GPI-2: a PGAS API for asynchronous and scalable parallel applications

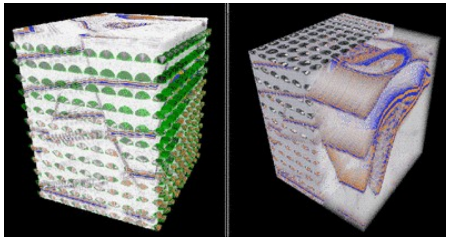
Rui Machado

CC-HPC, Fraunhofer ITWM

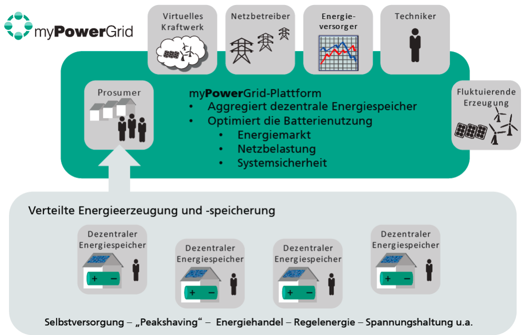
Barcelona, 13 Jan. 2014

Fraunhofer ITWM – CC-HPC

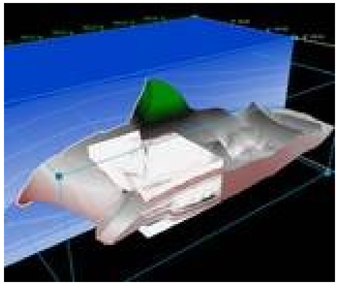
- Fraunhofer Institute for Industrial Mathematics
- Kaiserslautern, Germany
- CC-HPC: Competence Center for High Performance Computing and Visualization



Seismic Imaging



Green by IT

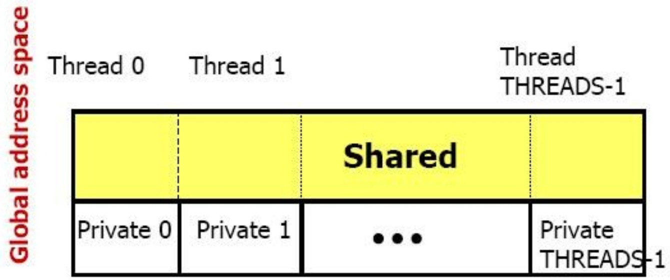


Outline

- Background
- GPI-2
  - Introduction
  - Functionality
  - Small examples
- Performance / Applications
- Final remarks

Background

- GPI: **G**lobal Address Space **P**rogramming **I**nterface
  - Previously: Fraunhofer Virtual Machine, CellBE
  - Started around 2006 in industry codes
  - Completely replaced MPI
- PGAS: **P**artitioned **G**lobal **A**ddress **S**pace
  - Ex: UPC (extension), Chapel, X10 (new)



- **GPI-2**: second generation of GPI based on GASPI specification
  - More complete functionality set
  - Preparing for the future

Fraunhofer ITWM

GPI-2: Implementation of GASPI specification

Scalability
- Asynchronous one-sided communication
- Weak synchronization
- Low and explicit resource usage
- Threaded model

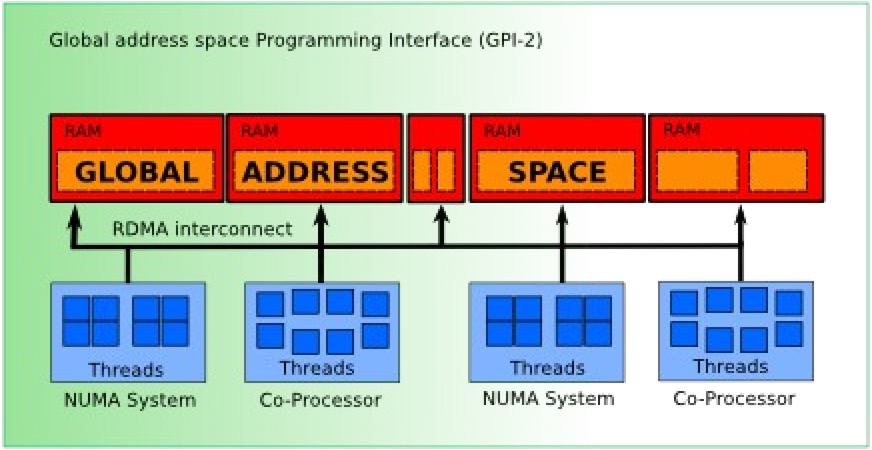
Flexibility
- Dynamic process set
- Groups
- Compact API

GPI-2

Versatility
- Multi-segment support
  - Co-processor/Accelerator
  - Multiple domains
- MPI interoperability

Fault tolerant
- Timeout mechanism
- State vector
- Locally well-defined state

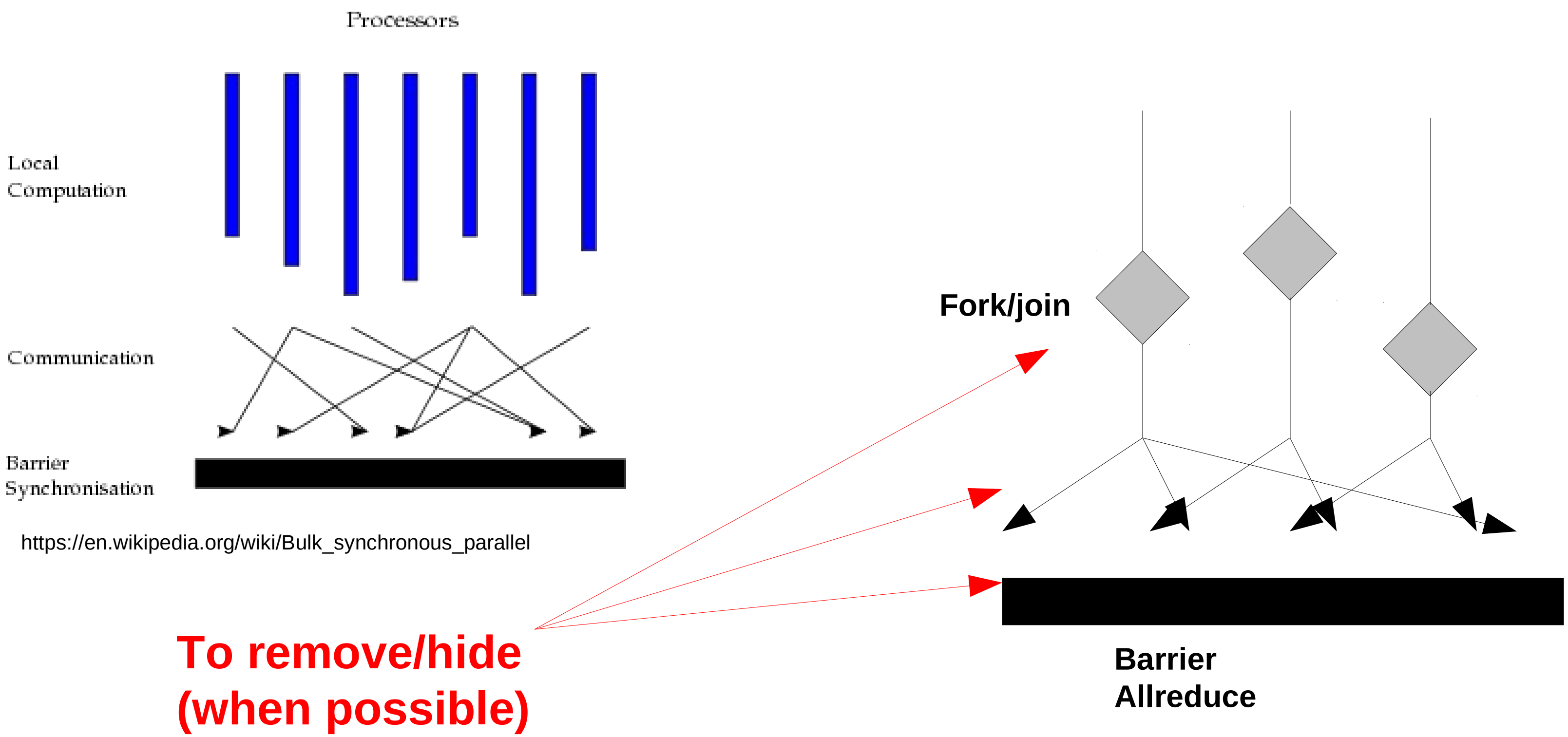
GPI-2: Main attributes and goals



- PGAS library
- Asynchronous, non-blocking one-sided communication
- Wire speed, efficient communication
- Zero-copy, no CPU cycles
- **Goal**: full overlap of computation and communication
- **Goal:** from bulk-synchronous 2-sided comm to asynchronous 1-sided
- Thread-safe: threads are rule, not exception
- Fault tolerance mechanisms
- Heterogeneous by construction

Fraunhofer ITWM

Programming model change



- Changes to program/algorithm to accommodate programming model
- *“A new model of computation has to change the synchronization semantics, removing global barriers, and exposing new levels of algorithmic parallelism.”*

Fraunhofer ITWM

Functionality - Overview

- Compact API
  - Easy to learn
- One-sided communication
- Weak synchronization
- Segments
- Groups
- Collectives
- Atomic operations
- Lists and Passive communication
- Connection management
- Fault tolerance

Fraunhofer ITWM

One-sided Communication

- It's the standard way of communication
  - Advantage: no implicit synchronization on each communication request
- One side specifies all parameters
  - Local location, remote location, size, ...
- Remote side does not get involved
- Two operations: WRITE and READ
- Split phases: write/read, wait
- Queues
  - Scalability
  - Separation of concerns
  - Configurable number and depth (with limits)

Weak synchronization

- In one-sided communication:
  - Remote rank is not involved
- In some cases and at some point, the remote rank needs some information of whether data is valid
- Enter notifications:
  - Notify remote rank of data transfer(s)
  - Efficient wait operation on remote side
- Independent notification
  - After several write operations
  - Guarantee of non-overtaking
- Single command:
  - write_notify

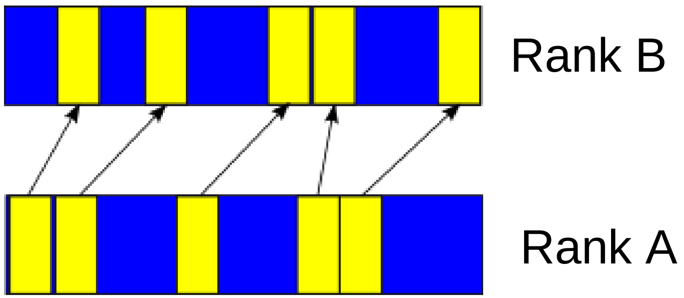
Segments

- Contiguous blocks of memory
  - Partitions of global space
  - May be globally accessible by all threads on all nodes
  - May represent different levels: RAM, NVRAM, Co-Processor
- Access:
  - Locally: with common memory operations
  - Remotely: with GPI-2 one-sided communication
- Management done by application
- 2 options:
  - Finely controlled: Allocate and register
  - Easy: Create within a group

Groups / Collectives

- Group is a subset of total ranks
  - Default: GASPI_GROUP_ALL
- A rank can be member of several groups
- Use-cases:
  - Share collectives
  - Create segments
- Supported collectives:
  - Barrier
  - Allreduce (MIN, MAX, SUM)
  - Allreduce with user defined reduction

Lists, Passive Comm and Atomic Ops



- Write/Read Lists
  - With/Without notification
  - *1* vs *N* requests

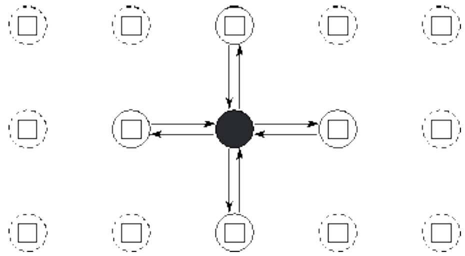
- CAS operations on global data (segments)
- FETCH-ADD and COMPARE-SWAP
- Use as:
  - Global shared variables
  - Build mechanisms to sync processes or events

- Passive communication
  - 2-sided semantics (send/recv)
  - No busy-waiting
  - Non-critical communication

Fraunhofer ITWM

Connection Management

- Communication infrastructure must be initialized
- By default, done at initialization
- Configurable by user
- In most applications, ranks communicate with a subset
- Create own topology
  - Connect/Disconnect with particular rank
- Resource consumption at large scale
  - 1000's nodes
- Flexibility



Fault tolerance

- Timeout mechanism
  - GASPI_BLOCK, GASPI_TEST, time (msecs)
  - In most non-local operations
- If a node fails, application is still alive
  - Important for long running computations
- State vector
  - GASPI_STATE_HEALTHY / GASPI_STATE_CORRUPT
- Addition of spare nodes (*tbd*)

Small Examples

GPI-2: Hello World

```
#include <GASPI.h>

 int main (int argc, char *argv[])
 {
   gaspi_proc_init (GASPI_BLOCK);

   gaspi_rank_t iProc;
   gaspi_proc_rank(&iProc);
   gaspi_rank_t nProc;
   gaspi_proc_num(&nProc);

   /* creation of global segment for communication */
   gaspi_segment_create(0, (1 << 31), GASPI_GROUP_ALL, GASPI_BLOCK, GASPI_DEFAULT_ALLOC);

   gaspi_printf(„Hello world from %u (of %u)\n“, iProc, nProc);

   gaspi_barrier(GASPI_GROUP_ALL, GASPI_BLOCK);

   gaspi_proc_term(GASPI_BLOCK);

   return 0;
 }
```

```
 > gcc alltoall.c -I $GPI2_HOME/include -L $GPI2_HOME/lib64 -lGPI2
-libverbs15 -o $GPI2_HOME/bin/helloworld.bin
 > getnode -n 4
 > gaspi_run -m machinefile $GPI_HOME/bin/alltoall
 Hello world from 0 (of 4)
 Hello world from 1 (of 4)
 Hello world from 2 (of 4)
 Hello world from 3 (of 4)
```


Fraunhofer ITWM

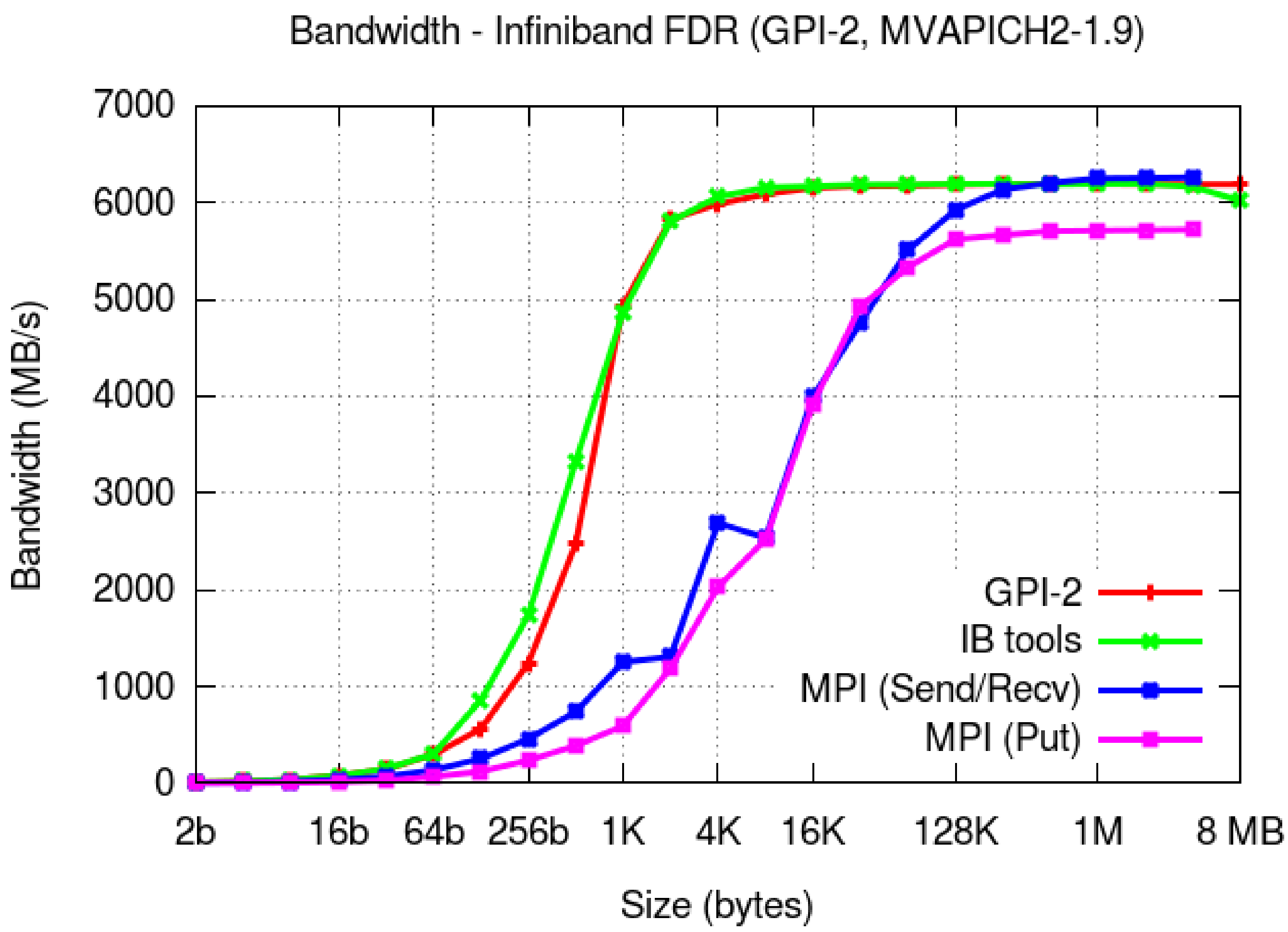
GPI-2: Alltoall in C

```
#include <GASPI.h>
int main(void)
{
  SUCCESS_OR_DIE (gaspi_proc_init (GASPI_BLOCK));
  gaspi_rank_t iProc, nProc;
  SUCCESS_OR_DIE (gaspi_proc_rank (&iProc));
  SUCCESS_OR_DIE (gaspi_proc_num (&nProc));
  gaspi_segment_id_t segment_id[2] = {0,1};
  gaspi_pointer_t segment_ptr[2];
  for (int i = 0; i < 2; ++i) /* create two segments and retrieve pointers */
  {
    SUCCESS_OR_DIE (gaspi_segment_create ( segment_id[i], nProc * sizeof(int)
                                         ,GASPI_GROUP_ALL, GASPI_BLOCK, GASPI_MEM_INITIALIZED
                                         )
                   );
    SUCCESS_OR_DIE (gaspi_segment_ptr (segment_id[i], segment_ptr + i));
  }
  for (gaspi_rank_t other = (iProc + 1) % nProc; other != iProc; other = (other + 1) % nProc)
  {
    ((int *)segment_ptr[0])[other] = iProc * nProc + other;
    SUCCESS_OR_DIE (gaspi_write ( segment_id[0], other * sizeof (int) /* local address */
                                , other, segment_id[1], iProc * sizeof (int) /* remote address */
                                , sizeof (int) /* sizeof message */
                                , (gaspi_queue_id_t) 0, GASPI_BLOCK
                                )
                   );
  }
  /* IMPORTANT: do some work here while the communication is in progress */
  SUCCESS_OR_DIE (gaspi_wait ((gaspi_queue_id_t) 0, GASPI_BLOCK));
  SUCCESS_OR_DIE (gaspi_barrier (GASPI_GROUP_ALL, GASPI_BLOCK));
  SUCCESS_OR_DIE (gaspi_proc_term (GASPI_BLOCK));
}
```

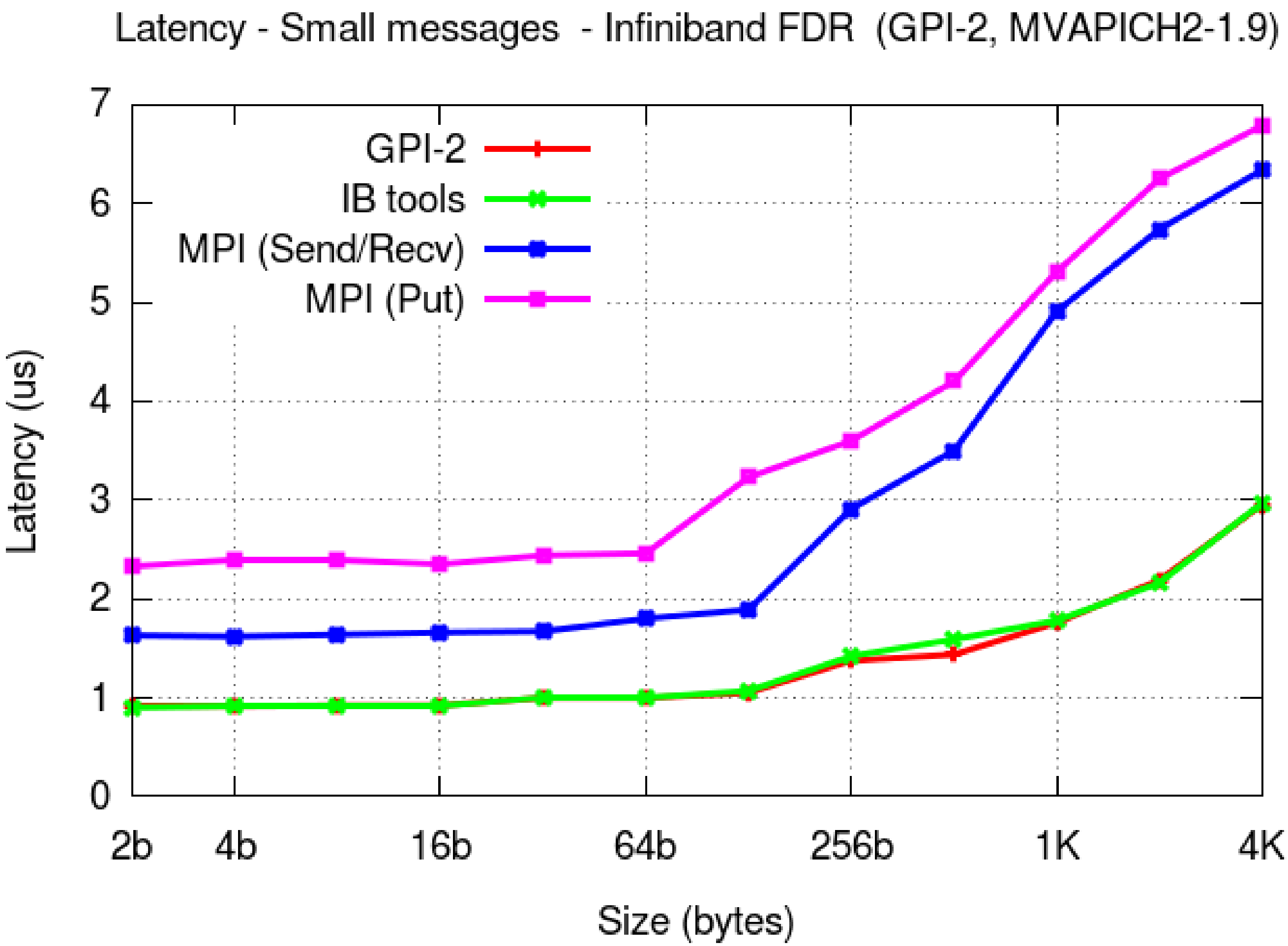

Fraunhofer ITWM

Performance

GPI-2: Bandwidth

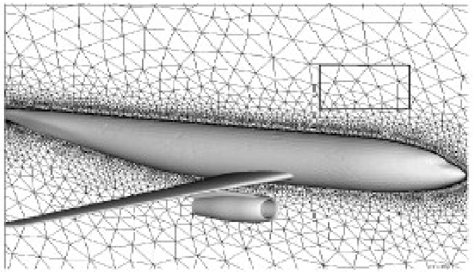


GPI–2: Latency (small messages)

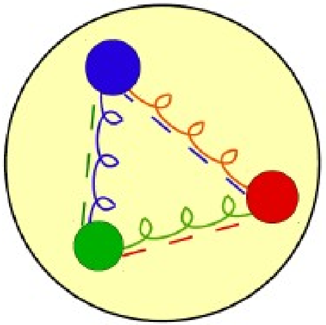


Fraunhofer ITWM

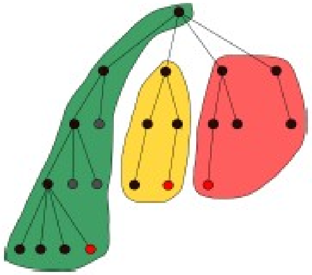
GPI Application examples



- TAU: Computational fluid dynamics (DLR, T-Systems SfR), classical MPI decomposition
  - 3D Finite Volume Reynolds Averaged Navier Stokes Solver
  - "key enabler for meeting strategic goals of future air transportation"



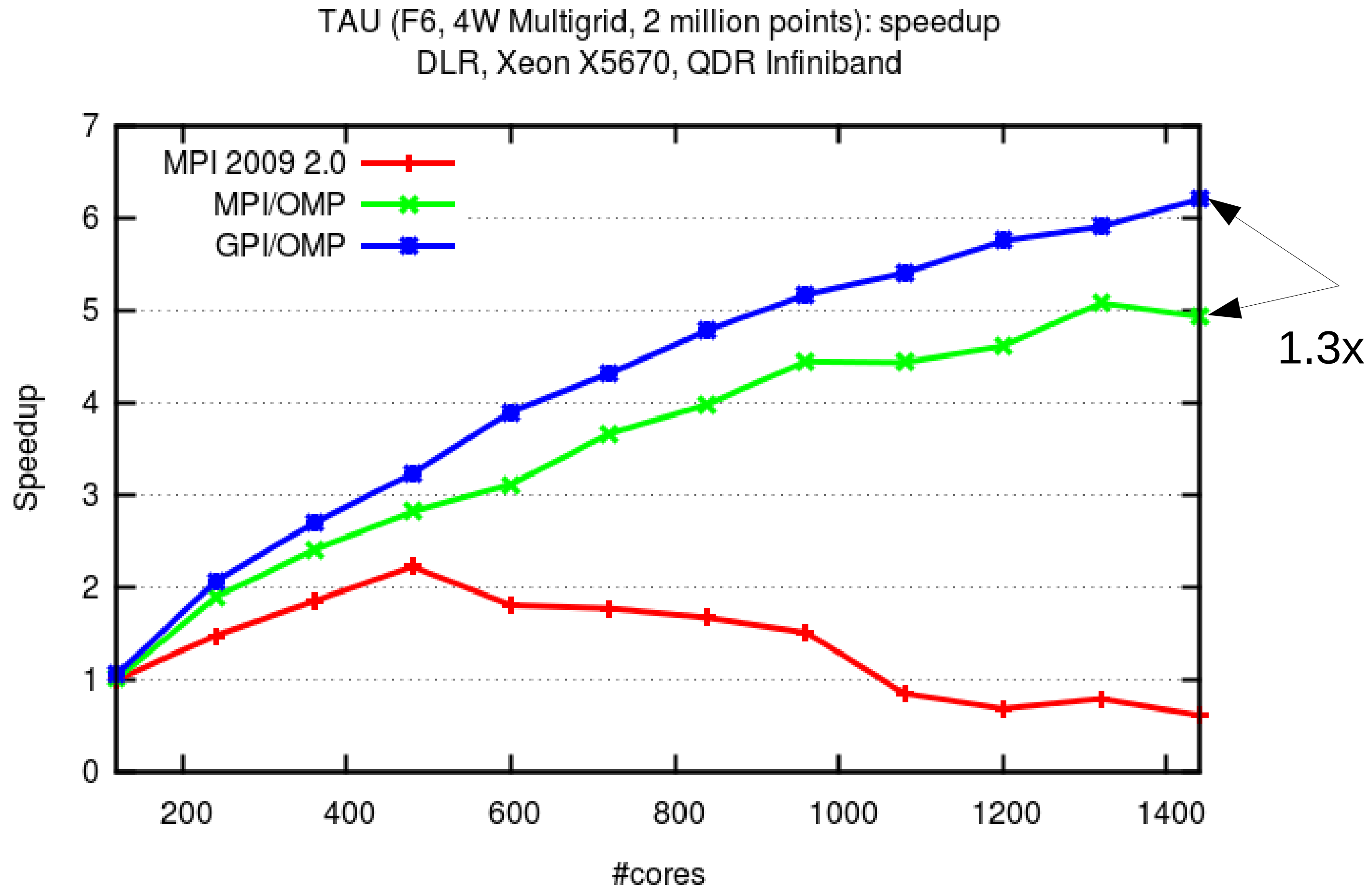
- BQCD: theory of strong interaction (Nakamura, Stüben), classical Stencil code
  - prediction of particle masses
  - Fortran90, MPI/OMP hybrid and shmem



- UTS: Unstructured tree search benchmark (Ohio State et. al.), generic parallel graph algorithm
  - dynamically generated search space
  - static scheduling is unlikely to work well

Fraunhofer ITWM

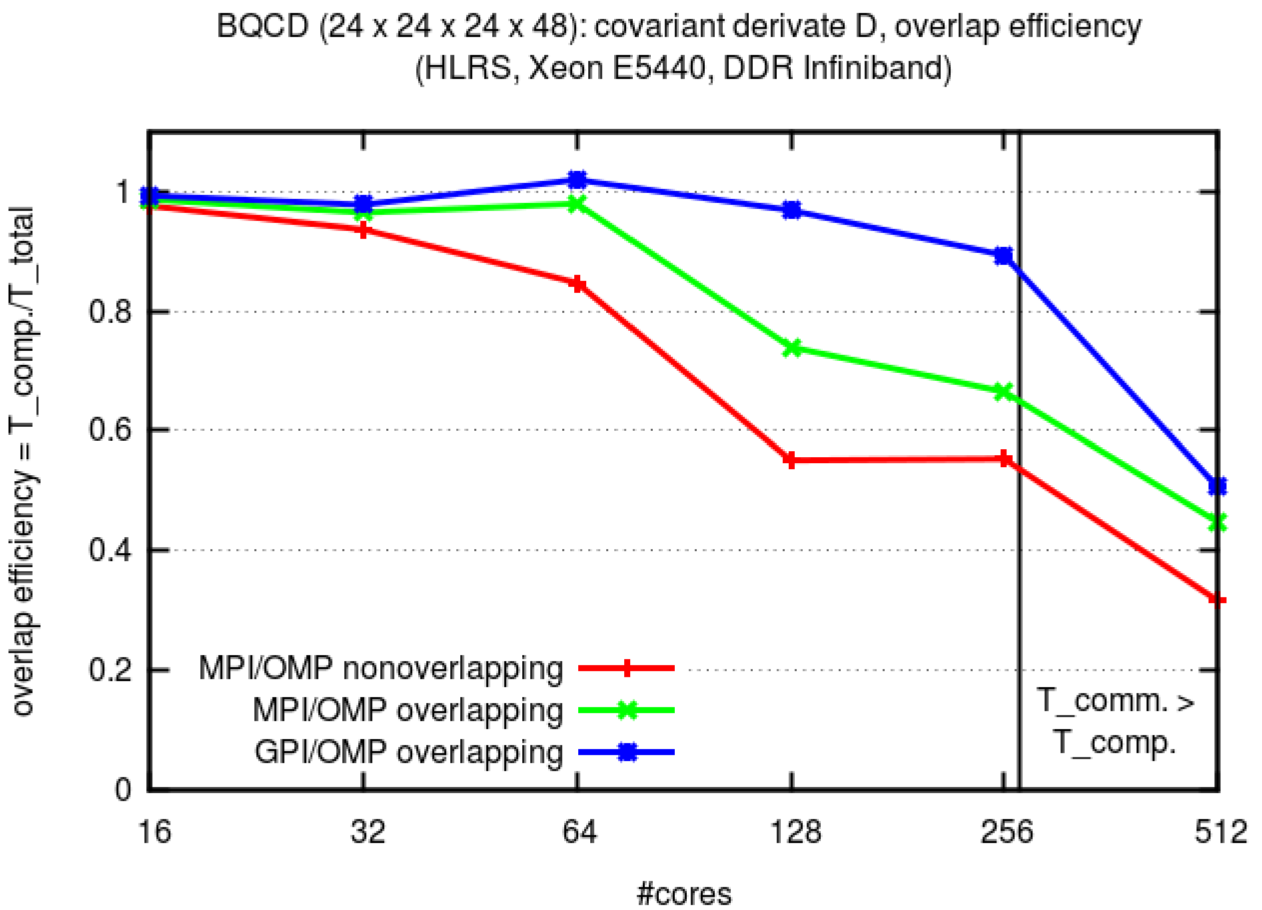
GPI – TAU (CFD solver) with increased speed-up



Fraunhofer ITWM

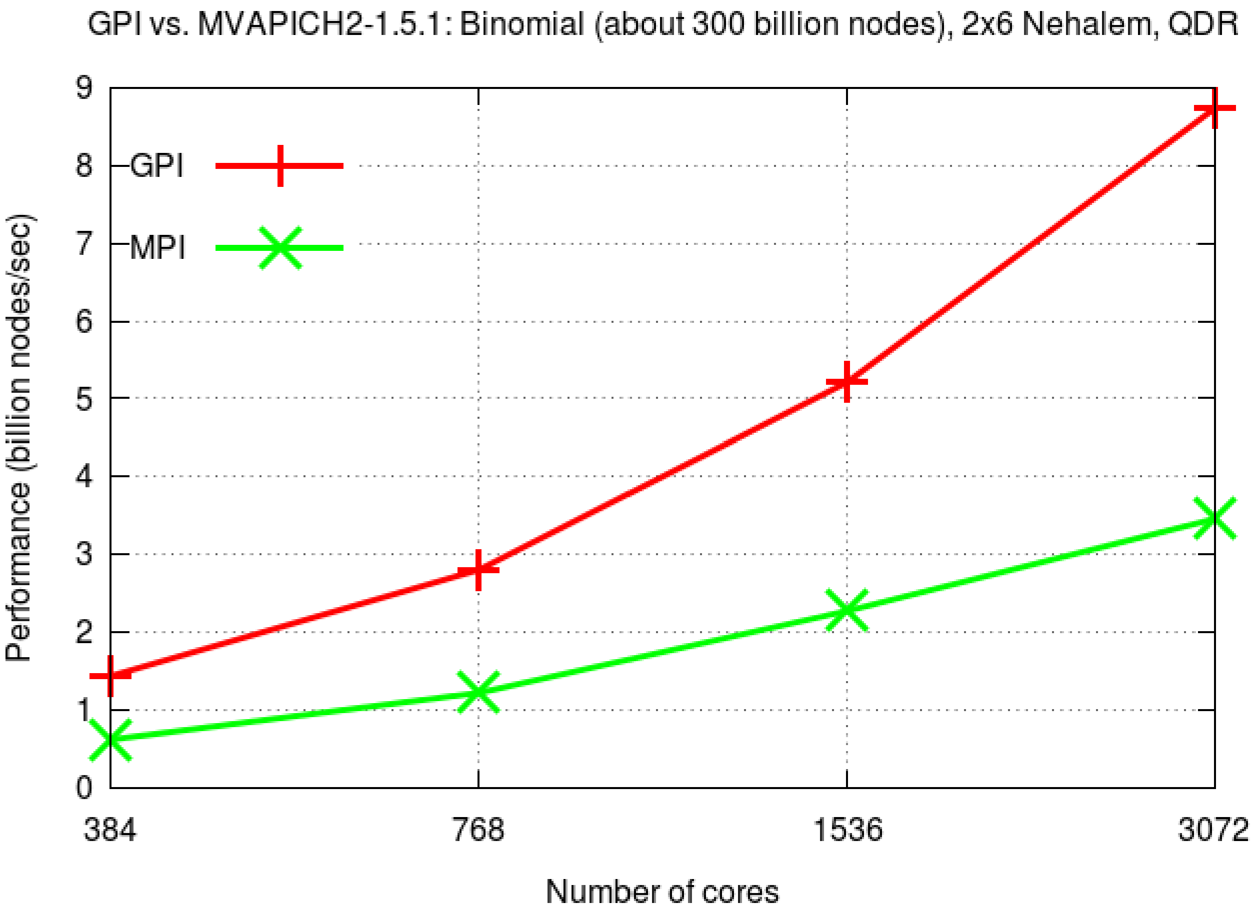
GPI – BQCD overlap efficiency

- Full overlap of communication with computation



Fraunhofer ITWM

UTS – Unbalanced Tree Search



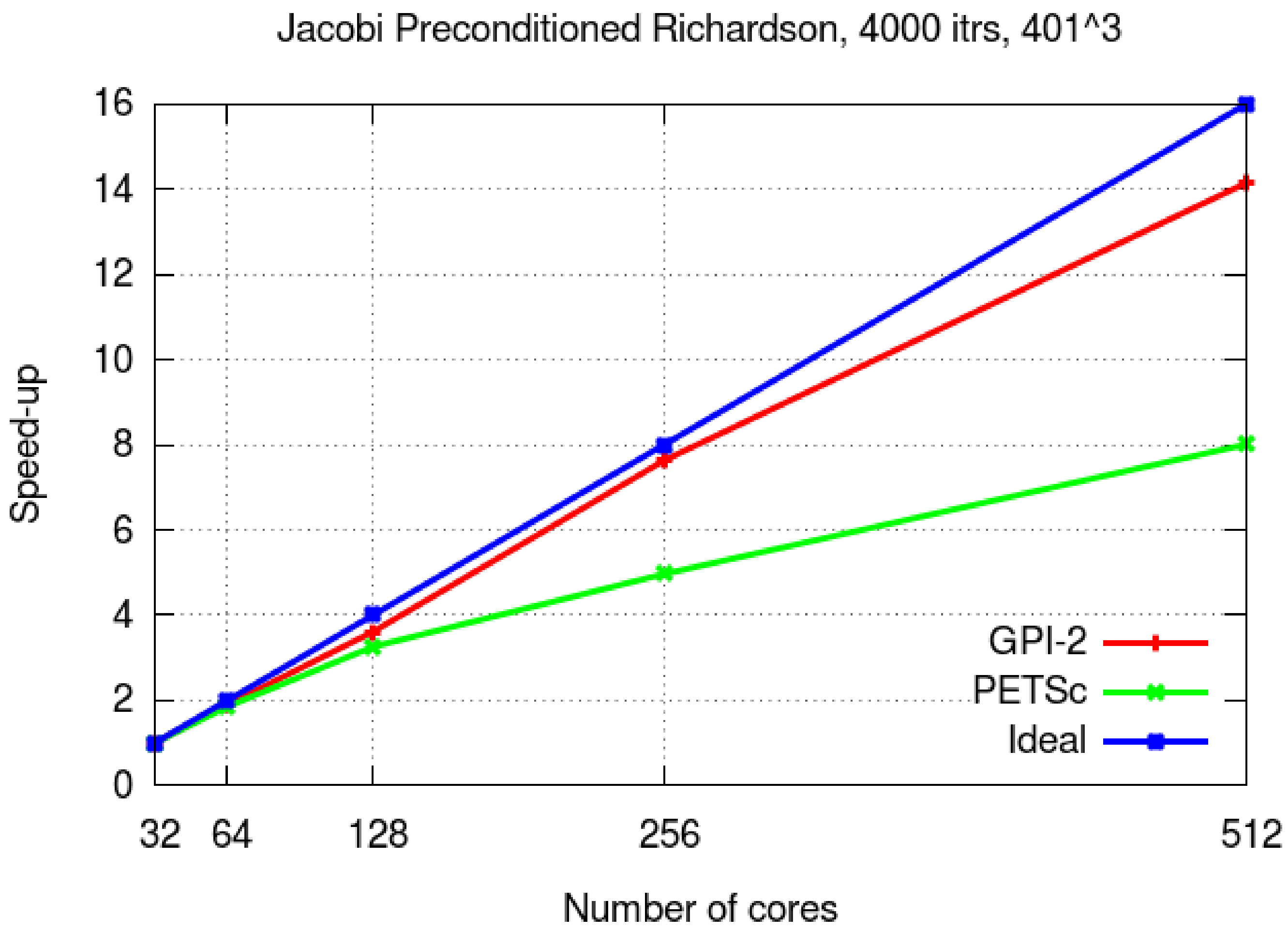
2.5x over MPI

89% relative efficiency

7.12 relative speedup

Fraunhofer ITWM

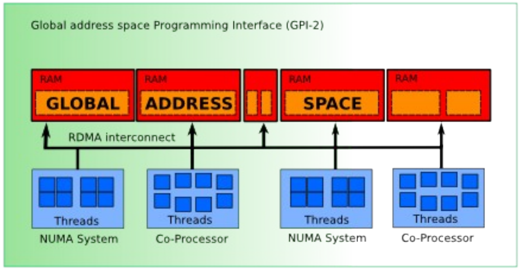
Jacobi preconditioned Richardson method



More information at:

www.gpi-site.com

github.com/cc-hpc-itwm/GPI-2



OPENFABRICS ALLIANCE

Fraunhofer ITWM

Final remarks

- Provided an overview of GPI-2 and its functionality
- API for asynchronous and scalable parallel applications
- Motivation to try it out
- Looking for users, cooperations, projects

Fraunhofer ITWM

Thank you for your attention

rui.machado [at] itwm.fraunhofer.de