Performance Evaluation of MPI, UPC and OpenMP on Multicore Architectures

Damián A. Mallón[1]*, Guillermo L. Taboada[2], Carlos Teijeiro[2], Juan Touriño[2], Basilio B. Fraguela[2], Andrés Gómez[1], Ramón Doallo[2], José Carlos Mouriño[1]

[1] Galicia Supercomputing Center (CESGA)
Santiago de Compostela (Spain)
{dalvarez,agomez,jmourino}@cesga.es

[2] Computer Architecture Group
University of A Coruña (Spain)
{taboada,cteijeiro,juan,basilio,doallo}@udc.es

16th European PVM/MPI Users' Group Conference (EuroPVM/MPI '09),
CSC-IT Center for Science, Espoo, Finland

High-Performance Computing Systems

Nowadays HPC Systems:

- Mainly multicore clusters
- Usually programmed with:
  - MPI for Distributed Memory systems
  - OpenMP for Shared Memory systems
  - MPI+OpenMP for hybrid systems (multicore clusters)

Programming Alternatives:

- Partitioned Global Array Space (PGAS) languages

High-Performance Computing Systems

Nowadays HPC Systems:

- Mainly multicore clusters
- Usually programmed with:
  - MPI for Distributed Memory systems
  - OpenMP for Shared Memory systems
  - MPI+OpenMP for hybrid systems (multicore clusters)

Programming Alternatives:

- Partitioned Global Array Space (PGAS) languages

PGAS languages

PGAS Memory Model:

2 Memory spaces:

- Shared memory space
  - Addressable by all threads
  - Affinity to one thread
- Private memory space

PGAS Languages:

- CoArray Fortran (Fortran based)
- Titanium (Java based)
- UPC (C based)*

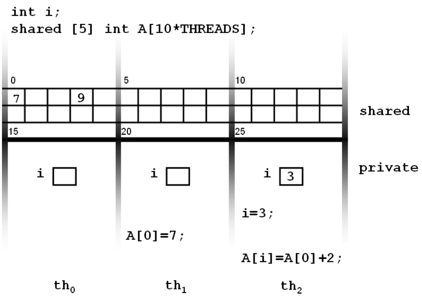


Figure: Shared and private spaces in PGAS

PGAS languages

PGAS Memory Model:

2 Memory spaces:

- Shared memory space
  - Addressable by all threads
  - Affinity to one thread
- Private memory space

PGAS Languages:

- CoArray Fortran (Fortran based)
- Titanium (Java based)
- UPC (C based)*

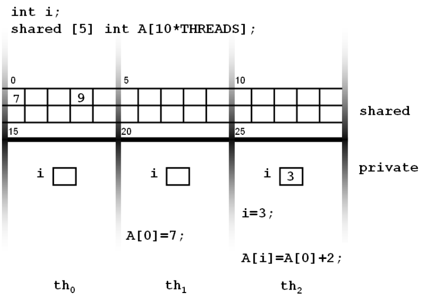


Figure: Shared and private spaces in PGAS

Unified Parallel C (UPC)

UPC:

- Support from academia and industry
- Several free and commercial implementations:
  - Berkeley UPC
  - Intrepid (GCC-UPC)
  - HP UPC
  - IBM UPC
  - Cray UPC
  - ...
- Question: Is it suitable for HPC? How does it perform?

Unified Parallel C (UPC)

UPC:

- Support from academia and industry
- Several free and commercial implementations:
  - Berkeley UPC
  - Intrepid (GCC-UPC)
  - HP UPC
  - IBM UPC
  - Cray UPC
  - ...
- Question: Is it suitable for HPC? How does it perform?

Unified Parallel C (UPC)

UPC:

- Support from academia and industry
- Several free and commercial implementations:
  - Berkeley UPC
  - Intrepid (GCC-UPC)
  - HP UPC
  - IBM UPC
  - Cray UPC
  - ...
- Question: Is it suitable for HPC? How does it perform?

MPI/OpenMP/UPC Benchmarking

MPI and OpenMP Benchmarking:

MPI and OpenMP are well established approaches:

- Tons of available benchmarks
  - NAS Parallel Benchmarks (NPB)
  - SPEC Benchmarks
  - Intel MPI Benchmarks
  - PARKBENCH
  - Just to name a few...

MPI/OpenMP/UPC Benchmarking

MPI and OpenMP Benchmarking:

MPI and OpenMP are well established approaches:

- Tons of available benchmarks
  - NAS Parallel Benchmarks (NPB)
  - SPEC Benchmarks
  - Intel MPI Benchmarks
  - PARKBENCH
  - Just to name a few...

MPI/OpenMP/UPC Benchmarking

MPI and OpenMP Benchmarking:

MPI and OpenMP are well established approaches:

- Tons of available benchmarks
  - NAS Parallel Benchmarks (NPB)
  - SPEC Benchmarks
  - Intel MPI Benchmarks
  - PARKBENCH
  - Just to name a few...

MPI/OpenMP/UPC Benchmarking

MPI and OpenMP Benchmarking:

MPI and OpenMP are well established approaches:

- Tons of available benchmarks
  - NAS Parallel Benchmarks (NPB)
  - SPEC Benchmarks
  - Intel MPI Benchmarks
  - PARKBENCH
  - Just to name a few...

MPI/OpenMP/UPC Benchmarking

MPI and OpenMP Benchmarking:

MPI and OpenMP are well established approaches:

- Tons of available benchmarks
  - NAS Parallel Benchmarks (NPB)
  - SPEC Benchmarks
  - Intel MPI Benchmarks
  - PARKBENCH
  - Just to name a few...

MPI/OpenMP/UPC Benchmarking

MPI and OpenMP Benchmarking:

MPI and OpenMP are well established approaches:

- Tons of available benchmarks
  - NAS Parallel Benchmarks (NPB)
  - SPEC Benchmarks
  - Intel MPI Benchmarks
  - PARKBENCH
  - Just to name a few...

MPI/OpenMP/UPC Benchmarking

UPC Benchmarking:

UPC is an emerging alternative:

- Few benchmarks available
- The main source is the Berkeley UPC distribution
- Moreover, the benchmarks have to be available in MPI, OpenMP and UPC
- Even less alternatives
- However, the main parallel benchmarking suite is available (NPB)
- In addition, this work uses two more kernels: Matrix Multiplication and Sobel Edge Detection (MPI/OpenMP/UPC)

MPI/OpenMP/UPC Benchmarking

UPC Benchmarking:

UPC is an emerging alternative:

- Few benchmarks available
- The main source is the Berkeley UPC distribution
- Moreover, the benchmarks have to be available in MPI, OpenMP and UPC
- Even less alternatives
- However, the main parallel benchmarking suite is available (NPB)
- In addition, this work uses two more kernels: Matrix Multiplication and Sobel Edge Detection (MPI/OpenMP/UPC)

MPI/OpenMP/UPC Benchmarking

UPC Benchmarking:

UPC is an emerging alternative:

- Few benchmarks available
- The main source is the Berkeley UPC distribution
- Moreover, the benchmarks have to be available in MPI, OpenMP and UPC
- Even less alternatives
- However, the main parallel benchmarking suite is available (NPB)
- In addition, this work uses two more kernels: Matrix Multiplication and Sobel Edge Detection (MPI/OpenMP/UPC)

MPI/OpenMP/UPC Benchmarking

UPC Benchmarking:

UPC is an emerging alternative:

- Few benchmarks available
- The main source is the Berkeley UPC distribution
- Moreover, the benchmarks have to be available in MPI, OpenMP and UPC
- Even less alternatives
- However, the main parallel benchmarking suite is available (NPB)
- In addition, this work uses two more kernels: Matrix Multiplication and Sobel Edge Detection (MPI/OpenMP/UPC)

MPI/OpenMP/UPC Benchmarking

UPC Benchmarking:

UPC is an emerging alternative:

- Few benchmarks available
- The main source is the Berkeley UPC distribution
- Moreover, the benchmarks have to be available in MPI, OpenMP and UPC
- Even less alternatives
- However, the main parallel benchmarking suite is available (NPB)
- In addition, this work uses two more kernels: Matrix Multiplication and Sobel Edge Detection (MPI/OpenMP/UPC)

MPI/OpenMP/UPC Benchmarking

UPC Benchmarking:

UPC is an emerging alternative:

- Few benchmarks available
- The main source is the Berkeley UPC distribution
- Moreover, the benchmarks have to be available in MPI, OpenMP and UPC
- Even less alternatives
- However, the main parallel benchmarking suite is available (NPB)
- In addition, this work uses two more kernels: Matrix Multiplication and Sobel Edge Detection (MPI/OpenMP/UPC)

MPI/OpenMP/UPC Benchmarking

NAS Parallel Benchmarks (NPB)

- Composed of kernels and pseudo-applications extracted from CFD codes.
- They have various *classes*, to increase the datasets and hence allow the researchers to use the NPB to evaluate small and large systems

MPI/OpenMP/UPC Benchmarking

NAS Parallel Benchmarks (NPB)

- Composed of kernels and pseudo-applications extracted from CFD codes.
- They have various *classes*, to increase the datasets and hence allow the researchers to use the NPB to evaluate small and large systems

MPI/OpenMP/UPC Benchmarking

NPB Kernels

- CG: an iterative solver that tests regular communications in sparse matrix-vector multiplications
- EP: an embarrassingly parallel code that assesses the floating point performance
- FT: performs series of 1-D FFTs on a 3-D mesh that tests aggregated communication performance
- IS: a large integer sort that evaluates both integer computation performance and the aggregated communication throughput
- MG: a simplified multigrid kernel that performs both short and long distance communications

MPI/OpenMP/UPC Benchmarking

NPB Kernels

- CG: an iterative solver that tests regular communications in sparse matrix-vector multiplications
- EP: an embarrassingly parallel code that assesses the floating point performance
- FT: performs series of 1-D FFTs on a 3-D mesh that tests aggregated communication performance
- IS: a large integer sort that evaluates both integer computation performance and the aggregated communication throughput
- MG: a simplified multigrid kernel that performs both short and long distance communications

MPI/OpenMP/UPC Benchmarking

NPB Kernels

- CG: an iterative solver that tests regular communications in sparse matrix-vector multiplications
- EP: an embarrassingly parallel code that assesses the floating point performance
- FT: performs series of 1-D FFTs on a 3-D mesh that tests aggregated communication performance
- IS: a large integer sort that evaluates both integer computation performance and the aggregated communication throughput
- MG: a simplified multigrid kernel that performs both short and long distance communications

MPI/OpenMP/UPC Benchmarking

NPB Kernels

- CG: an iterative solver that tests regular communications in sparse matrix-vector multiplications
- EP: an embarrassingly parallel code that assesses the floating point performance
- FT: performs series of 1-D FFTs on a 3-D mesh that tests aggregated communication performance
- IS: a large integer sort that evaluates both integer computation performance and the aggregated communication throughput
- MG: a simplified multigrid kernel that performs both short and long distance communications

MPI/OpenMP/UPC Benchmarking

NPB Kernels

- CG: an iterative solver that tests regular communications in sparse matrix-vector multiplications
- EP: an embarrassingly parallel code that assesses the floating point performance
- FT: performs series of 1-D FFTs on a 3-D mesh that tests aggregated communication performance
- IS: a large integer sort that evaluates both integer computation performance and the aggregated communication throughput
- MG: a simplified multigrid kernel that performs both short and long distance communications

MPI/OpenMP/UPC Benchmarking

NPB Kernels

- CG: an iterative solver that tests regular communications in sparse matrix-vector multiplications
- EP: an embarrassingly parallel code that assesses the floating point performance
- FT: performs series of 1-D FFTs on a 3-D mesh that tests aggregated communication performance
- IS: a large integer sort that evaluates both integer computation performance and the aggregated communication throughput
- MG: a simplified multigrid kernel that performs both short and long distance communications

MPI/OpenMP/UPC Benchmarking

NPB-UPC

The NPB-UPC are optimized with:

- Privatization: Casts local shared memory accesses to private memory accesses (avoids the indirection required for shared memory accesses)
- Prefetching: Copies non-local shared memory blocks into private memory (avoids the slow shared memory accesses)

MPI/OpenMP/UPC Benchmarking

NPB-UPC

The NPB-UPC are optimized with:

- Privatization: Casts local shared memory accesses to private memory accesses (avoids the indirection required for shared memory accesses)
- Prefetching: Copies non-local shared memory blocks into private memory (avoids the slow shared memory accesses)

MPI/OpenMP/UPC Benchmarking

NPB-UPC

The NPB-UPC are optimized with:

- Privatization: Casts local shared memory accesses to private memory accesses (avoids the indirection required for shared memory accesses)
- Prefetching: Copies non-local shared memory blocks into private memory (avoids the slow shared memory accesses)

1 Experimental Testbed, 2 Scenarios:

Finis Terrae (CESGA, # 427 11/08 TOP500)

142 HP Integrity rx7640 nodes, each:

- 16 Montvale Itanium2 (IA64) cores at 1.6 GHz
- 128 GB RAM
- Mellanox InfiniBand DDR 4x HCA (16 Gbps bandwidth)

1 HP Integrity Superdome:

- 128 Montvale Itanium2 (IA64) cores at 1.6 GHz
- 1 TB RAM

SW Configuration:

- Linux SuSE 10, C compiler Intel icc 11 with OpenMP, HP MPI 2.2.5.1
- Berkeley UPC (BUPC) 2.8

Hybrid (Shared/Distributed Memory Scenario)

Up to 8 nodes and up to 16 cores per node (cores per node = $\lceil n/8 \rceil$)

SMP (Shared Memory Scenario)

Up to 128 threads on shared memory on the Superdome

1 Experimental Testbed, 2 Scenarios:

Finis Terrae (CESGA, # 427 11/08 TOP500)

142 HP Integrity rx7640 nodes, each:

- 16 Montvale Itanium2 (IA64) cores at 1.6 GHz
- 128 GB RAM
- Mellanox InfiniBand DDR 4x HCA (16 Gbps bandwidth)

1 HP Integrity Superdome:

- 128 Montvale Itanium2 (IA64) cores at 1.6 GHz
- 1 TB RAM

SW Configuration:

- Linux SuSE 10, C compiler Intel icc 11 with OpenMP, HP MPI 2.2.5.1
- Berkeley UPC (BUPC) 2.8

Hybrid (Shared/Distributed Memory Scenario)

Up to 8 nodes and up to 16 cores per node (cores per node = $\lceil n/8 \rceil$)

SMP (Shared Memory Scenario)

Up to 128 threads on shared memory on the Superdome

1 Experimental Testbed, 2 Scenarios:

Finis Terrae (CESGA, # 427 11/08 TOP500)

142 HP Integrity rx7640 nodes, each:

- 16 Montvale Itanium2 (IA64) cores at 1.6 GHz
- 128 GB RAM
- Mellanox InfiniBand DDR 4x HCA (16 Gbps bandwidth)

1 HP Integrity Superdome:

- 128 Montvale Itanium2 (IA64) cores at 1.6 GHz
- 1 TB RAM

SW Configuration:

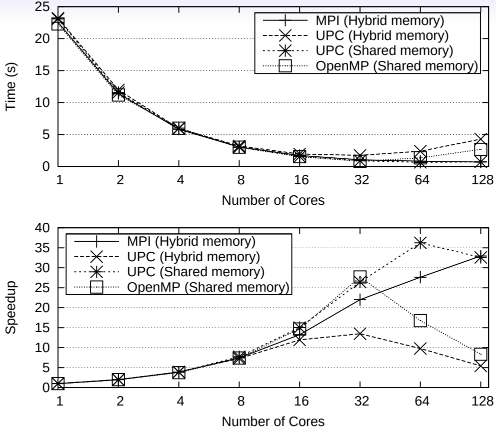
- Linux SuSE 10, C compiler Intel icc 11 with OpenMP, HP MPI 2.2.5.1
- Berkeley UPC (BUPC) 2.8

Hybrid (Shared/Distributed Memory Scenario)

Up to 8 nodes and up to 16 cores per node (cores per node = $\lceil n/8 \rceil$)

SMP (Shared Memory Scenario)

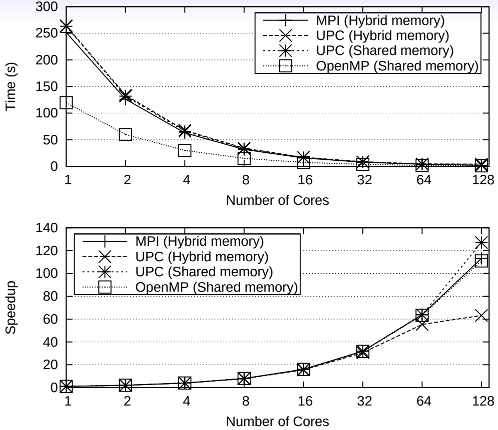
Up to 128 threads on shared memory on the Superdome



The lower the better

The higher the better

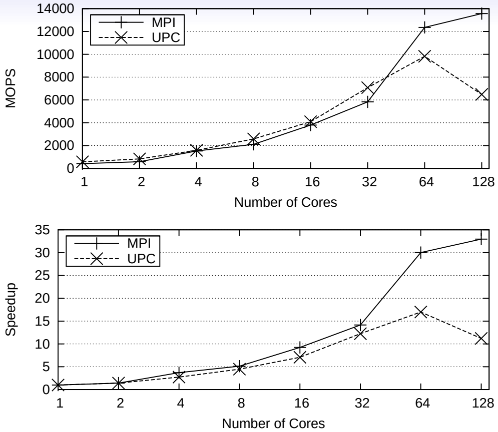
Figure: Matrix Multiplication Kernel



The lower the better

The higher the better

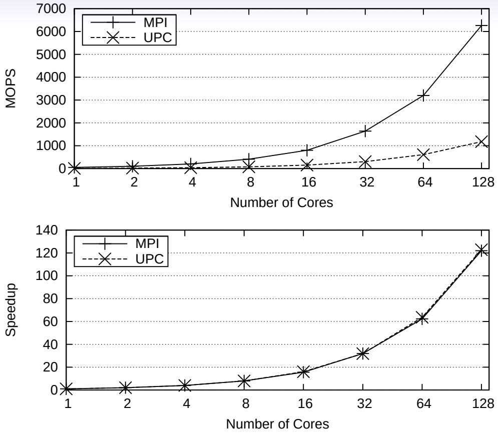
Figure: Sobel Edge Detector Kernel



The higher the better

The higher the better

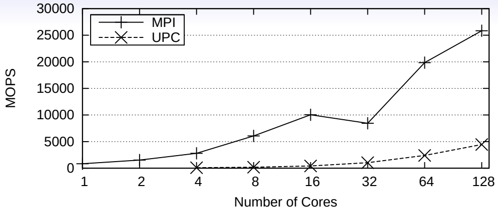
Figure: CG Kernel in Hybrid Memory (Class C)



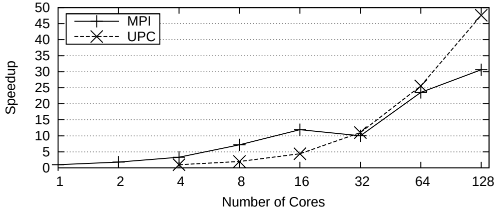
The higher the better

The higher the better

Figure: EP Kernel in Hybrid Memory (Class C)

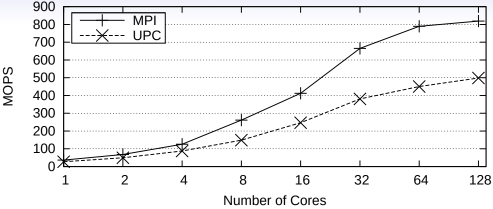


The higher the better

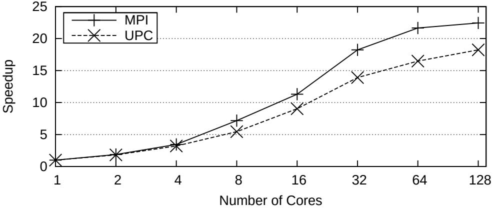


The higher the better

Figure: FT Kernel in Hybrid Memory (Class C)

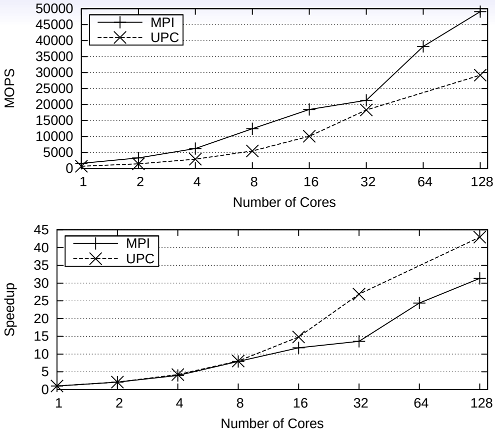


The higher the better



The higher the better

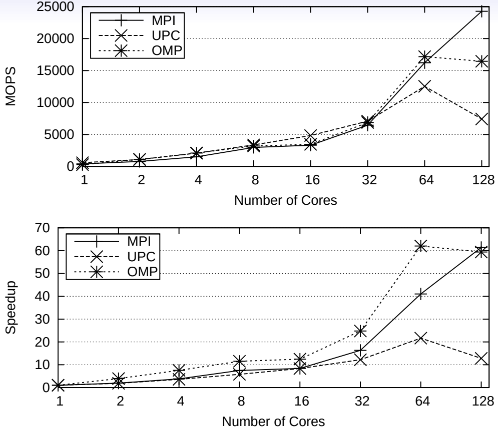
Figure: IS Kernel in Hybrid Memory (Class C)



The higher the better

The higher the better

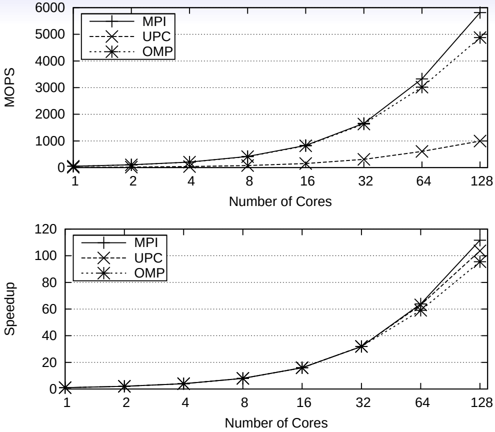
Figure: MG Kernel in Hybrid Memory (Class C)



The higher the better

The higher the better

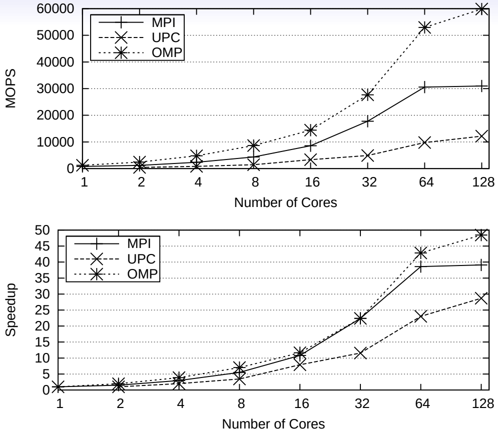
Figure: CG Kernel in Shared Memory (Class C)



The higher the better

The higher the better

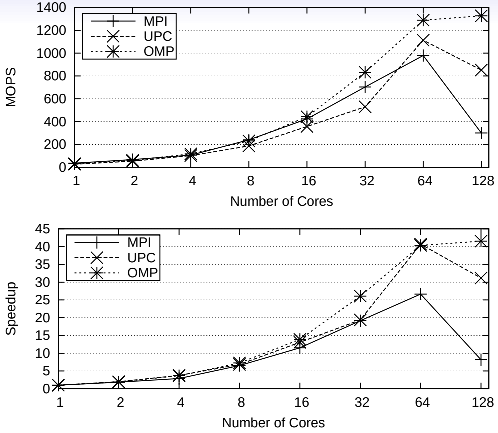
Figure: EP Kernel in Shared Memory (Class C)



The higher the better

The higher the better

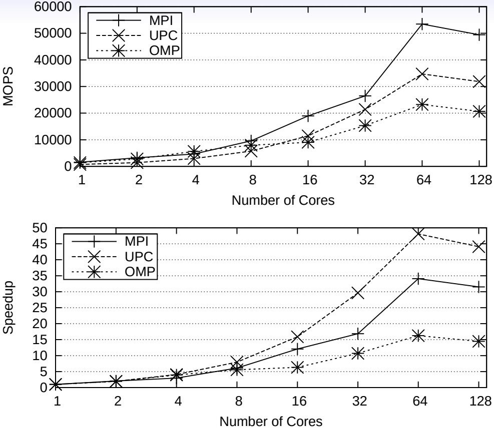
Figure: FT Kernel in Shared Memory (Class C)



The higher the better

The higher the better

Figure: IS Kernel in Shared Memory (Class C)



The higher the better

The higher the better

Figure: MG Kernel in Shared Memory (Class C)

Summary

Main Contribution:

- An up-to-date performance evaluation of the two preferred choices for HPC programming (MPI and OpenMP) and one emerging alternative (UPC), on two configurations: hybrid (shared and distributed) and shared memory, using a big multicore cluster.

Main Conclusions (I):

- Data locality is the key for good performance in hybrid memory $\rightarrow$ MPI is the top performer
- Nevertheless UPC speedup is better in some hybrid setups (mainly due its poor 1 thread performance/not mature compiler technology)
- 16 threads/processes per node are too much for IB. Even worst in the future

Summary

Main Conclusions (and II):

- Speedups for MPI and UPC up to 64 cores are better in shared memory than in hybrid memory (as expected)
- Too much remote memory accesses harm the performance. It does not worth using 128 cores in shared memory
- OpenMP benefits from its direct memory access and can outperform MPI, despite its poor data locality support
- UPC has better data locality support and direct memory access. However, it suffers from its compiler technology
- Despite UPC's performance, the gap between MPI/OpenMP and UPC is narrow enough to take UPC into account in most cases, due to its programmability/performance ratio

Summary

Final Conclusion

- Best performance → MPI
- Good performance and good time-to-solution → UPC
- Good-to-best performance and best time-to-solution (shared memory) → OpenMP

Summary

Final Conclusion

- Best performance → MPI
- Good performance and good time-to-solution → UPC
- Good-to-best performance and best time-to-solution (shared memory) → OpenMP

Summary

Final Conclusion

- Best performance → MPI
- Good performance and good time-to-solution → UPC
- Good-to-best performance and best time-to-solution (shared memory) → OpenMP

Questions?

Contact: Damián A. Mallón
dalvarez@cesga.es

Galicia Supercomputing Center, Spain
http://www.cesga.es