HPC-FF – Overview and Experience

eGOTiT Seminar, 15th Dec

Nitya Hariharan
High Level Support Team
Max-Planck-Institut für Plasmaphysik, München, Germany

With contributions from:
IPP, Germany; University of Alicante, Spain

Outline

- HPC-FF
- Intel Nehalem Architecture
- Infiniband network
- BEUPACK Benchmark
- Shared Memory Segments
- CPU affinity

HPC-FF

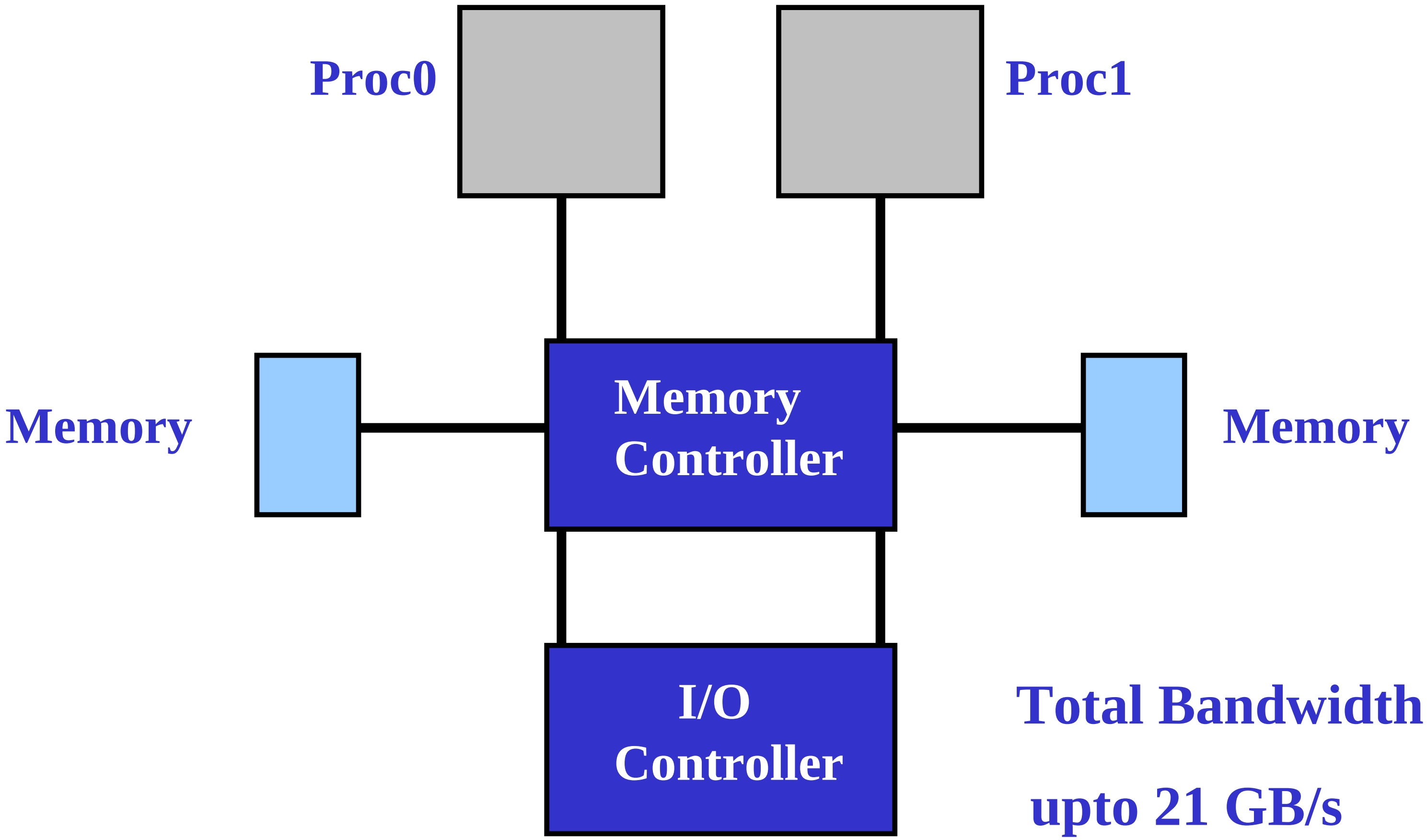
- Start of production – Aug 2009
- 1080 nodes @ 8 cores,
  peak performance 101 TFlop/s
- Quad core Intel Xeon 5570 (Nehalem) processors
- 24 Gb per node, Lustre filesystem
- Infiniband network with Mellanox switch
- ParTec MPI, Intel MPI

Intel Nehalem

- Supports SSE4.2 instructions
- QPI – Quick Path Interconnect
- Theoretical raw bandwidth – 25.6 GB/s
- Dynamic resource scaling
- NUMA architecture - low latency for local memory

Source - Intel

Before QPI

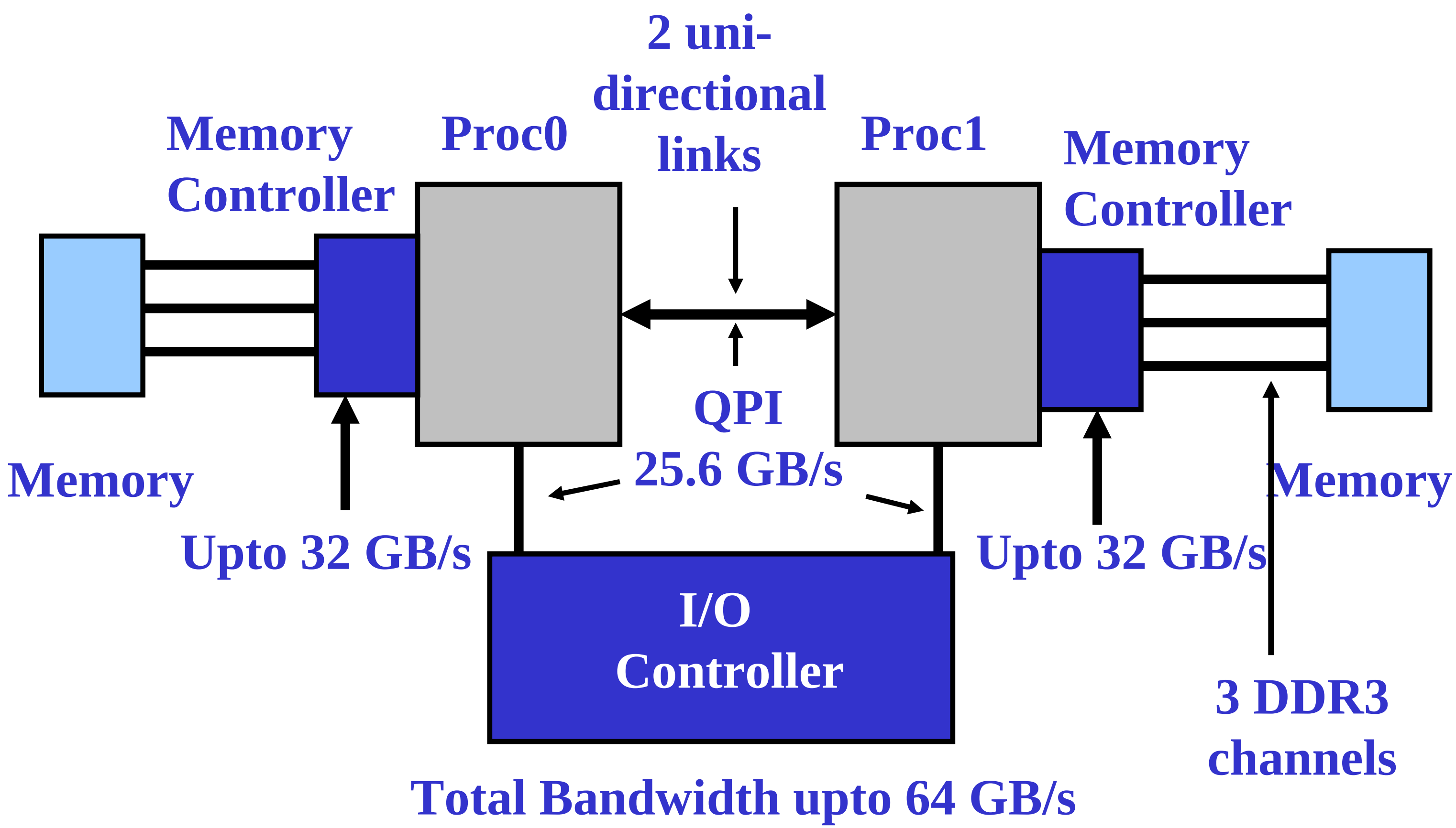


Total Bandwidth

upto 21 GB/s

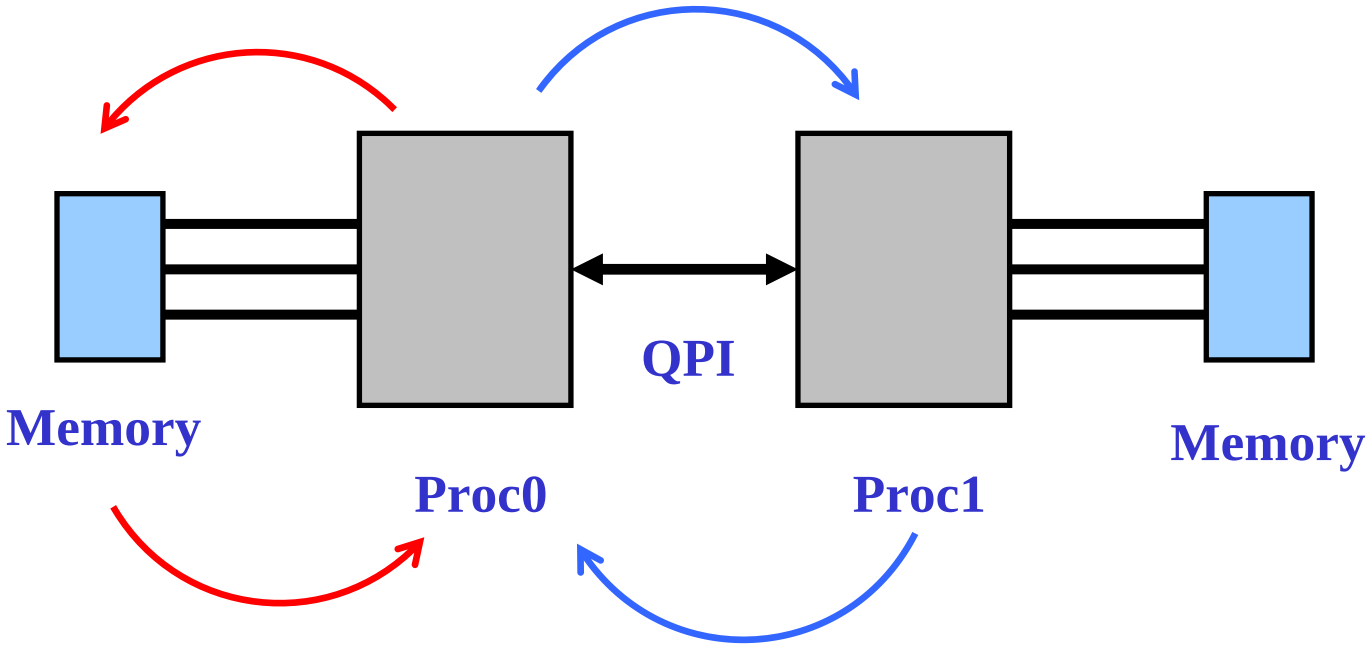
Source - Intel

QPI



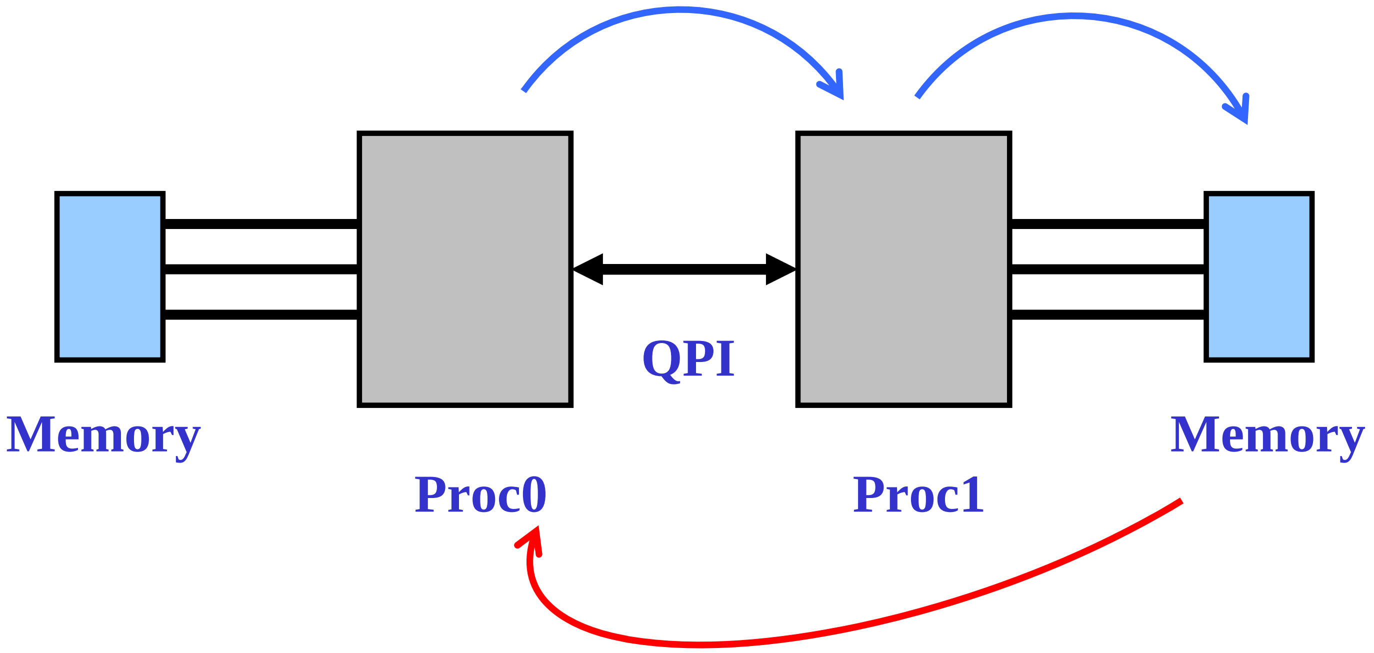
Source - Intel

Local Memory access



Source - Intel

Remote Memory access



Source - Intel

Infiniband(IB) Network

- Fabric/network topology for interconnecting compute and I/O nodes
- Nodes connected to the network through Host Channel Adaptors (HCAs)
- Queue based model, Send and Receive Queue (Queue Pair – QP)
- RDMA (Remote Direct Memory Access) also supported

RDMA

- Direct access to memory location of remote processes
- Uses zero-copy mechanism – origin and destination buffers are registered prior to use
- Virtual address and memory access key of remote process reqd
- Lower end-to-end latencies
- RDMA Read and Write

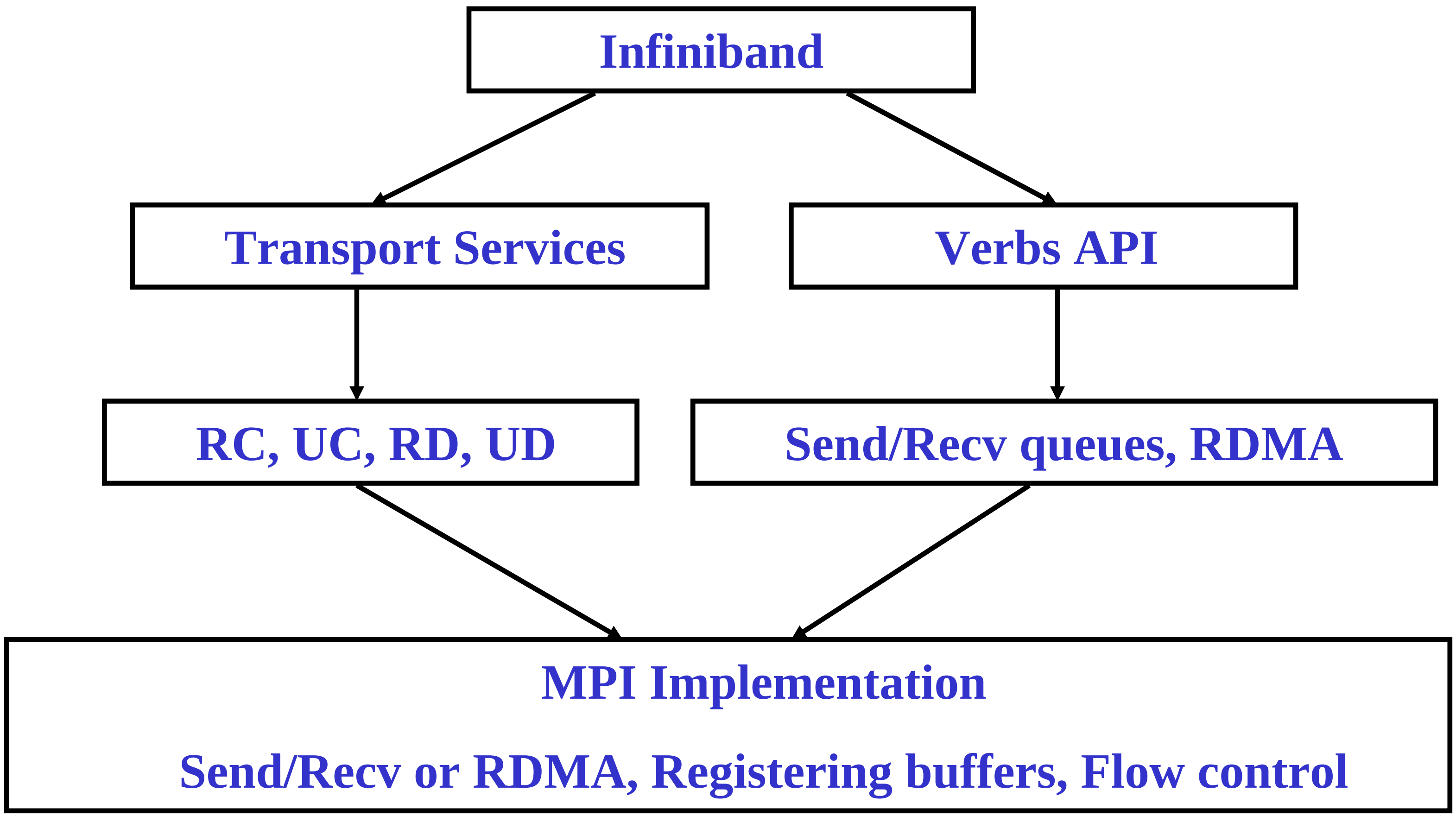
Infiniband

- **Different types of transport service**
  - ✓ **Reliable connection**
  - ✓ **Unreliable connection**
  - ✓ **Reliable datagram**
  - ✓ **Unreliable datagram**

- **On HPC-FF, RDMA writes with reliable connections as of now**

- **Beta implementation of Unreliable datagram**

IB and MPI


Infiniband
Transport Services
Verbs API
RC, UC, RD, UD
Send/Recv queues, RDMA
MPI Implementation
Send/Recv or RDMA, Registering buffers, Flow control

BEUPACK Benchmark

Fusion codes used as benchmark for IFERC procurement

- **ORB5**
- **GENE**
- **GEMR**
- **JOREK**
- **MDCASK**
- **GYSELA**

ORB5 benchmark

- Strong scaling – 2048×10^6 ions
- Weak scaling – 256×10^6 ions (256 cores) and then quadrupled
- Memory issues on 4096 cores, used PSP_ONDEMAND

Test cases by A. Bottino

PSP_ONDEMAND

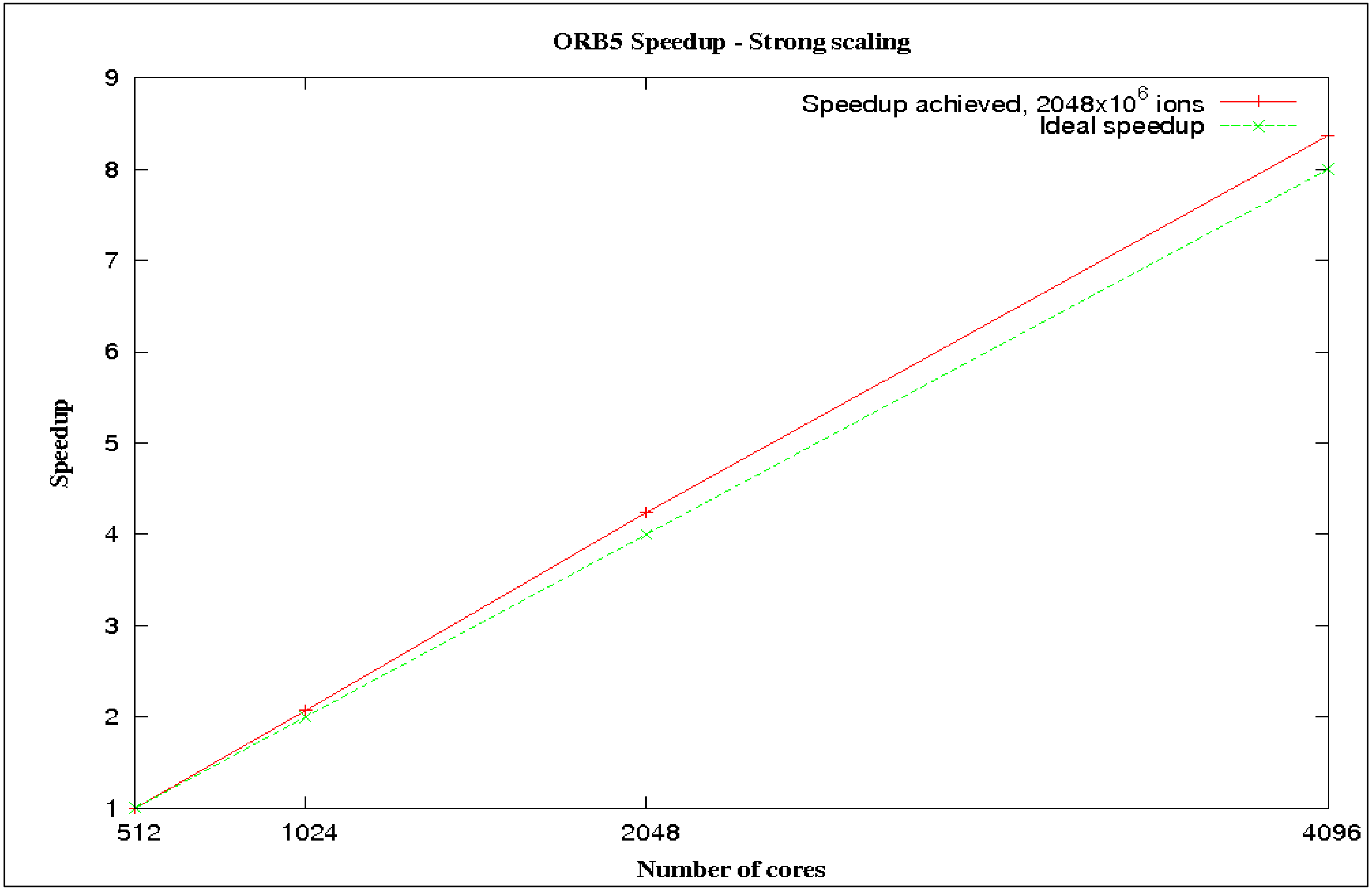
- ParTec MPI requires 0.55 MB per connection
- 16 buffers for Send and Receive
- ~2.2 GB per core for 4096 cores -> 500 MB left for application
- PSP_ONDEMAND=1, dynamic memory allocation

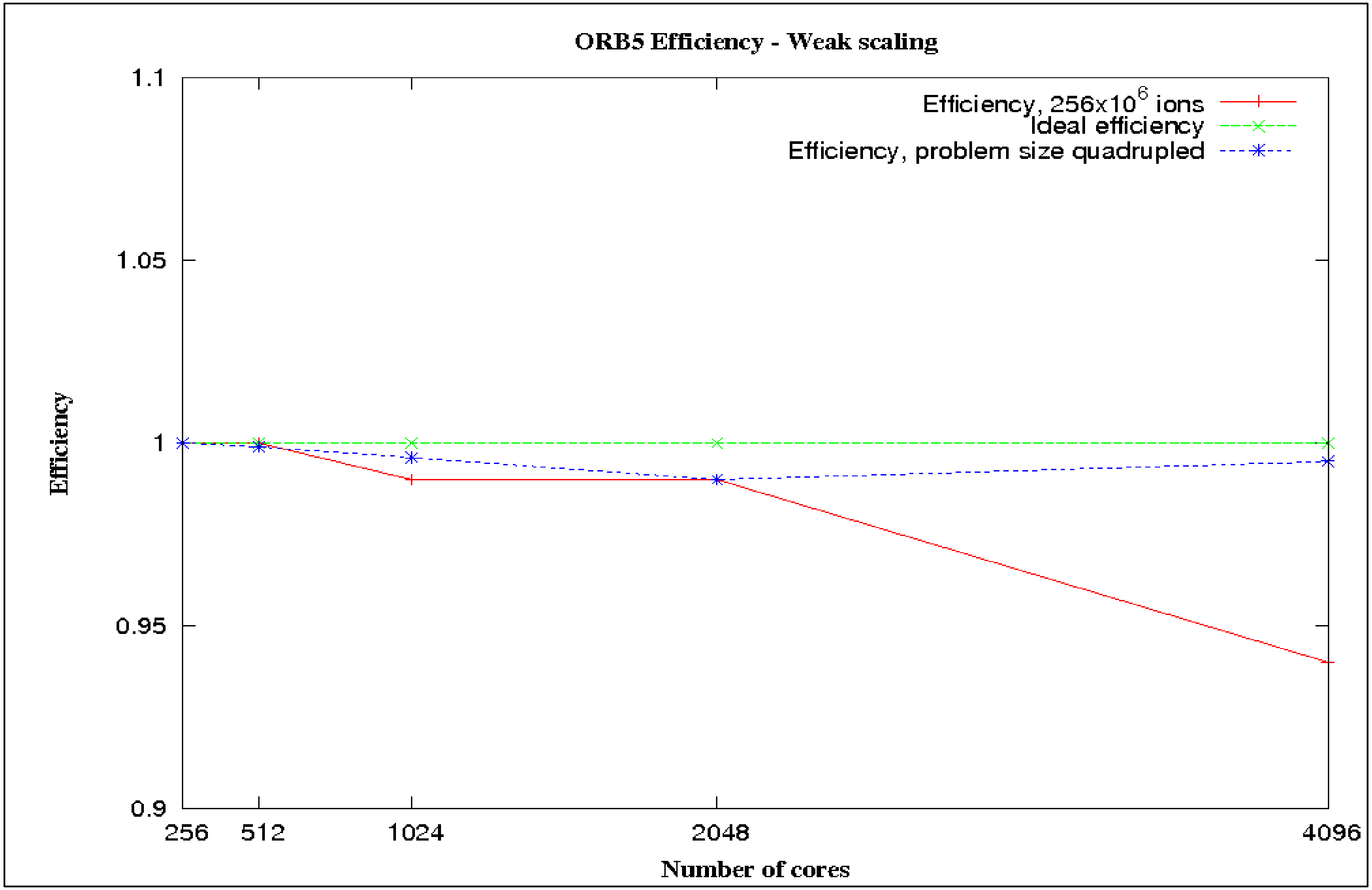
PSP_ONDEMAND

- **Works well as long as no all-to-all communication or large core numbers**

- **PSP_OPENIB_SENDQ_SIZE and PSP_OPENIB_RECVQ_SIZE**

- **Can degrade MPI's throughput and messaging rate,**
should be >= 3 to prevent deadlocks.


ORB5 Speedup - Strong scaling
Speedup achieved, 2048x10^6 ions
Ideal speedup
Speedup
Number of cores
1
2
3
4
5
6
7
8
9
512
1024
2048
4096


ORB5 Efficiency - Weak scaling
Efficiency, 256x10^6 ions
Ideal efficiency
Efficiency, problem size quadrupled
Efficiency
1.1
1.05
1
0.95
0.9
256
512
1024
2048
4096
Number of cores

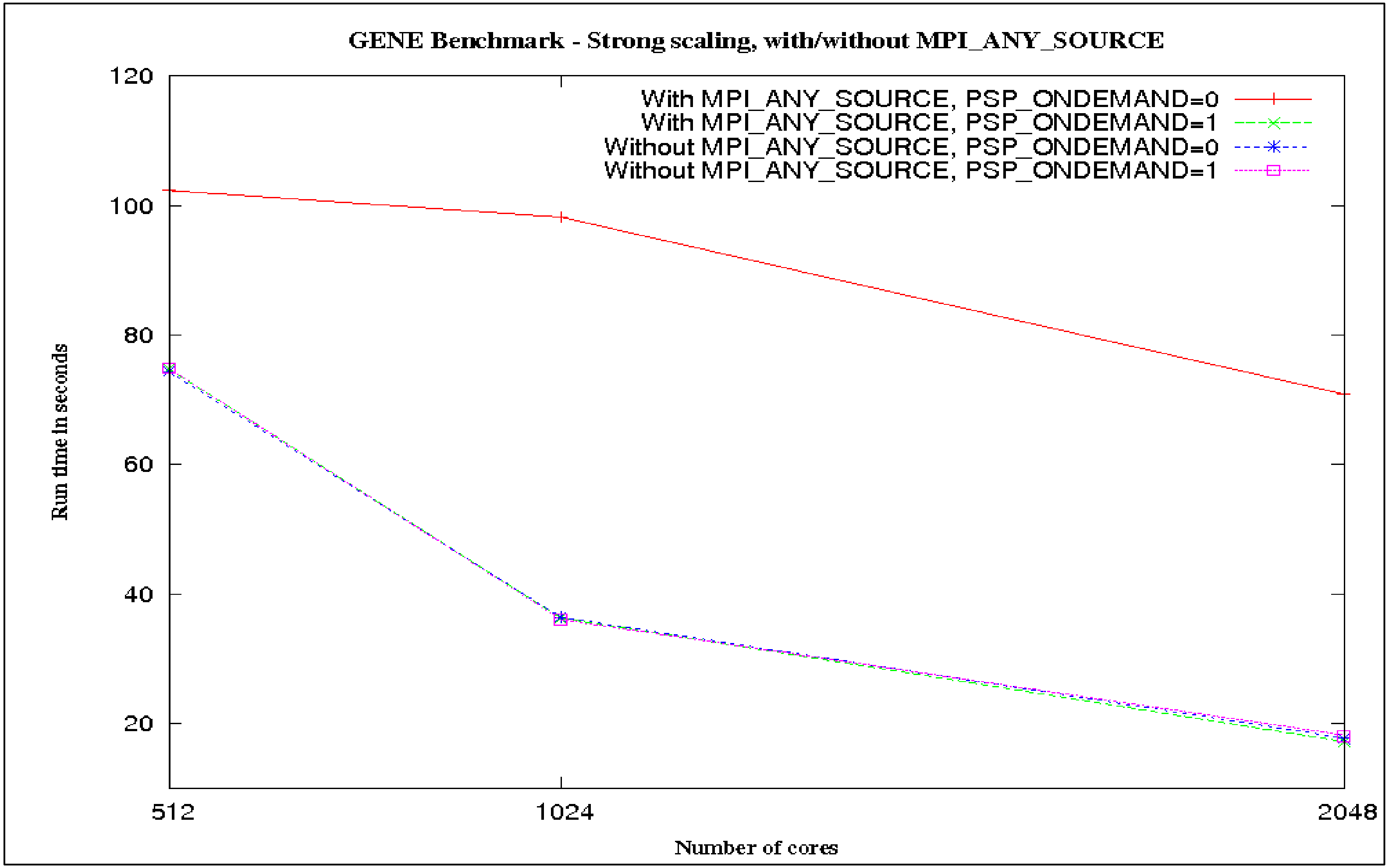
GENE benchmark

- **Strong scaling – 1.752 GB for 512 cores**
- **Weak scaling – 1.752 GB (constant)**
- **Usage of MPI_ANY_SOURCE**

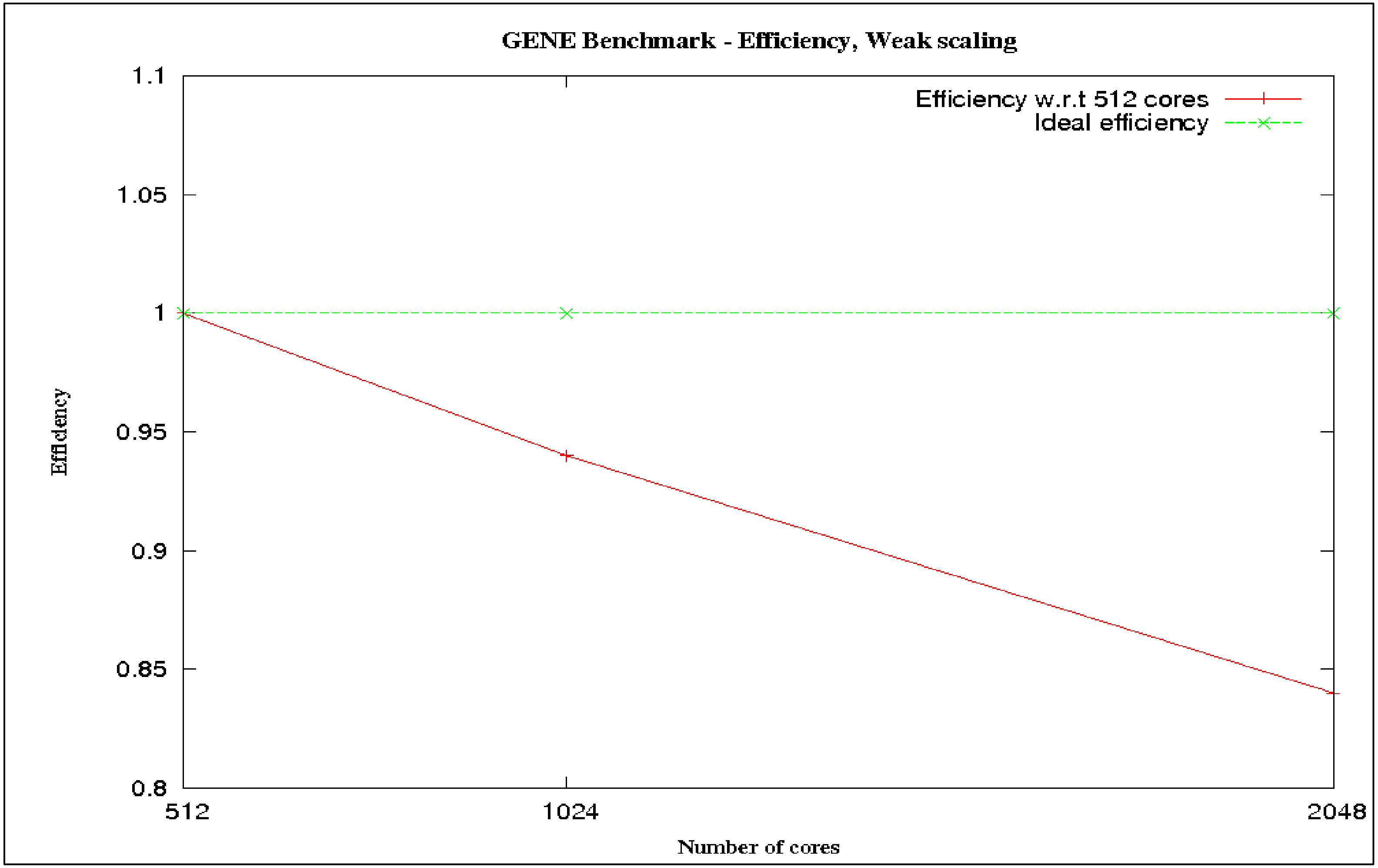
Test cases by T. Dannert

MPI_ANY_SOURCE

- **Communication lib has to poll a long list of receive queues**
- **Avoid it!**
- **When PSP_ONDEMAND=1, only one File Descriptor (FD) has all unestablished connections**
- **Few IB buffers from established connections and one FD**



GENE Benchmark - Strong scaling, with/without MPI_ANY_SOURCE
With MPI_ANY_SOURCE, PSP_ONDEMAND=0
With MPI_ANY_SOURCE, PSP_ONDEMAND=1
Without MPI_ANY_SOURCE, PSP_ONDEMAND=0
Without MPI_ANY_SOURCE, PSP_ONDEMAND=1
Run time in seconds
120
100
80
60
40
20
512
1024
2048
Number of cores


GENE Benchmark - Efficiency, Weak scaling
Efficiency w.r.t 512 cores
Ideal efficiency
1.1
1.05
1
0.95
0.9
0.85
0.8
Efficiency
512
1024
2048
Number of cores

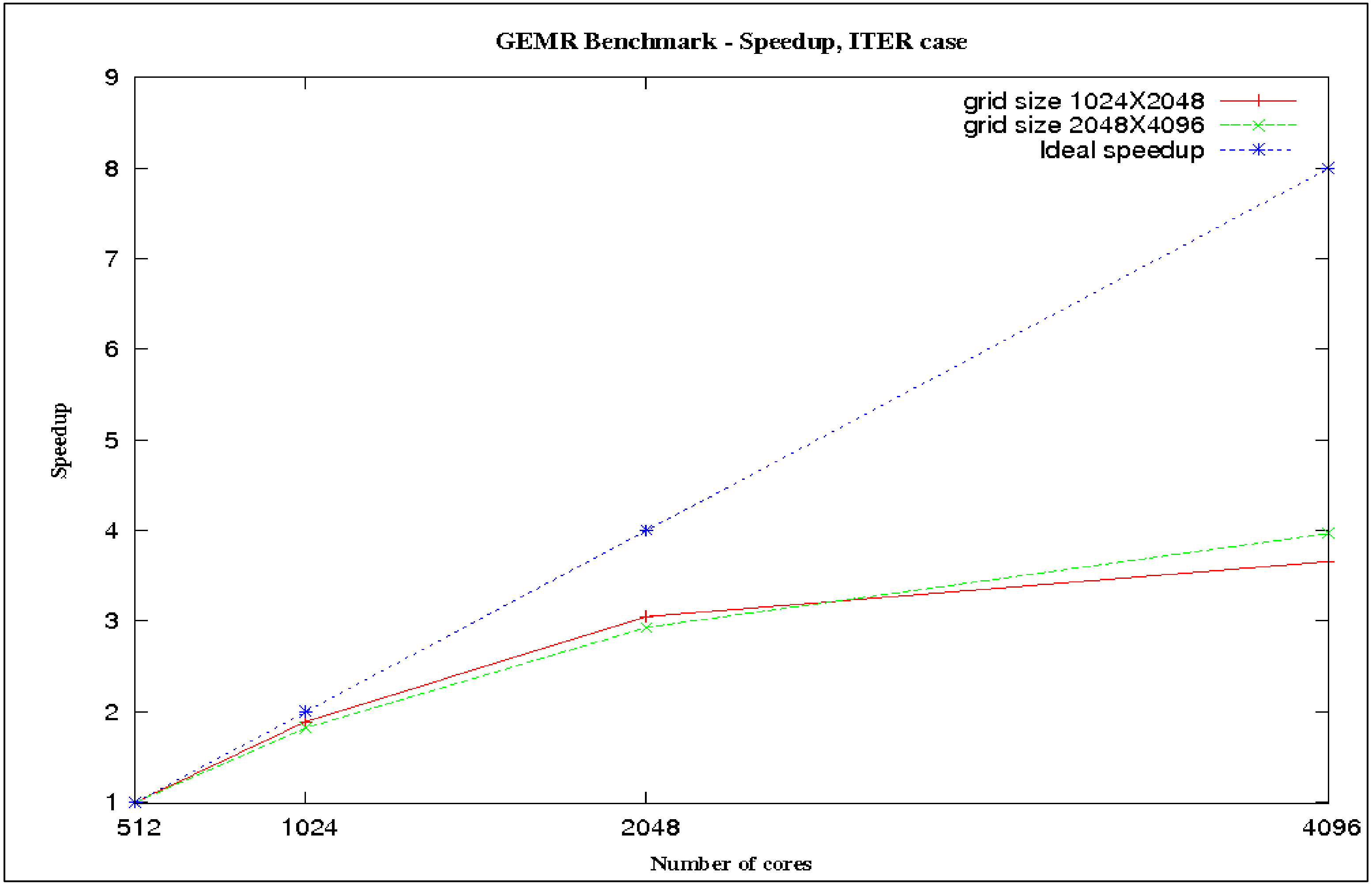
GEMR benchmark

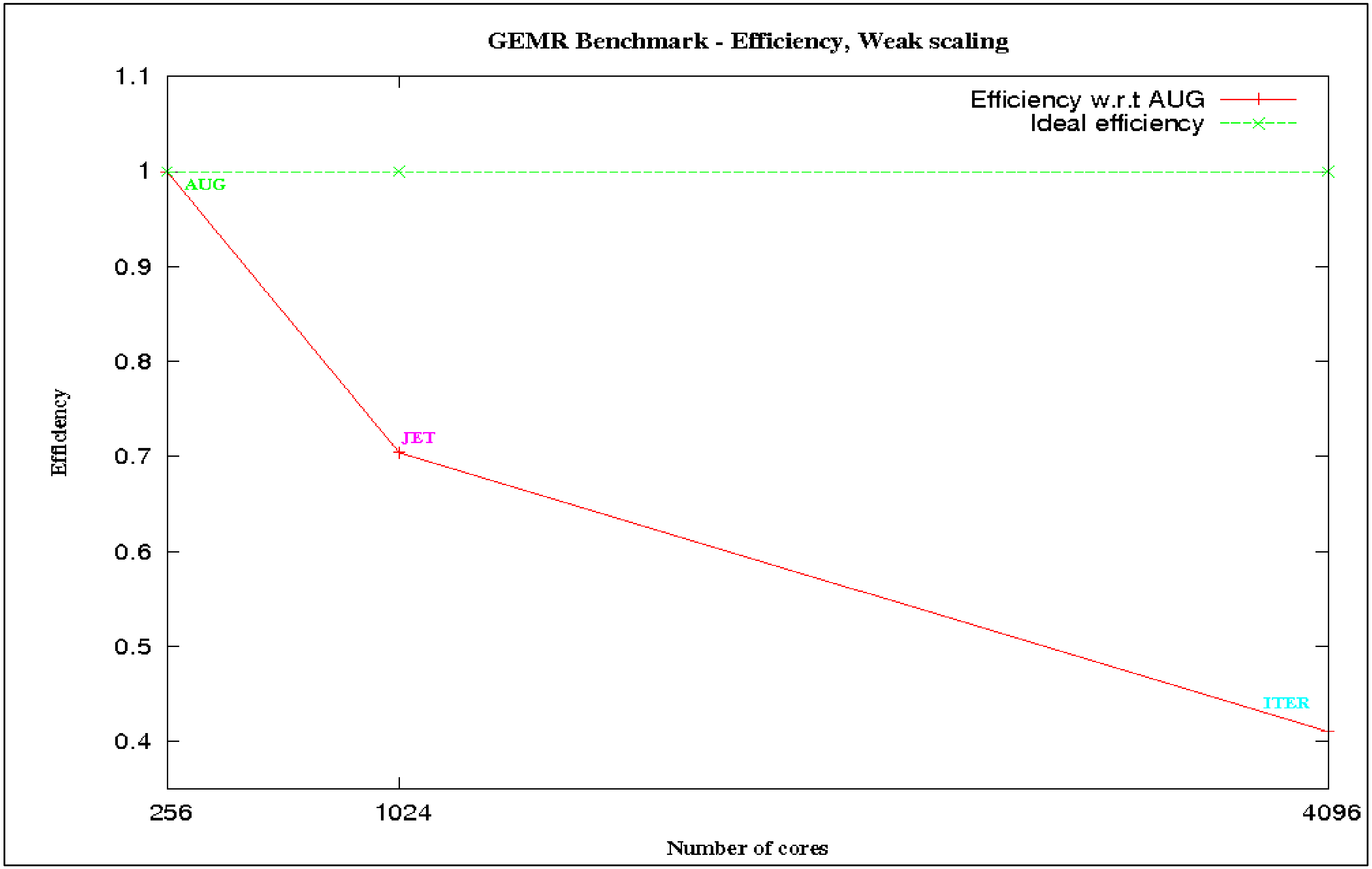
- **Strong scaling – ITER cases, grid size 1024x2048, 2048x4096**

- **Weak scaling – AUG, JET and ITER cases**
  - ✓ **AUG – 256x512**
  - ✓ **JET – 512x1024**
  - ✓ **ITER – 1024x2048**

- **No system related issues**

Test cases by B.D. Scott


GEMR Benchmark - Speedup, ITER case
grid size 1024X2048
grid size 2048X4096
Ideal speedup
Speedup
9
8
7
6
5
4
3
2
1
512
1024
2048
4096
Number of cores


GEMR Benchmark - Efficiency, Weak scaling
Efficiency w.r.t AUG
Ideal efficiency
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
Efficiency
AUG
JET
ITER
256
1024
4096
Number of cores

JOREK benchmark

- Only hybrid code as part of the benchmark
- Uses PastiX library – parallel solver for large sparse systems
- MPI_THREAD_MULTIPLE (MTM)
- MPI_THREAD_FUNNELED
  - errors with MPI_Start/MPI_Startall
- Test runs on IBM AIX, Sun Linux cluster AIMS (Intel MPI)
- Intel MPI installed on HPC-FF - /usr/local/impi/3.2.2.006/bin64/
- Working version of ParTec MPI with MTM support available

MDCASK benchmark

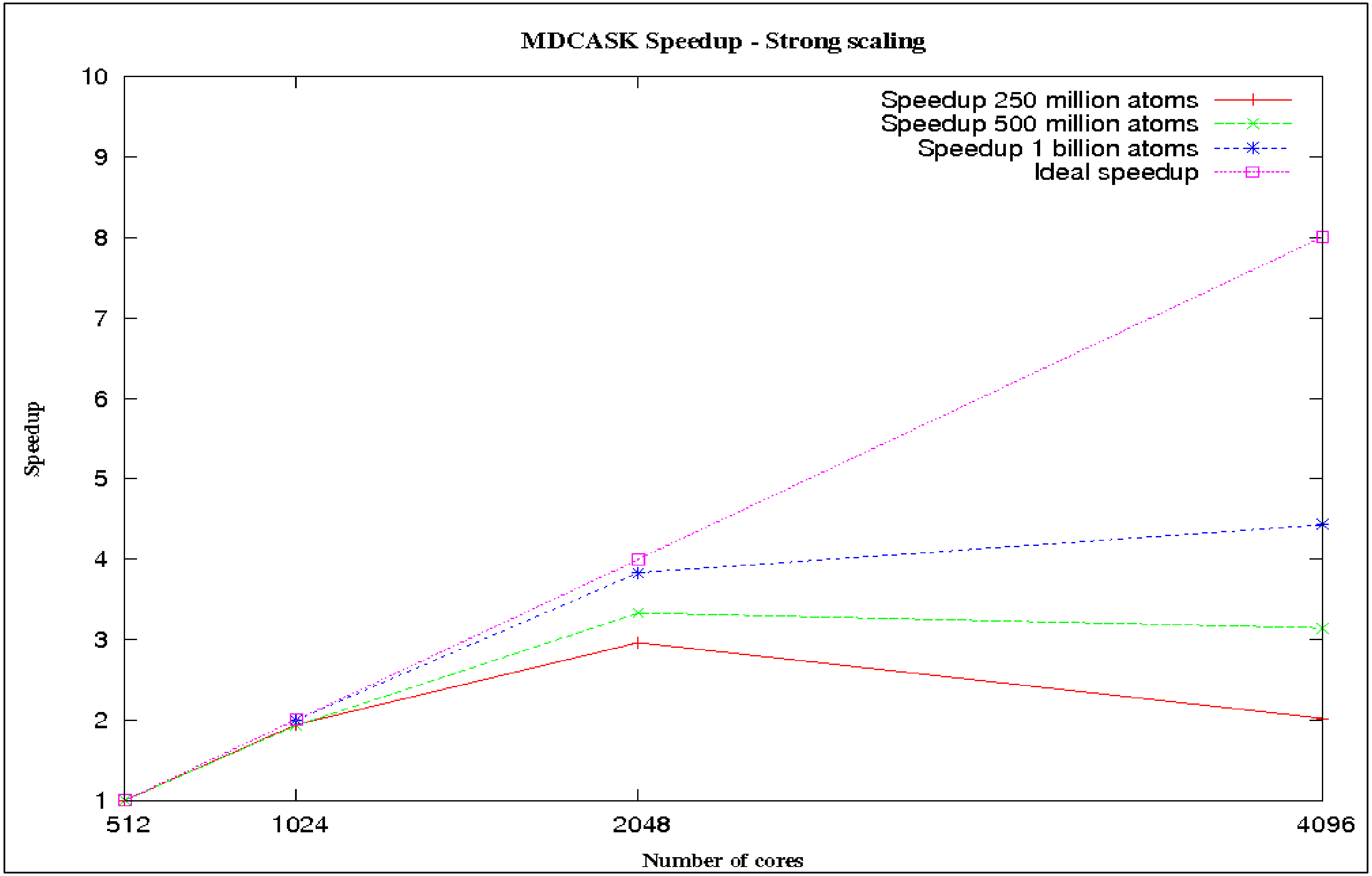
- Strong scaling – upto 1 billion atoms, too large for 512 cores (memory issues)

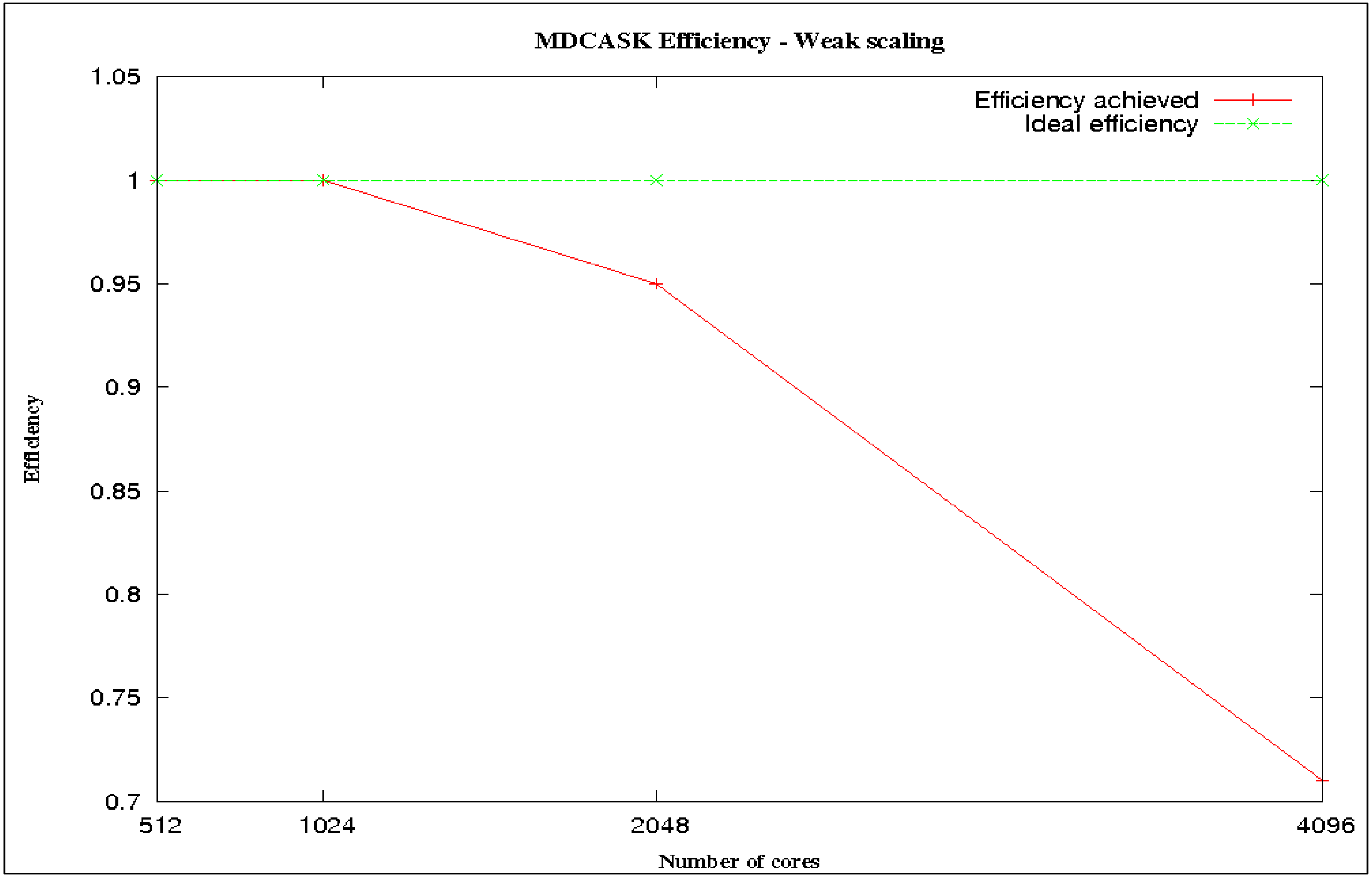
- Weak scaling – upto 2 billion atoms

- PSP_ONDEMAND – Intel MPI

- Intel does dynamic buffer allocation by default
I_MPI_USE_DYNAMIC_CONNECTIONS

Test cases by M.J. Caturla


MDCASK Speedup - Strong scaling
Speedup 250 million atoms
Speedup 500 million atoms
Speedup 1 billion atoms
Ideal speedup
Speedup
Number of cores
1
2
3
4
5
6
7
8
9
10
512
1024
2048
4096


MDCASK Efficiency - Weak scaling
Efficiency achieved
Ideal efficiency
1.05
1
0.95
0.9
0.85
0.8
0.75
0.7
Efficiency
512
1024
2048
4096
Number of cores

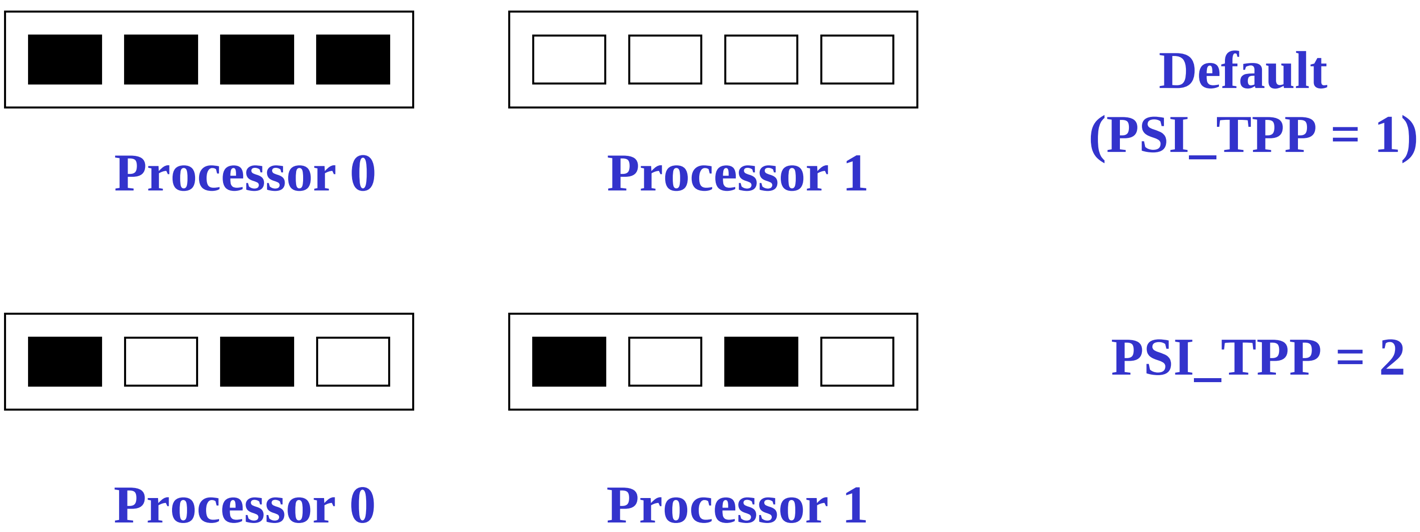
ParTec and Intel MPI

Memory consumption with static buffer allocation

Number of cores	ParTec MPI	Intel MPI
128	68 MB	68 MB
256	138 MB	139 MB
512	278 MB	281 MB
1024	558 MB	565 MB
2048	1118 MB	Program crash

PSI_TPP

- NUMA design
- Memory separated into two nodes of 12 GB each



PSI_TPP

ppn	PSI_TPP	Number of cores per compute node	Memory per core	Explanation
8	1 (default)	8	3	1 core allocated per task.
8	2	4	6	2 cores allocated per task, 2*3GB memory available per core. Doubled bandwidth.
8	4	2	12	4 cores allocated per task.
4	1 (default)	4	3	1 core allocated per task.

PSI_TPP

Benchmark (512 cores)	Run time in seconds (PSI_TPP=1)	Run time in seconds (PSI_TPP=2)	%Gain
ORB5	309.34	280.12	9.45
MDCASK – 250 Million atoms	4.80 (per time step)	4.64 (per time step)	3.33
MDCASK – 500 Million atoms	9.26 (per time step)	8.82 (per time step)	4.75
GEMR – ITER case (problem size 4096x2048)	2730.28	1985.62	27.27
GEMR – ITER case (problem size 2048x1024)	682.37	428.7	37.17
GENE	74.87	49.99	33.23

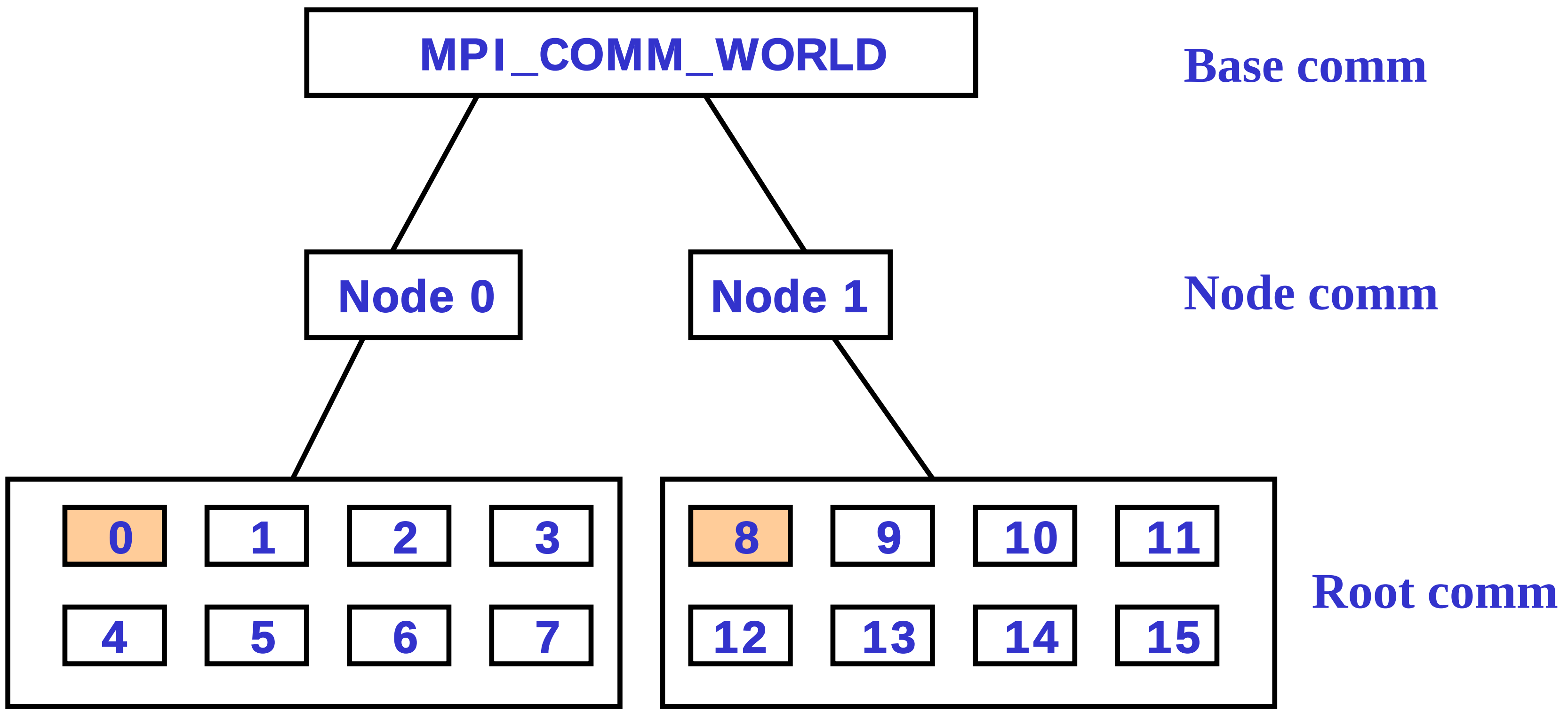
Shared Memory Segments (SMS)

- SMS – area of memory accessible by more than one process on the same node
- Based on System V IPC (Inter Process Communication) framework
- FIPC (Fortran IPC) module by Ian Bush
- Fortran and C interoperability
- Hybrid codes – MPI and OpenMP, effort in adding OpenMP to existing MPI codes
- Useful for machines like HPC-FF
- More portable, no thread-safe MPI reqd

FIPC

- **Create a context –** ***fipc_init***
- **Create a SMS in this context –** ***fipc_seg_create***
- **Processes with an SMP node share the memory segment**
- **Need semaphores –** ***fipc_critical_start/end***
- **Routines like** ***fipc_allreduce*****,** ***fipc_ctxt_rank*****,** ***fipc_ctxt_size***

FIPC Context



Splines and SMS

- Tested with Cubic Splines from Numerical Recipes

- Two functions *spline* and *splint*

- Input data *x*, *f(x)* and output of *spline* can be used as SMS.

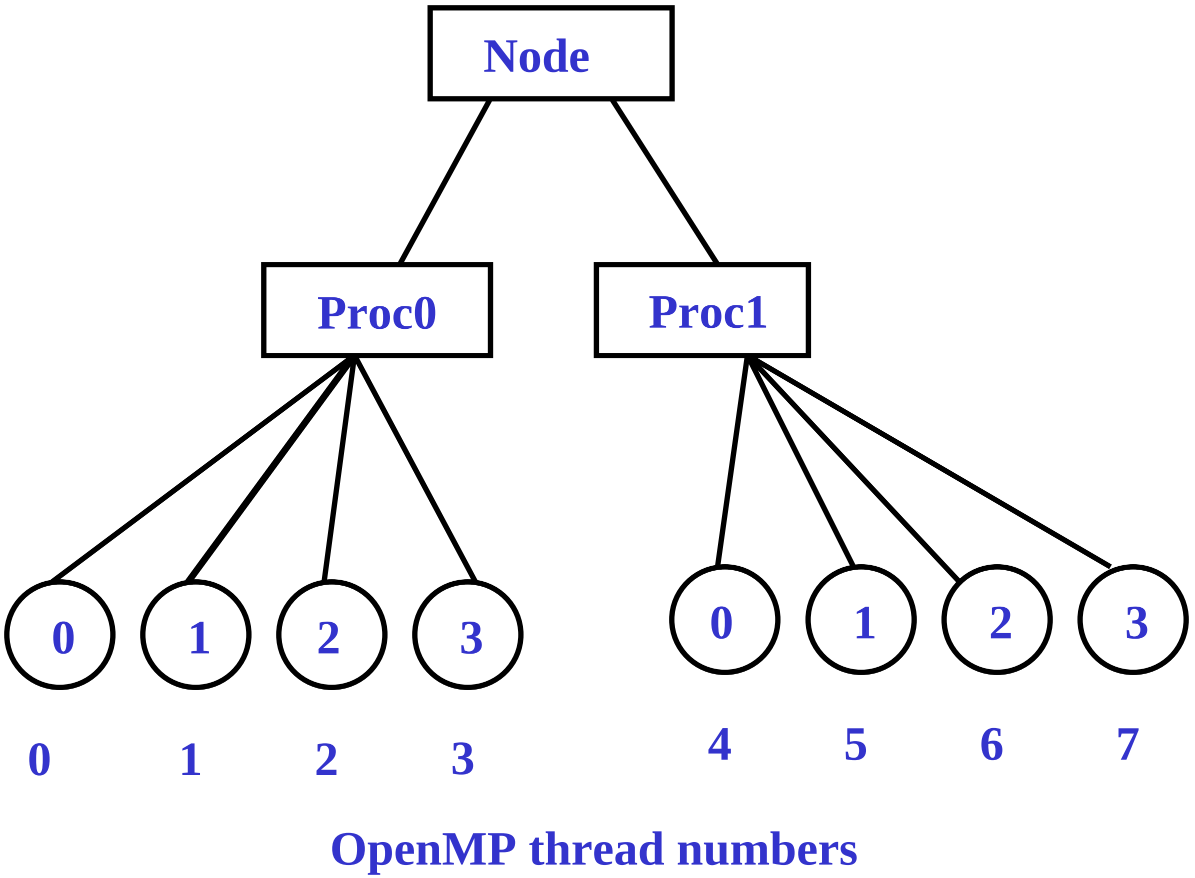
- Will be used for EZspline library for ASCOT.

- Downside, SMS not deleted after abnormal termination. Need additional processing. On batch nodes, cleaned up by ParTec software, but can still be a problem with multiple jobs.

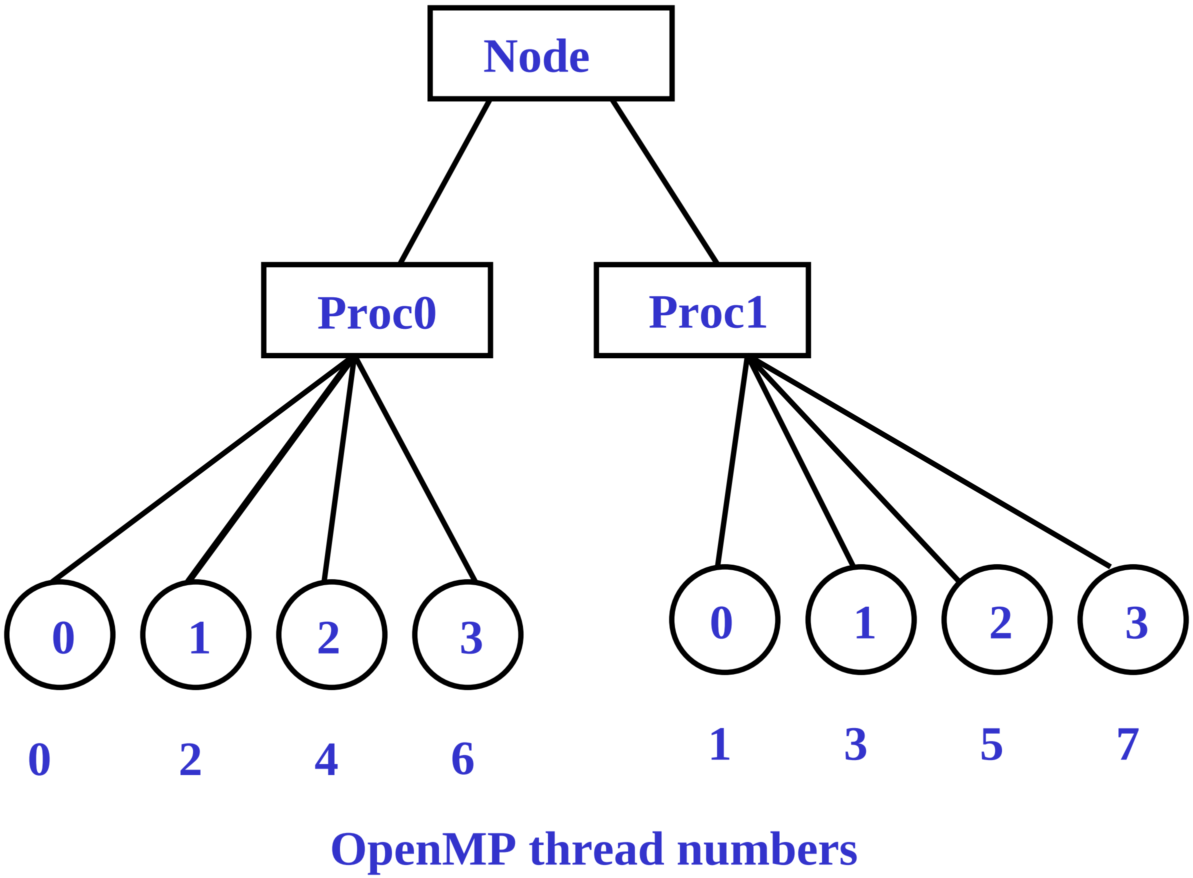
CPU Affinity

- Intel's OpenMP runtime library allows OpenMP threads to be bound to physical processing units
- Can have a dramatic impact on the runtime of the code
- Need to use KMP_AFFINITY
  - ✓ Can determine machine topology and assign OpenMP threads to processors
- On HPC-FF, KMP_AFFINITY overwritten by PSI daemon
- Need to export two other variables
  - ✓ __PSI_NO_PINPROC -> 1
  - ✓ __PSI_NO_MEMBIND -> 1

KMP_AFFINITY=compact



KMP_AFFINITY=scatter



Additional information

- Use module commands in the batch script
- To check correct paths of the executable, use ldd in the batch script
- Not all .a files found in the compute nodes, compile in the login node while debugging.
  - Statically linked files have a '.a' extension, so an executable compiled in the login node can still run on the compute node.
  - Compiling on the compute node will not work as all the .a files are required during the compilation process.

Summary

- Two versions of ParTec MPI, Intel MPI

- MPI consumes memory, use PSP_ONDEMAND where ever possible

- Avoid MPI_ANY_SOURCE

- Use PSI_TPP to increase available memory and bandwidth

Acknowledgements

- Juelich Support - Alexander Schnurpfeil, ParTec - Jens Hauke
- ORB5 – Alberto Bottino, IPP
- GENE – Tilman Dannert, IPP
- GEMR – Bruce D Scott, IPP
- JOREK – Guido Huysmans, CEA; Florent Sourbier (Benchmark), CEA
- MDCASK – Maria Jose Caturla, Univ of Alicante
- GYSELA - Virginie Grandgirard (Benchmark on HPC-FF), CEA
- FIPC module - Ian Bush, NAG