
<code>starpu_pthread_cond_t</code>	<code>posted_cond</code>	
<code>struct _starpu_mpi_req *</code>	<code>other_request</code>	In the case of a Wait/Test request, we are going to post a request to test the completion of another request
<code>MPI_Request</code>	<code>size_req</code>	
<code>struct _starpu_mpi_envelope *</code>	<code>envelope</code>	
<code>unsigned</code>	<code>is_internal_req:1</code>	
<code>unsigned</code>	<code>to_destroy:1</code>	
<code>struct _starpu_mpi_req *</code>	<code>internal_req</code>	
<code>struct _starpu_mpi_early_data_handle *</code>	<code>early_data_handle</code>	
<code>UT_hash_handle</code>	<code>hh</code>	
<code>nm_gate_t</code>	<code>gate</code>	
<code>nm_session_t</code>	<code>session</code>	
<code>nm_sr_request_t</code>	<code>data_request</code>	
<code>int</code>	<code>waited</code>	
<code>piom_cond_t</code>	<code>req_cond</code>	
<code>nm_sr_request_t</code>	<code>size_req</code>	

## 7.12 `starpu_mpi_private.h` File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/uthash.h>
#include <starpu_mpi.h>
#include <starpu_mpi_fxt.h>
#include <common/list.h>
#include <common/prio_list.h>
#include <common/starpu_spinlock.h>
#include <core/simgrid.h>
```

## Data Structures

- struct [\\_starpu\\_simgrid\\_mpi\\_req](#)
- struct [\\_starpu\\_mpi\\_node](#)
- struct [\\_starpu\\_mpi\\_node\\_tag](#)
- struct [\\_starpu\\_mpi\\_coop\\_sends](#)
- struct [\\_starpu\\_mpi\\_data](#)
- struct [\\_starpu\\_mpi\\_req](#)
- struct [\\_starpu\\_mpi\\_argc\\_argv](#)
- struct [\\_starpu\\_mpi\\_backend](#)

## Macros

- #define **STARPU\_MPI\_ASSERT\_MSG**(x, msg, ...)
- #define **\_STARPU\_MPI\_MALLOC**(ptr, size)
- #define **\_STARPU\_MPI\_CALLOC**(ptr, nmemb, size)
- #define **\_STARPU\_MPI\_REALLOC**(ptr, size)
- #define **\_STARPU\_MPI\_COMM\_DEBUG**(ptr, count, datatype, node, tag, utag, comm, way)
- #define **\_STARPU\_MPI\_COMM\_TO\_DEBUG**(ptr, count, datatype, dest, tag, utag, comm)
- #define **\_STARPU\_MPI\_COMM\_FROM\_DEBUG**(ptr, count, datatype, source, tag, utag, comm)
- #define **\_STARPU\_MPI\_DEBUG**(level, fmt, ...)
- #define **\_STARPU\_MPI\_DISP**(fmt, ...)
- #define **\_STARPU\_MPI\_MSG**(fmt, ...)
- #define **\_STARPU\_MPI\_LOG\_IN**()
- #define **\_STARPU\_MPI\_LOG\_OUT**()

## Enumerations

- enum **\_starpu\_mpi\_request\_type** {  
**SEND\_REQ** , **RECV\_REQ** , **WAIT\_REQ** , **TEST\_REQ** ,  
**BARRIER\_REQ** , **PROBE\_REQ** , **UNKNOWN\_REQ** }

## Functions

- int **\_starpu\_mpi\_simgrid\_mpi\_test** (unsigned \*done, int \*flag)
- void **\_starpu\_mpi\_simgrid\_wait\_req** (MPI\_Request \*request, MPI\_Status \*status, starpu\_thread\_queue\_t \*queue, unsigned \*done)
- char \* **\_starpu\_mpi\_get\_mpi\_error\_code** (int code)
- void **\_starpu\_mpi\_env\_init** (void)
- [struct \\_starpu\\_mpi\\_data](#) \* **\_starpu\_mpi\_data\_get** (starpu\_data\_handle\_t data\_handle)
- void **\_starpu\_mpi\_submit\_ready\_request** (void \*arg)
- void **\_starpu\_mpi\_release\_req\_data** ([struct \\_starpu\\_mpi\\_req](#) \*req)
- void **\_starpu\_mpi\_coop\_sends\_build\_tree** ([struct \\_starpu\\_mpi\\_coop\\_sends](#) \*coop\_sends)
- void **\_starpu\_mpi\_coop\_send** (starpu\_data\_handle\_t data\_handle, [struct \\_starpu\\_mpi\\_req](#) \*req, enum starpu\_data\_access\_mode mode, int sequential\_consistency)
- void **\_starpu\_mpi\_submit\_coop\_sends** ([struct \\_starpu\\_mpi\\_coop\\_sends](#) \*coop\_sends, int submit\_control, int submit\_data)
- void **\_starpu\_mpi\_submit\_ready\_request\_inc** ([struct \\_starpu\\_mpi\\_req](#) \*req)
- void **\_starpu\_mpi\_request\_init** ([struct \\_starpu\\_mpi\\_req](#) \*\*req)
- [struct \\_starpu\\_mpi\\_req](#) \* **\_starpu\_mpi\_request\_fill** (starpu\_data\_handle\_t data\_handle, int srcdst, starpu\_mpi\_tag\_t data\_tag, MPI\_Comm comm, unsigned detached, unsigned sync, int prio, void(\*callback)(void \*), void \*arg, enum \_starpu\_mpi\_request\_type request\_type, void(\*func)([struct \\_starpu\\_mpi\\_req](#) \*), int sequential\_consistency, int is\_internal\_req, starpu\_ssize\_t count)
- void **\_starpu\_mpi\_request\_destroy** ([struct \\_starpu\\_mpi\\_req](#) \*req)
- void **\_starpu\_mpi\_isend\_size\_func** ([struct \\_starpu\\_mpi\\_req](#) \*req)
- void **\_starpu\_mpi\_irecv\_size\_func** ([struct \\_starpu\\_mpi\\_req](#) \*req)

- `int _starpu_mpi_wait (starpu_mpi_req *public_req, MPI_Status *status)`
- `int _starpu_mpi_test (starpu_mpi_req *public_req, int *flag, MPI_Status *status)`
- `int _starpu_mpi_barrier (MPI_Comm comm)`
- `void _starpu_mpi_progress_shutdown (void **value)`
- `int _starpu_mpi_progress_init (struct _starpu_mpi_argc_argv *argc_argv)`
- `void _starpu_mpi_wait_for_initialization ()`
- `void _starpu_mpi_data_flush (starpu_data_handle_t data_handle)`

## Variables

- `starpu_pthread_wait_t _starpu_mpi_thread_wait`
- `starpu_pthread_queue_t _starpu_mpi_thread_dontsleep`
- `int _starpu_debug_rank`
- `int _starpu_mpi_comm_debug`
- `int _starpu_mpi_fake_world_size`
- `int _starpu_mpi_fake_world_rank`
- `int _starpu_mpi_use_prio`
- `int _starpu_mpi_nobind`
- `int _starpu_mpi_thread_cpuid`
- `int _starpu_mpi_use_coop_sends`
- `PRIO_struct _starpu_mpi_req`
- `struct _starpu_mpi_backend _mpi_backend`

## 7.12.1 Data Structure Documentation

### 7.12.1.1 struct \_starpu\_simgrid\_mpi\_req

#### Data Fields

MPI_Request *	request	
MPI_Status *	status	
starpu_pthread_queue_t *	queue	
unsigned *	done	

### 7.12.1.2 struct \_starpu\_mpi\_node

#### Data Fields

MPI_Comm	comm	
int	rank	

### 7.12.1.3 struct \_starpu\_mpi\_node\_tag

#### Data Fields

<a href="#">struct _starpu_mpi_node</a>	node	
starpu_mpi_tag_t	data_tag	

### 7.12.1.4 struct \_starpu\_mpi\_coop\_sends

#### Data Fields

<a href="#">struct _starpu_mpi_req_multilist_coop_sends</a>	reqs	
---	------	--

## Data Fields

<a href="#">struct_starpu_mpi_data *</a>	mpi_data	
<a href="#">struct_starpu_spinlock</a>	lock	
<a href="#">struct_starpu_mpi_req **</a>	reqs_array	
unsigned	n	
unsigned	redirects_sent	

## 7.12.1.5 struct\_starpu\_mpi\_data

## Data Fields

int	magic	
<a href="#">struct_starpu_mpi_node_tag</a>	node_tag	
char *	cache_sent	
int	cache_received	
<a href="#">struct_starpu_spinlock</a>	coop_lock	
<a href="#">struct_starpu_mpi_coop_sends *</a>	coop_sends	

## 7.12.1.6 struct\_starpu\_mpi\_argc\_argv

## Data Fields

int	initialize_mpi	
int *	argc	
char ***	argv	
MPI_Comm	comm	
int	fargc	Fortran argc
char **	fargv	Fortran argv
int	rank	
int	world_size	

## 7.13 starpu\_mpi\_tag.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

## Functions

- void **starpu\_mpi\_tag\_init** (void)
- void **starpu\_mpi\_tag\_shutdown** (void)
- void **starpu\_mpi\_tag\_data\_register** (starpu\_data\_handle\_t handle, starpu\_mpi\_tag\_t data\_tag)
- int **starpu\_mpi\_tag\_data\_release** (starpu\_data\_handle\_t handle)
- starpu\_data\_handle\_t **starpu\_mpi\_tag\_get\_data\_handle\_from\_tag** (starpu\_mpi\_tag\_t data\_tag)

## 7.14 starpu\_mpi\_datatype.h File Reference

```
#include <starpu_mpi.h>
#include <starpu_mpi_private.h>
```



## Functions

- void `_starpu_mpi_datatype_init` (void)
- void `_starpu_mpi_datatype_shutdown` (void)
- void `_starpu_mpi_datatype_allocate` (starpu\_data\_handle\_t data\_handle, [struct \\_starpu\\_mpi\\_req](#) \*req)
- void `_starpu_mpi_datatype_free` (starpu\_data\_handle\_t data\_handle, MPI\_Datatype \*datatype)
- MPI\_Datatype `_starpu_mpi_datatype_get_user_defined_datatype` (starpu\_data\_handle\_t data\_handle)

## 7.15 starpu\_mpi\_fxt.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/fxt.h>
```

## Macros

- #define `_STARPU_MPI_FUT_START`
- #define `_STARPU_MPI_FUT_STOP`
- #define `_STARPU_MPI_FUT_BARRIER`
- #define `_STARPU_MPI_FUT_ISEND_SUBMIT_BEGIN`
- #define `_STARPU_MPI_FUT_ISEND_SUBMIT_END`
- #define `_STARPU_MPI_FUT_IRecv_SUBMIT_BEGIN`
- #define `_STARPU_MPI_FUT_IRecv_SUBMIT_END`
- #define `_STARPU_MPI_FUT_ISEND_COMPLETE_BEGIN`
- #define `_STARPU_MPI_FUT_ISEND_COMPLETE_END`
- #define `_STARPU_MPI_FUT_DATA_SET_RANK`
- #define `_STARPU_MPI_FUT_IRecv_TERMINATED`
- #define `_STARPU_MPI_FUT_ISEND_TERMINATED`
- #define `_STARPU_MPI_FUT_TESTING_DETACHED_BEGIN`
- #define `_STARPU_MPI_FUT_TESTING_DETACHED_END`
- #define `_STARPU_MPI_FUT_TEST_BEGIN`
- #define `_STARPU_MPI_FUT_TEST_END`
- #define `_STARPU_MPI_FUT_IRecv_COMPLETE_BEGIN`
- #define `_STARPU_MPI_FUT_IRecv_COMPLETE_END`
- #define `_STARPU_MPI_FUT_SLEEP_BEGIN`
- #define `_STARPU_MPI_FUT_SLEEP_END`
- #define `_STARPU_MPI_FUT_DTESTING_BEGIN`
- #define `_STARPU_MPI_FUT_DTESTING_END`
- #define `_STARPU_MPI_FUT_UTESTING_BEGIN`
- #define `_STARPU_MPI_FUT_UTESTING_END`
- #define `_STARPU_MPI_FUT_UWAIT_BEGIN`
- #define `_STARPU_MPI_FUT_UWAIT_END`
- #define `_STARPU_MPI_FUT_POLLING_BEGIN`
- #define `_STARPU_MPI_FUT_POLLING_END`
- #define `_STARPU_MPI_FUT_DRIVER_RUN_BEGIN`
- #define `_STARPU_MPI_FUT_DRIVER_RUN_END`
- #define `_STARPU_MPI_FUT_DATA_SET_TAG`
- #define `_STARPU_MPI_TRACE_START`(a, b)
- #define `_STARPU_MPI_TRACE_STOP`(a, b)
- #define `_STARPU_MPI_TRACE_BARRIER`(a, b, c)
- #define `_STARPU_MPI_TRACE_ISEND_SUBMIT_BEGIN`(a, b, c)
- #define `_STARPU_MPI_TRACE_ISEND_SUBMIT_END`(a, b, c, d)
- #define `_STARPU_MPI_TRACE_IRecv_SUBMIT_BEGIN`(a, b)
- #define `_STARPU_MPI_TRACE_IRecv_SUBMIT_END`(a, b)

- `#define _STARPU_MPI_TRACE_ISEND_COMPLETE_BEGIN(a, b, c)`
- `#define _STARPU_MPI_TRACE_COMPLETE_BEGIN(a, b, c)`
- `#define _STARPU_MPI_TRACE_COMPLETE_END(a, b, c)`
- `#define _STARPU_MPI_TRACE_TERMINATED(a, b, c)`
- `#define _STARPU_MPI_TRACE_ISEND_COMPLETE_END(a, b, c)`
- `#define _STARPU_MPI_TRACE_IRecv_COMPLETE_BEGIN(a, b)`
- `#define _STARPU_MPI_TRACE_IRecv_COMPLETE_END(a, b)`
- `#define _STARPU_MPI_TRACE_SLEEP_BEGIN()`
- `#define _STARPU_MPI_TRACE_SLEEP_END()`
- `#define _STARPU_MPI_TRACE_DTESTING_BEGIN()`
- `#define _STARPU_MPI_TRACE_DTESTING_END()`
- `#define _STARPU_MPI_TRACE_UTEStING_BEGIN(a, b)`
- `#define _STARPU_MPI_TRACE_UTEStING_END(a, b)`
- `#define _STARPU_MPI_TRACE_UWAIT_BEGIN(a, b)`
- `#define _STARPU_MPI_TRACE_UWAIT_END(a, b)`
- `#define _STARPU_MPI_TRACE_DATA_SET_RANK(a, b)`
- `#define _STARPU_MPI_TRACE_DATA_SET_TAG(a, b)`
- `#define _STARPU_MPI_TRACE_TESTING_DETACHED_BEGIN()`
- `#define _STARPU_MPI_TRACE_TESTING_DETACHED_END()`
- `#define _STARPU_MPI_TRACE_TEST_BEGIN(peer, data_tag)`
- `#define _STARPU_MPI_TRACE_TEST_END(peer, data_tag)`
- `#define _STARPU_MPI_TRACE_POLLING_BEGIN()`
- `#define _STARPU_MPI_TRACE_POLLING_END()`
- `#define _STARPU_MPI_TRACE_DRIVER_RUN_BEGIN()`
- `#define _STARPU_MPI_TRACE_DRIVER_RUN_END()`

## 7.16 starpu\_mpi\_select\_node.h File Reference

```
#include <mpi.h>
```

### Macros

- `#define _STARPU_MPI_NODE_SELECTION_MAX_POLICY`

### Functions

- `void _starpu_mpi_select_node_init ()`
- `int _starpu_mpi_select_node (int me, int nb_nodes, struct starpu_data_descr *descr, int nb_data, int policy)`

## 7.17 starpu\_mpi\_task\_insert.h File Reference

### Functions

- `int _starpu_mpi_find_executee_node (starpu_data_handle_t data, enum starpu_data_access_mode mode, int me, int *do_execute, int *inconsistent_execute, int *xrank)`
- `void _starpu_mpi_exchange_data_before_execution (starpu_data_handle_t data, enum starpu_data_access_mode mode, int me, int xrank, int do_execute, int prio, MPI_Comm comm)`
- `int _starpu_mpi_task_postbuild_v (MPI_Comm comm, int xrank, int do_execute, struct starpu_data_descr *descrs, int nb_data, int prio)`

## 7.18 load\_balancer\_policy.h File Reference

```
#include <starpu_mpi_lb.h>
```

## Data Structures

- struct [load\\_balancer\\_policy](#)

## Variables

- [struct load\\_balancer\\_policy](#) `load_heat_propagation_policy`

## 7.19 load\_data\_interface.h File Reference

```
#include <starpu.h>
```

## Data Structures

- struct [load\\_data\\_interface](#)

## Macros

- `#define LOAD_DATA_GET_NSUBMITTED_TASKS(interface)`
- `#define LOAD_DATA_GET_SLEEP_THRESHOLD(interface)`
- `#define LOAD_DATA_GET_WAKEUP_THRESHOLD(interface)`

## Functions

- void **load\_data\_data\_register** (starpu\_data\_handle\_t \*handle, unsigned home\_node, int sleep\_task\_threshold, double wakeup\_ratio)
- int **load\_data\_get\_sleep\_threshold** (starpu\_data\_handle\_t handle)
- int **load\_data\_get\_wakeup\_threshold** (starpu\_data\_handle\_t handle)
- int **load\_data\_get\_current\_phase** (starpu\_data\_handle\_t handle)
- int **load\_data\_get\_nsubmitted\_tasks** (starpu\_data\_handle\_t handle)
- int **load\_data\_get\_nfinished\_tasks** (starpu\_data\_handle\_t handle)
- int **load\_data\_inc\_nsubmitted\_tasks** (starpu\_data\_handle\_t handle)
- int **load\_data\_inc\_nfinished\_tasks** (starpu\_data\_handle\_t handle)
- int **load\_data\_next\_phase** (starpu\_data\_handle\_t handle)
- int **load\_data\_update\_elapsed\_time** (starpu\_data\_handle\_t handle)
- double **load\_data\_get\_elapsed\_time** (starpu\_data\_handle\_t handle)
- int **load\_data\_update\_wakeup\_cond** (starpu\_data\_handle\_t handle)
- int **load\_data\_wakeup\_cond** (starpu\_data\_handle\_t handle)

### 7.19.1 Data Structure Documentation

#### 7.19.1.1 struct load\_data\_interface

interface for load\_data

#### Data Fields

double	start	Starting time of the execution
double	elapsed_time	Elapsed time until the start time and the time when event "launch a load balancing phase" is triggered
int	phase	Current submission phase, i.e how many balanced steps have already happened so far.
int	nsubmitted_tasks	Number of currently submitted tasks
int	nfinished_tasks	Number of currently finished tasks
int	sleep_task_threshold	Task threshold to sleep the submission thread

## Data Fields

int	wakeup_task_threshold	Task threshold to wake-up the submission thread
double	wakeup_ratio	Ratio of submitted tasks to wait for completion before waking up the submission thread

## 7.20 data\_movements\_interface.h File Reference

```
#include <starpu.h>
```

### Data Structures

- struct [data\\_movements\\_interface](#)

### Macros

- #define **DATA\_MOVEMENTS\_GET\_SIZE\_TABLES**(interface)
- #define **DATA\_MOVEMENTS\_GET\_TAGS\_TABLE**(interface)
- #define **DATA\_MOVEMENTS\_GET\_RANKS\_TABLE**(interface)

### Functions

- void **data\_movements\_data\_register** (starpu\_data\_handle\_t \*handle, unsigned home\_node, int \*ranks, starpu\_mpi\_tag\_t \*tags, int size)
- starpu\_mpi\_tag\_t \*\* **data\_movements\_get\_ref\_tags\_table** (starpu\_data\_handle\_t handle)
- int \*\* **data\_movements\_get\_ref\_ranks\_table** (starpu\_data\_handle\_t handle)
- int **data\_movements\_reallocate\_tables** (starpu\_data\_handle\_t handle, int size)
- starpu\_mpi\_tag\_t \* **data\_movements\_get\_tags\_table** (starpu\_data\_handle\_t handle)
- int \* **data\_movements\_get\_ranks\_table** (starpu\_data\_handle\_t handle)
- int **data\_movements\_get\_size\_tables** (starpu\_data\_handle\_t handle)

### 7.20.1 Data Structure Documentation

#### 7.20.1.1 struct data\_movements\_interface

interface for data\_movements

## Data Fields

starpu_mpi_tag_t *	tags	Data tags table
int *	ranks	Ranks table (where to move the corresponding data)
int	size	Size of the tables



## Chapter 8

# StarPU Resource Manager File Documentation

### 8.1 starpurm\_private.h File Reference

#### Data Structures

- struct [s\\_starpurm](#)

#### Enumerations

- enum **e\_state** { **state\_uninitialized** , **state\_init** }
- enum **e\_starpurm\_unit\_type** { **starpurm\_unit\_cpu** , **starpurm\_unit\_opencl** , **starpurm\_unit\_cuda** , **starpurm\_unit\_mic** , **starpurm\_unit\_ntypes** }

#### 8.1.1 Data Structure Documentation

##### 8.1.1.1 struct s\_starpurm

#### Data Fields

hwloc_topology_t	topology	Machine topology as detected by hwloc.
unsigned	max_ncpus	Current upper bound on the number of CPU cores selectable for computing with the runtime system.
unsigned	selected_ncpus	Number of currently selected CPU workers
unsigned	selected_nworkers	Number of currently selected workers (CPU+devices)
int	state	Initialization state of the RM instance.
int	dynamic_resource_sharing	Boolean indicating the state of the dynamic resource sharing layer. !0 indicates that dynamic resource sharing is enabled. 0 indicates that dynamic resource sharing is disabled.
unsigned	sched_ctx_id	Id of the StarPU's sched_ctx used by the RM instance.
int	unit_ntypes	Number of unit types supported by this RM instance.
int *	nunits_by_type	Number of unitss available for each type.
int	nunits	Number of units.
int *	unit_offsets_by_type	Offset of unit numbering for each type.
<a href="#">struct s_starpurm_unit *</a>	units	Array of units.

## Data Fields

hwloc_cpuset_t	global_cpuset	Cpuset of all the StarPU's workers (CPU+devices).
hwloc_cpuset_t	all_cpu_workers_cpuset	Cpuset of all StarPU CPU workers.
hwloc_cpuset_t	all_opengl_device_workers_cpuset	Cpuset of all StarPU OpenGL workers.
hwloc_cpuset_t	all_cuda_device_workers_cpuset	Cpuset of all StarPU CUDA workers.
hwloc_cpuset_t	all_mic_device_workers_cpuset	Cpuset of all StarPU MIC workers.
hwloc_cpuset_t	all_device_workers_cpuset	Cpuset of all StarPU device workers.
hwloc_cpuset_t	selected_cpuset	Cpuset of all selected workers (CPU+devices).
hwloc_cpuset_t	initially_owned_cpuset_mask	Cpuset mask of initially owned cpuset or full if not used.
int	max_worker_id	maximum value among worker ids
int *	worker_unit_ids	worker id to unit id table
unsigned int	max_temporary_ctxs	Temporary contexts accounting.
unsigned int	avail_temporary_ctxs	
pthread_mutex_t	temporary_ctxs_mutex	
pthread_cond_t	temporary_ctxs_cond	
int	starpup_in_pause	Global StarPU pause state
pthread_t	event_thread	Event list.
pthread_mutex_t	event_list_mutex	
pthread_cond_t	event_list_cond	
pthread_cond_t	event_processing_cond	
int	event_processing_enabled	
int	event_processing_ended	
<a href="#">struct</a> s_starpurm_event *	event_list_head	
<a href="#">struct</a> s_starpurm_event *	event_list_tail	