HPC for Computational Physics at INFN: The COKA and SUMA projects

COKA and SUMA Collaboration

Introduction : COKA & SUMA

INFN has a large and lively community of scientists actively engaged in computational physics, working on areas such as Lattice Gauge Theories (LGT), the Physics of Complex Systems and Fluid Dynamics. These scientists use world-class computing resources, including INFN-own computing resources, and access programmes such as PRACE.

INFN also supports these efforts with a number of projects more directly focusing on new HPC architectures, new parallel programming models, massively parallel algorithm development and optimization; broadly speaking, the focus of these projects is to enable INFN scientists to use forthcoming Exaflops systems as efficiently as possible.

In this framework, the SUMA project, co-funded by the Italian Ministry of University and Science (MIUR), provides wide-spectrum support to the INFN computational communities, including the operation of Tier-1 HPC clusters and the funding of post-doc positions for HPC-oriented research.

The COKA project **Computing on Knights and Kepler Architectures** focuses on the use of accelerators in HPC general purpose computing, assessing the performance of accelerator-based systems and developing programming methodologies to effectively use all parallel features available on these processors.

The hardware targets for these tests are the recently released x86-based Intel Xeon PHI and the K20-based NVIDIA Tesla boards, as well as low-power architectures such as the CARMA boards.

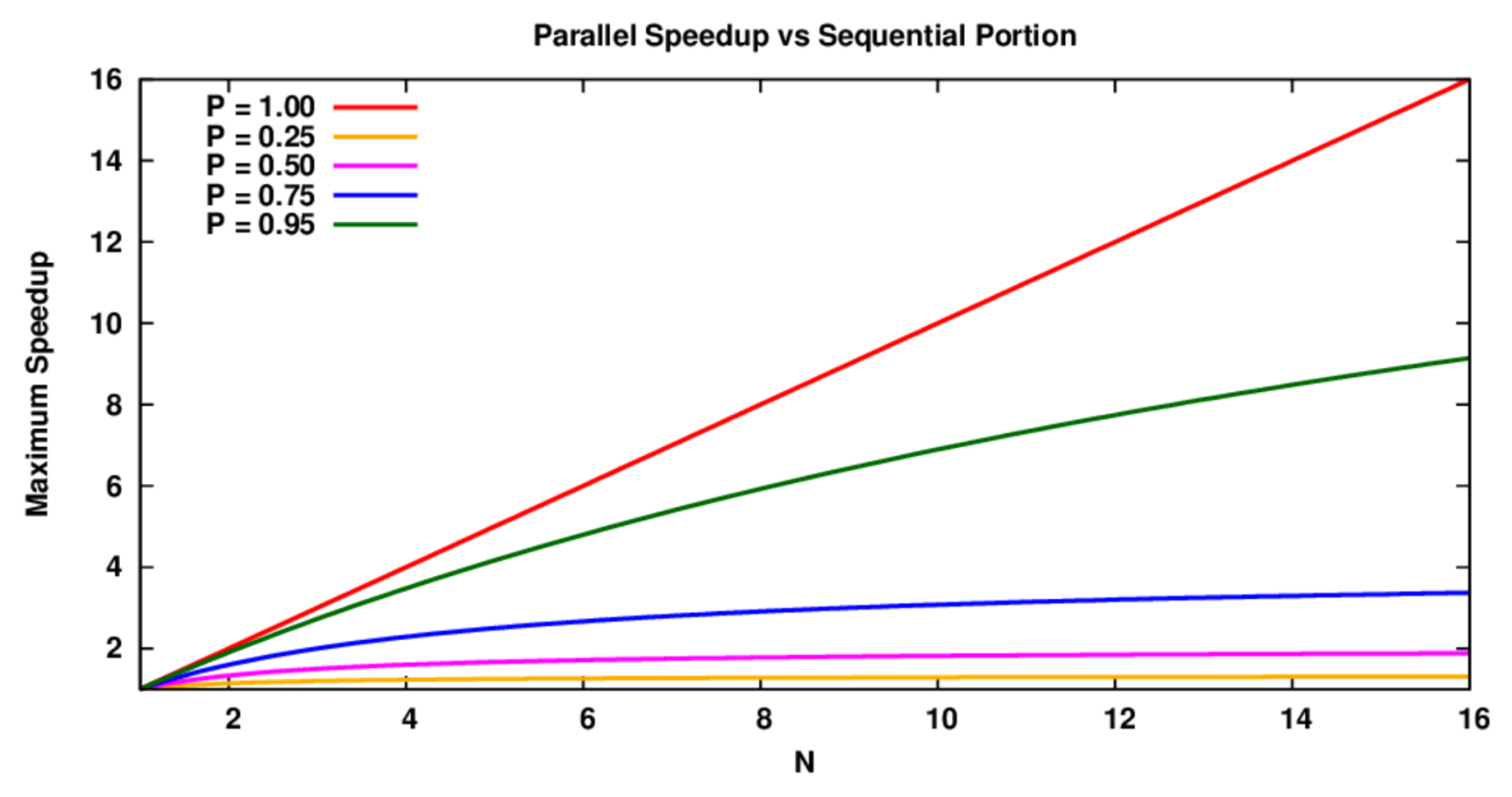
The project address the performances of applications relevant in theoretical and experimental physics. This poster presents a selection of results in such diverse areas as Lattice Boltzmann methods, Monte Carlo simulations of Spin Glass systems, and LGT, as well as data-analysis and trigger computing for high energy physics experiments.

Many-core Architectures and Issues

	i7-4930K	Tesla K20X	Xeon-Phi 7120P
#physical cores	6	14	61
#logical cores	12	2688	244
Frequency (GHz)	3.4	0.735	1.238
GFLOPS (DP)	163.2	1317	1208
SIMD	AVX 64-bit	N/A	AVX2 512-bit
cache (MB)	12	1.5	30.5
Mem BW (GB/s)	59.7	250	352
Power (W)	130	235	300

Issues:

- **core parallelism:**
  - keep all 60 cores (1 reserver for OS) busy
  - runs 2-3 (up-to) 4 threads/core is necessary to hide memory latency
- **vector parallelism:**
  - enable data-parallelism
  - enable use of 512-bit vector instructions
- **Amdhal's law:**
  - accelerator device clock period is O(1) ns
  - latency of PCI-E bus is



Code Portability: OpenCL

- programming framework for heterogeneous architectures: CPU + accelerators
- computing model:
  - host-code plus one or more kernels running on accelerators
  - kernels are executed by a set of work-items each processing an item of the data-set (data-parallelism)
  - work-items are grouped into work-groups, each executed by a compute-unit and processing K work-items in parallel using vector instructions
  - e.g.: on Xeon-Phi work-groups are mapped on (virtual-)cores processing each up to 8 double-precisions floating-point data
- memory model identifies a hierarchy of four spaces which differ for size and access-time : private, local, global and constant memory

$$C = s \cdot A \times B, \quad s \in \mathbb{R}, \quad A, B, C \in \mathbb{R}^n$$

```
__kernel void saxpy( __global double *A, __global double *B,
     __global double *C, const double s) {

  //get global thread ID
  int id = get_global_id(0);

  C[id] = s * A[id] + B[id];
}
```

OCL aims to guarantee portability of both code and performances across several architectures

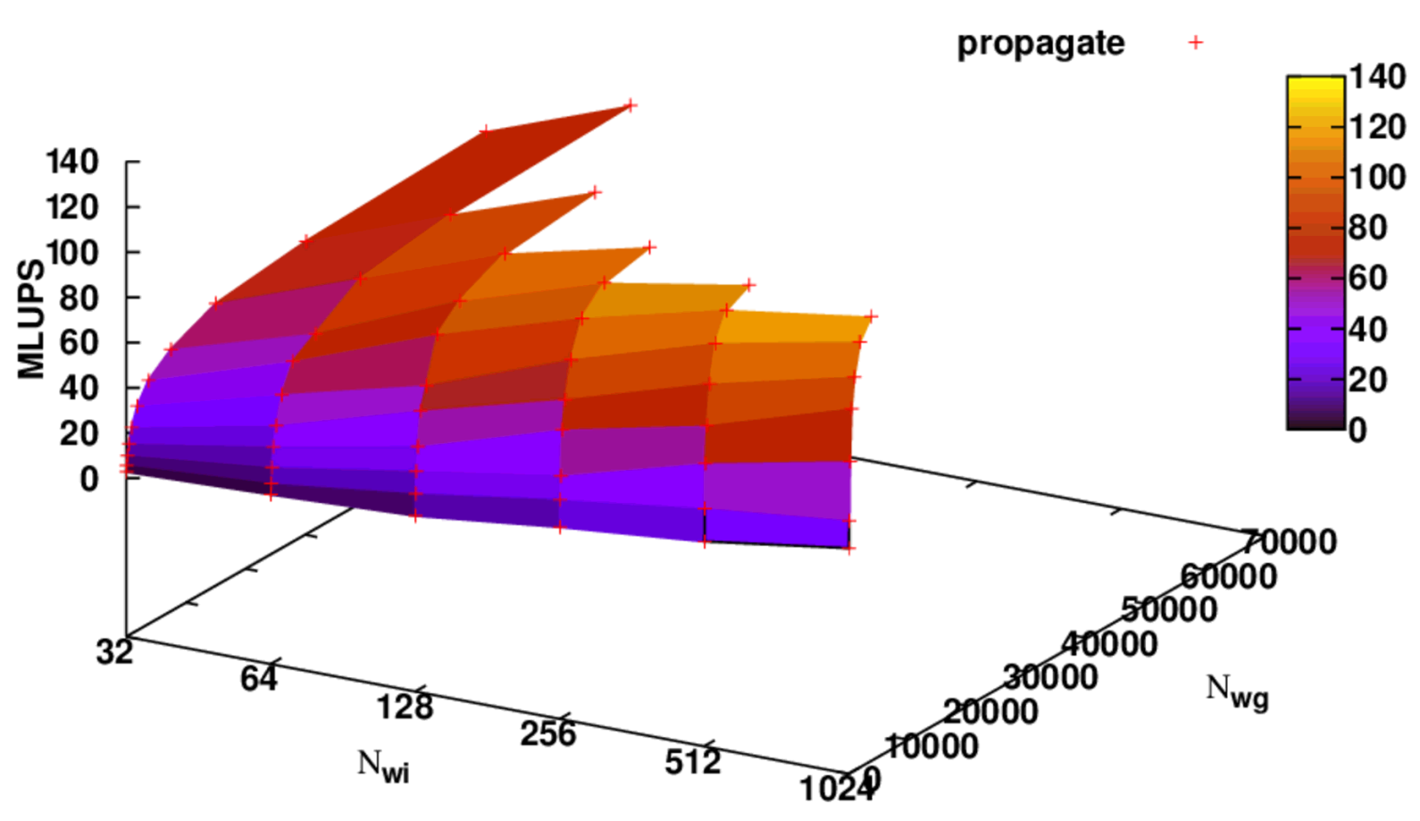
- each work-item executes the *saxpy* kernel computing just one data-item of the output array
- first it computes its unique global identifier id
- and then uses it to address the idth data-item of arrays A, B and C.

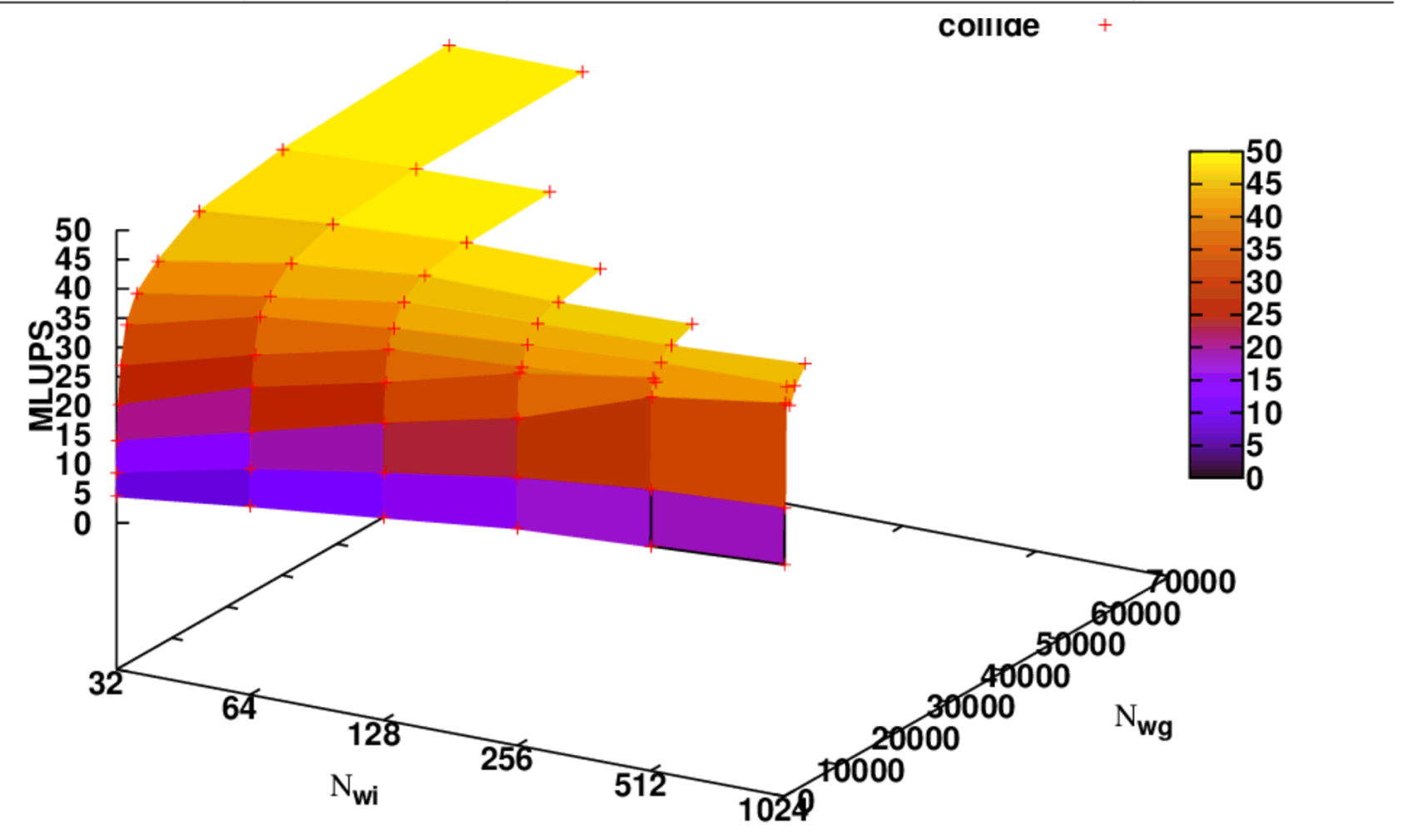
D2Q37

- Lattice size 1920x 2048
- 7600 DP operations /site

.

	Xeon-Phi 7120		Tesla K20Xm			i7-4930K	
Code Version	OCL	C$^{(*)}$	OCL	CUDA SM_20	CUDA SM_35	OCL	C$^{(*)}$
propagate T/iter [msec]	30.46	37.67	14.89	15.40	15.38	186.42	162.00
GB/s	76.42	61.8	156.33	151.16	151.36	12.48	14.54
$\mathcal{E}_p$	22%	17%	62%	60%	60%	21%	24%
bc T/iter [msec]	3.20	4.61	7.08	5.68	5.70	4.30	4.87
collide T/iter [msec]	72.79	79.14	93.27	83.33	43.96	440.18	307.42
GFLOPS (DP)	410	377	320	358	680	68	97
MLUPS	54.02	49.69	42.16	47.19	89.44	8.93	12.94
$\mathcal{E}_c$	34%	31%	24%	27%	52%	42%	59%
μJ / site	5.55	6.03	5.57	4.98	2.63	14.55	10.04
T_{WC}/iter [msec]	106.45	121.42	115.24	104.42	65.03	630.90	489.98
MLUPS	36.94	32.38	34.12	37.65	60.46	6.23	8.12





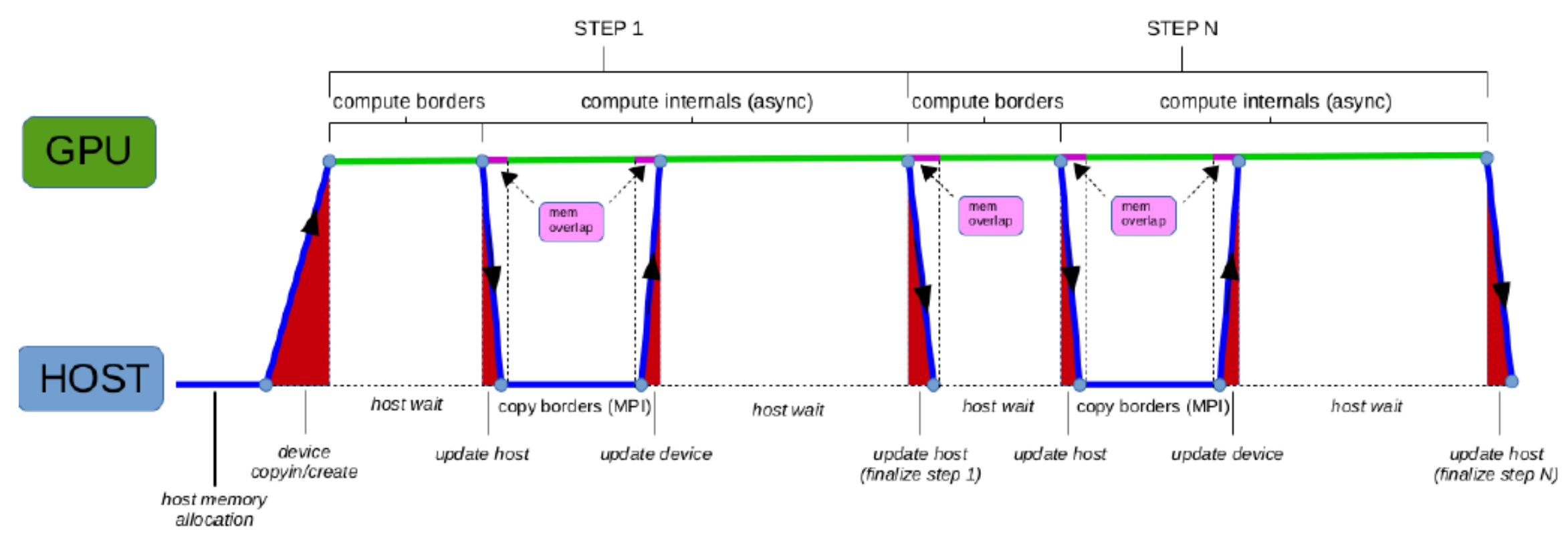
LQCD Pisa

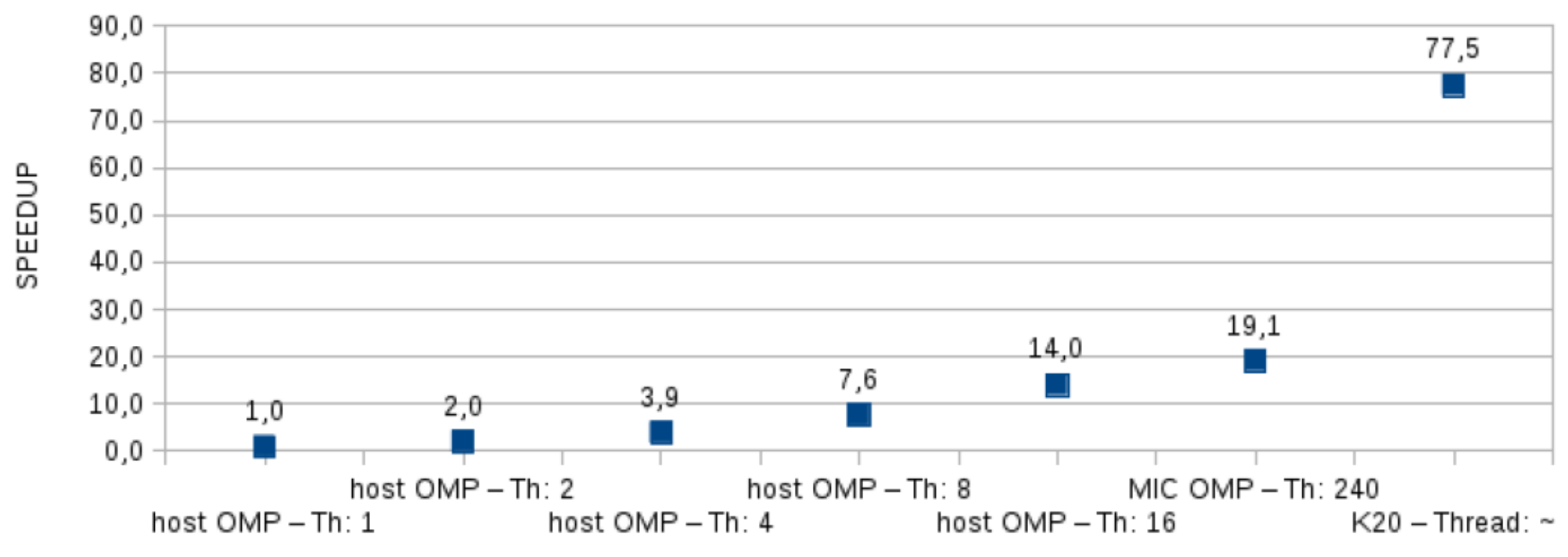
	Deo + Doe		DeoDoe
Block-size	*CUDA*	*OpenACC(PGI14.6)*	CUDA
8,8,8	7.58	9.29	-
16,1,1	8.43	16.16	47.1
16,2,1	7.68	9.92	30.4
16,4,1	7.76	9.96	30.4
16,8,1	7.75	10.11	30.5
16,16,1	7.64	10.46	30.9

Table: Time in [ns per site] run on an NVIDIA K20m GPU using double precision

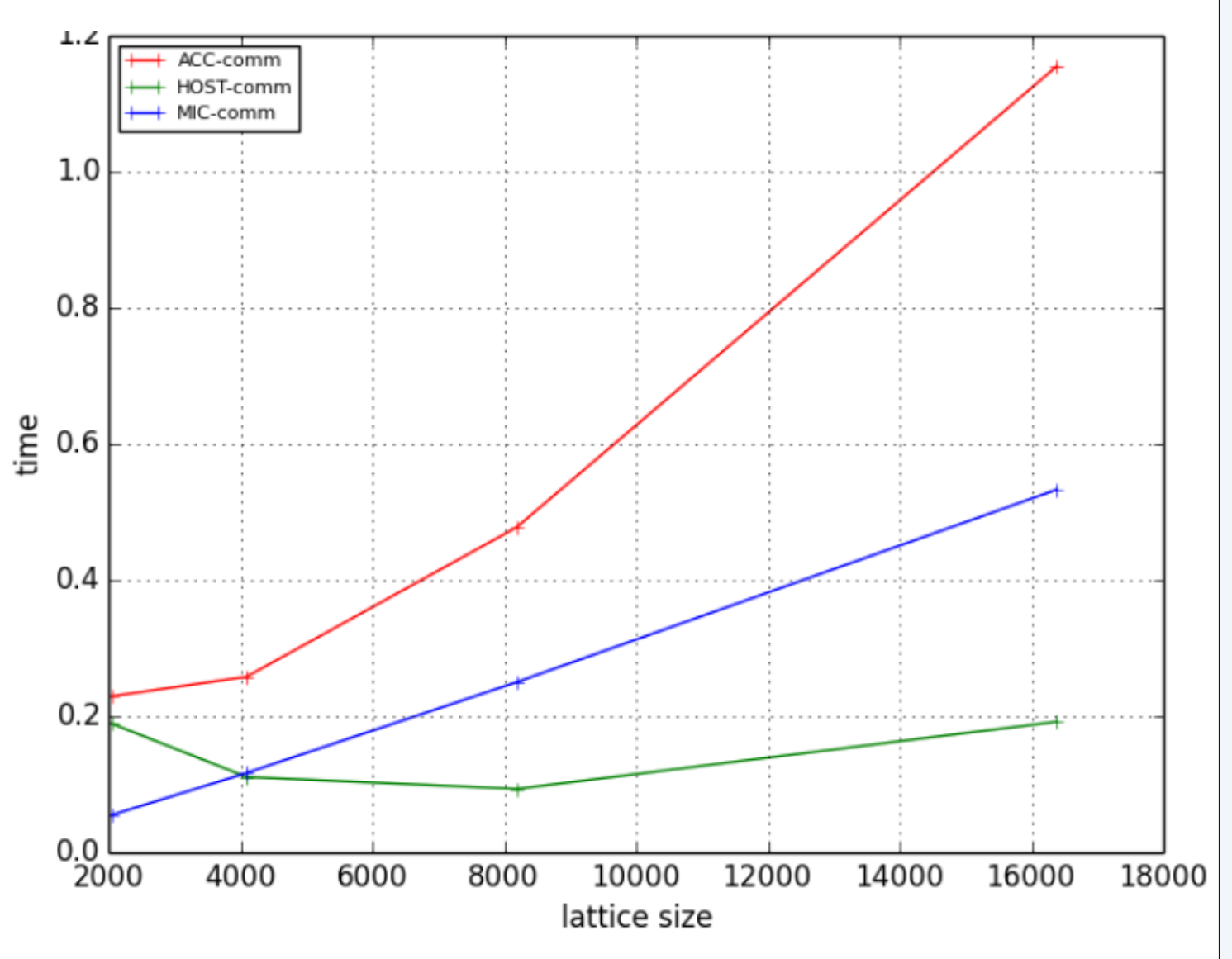
Assessment of the efficiencies and effectivness of directive-based programming paradigms for scientific HPC applications

- Validation tool based on the Game of life 2D
- Computation enforced through a parametric Ncomp() function (Double Precision vectorization)
- Parallelization with directive based languages for GPU NVIDIA K20 (OpenACC) and MIC (openMP)
- Overlapped communication/computation with independent processing of lattice borders and internals.



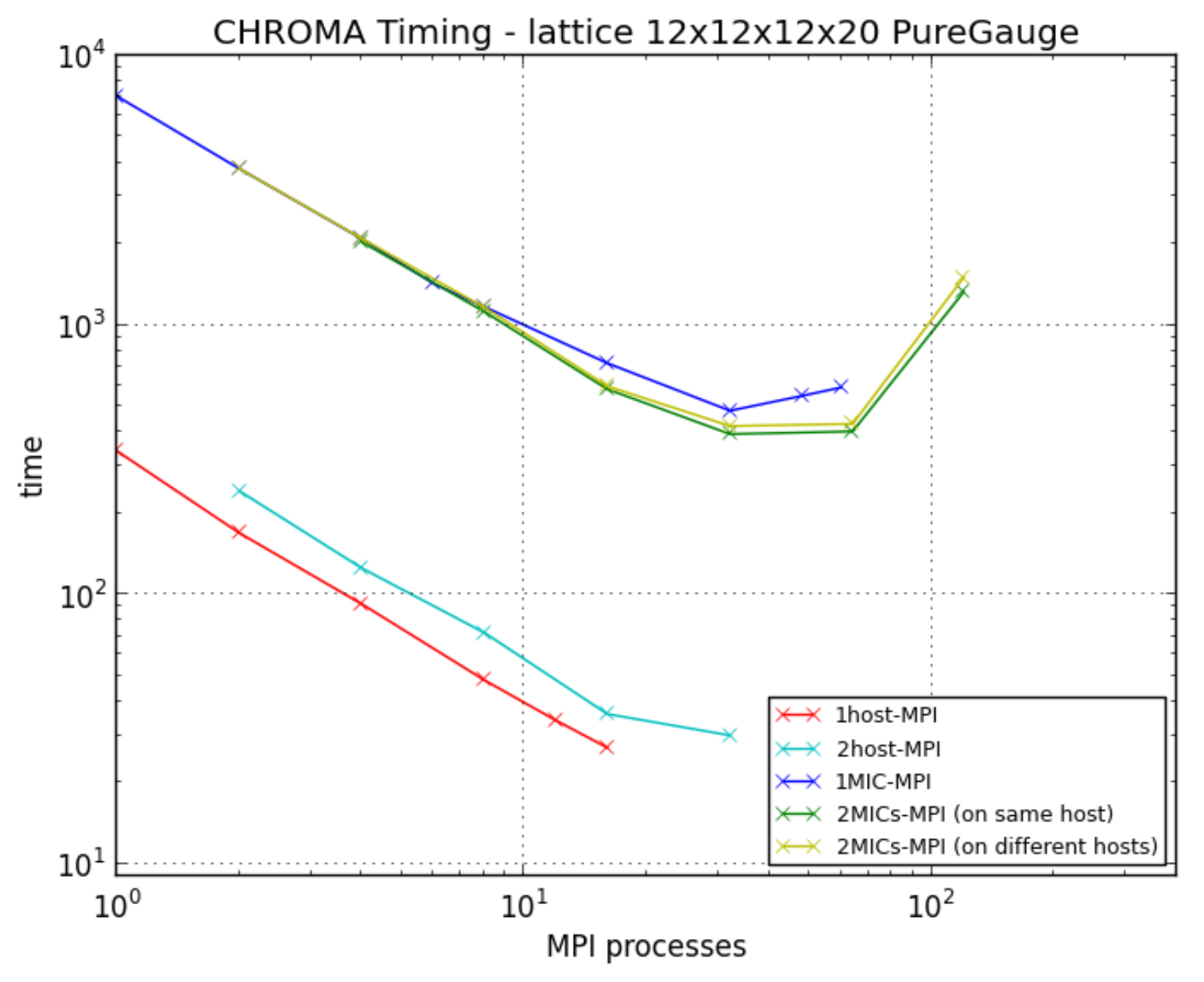


Single Host/device speed-up in relation to the host serial version



Multinode (8 MPI procs) communication time

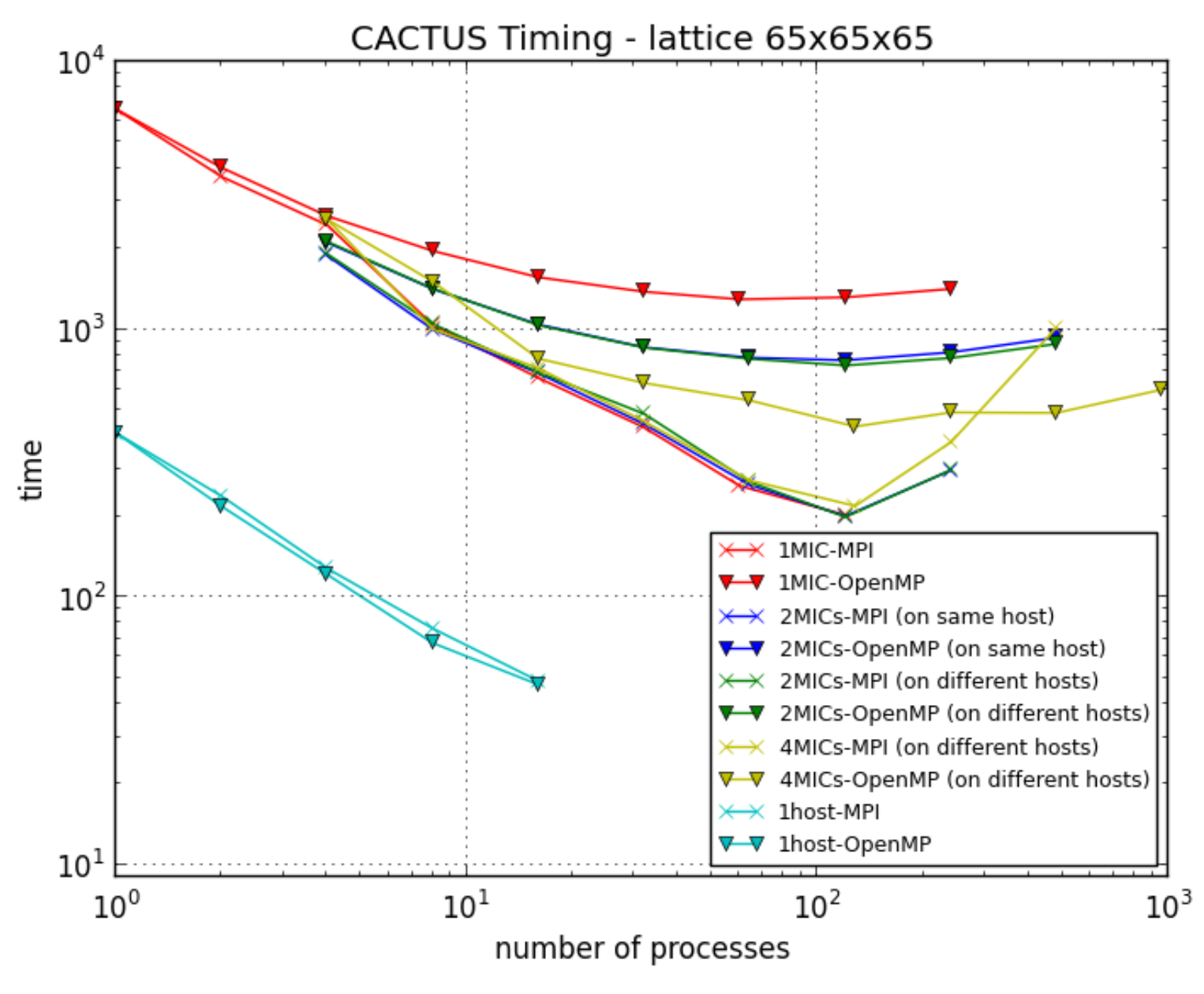
Native Mode performance of complex codes based on community software used by large collaborations in computational physics



The CHROMA (http://usqcd.jlab.org/usqcd-docs/chroma/) application has been re-compiled on the MIC environment without any code customization; this is a key point of the MIC architecture. The timing refers to the execution of 200 sweeps on a 12x12x12x20 lattice.

The drawback of this approach is the poor performance in comparison with the execution on the computing host (2x Sandy Bridge, E5-2687W 3.10 GHz, 8 cores).

Time of execution is 412 sec on a host core while it is 7033 sec on a PHI core.



The Einstein Toolkit (http://einsteintoolkit.org/)

Timing refer to the execution of 32 evolution steps on the evolution of a single General Relativistic Star on a 65x65x65 3-dimensional grid (0.6 total GBytes allocated memory). Single host core time is 410 sec while single PHI core time is 6857 sec.

Pure MPI execution on a PHI-card shows good scaling while OpenMP parallelization do not.

Best performance obtained using 60 MPI processes with 4 OpenMP threads each, where the timing is 189 sec (to be compared of the 47 sec obtained on the 16 cores of a host, 2x Sandy Bridge, E5-2687W 3.10 GHz).

References

[1] G. Crimi, F. Mantovani, M. Pivanti, S.F. Schifano, R. Tripiccione, Early Experience on Porting and Running a Lattice Boltzmann Code on the Xeon-Phi Co-Processor, Proceedings of the International Conference on Computational Science, ICCS 2013 Procedia Computer Science, Volume 18, 2013, Pages 551-560.

[2] F. Mantovani, M. Pivanti, S.F. Schifano, R. Tripiccione, Exploiting parallelism in many-core architectures: a test case based on Lattice Boltzmann Models, Proceedings of Conference on Computational Physics October 14-18, 2012 Kobe, Japan (in press).

[3] Implementation and Optimization of a Thermal Lattice Boltzmann Algorithm on a multi-GPU cluster, A. Bertazzo, F. Mantovani, M. Pivanti, F. Pozzati, S.F. Schifano, R. Tripiccione, Proceedings of Innovative Parallel Computing (INPAR) 2012, May 13-14, 2012 San Jose, CA (USA).

[4] F. Mantovani, M. Pivanti, S.F. Schifano, R. Tripiccione, Performance issues on many-core processors: A D2Q37 Lattice Boltzmann scheme as a test-case, Proceedings of 24rd International Conference on Parallel Computation Fluid Dynamics (PARCFD), May 21-25, 2012, Atlanta, GE (USA), Computers and Fluids (2013), 10.1016/j.compfuid.2013.05.014

[5] L. Biferale, F. Mantovani, M. Pivanti, F. Pozzati, M. Sbragaglia, A. Scagliarini, S. F. Schifano, F. Toschi, R. Tripiccione, Optimization of Multi-Phase Compressible Lattice Boltzmann Codes on Massively Parallel Multi-Core Systems, International Conference on Computational Science (ICCS), June 1-3, 2011, Singapore Procedia Science Vol. 4, pp. 994-1003, 2011.

[6] L. Biferale, F. Mantovani, M. Pivanti, F. Pozzati, M. Sbragaglia, A. Scagliarini, S. F. Schifano, F. Toschi, R. Tripiccione, An Optimized D2Q37 Lattice Boltzmann Code on GP-GPUs Proceedings of 23rd International Conference on Parallel Computation Fluid Dynamics (PARCFD) May 16-20 Barcelona (Spain), Computers and Fluids Vol. 80 (2013), pp. 55-62,10.1016/j.compfuid.2012.06.003.