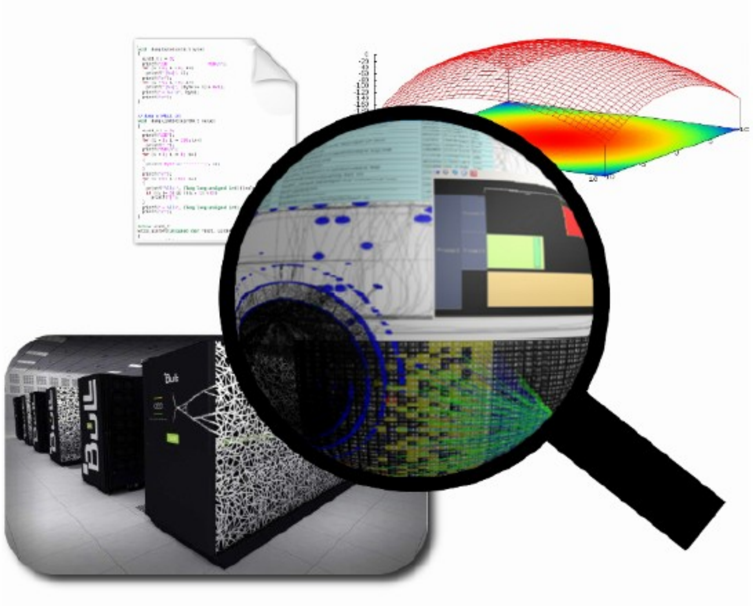
energie atomique • energies alternatives

MPC Trace Library



Jean-Baptiste BESNARD

PHD Student

CEA, DAM, DIF

F-91297 Arpajon, FRANCE

MPC Trace Library

Summary

- MPC Framework
- Fighting Complexity
- MPC Trace Library:
  - Overview
  - Architecture
  - Tools:
    - MPC Trace Debugger
    - MPC Trace Analyzer
  - Performance
- Conclusion/Perspectives

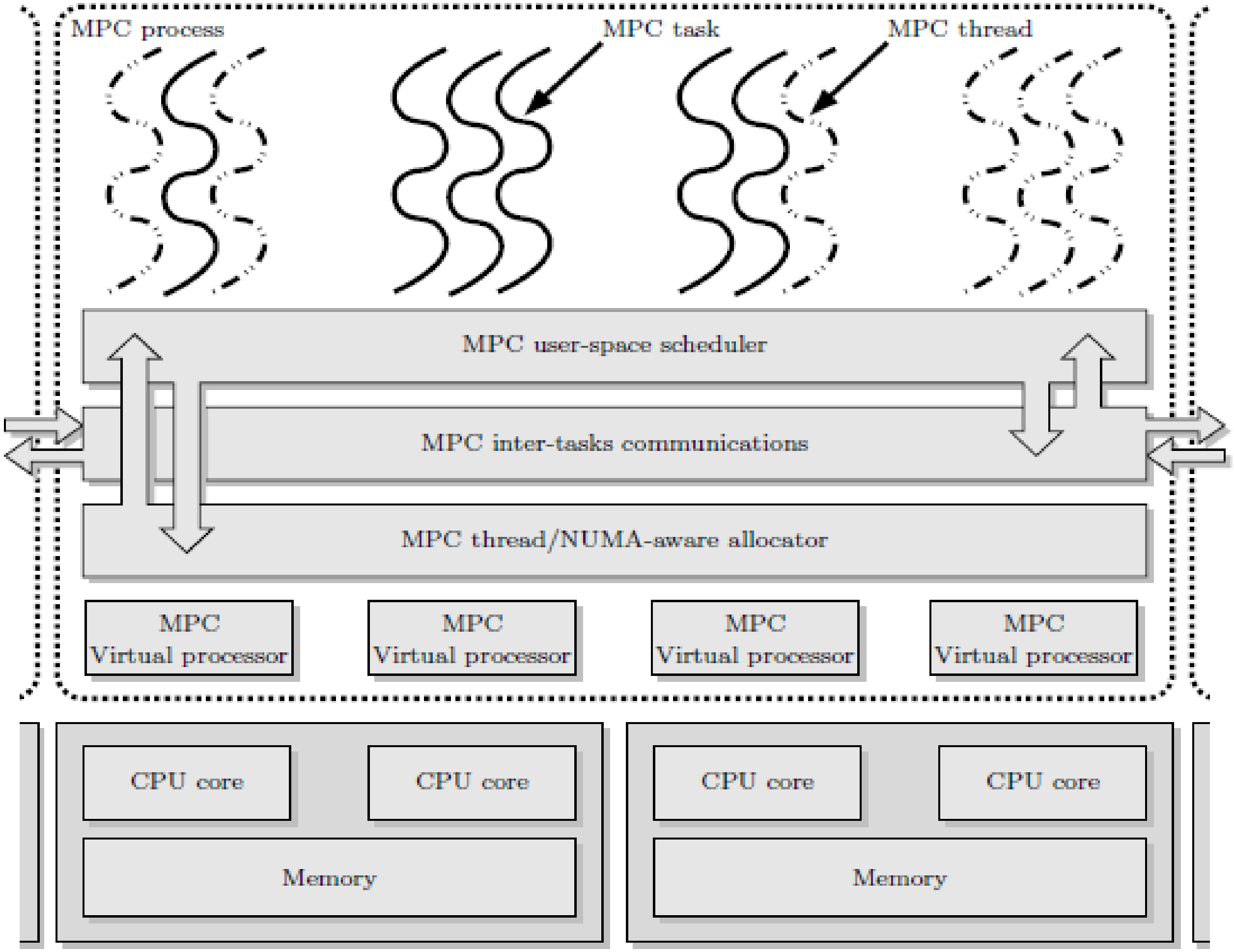
MPC Trace Library

MPC Framework

- Unified parallel runtime
  - MPI 1.3 fully supporting *MPI_THREAD_MULTIPLE*
    - Thread Based
    - Various interconnects *(IB, TCP, SHM)*
  - OpenMP 2.5
  - Tested on petaflopic range clusters (~100k cores)
  - Successfully ported to Intel MIC architecture
- The MPC Framework Provides …
  - User level *MxN* thread library and scheduler
  - Numa aware parallel Allocator
  - Patched GCC for OpenMP and automatic privatization
  - Patched *libthread_db* for GDB
  - Optimized support for Hierarchical Local Storage
  - […]
  - ***… and soon*** a parallel trace-based debugger and profiler

MPC Trace Library

MPC Framework



MPC Trace Library

MPC Framework

- The MPC framework involves 14 People working on ...
  - MPI Communication
  - Developer tools for debugging and optimization:
    - Static Analysis
    - Post-Mortem / Online
  - Programming models:
    - OpenMP
    - Accelerators
    - Code optimization
    - Upcoming models
  - Runtime services:
    - Parallel allocator
    - Hierarchical Local Storage
    - Threading model and scheduler
- Available at **mpc.sourceforge.net**

MPC Trace Library

energie atomique • energies alternatives

Fighting Complexity

“Intelligibility of complicatedness is obtained by simplification [...]

Intelligibility of complexness is obtained by modelization”

Jean-Louis le Moigne (translated)

in La Modélisation des systèmes complexes (p.10)

MPC Trace Library

Fighting Complexity

Explanation (disjunction)

“Intelligibility of complicatedness is obtained by simplification [...]

Intelligibility of complexness is obtained by modelization”

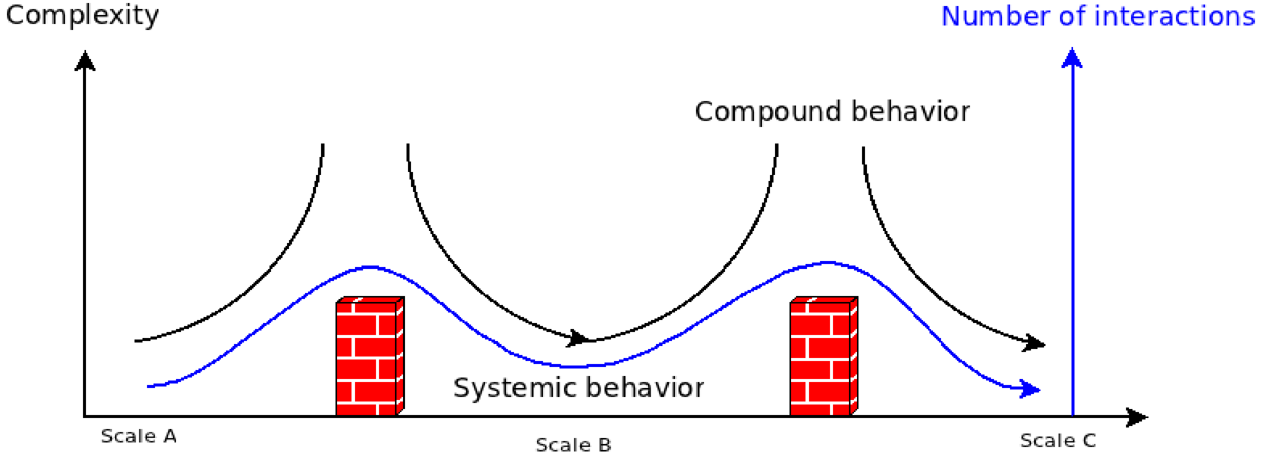
Jean-Louis le Moigne (translated)

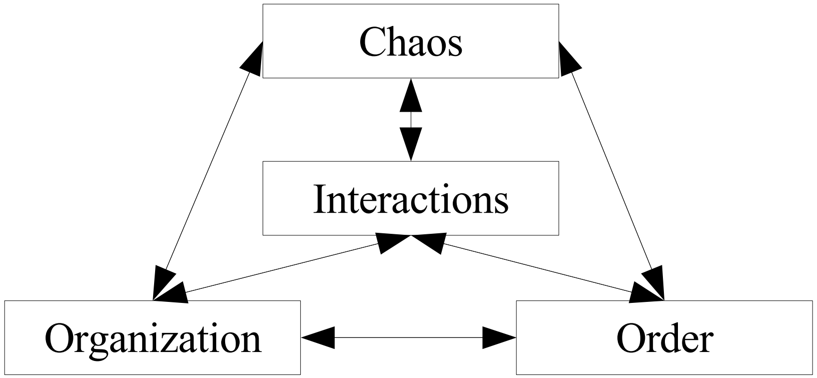
in La Modélisation des systèmes complexes (p.10)

Understanding (conjunction)

MPC Trace Library

Fighting Complexity




Chaos

Interactions

Organization

Order

The tetralogic loop

Edgar Morin (translated)
in La méthode Tome 1 (p.56)

MPC Trace Library

Fighting Complexity

- Still no unified representation of a parallel computation:
  - Handle every entities and interactions:
    - Combinatory → Large trace
  - Projection of behaviours on lower level rules:
    - A “point of view” problem
    - No abstraction → complexity
- Which organization can abstract the combination of programming models ?[...]??
  - For now we can only focus on what we know:
    - Project on individual existing models
    - Observe on the real substrate

→ **Optimization is then a trial and error process**

MPC Trace Library

energie atomique • energies alternatives

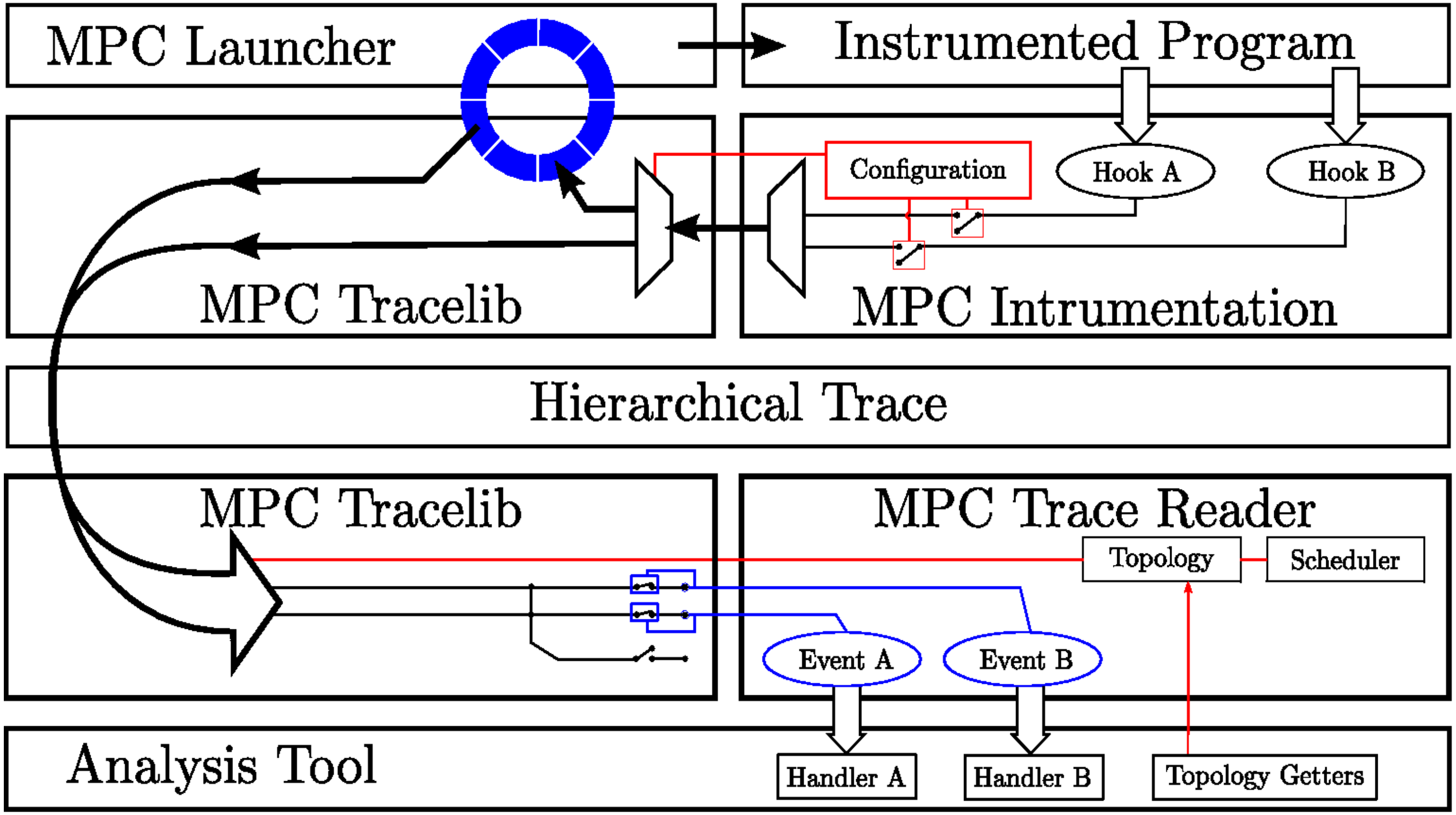
Overview

- Part of the MPC Framework
- Includes the whole instrumentation chain:
  - Instrumentation (MPI, libc, pthread)
    - MPC + MPI
  - Hierarchical trace format:
    - Simplifies meta-data handling
    - Event-centric
  - Parallel trace reader:
    - Compact interface
    - Abstracts parallelism and meta-data handling
  - Analysis tools:
    - All relying on the trace reader
- Intended for **both** debugging and profiling

MPC Trace Library

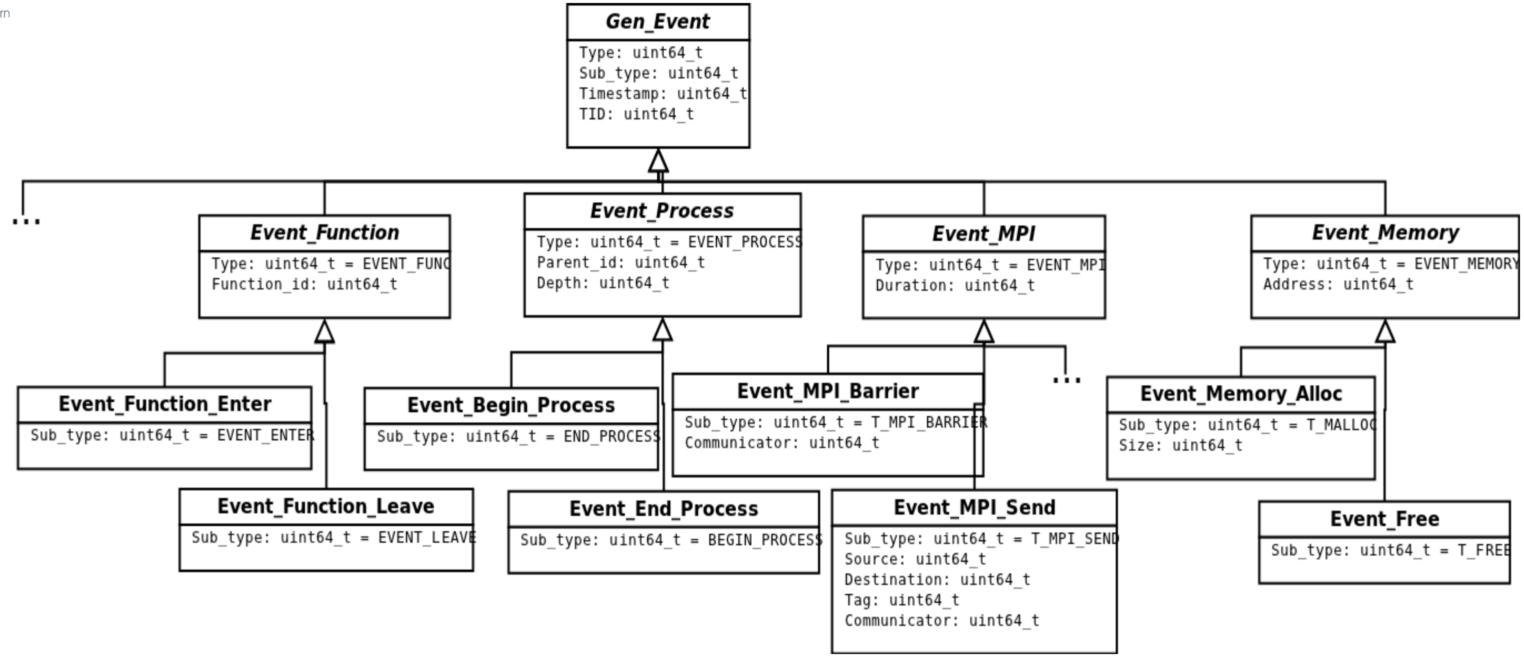
energie atomique • energies alternatives

Architecture



MPC Trace Library

Generic Event Representation



Gen_Event
Type: uint64_t
Sub_type: uint64_t
Timestamp: uint64_t
TID: uint64_t

...

Event_Function
Type: uint64_t = EVENT_FUNC
Function_id: uint64_t

Event_Process
Type: uint64_t = EVENT_PROCESS
Parent_id: uint64_t
Depth: uint64_t

Event_MPI
Type: uint64_t = EVENT_MPI
Duration: uint64_t

Event_Memory
Type: uint64_t = EVENT_MEMORY
Address: uint64_t

Event_Function_Enter
Sub_type: uint64_t = EVENT_ENTER

Event_Function_Leave
Sub_type: uint64_t = EVENT_LEAVE

Event_Begin_Process
Sub_type: uint64_t = END_PROCESS

Event_End_Process
Sub_type: uint64_t = BEGIN_PROCESS

Event_MPI_Barrier
Sub_type: uint64_t = T_MPI_BARRIER
Communicator: uint64_t

...

Event_MPI_Send
Sub_type: uint64_t = T_MPI_SEND
Source: uint64_t
Destination: uint64_t
Tag: uint64_t
Communicator: uint64_t

Event_Memory_Alloc
Sub_type: uint64_t = T_MALLOC
Size: uint64_t

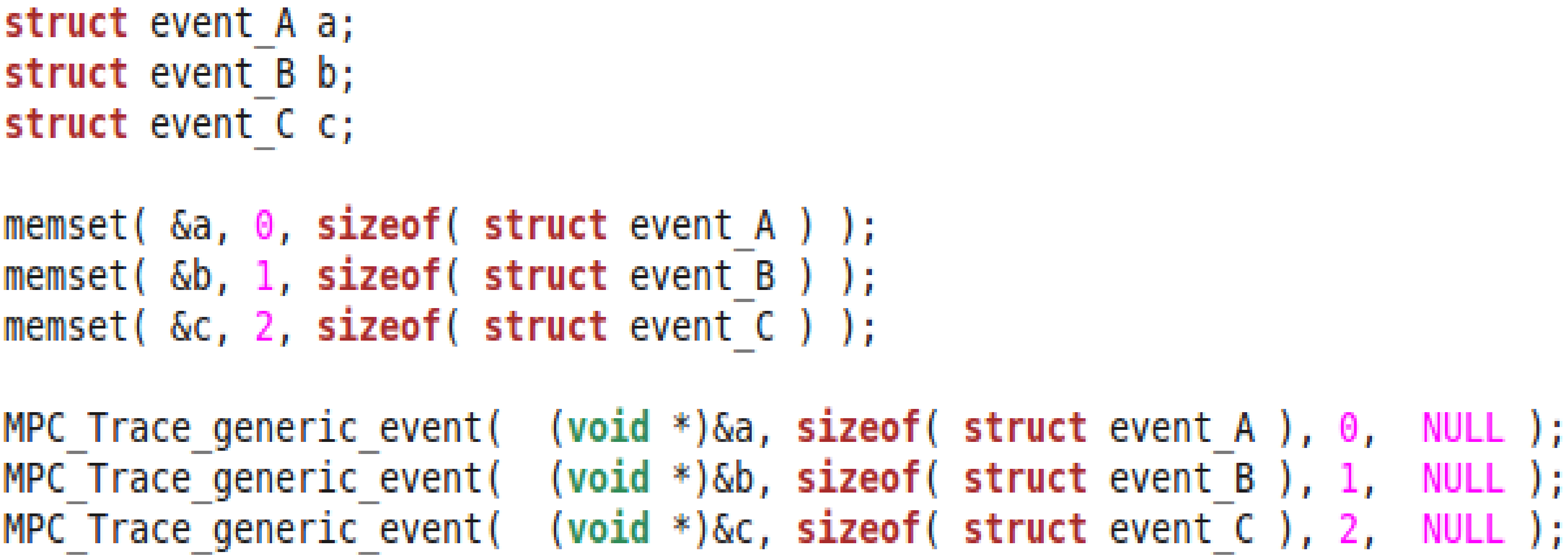
Event_Free
Sub_type: uint64_t = T_FREE

Allows a compact event submission interface:

void Submit_Event(**struct Gen_Event_t** *event, **struct m_Trace_Module** *m_module)

MPC Trace Library

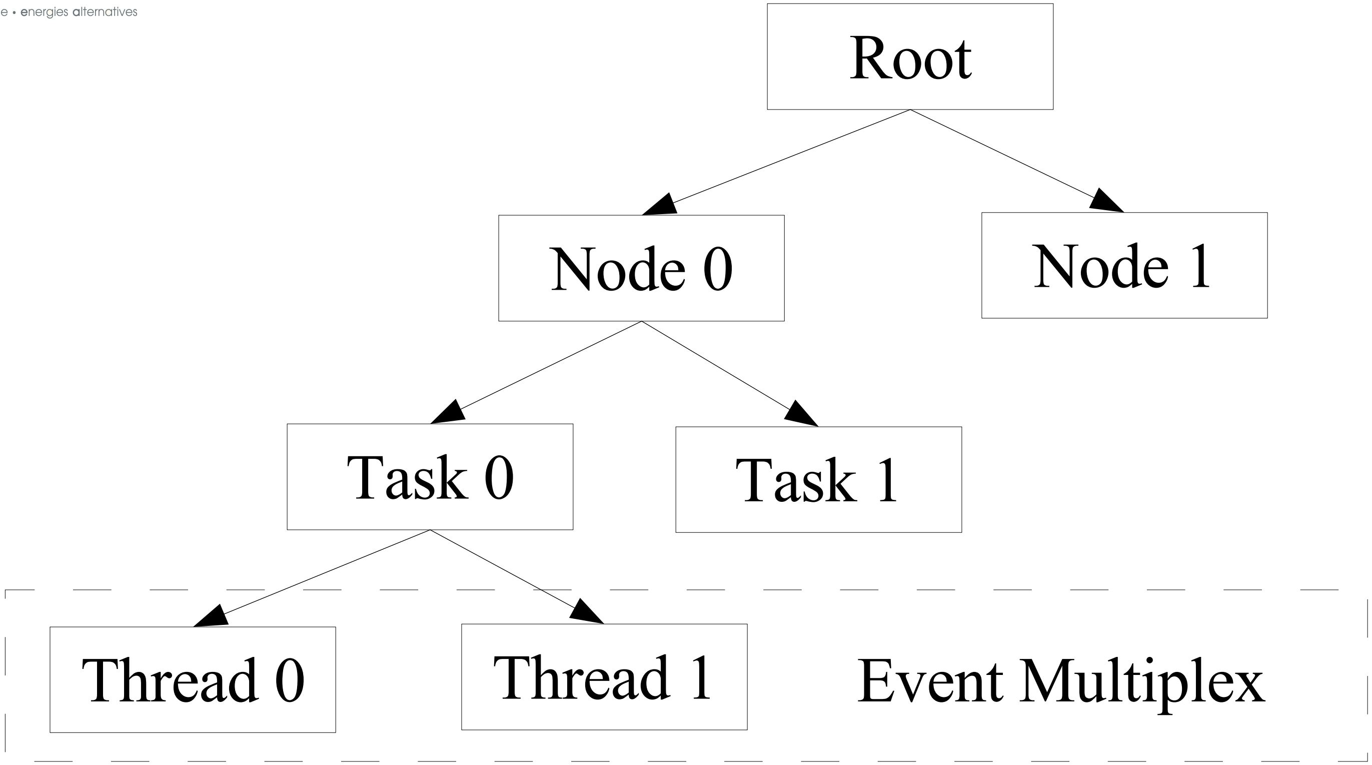
MPC Trace: user defined generic events



```
struct event_A a;
struct event_B b;
struct event_C c;

memset( &a, 0, sizeof( struct event_A ) );
memset( &b, 1, sizeof( struct event_B ) );
memset( &c, 2, sizeof( struct event_C ) );

MPC_Trace_generic_event(  (void *)&a, sizeof( struct event_A ), 0,  NULL );
MPC_Trace_generic_event(  (void *)&b, sizeof( struct event_B ), 1,  NULL );
MPC_Trace_generic_event(  (void *)&c, sizeof( struct event_C ), 2,  NULL );
```

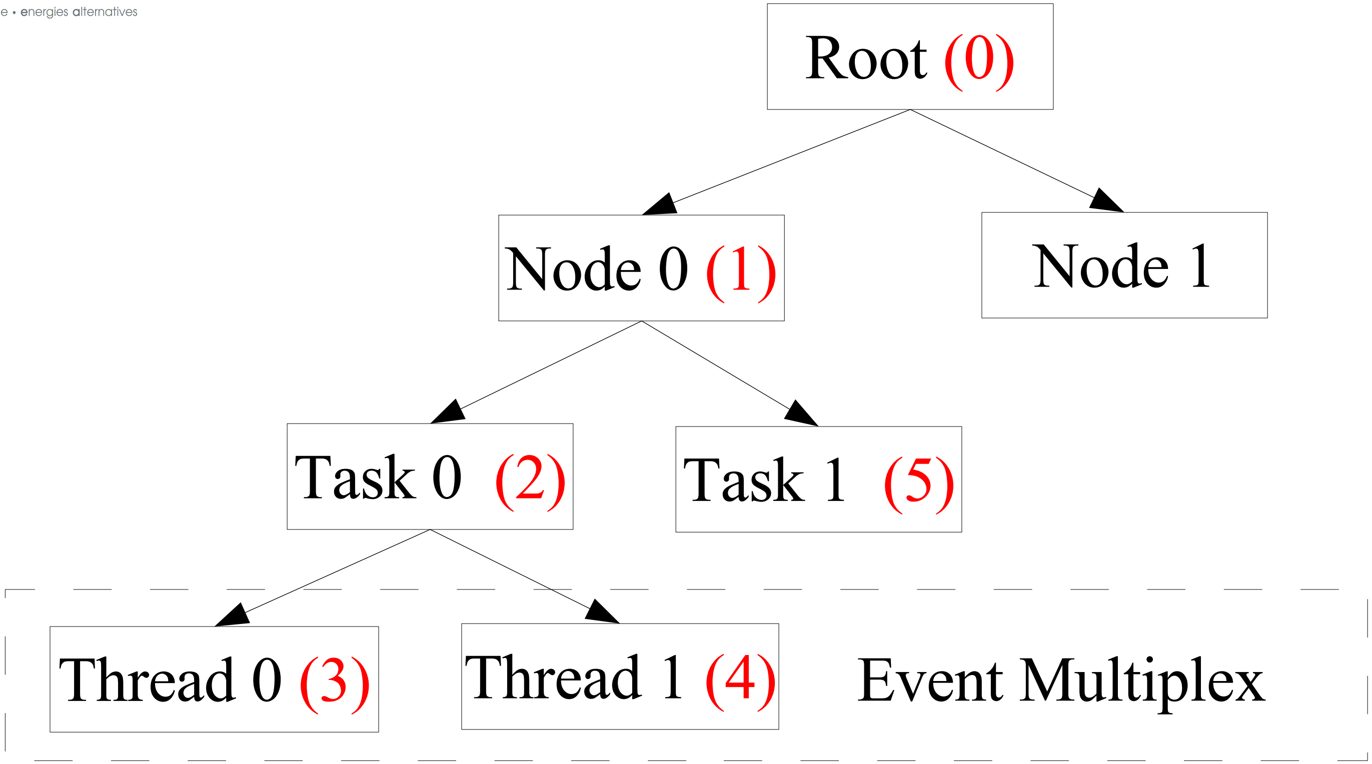
MPC Trace Library

Meta-data handling



MPC Trace Library

Meta-data handling after DFS



Global identifiers are computed on the fly by adding to each “job” local ids its container id.

MPC Trace Library

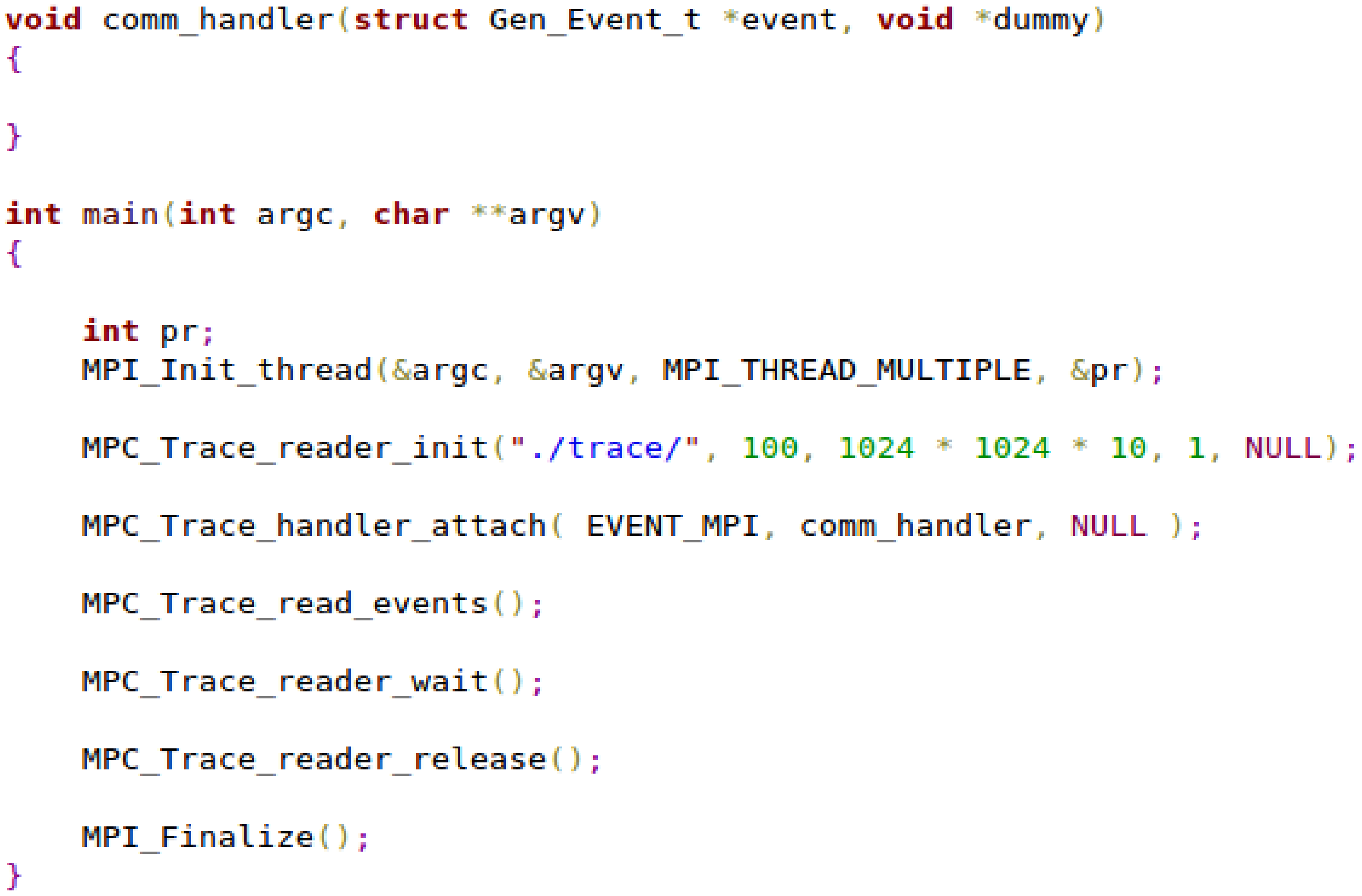
MPC Trace reader

- MPI + PThread based parallelism:
  - Simple dispatch of event files over processing ranks.
  - Also handles meta-data retrieval and dispatch
- Event are sent to the analysis tool via handlers:
  - Using a single handler footprint:

    *void (*handler)(struct Gen_Event_t *evt, void *arg)*
  - Possibility to register multiple handlers on the same type of event.
  - Possibility to register an handler on every events
  - Only event files with associated handlers are processed.

MPC Trace Library

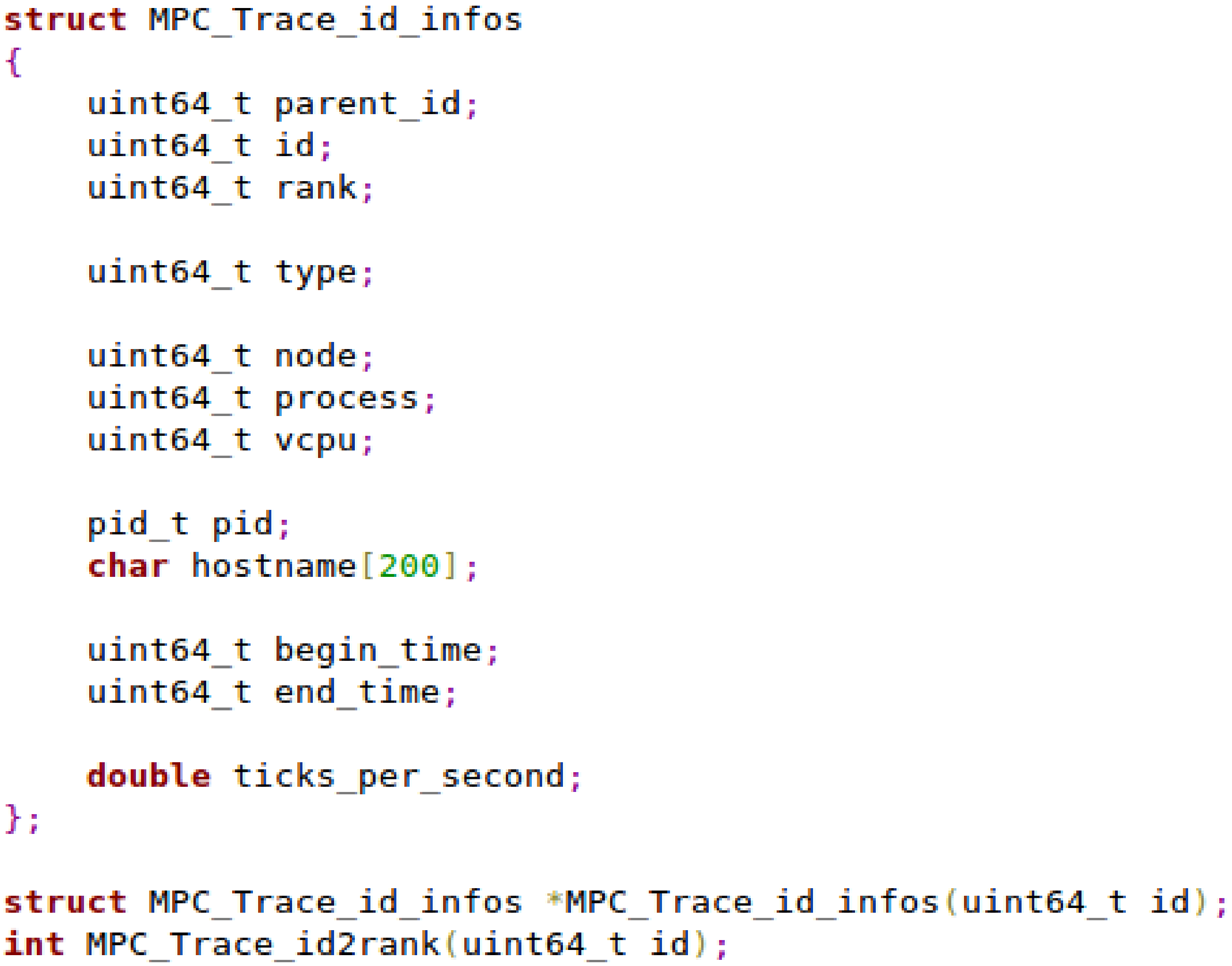
MPC Trace reader: simple example



```
void comm_handler(struct Gen_Event_t *event, void *dummy)
{

}

int main(int argc, char **argv)
{

    int pr;
    MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &pr);

    MPC_Trace_reader_init("./trace/", 100, 1024 * 1024 * 10, 1, NULL);

    MPC_Trace_handler_attach( EVENT_MPI, comm_handler, NULL );

    MPC_Trace_read_events();

    MPC_Trace_reader_wait();

    MPC_Trace_reader_release();

    MPI_Finalize();
}
```

MPC Trace Library

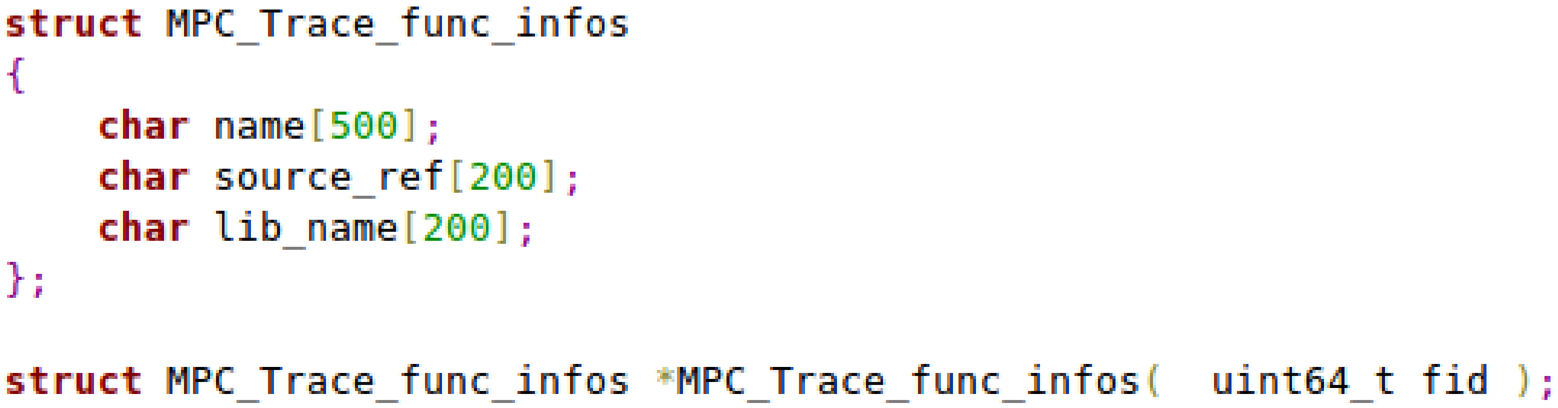
MPC Trace reader: topology informations



```
struct MPC_Trace_id_infos
{
    uint64_t parent_id;
    uint64_t id;
    uint64_t rank;

    uint64_t type;

    uint64_t node;
    uint64_t process;
    uint64_t vcpu;

    pid_t pid;
    char hostname[200];

    uint64_t begin_time;
    uint64_t end_time;

    double ticks_per_second;
};

struct MPC_Trace_id_infos *MPC_Trace_id_infos(uint64_t id);
int MPC_Trace_id2rank(uint64_t id);
```

MPC Trace Library

MPC Trace reader: symbol informations



```
struct MPC_Trace_func_infos
{
    char name[500];
    char source_ref[200];
    char lib_name[200];
};

struct MPC_Trace_func_infos *MPC_Trace_func_infos(  uint64_t fid );
```

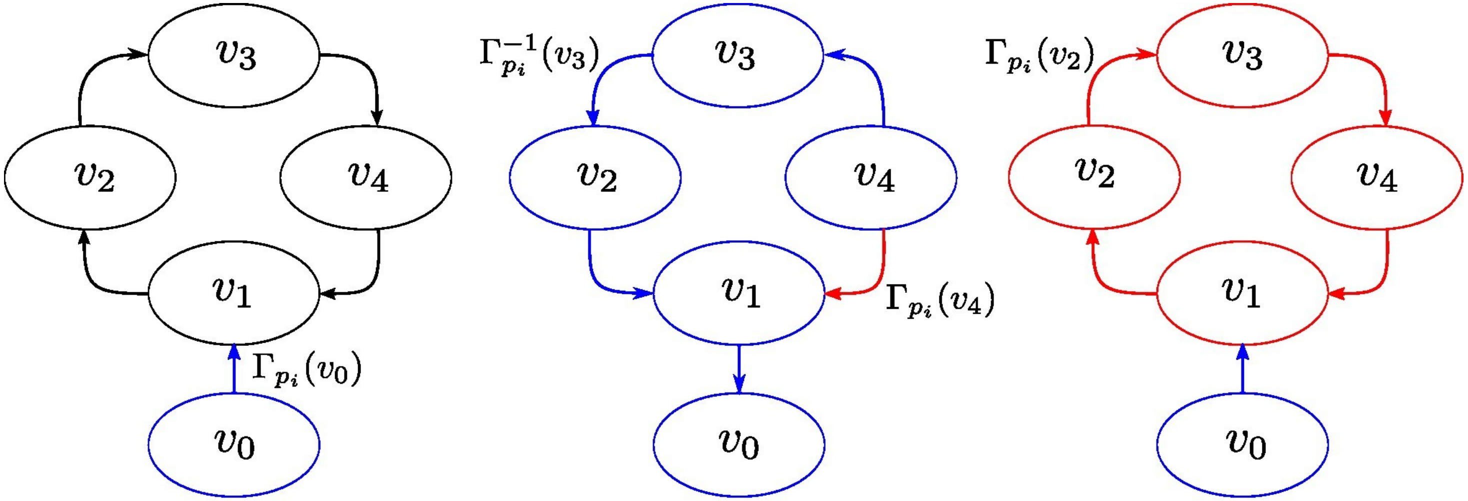
MPC Trace Library

MPC Trace debugger: trace based backtrace

```
10  malloc at 0x10441f040 size 72
9    malloc at 0x10441f100 size 16
8    malloc at 0x10441f1e0 size 16
7 « Parameters::SetParameters()
6 » DomainDecomposition(Parameters&)
5 « DomainDecomposition(Parameters&)
4 » Parameters::AllocateTables()
3    » Parameters::AllocateTables()
2    « Parameters::AllocateTables()
1 BEGIN MPI_ALLREDUCE with MPI_COMM_WORLD
0 Process exited badly with signal Segmentation fault (11)
```

MPC Trace Library

MPC Trace debugger: deadlock detection

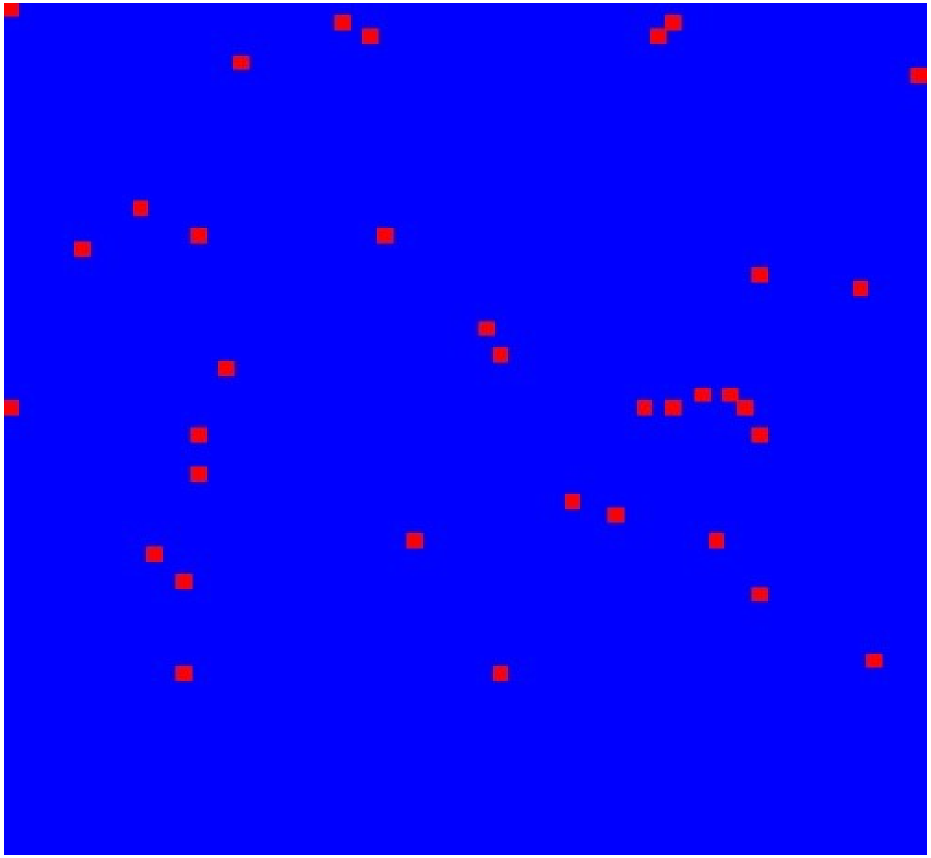


Deadlock detection : a simple coloration over the lock dependency graph generated from a trace-based crash-dump.

MPC Trace Library

energie atomique • energies alternatives

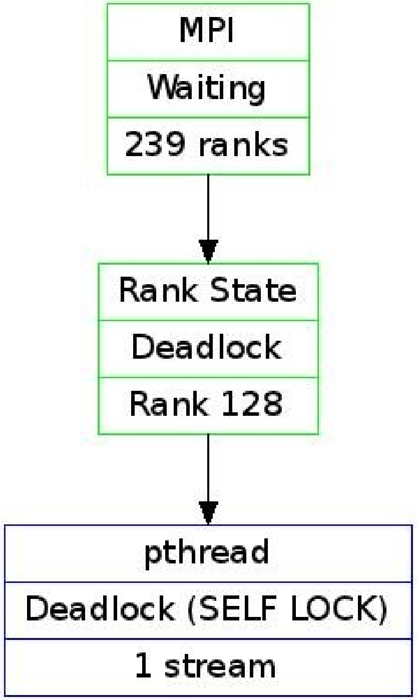
MPC Trace debugger: deadlock detection



Missing ranks in an MPI_Reduce over 4096 MPI processes

MPC Trace Library

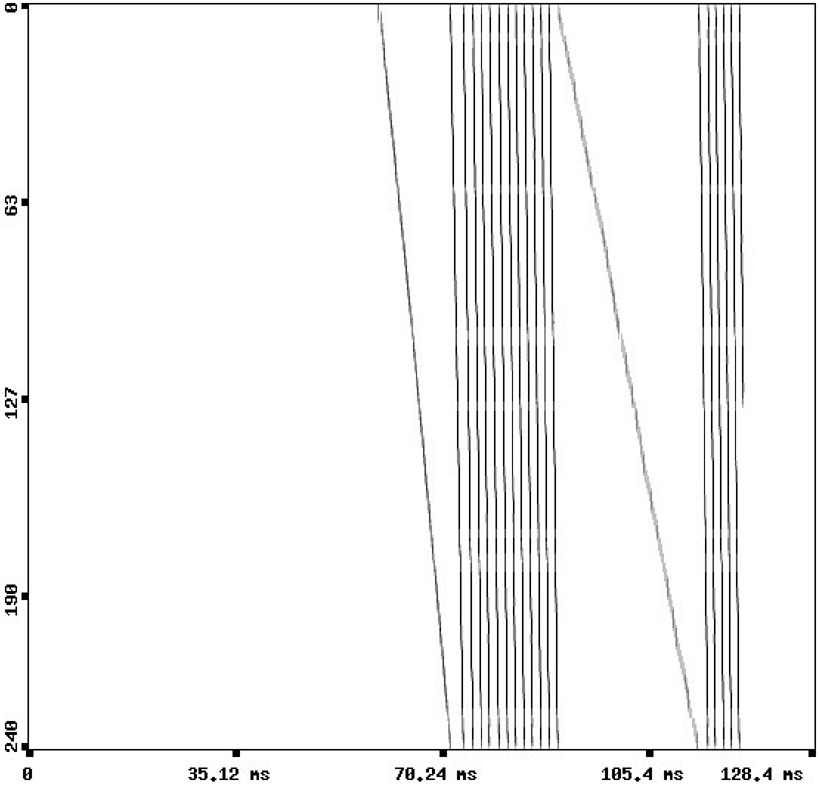
MPC Trace debugger: deadlock detection



MPI
Waiting
239 ranks

Rank State
Deadlock
Rank 128

pthread
Deadlock (SELF LOCK)
1 stream



Deadlock on a 240 processes ring.

MPC Trace Library

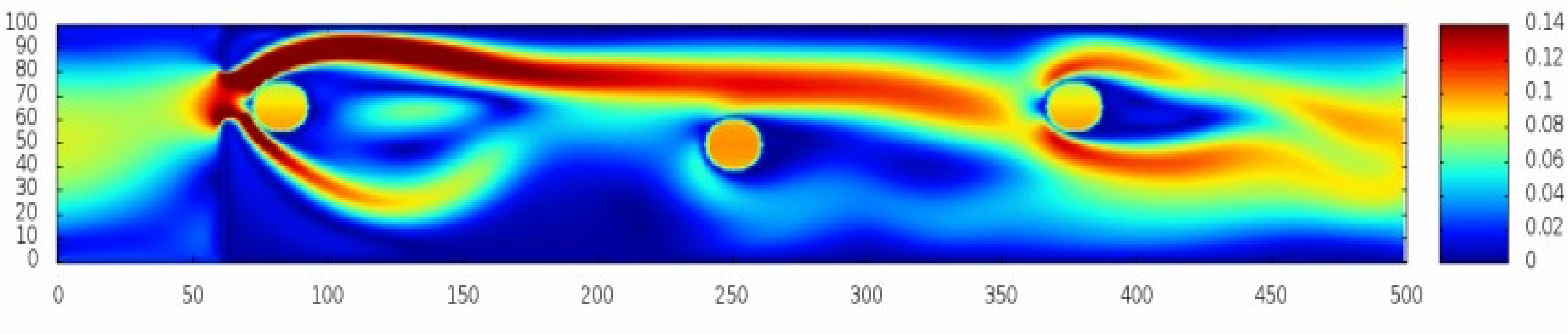
MPC Trace analyzer

- Based on the MPC Trace reader:
  - Immediate data parallelism
  - Trace processing at scale
- Produces configurable PDF reports:
  - Around 150 options
  - Implements 11 concurrent analysis
- Based on a simple “Map Reduce” approach
- Tested up to 4096 cores on real C++ codes
- Compatible with most MPI flavours and MPC.

MPC Trace Library

MPC Trace analyzer

- First example **lbm**:
  - This year optimization project for our MIHPS students
  - Simulates a Kármán vortex street
  - Carefully desoptimized



MPC Trace Library

Comm Mapper

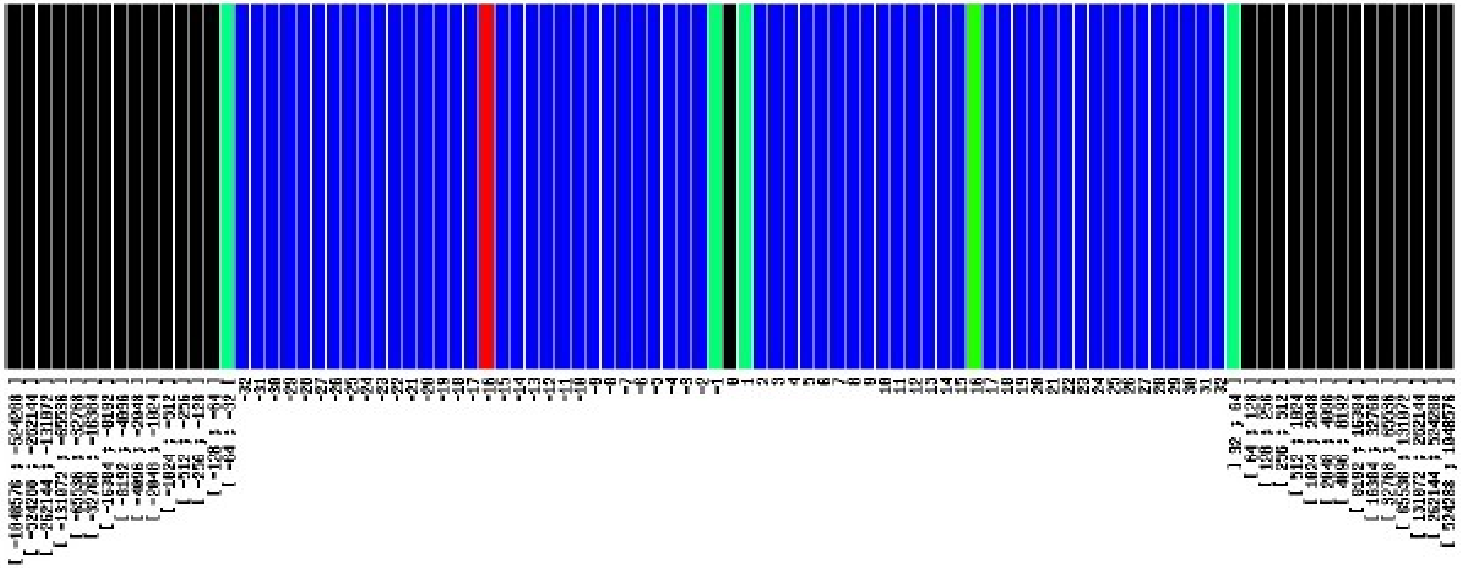
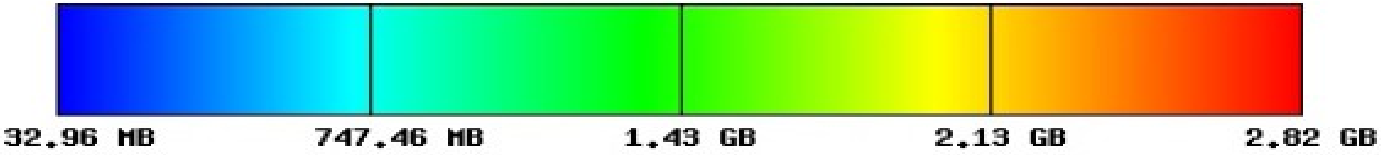


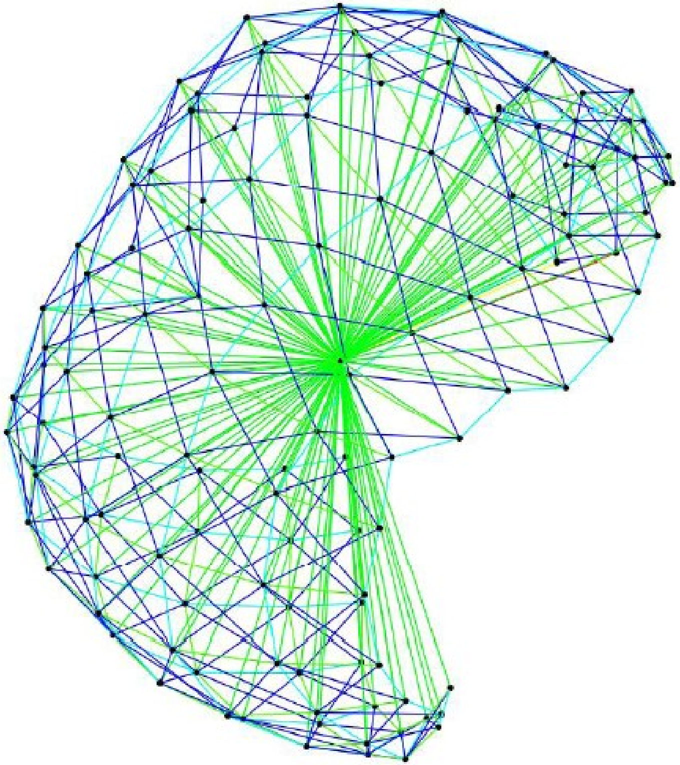
Figure 3.3: Mapping of MPI Calls in size



Communication mapping on a voluntarily unbalanced benchmark

MPC Trace Library

Comm Matrix



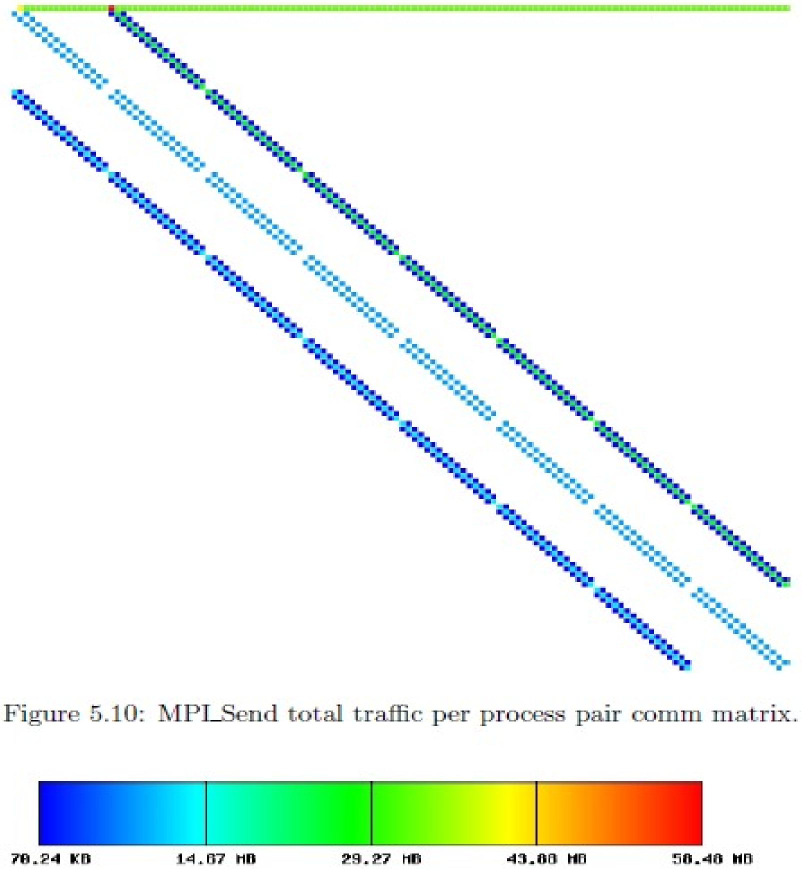


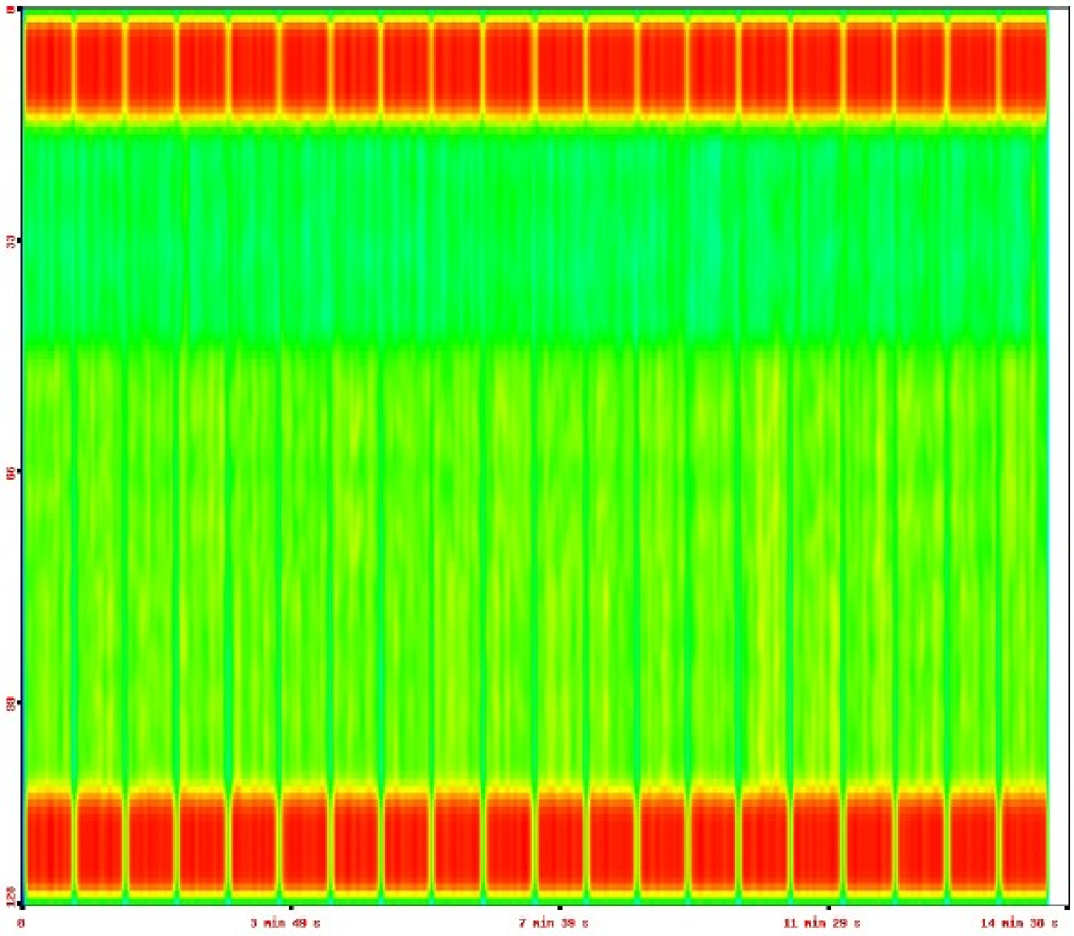
Figure 5.10: MPI_Send total traffic per process pair comm matrix.

MPI_Send topology on a voluntarily unbalanced benchmark (128 tasks)

MPC Trace Library

energie atomique • energies alternatives

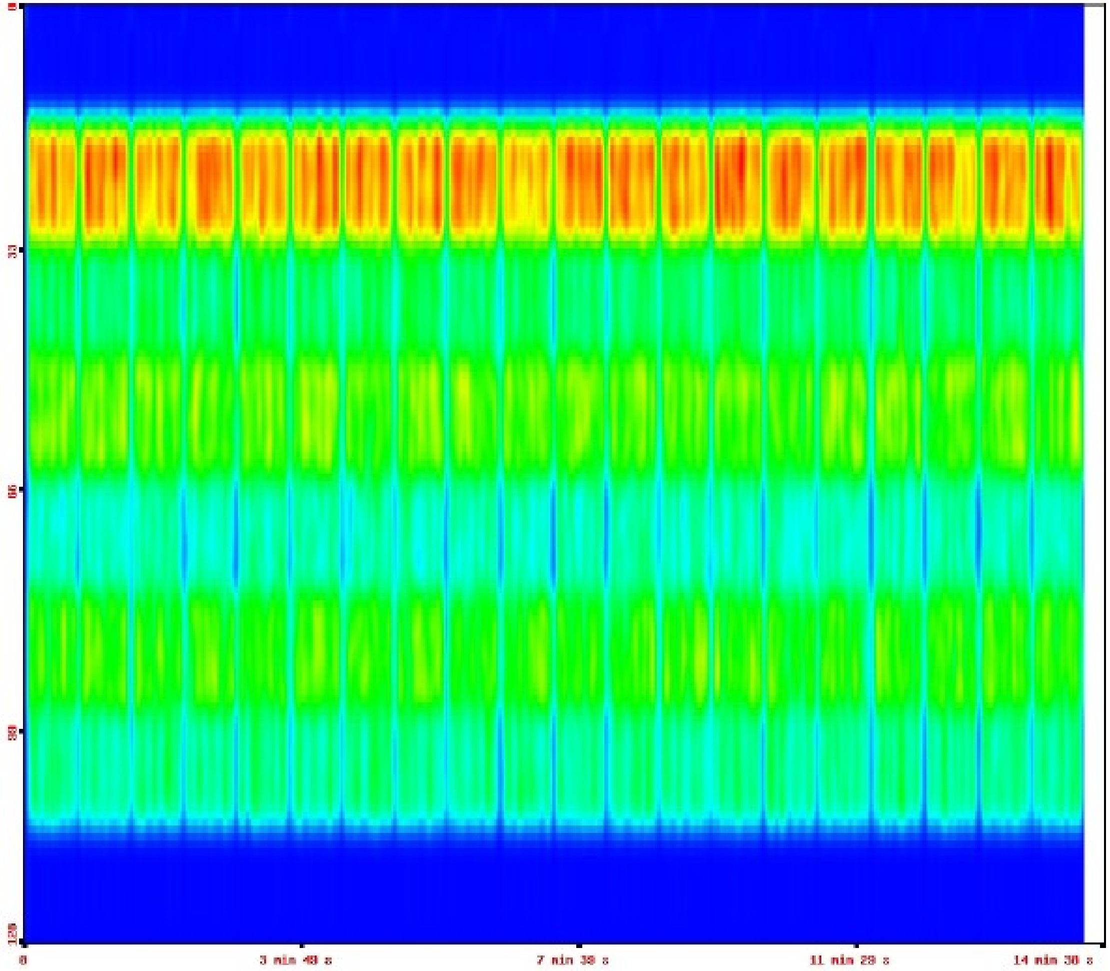
Time Matrix



Collectives time matrix on a voluntarily unbalanced benchmark (128 tasks)

MPC Trace Library

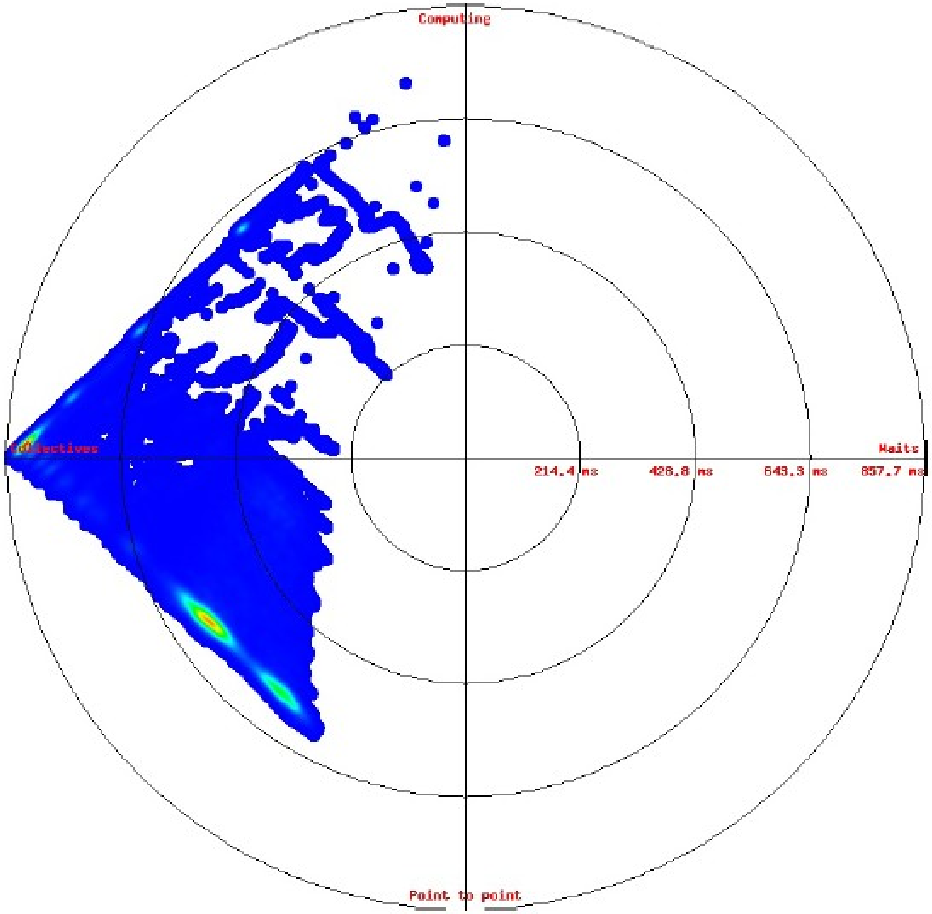
Time Matrix



MPI_Recv time matrix on a voluntarily unbalanced benchmark (128 tasks)

MPC Trace Library

MPI Quadrant



MPI Quadrant on a voluntarily unbalanced benchmark (128 tasks)

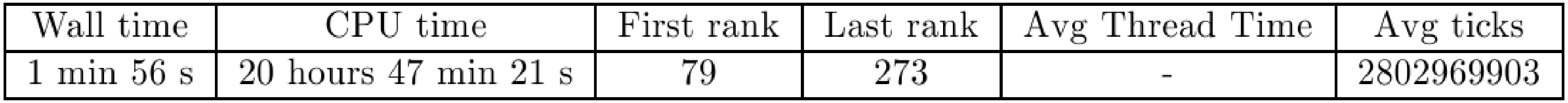
MPC Trace Library

MPC Trace analyzer

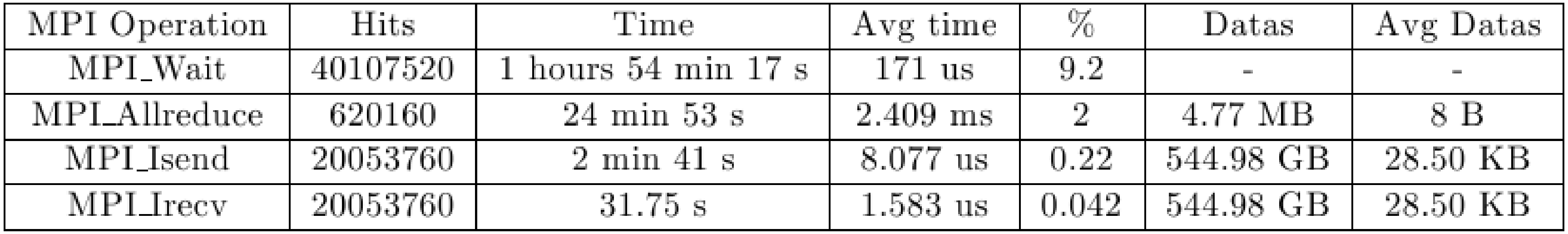
- Second example **EulerMHD**:
  - Middle sized C++ MPI code
  - Simulates Euler and ideal Magneto HydroDynamic equations at high order on a 2D Cartesian mesh.
  - Code scales pretty well thanks to its regular communication topology:
    - Up to 80k cores with MPC

MPC Trace Library

MPI Profile



Wall time	CPU time	First rank	Last rank	Avg Thread Time	Avg ticks
1 min 56 s	20 hours 47 min 21 s	79	273	-	2802969903

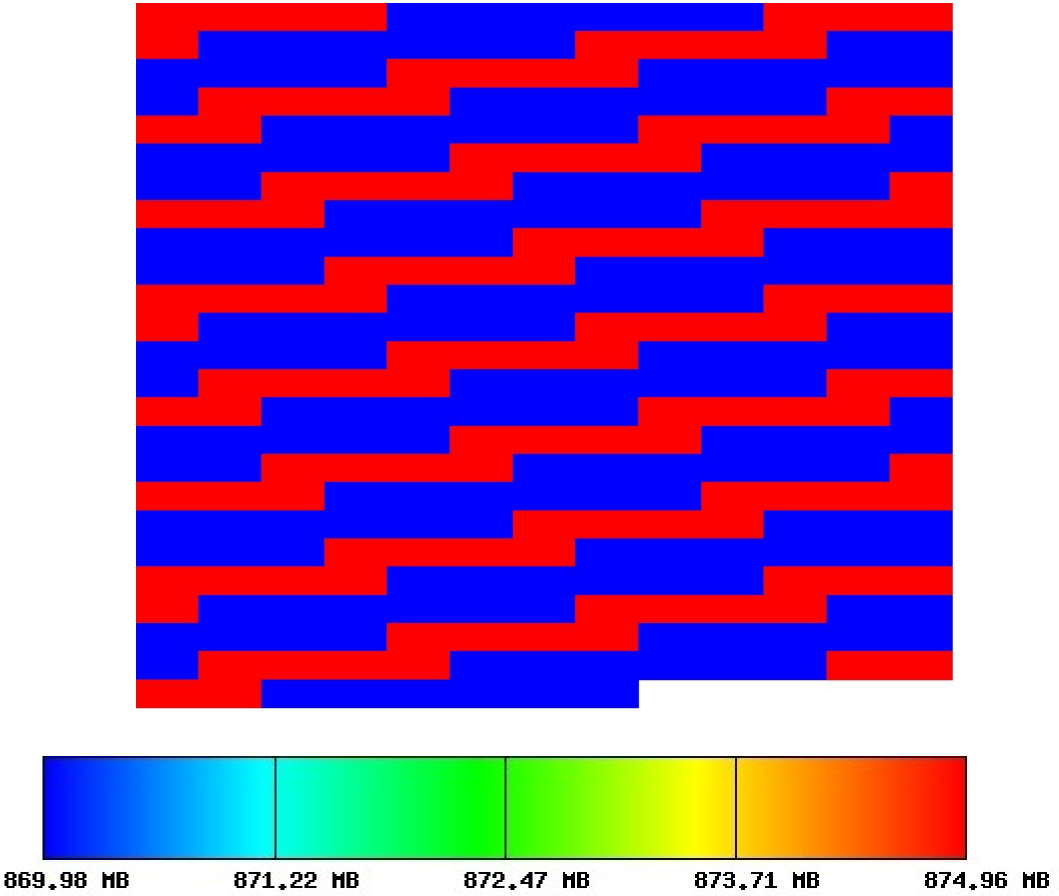


MPI Operation	Hits	Time	Avg time	%	Datas	Avg Datas
MPI_Wait	40107520	1 hours 54 min 17 s	171 us	9.2	-	-
MPI_Allreduce	620160	24 min 53 s	2.409 ms	2	4.77 MB	8 B
MPI_Isend	20053760	2 min 41 s	8.077 us	0.22	544.98 GB	28.50 KB
MPI_Irecv	20053760	31.75 s	1.583 us	0.042	544.98 GB	28.50 KB

Unbalanced case on 640 processes

MPC Trace Library

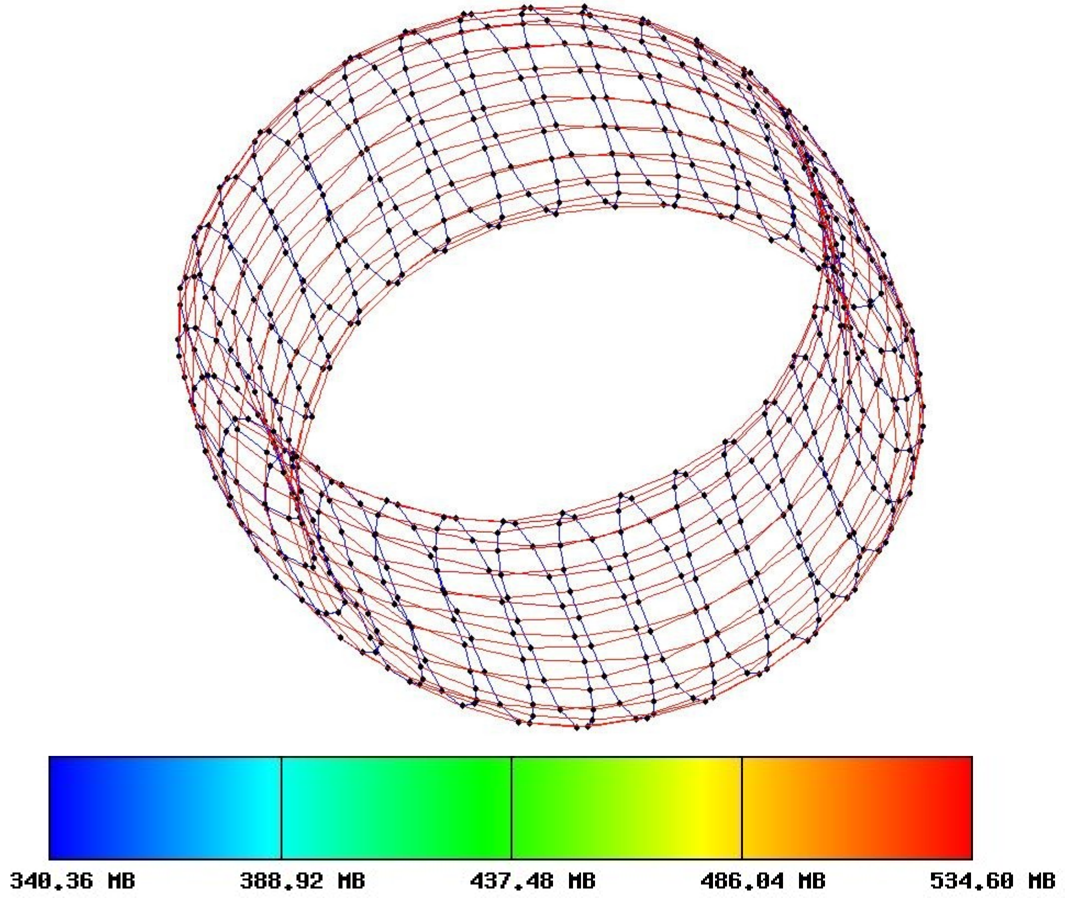
MPI Comm Mapper



MPI_Send total size for an unbalanced case on 640 processes

MPC Trace Library

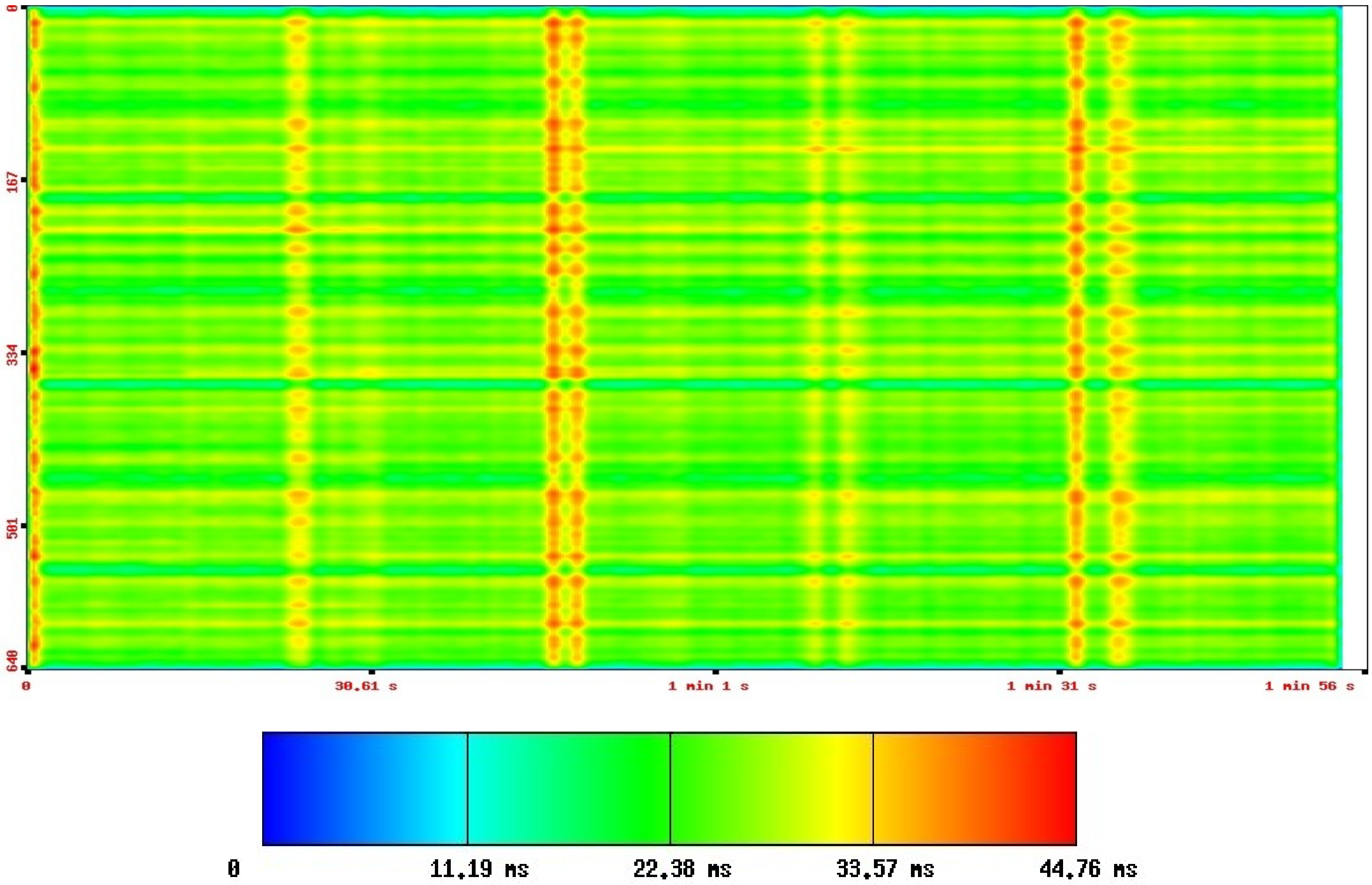
MPI Comm Matrix



MPI_Send total size for an
unbalanced case on 640 processes

MPC Trace Library

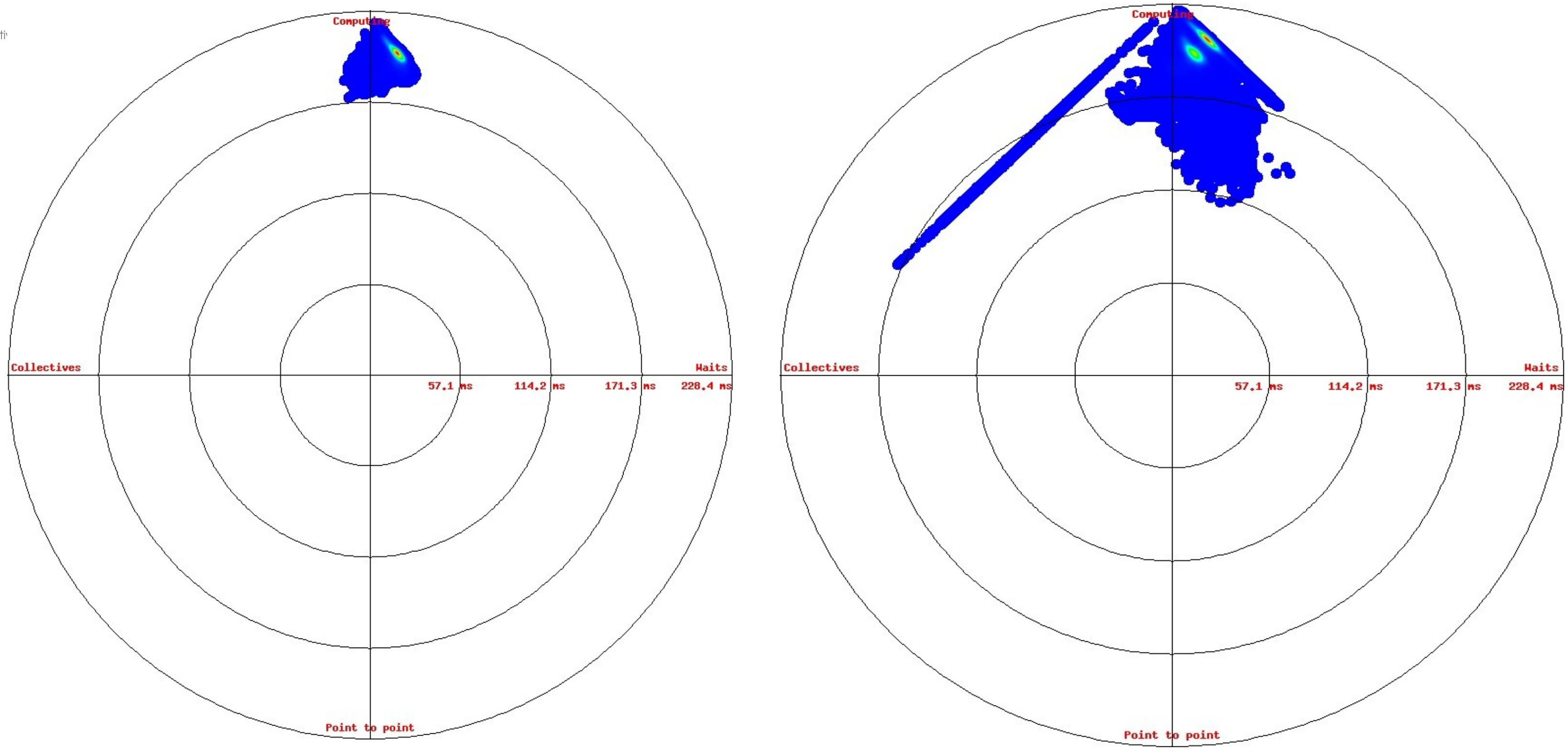
MPI Time matrix



MPI Time for an unbalanced case on 640 processes

MPC Trace Library

MPI Quadrant

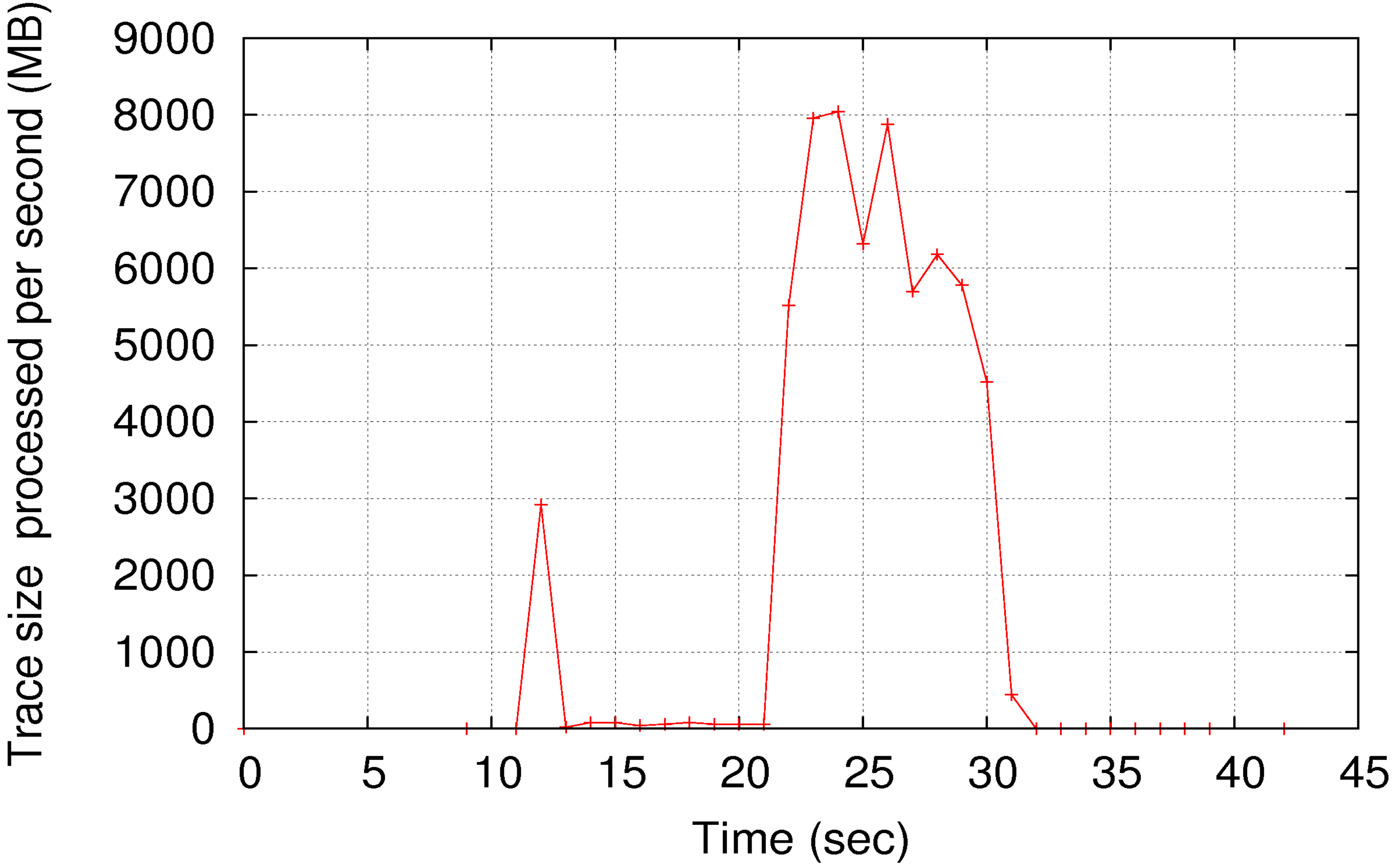


MPI Quadrant respectively correlated and decorrelated cases
for an unbalanced case on EulerMHD 640 processes

MPC Trace Library

Performances: MPC trace analyzer (128 tasks lbm)

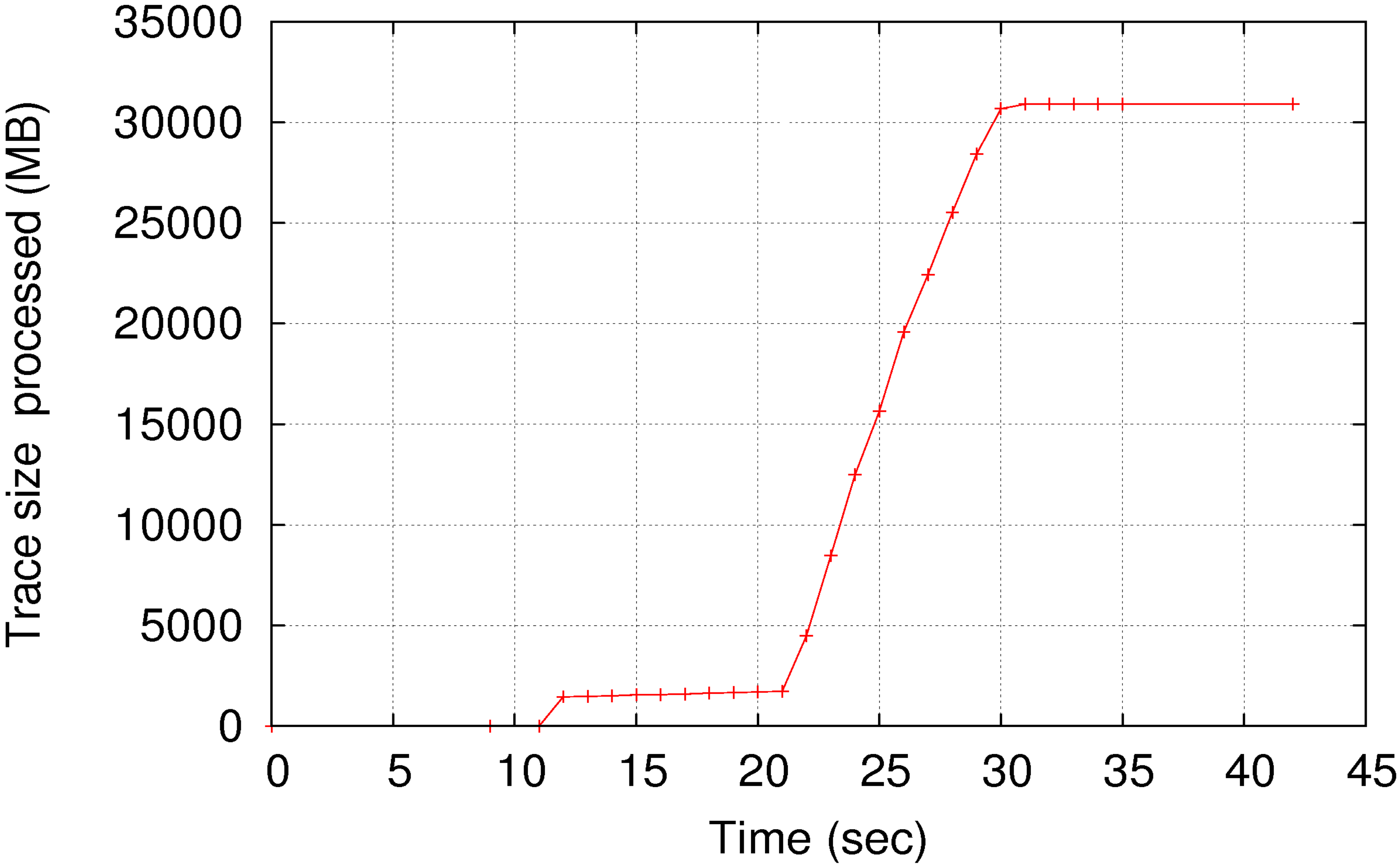
Processing throughput



MPC Trace Library

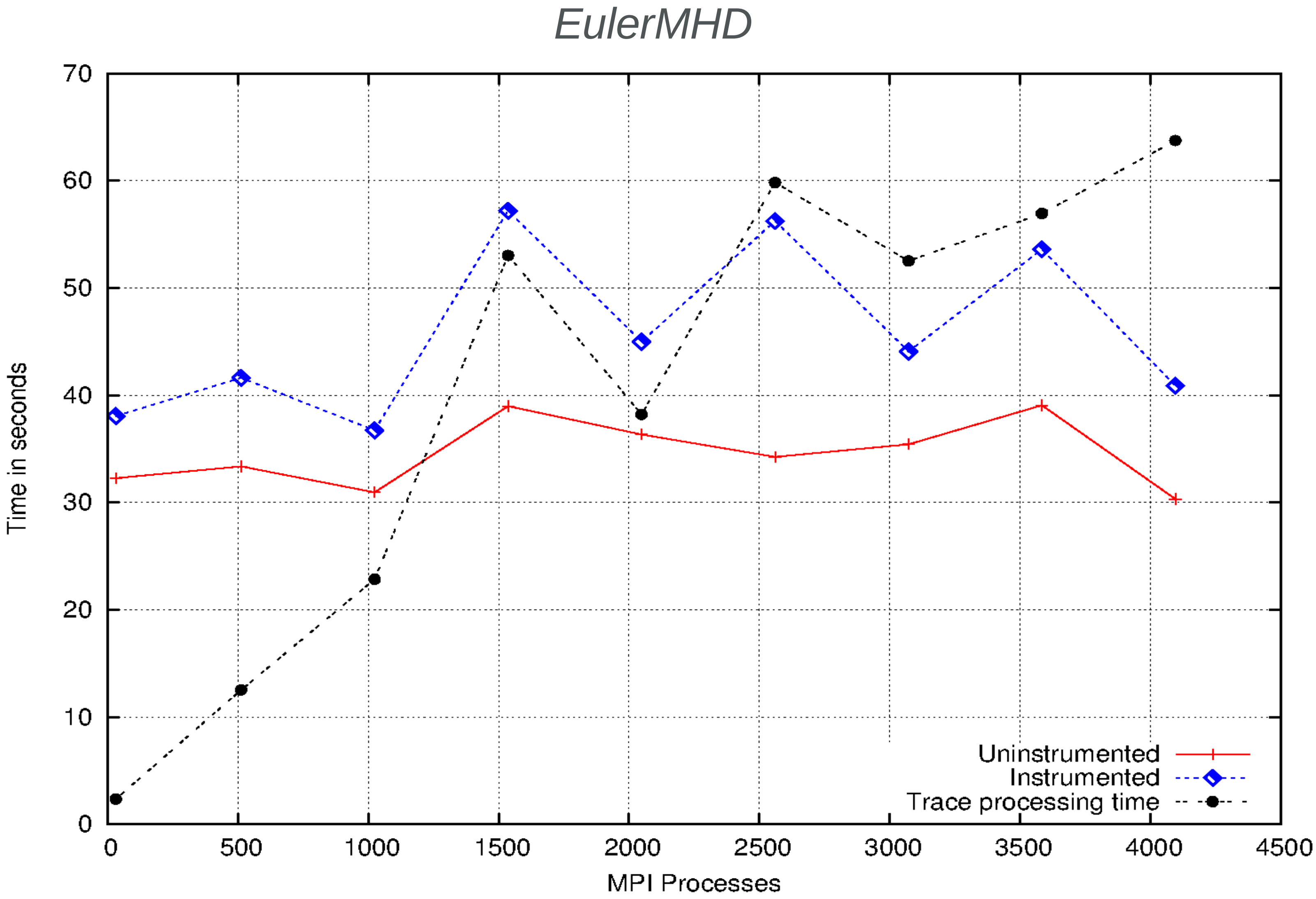
Performances: mpc trace analyzer (128 tasks lbm)

Total trace size processed



MPC Trace Library

Performances: MPC trace analyzer (debug 1024 events)

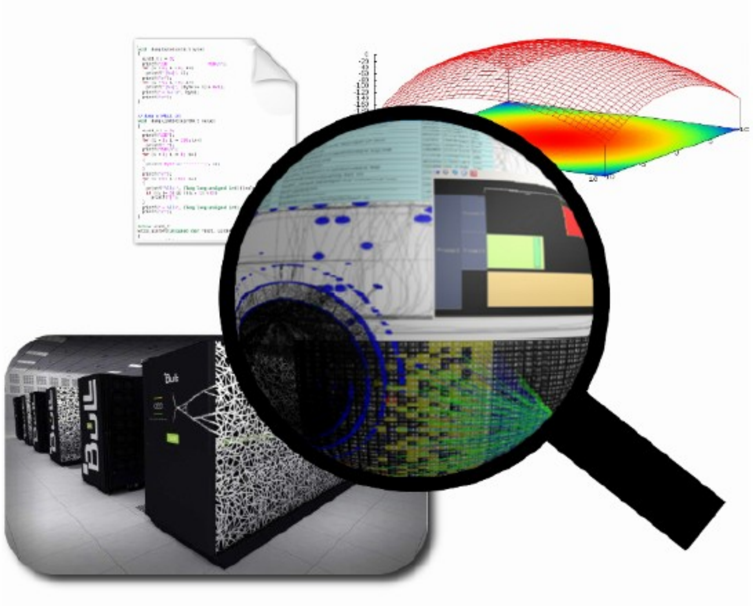


MPC Trace Library

Conclusion / Perspectives

- MPC Trace library:
  - Compact interface
  - Parallel analysis
  - Tested up to 30 000 cores in trace and 4096 cores with the MPC trace analyzer
- Perspectives:
  - Model based approach
  - Code characterization
- Not open sourced for now might come in a next release of the MPC framework.

MPC Trace Library



Jean-Baptiste BESNARD

PHD Student

CEA, DAM, DIF

F 91297 Arpajon, FRANCE