On usability of HPC systems

Sebastiano Fabio Schifano

University of Ferrara and INFN-Ferrara

SM&FT 2017 High Performance Computing in Theoretical Physics

December 13-15, 2017

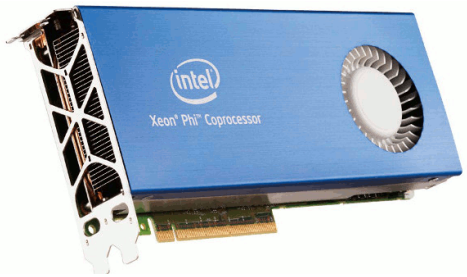
Bari, Italy

Outline

Focus

On the usability of HPC systems for developers of scientific applications.

Panorama of “modern” processors ... neglecting many details



	apeNEXT	P100	V100	Xeon Phy 7230	Xeon E5-2697 v4	Xeon 8160
Year	2002	2016	2017	2016	2016	2017
f [GHZ]	0.2	1.3	1.3	1.3	2.3	2.1
#cores / SMs	1	56	80	64	18	24
#threads / CUDA-cores	1	3584	5120	256	36	48
$\mathcal{P}_{DP}$ [GFlops]	1.6	4759	7000	2662	650	1533
β_{mem} [GB/s]	3.2	732	900	400	76.8	119.21
β_{net} [GB/s]	1.2	12.5	12.5	12.5	12.5	12.5
Watt	4	250	250	215	145	150
$\mathcal{P}/W$	0.4	19	28	12	4.5	10
$\mathcal{P}/\beta_{mem}$	0.5	6.5	7.7	6.6	8.5	13
$\mathcal{P}/\beta_{net}$	1.3	381	560	213	52	123
$\mathcal{P} \times \lambda$	240	2379776	1050000	1331000	331200	766500

β_{net} for apeNEXT is 200 MB/s x 6, $\lambda = 150$ ns

β_{net} for P100 and V100 is Mellanox 4X EDR $\approx$ 12.5 GB/s, $\lambda = 500$ ns

β_{net} for Xeon is OmniPath 100 Gbit/s = 12.5 GB/s, $\lambda = 500$ ns

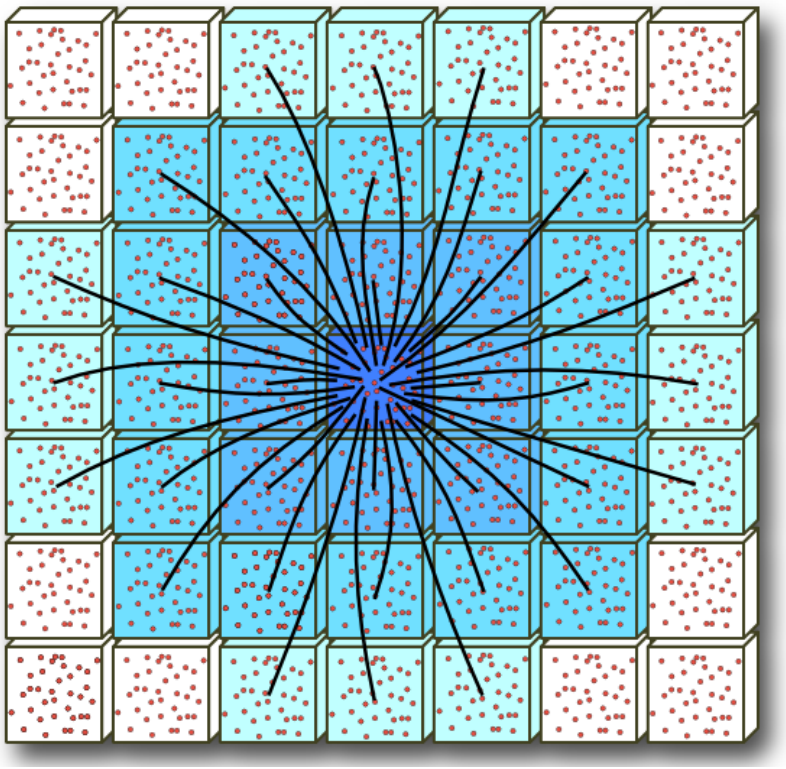
- several level of parallelism: $\mathcal{P} = f \times$ #cores $\times$ #opPerCycle $\times$ #flopPerOp
- memory layouts play an important role also for computing performances.

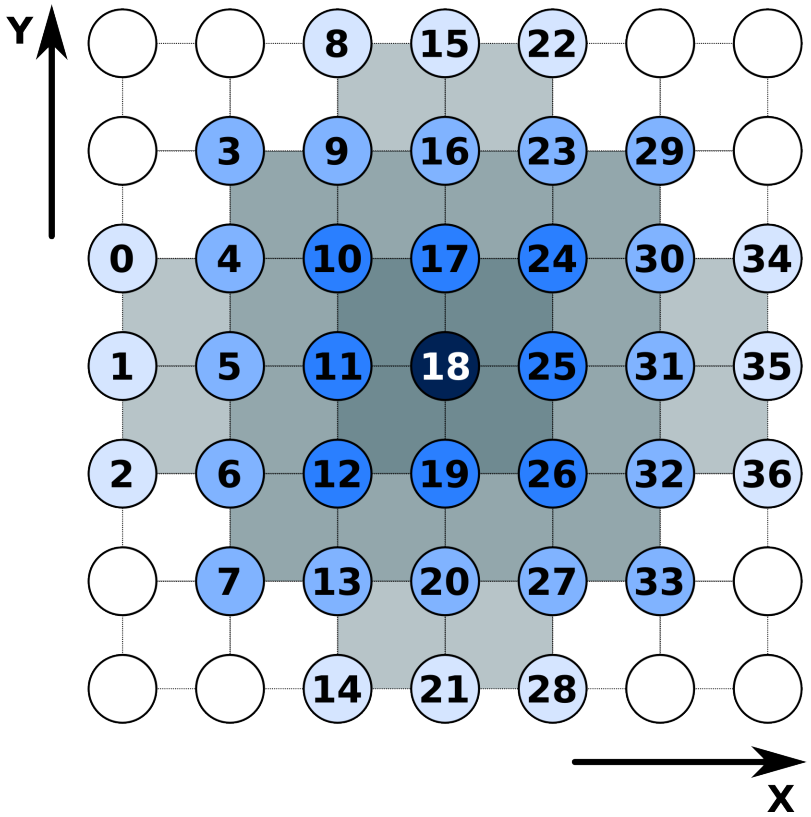
Lattice Boltzmann Methods

- Lattice Boltzmann method (LBM) is a class of computational fluid dynamics (CFD) methods
- simulation of synthetic dynamics is described by discrete **Boltzmann** equation, instead of **Navier-Stokes** equations
- a set of **virtual particles** called **populations** is arranged at edges of a discrete and regular grid
- interacting by **propagation** and **collision** reproduce – after appropriate averaging – the dynamics of fluids
- D2Q37 is a D2 LBM model with 37 components of velocity (populations)
- suitable to study behaviour of **compressible** gas and fluids optionally in presence of **combustion** [1] effects
- include correct treatment of *Navier-Stokes*, heat transport and perfect-gas ($P = \rho T$) equations

[1] chemical reactions turning cold-mixture of reactants into hot-mixture of burnt product.

D2Q37: propagation scheme

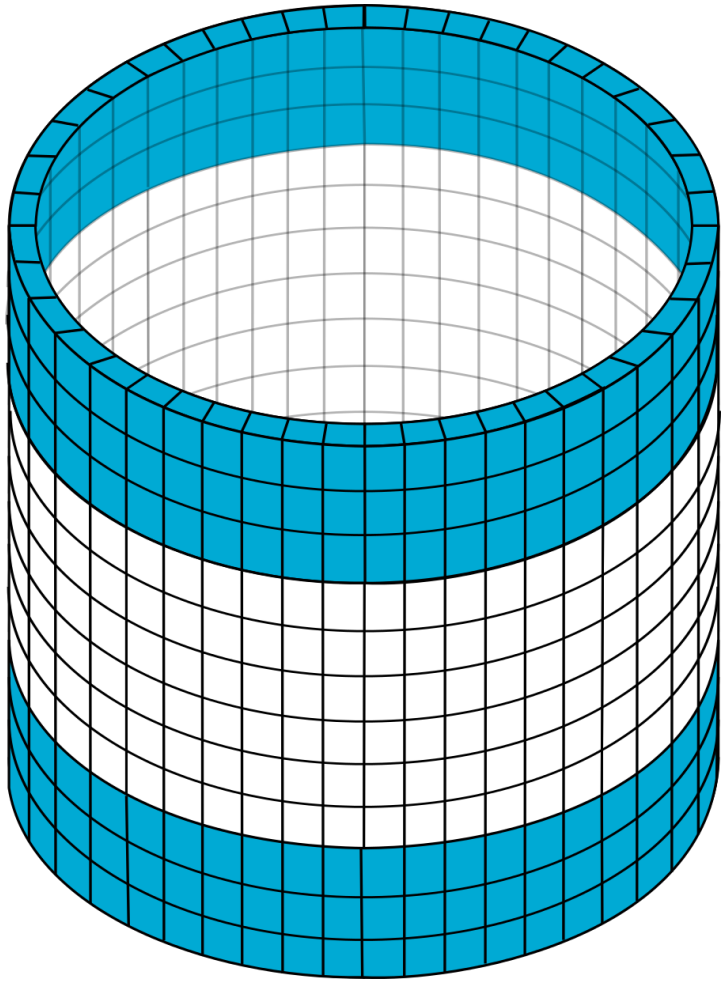




- require to access neighbours cells at distance 1,2, and 3,
- generate memory-accesses with **sparse** addressing patterns.

D2Q37: boundary-conditions

- we simulate a 2D lattice with periodic-boundaries along x-direction
- at the top and the bottom boundary conditions are enforced:
  - to adjust some values at sites $y = 0 \ldots 2$ and $y = N_y - 3 \ldots N_y - 1$
  - e.g. set vertical velocity to zero



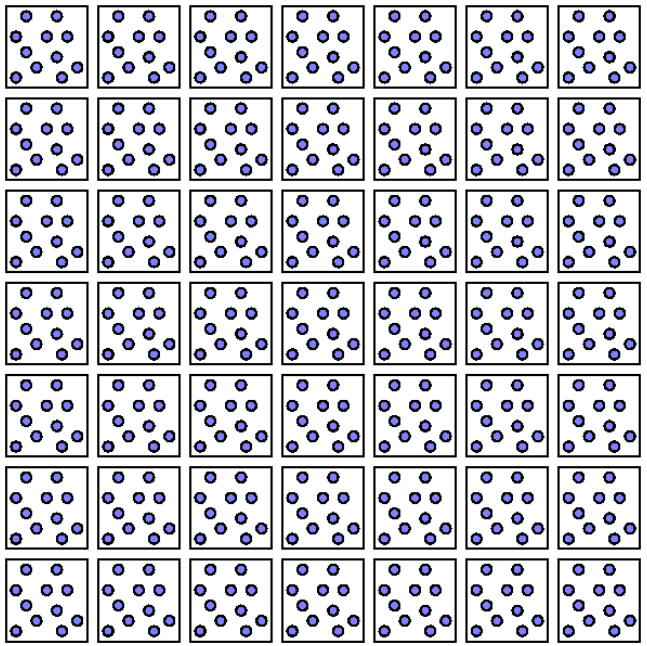
This step (bc) is computed before the collision step.

D2Q37 collision

- collision is computed at each lattice-cell site
- computational intensive:
  for the D2Q37 model requires $\approx$ 6500 DP floating point operations
- computation is completely **local**:
  arithmetic operations require only the populations associated to the site

D2Q37 LBM: Computational Requirements

```
foreach time-step

  foreach lattice-point
    propagate();
  endfor

  foreach lattice-point
    collide();
  endfor

endfor
```

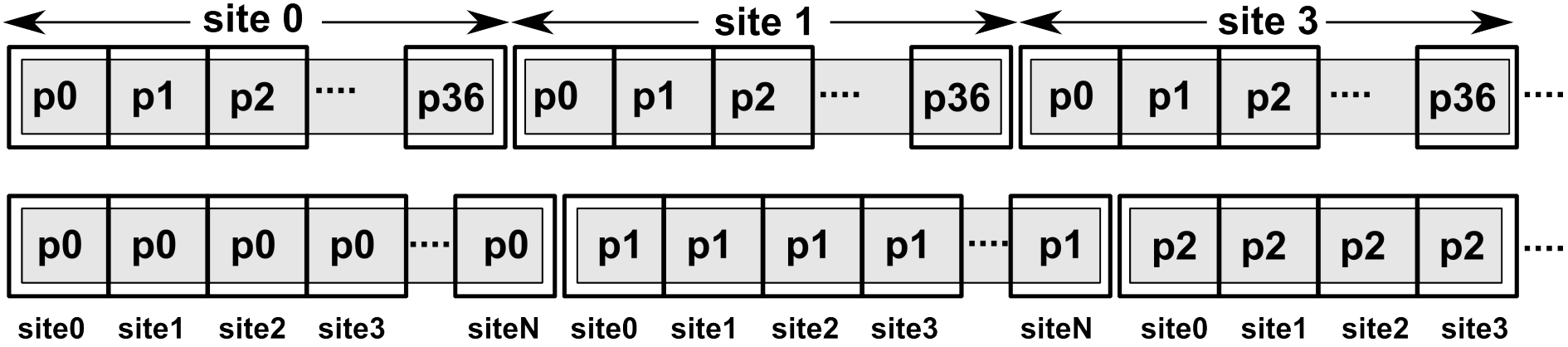


- embarassing parallelism: all sites can be processed in parallel applying in sequence propagate and collide
- two relevant kernels: *propagate* memory-bound, *collide* compute-bound
- good tool to stress, test and benchmark computing systems.

Challenge

Design efficient implementations to exploit all level of parallelism and a large fraction of available peak performance.

Memory Layout for vectorizing LBM: AoS vs SoA

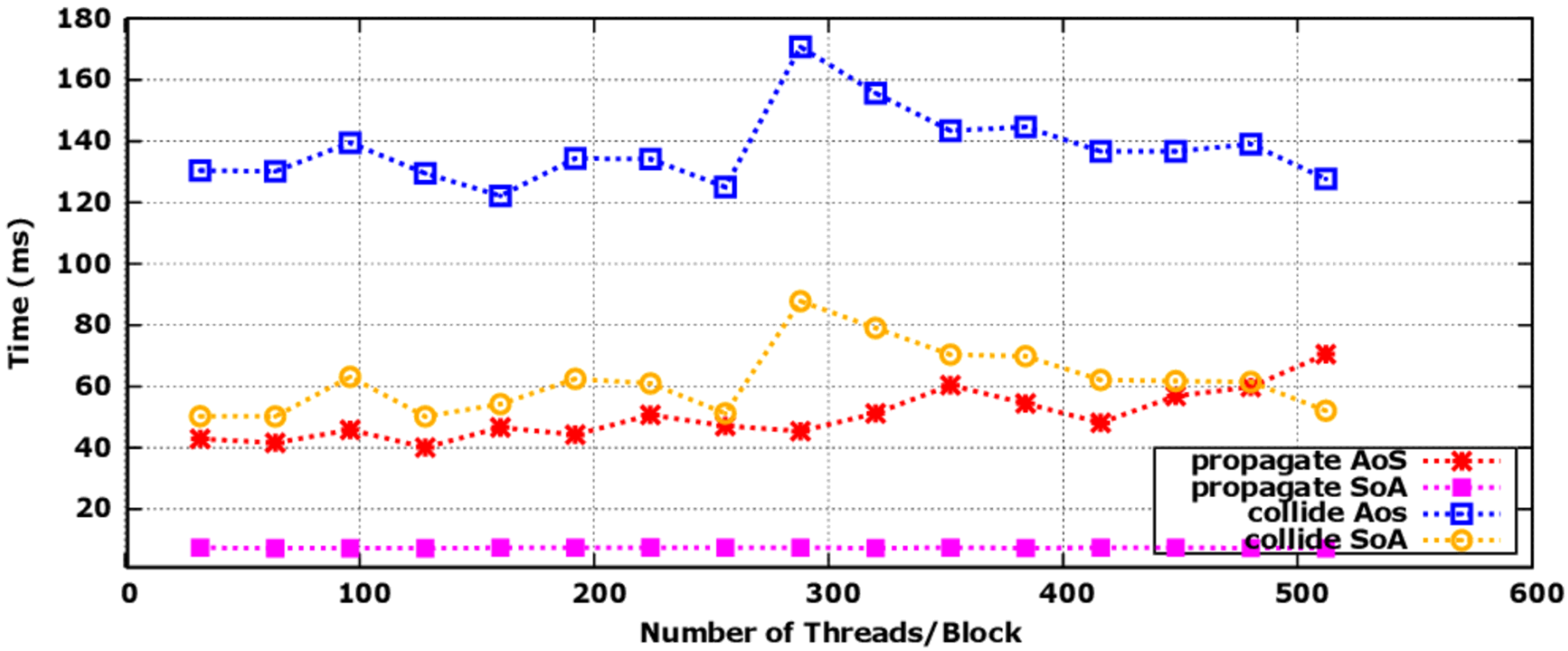



```
#define N (LX*LY)
typedef struct {
  double  p1; // population  1
  double  p2; // population  2
  ...
  double p37; // population 37
} pop_t;

pop_t lattice[N];
```

```
#define N (LX*LY)
typedef struct {
  double  p1[N]; // population  1
  double  p2[N]; // population  2
  ...
  double p37[N]; // population 37
} pop_t;

pop_t lattice;
```

- AoS (upper) site pop. data of each site close in memory
- SoA (lower) same-index pop. data of different site close in memory

Memory layout for GPUs: AoS vs SoA



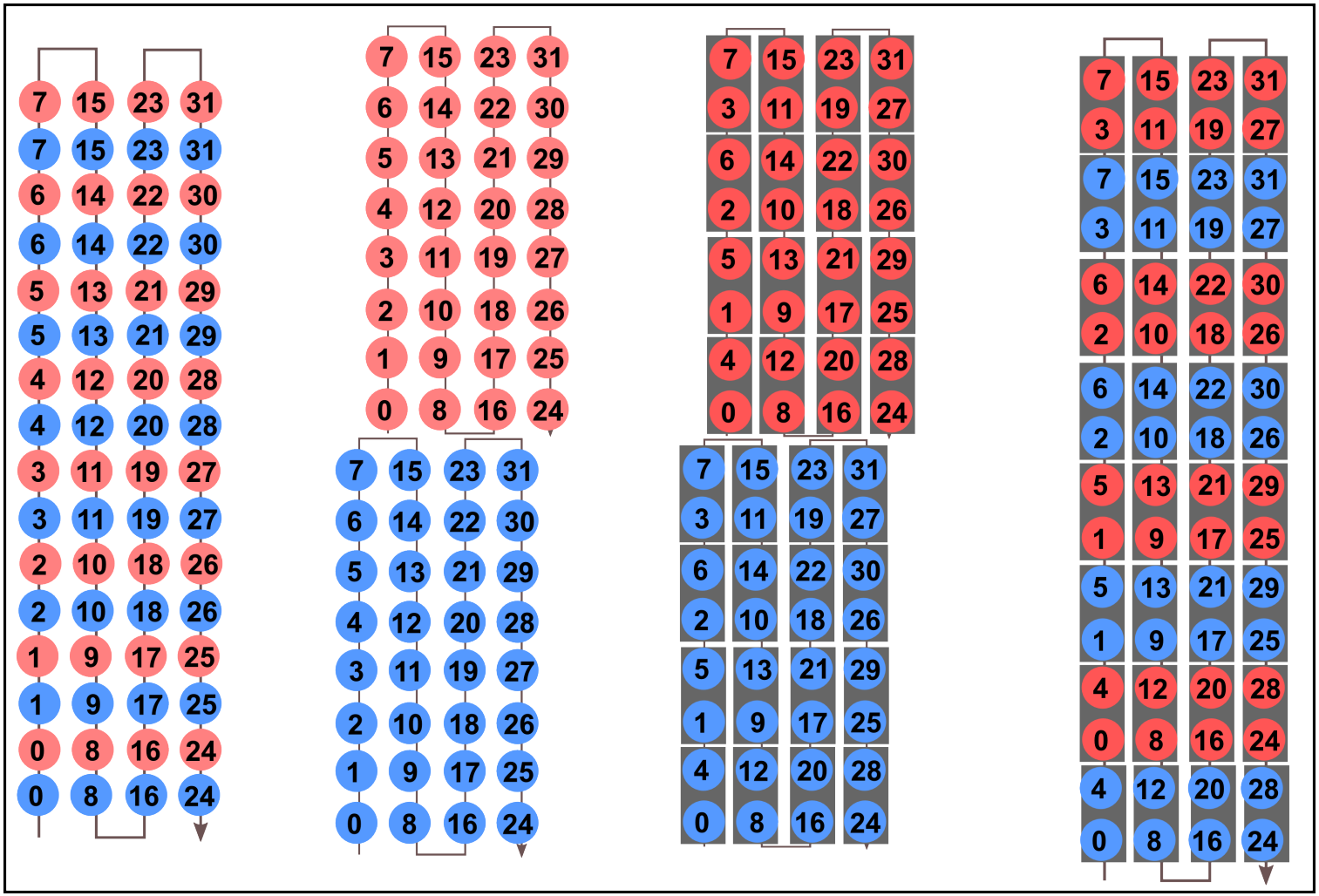
NVIDIA K40 GPU

propagate $\approx$ 10X faster, collide $\approx$ 2X faster

On GPU SoA gives the best results because enable vector processing and coalescing access to memory.

Memory Layouts for LBM on CPUs

Lattice 4 $\times$ 8 with two (blu and red) population per site.



Left to right: Array of Structures (AoS), Structure of Arrays (SoA),

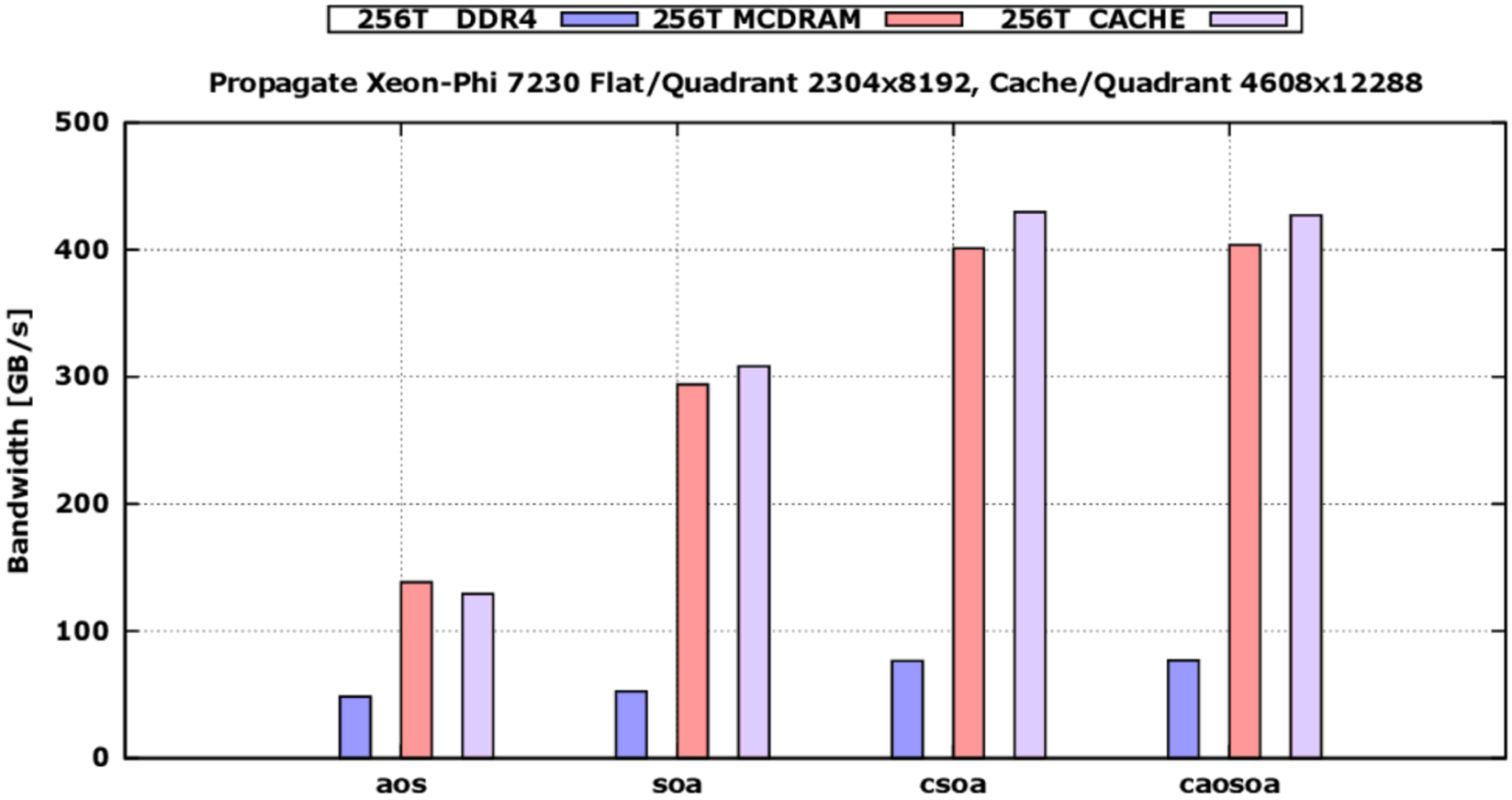
Clustered Structure of Arrays (CSoA), Clustered Array of Structure of Arrays (CAoSoA).

Results: VTUNE Analysis on KNL

Metric	*AoS*	*SoA*	*CSoA*	*CAoSoA*	Threshold
propagate L2 CACHE Miss Rate	0.50	0.10	0.05	0.00	< 0.20
collide L2 TLB Miss Overhead	0.00	0.21	1.00	0.00	< 0.05

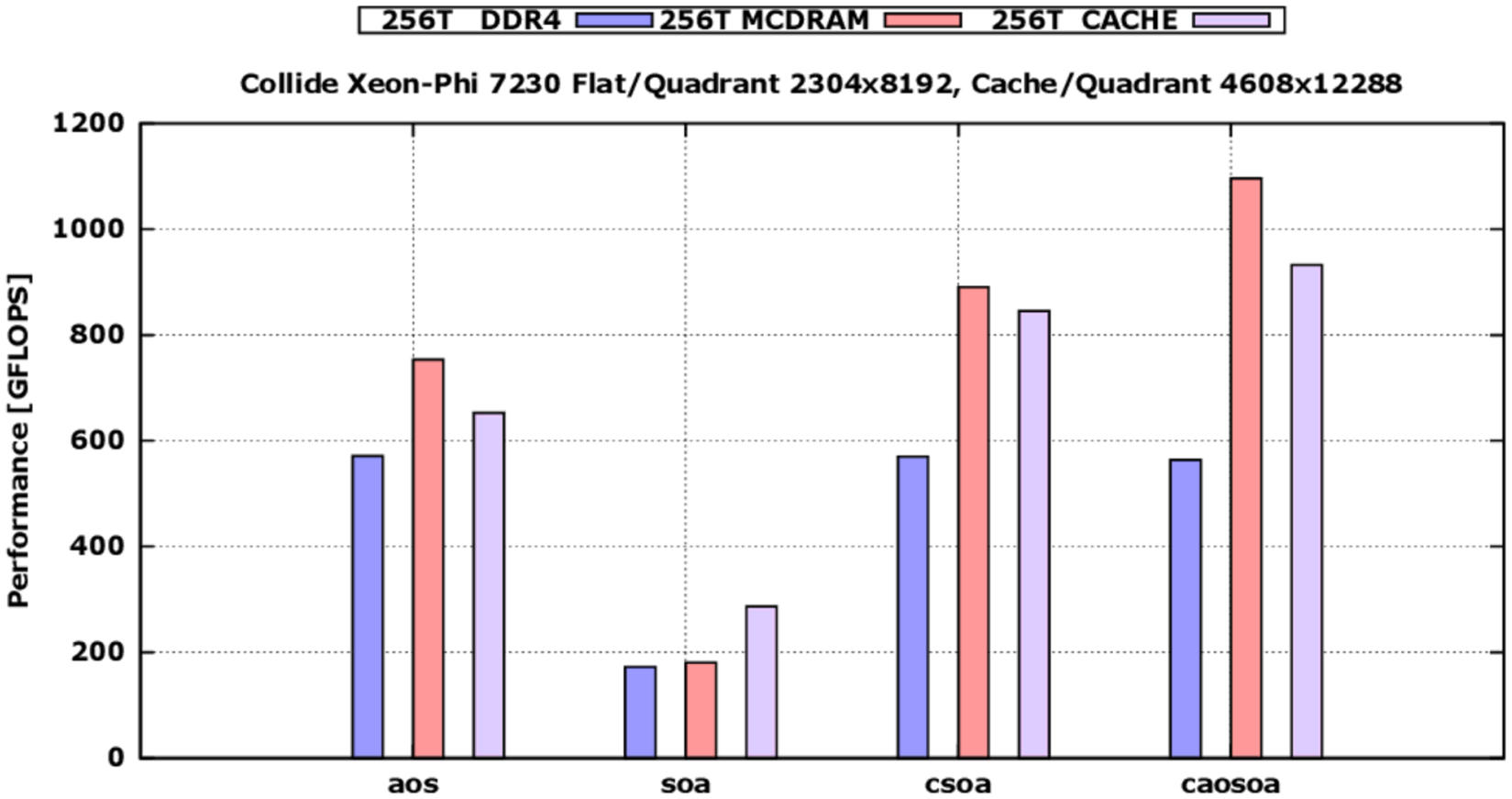
- Threshold value suggested by the Intel VTUNE profiler
- `L2 CACHE Miss Rate`: fraction of memory references not found in `L2 CACHE`
- `L2 TLB Miss Overhead`: fraction of CPU cycles spent for data page walks

Results: Propagate Performance on KNL



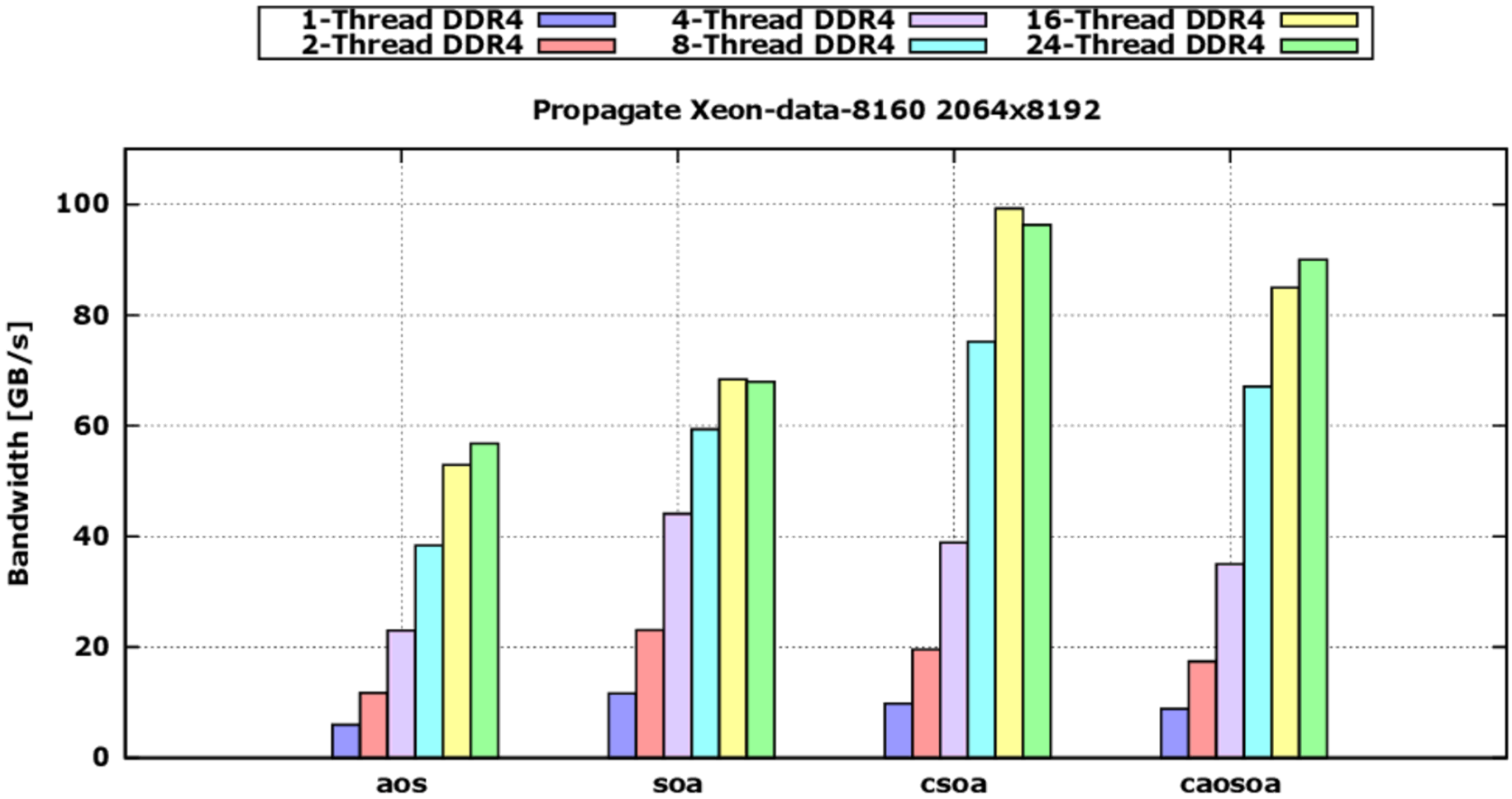
- FLAT-mode performances increases from *AoS* $\rightarrow$ *SoA* $\rightarrow$ *CSoA*
  - 138, 314, and 433 GB/s using the MCDRAM
  - 51, 56, and 81 GB/s using the DDR4
- for CACHE-mode we measure 59, 60 and 62 GB/s with a lattice not fitting into MCDRAM

Results: Collide Performance on KNL



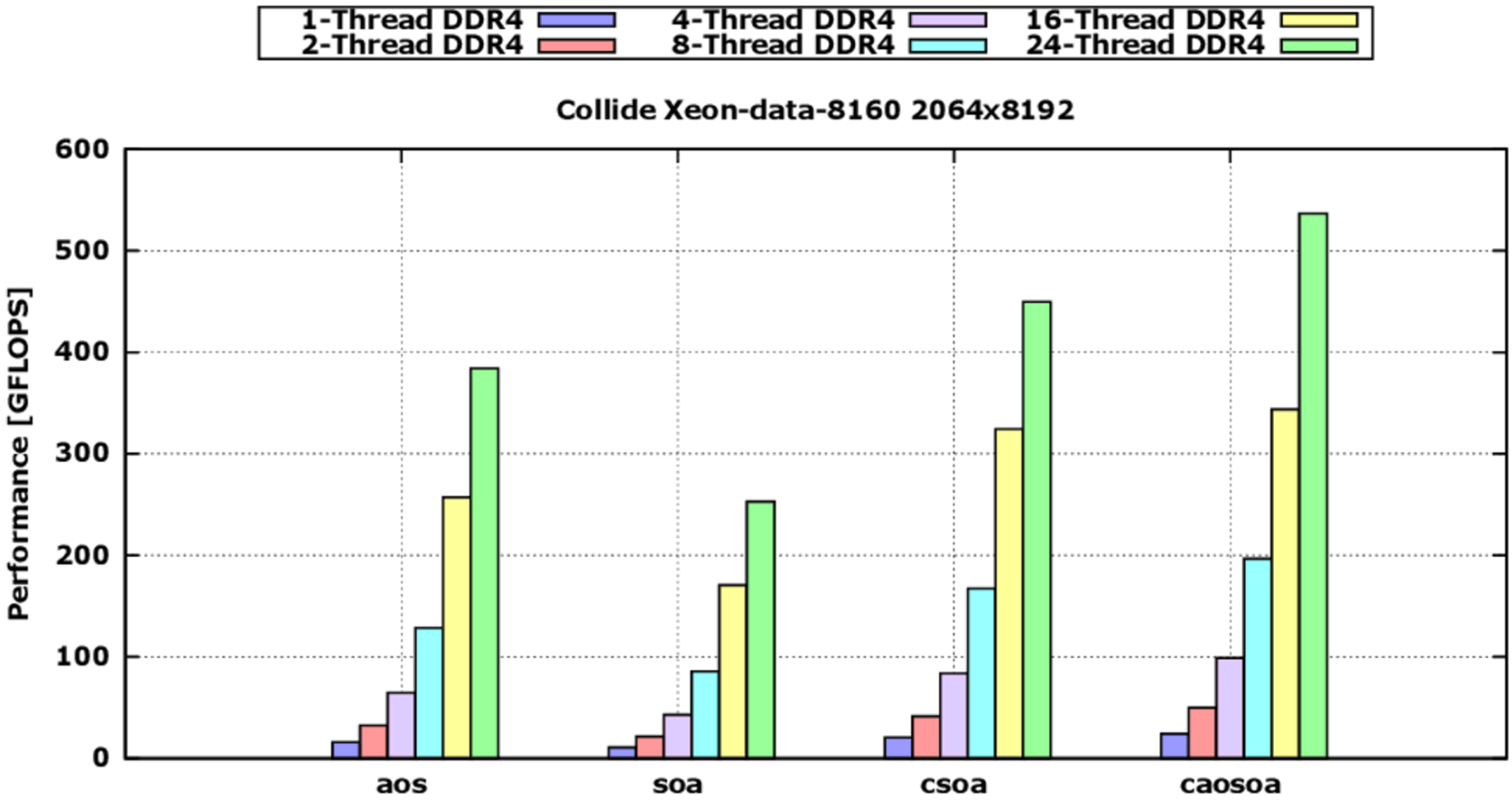
- for FLAT-MCDRAM configuration performance increases from *AoS* $\rightarrow$ *CSoA* $\rightarrow$ *CAoSoA*
- using *CAoSoA* we measure a sustained performance of 1 Tflops ($\approx$ 30% of raw peak)
- using FLAT-DDR4 and Cache configurations performances are limited by memory bandwidth

Results: Propagate Performance on SkyLake



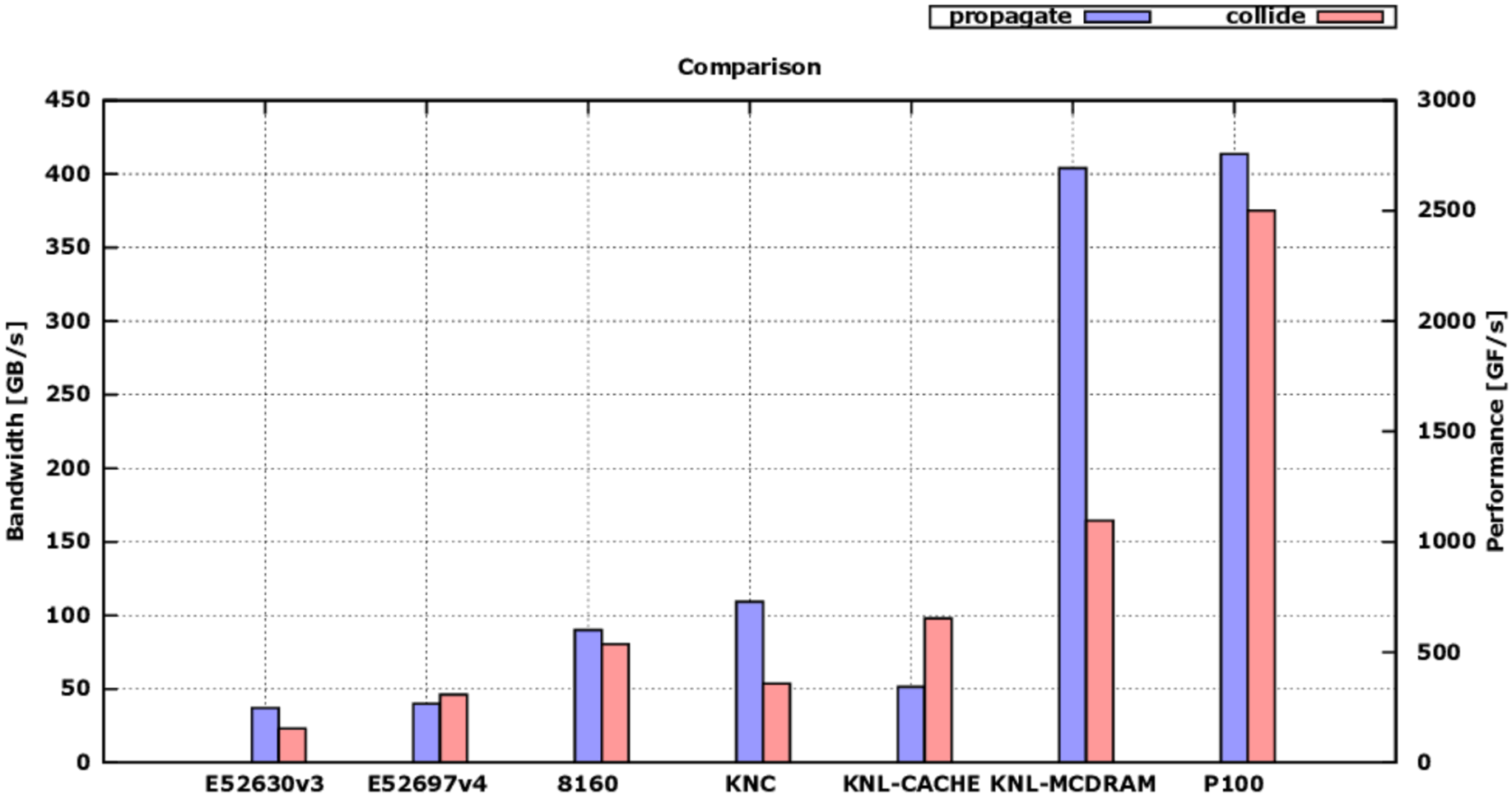
propagate $\approx$ 100 GB/s, *approx*85% of raw peak.

Results: Collide Performance on KNL



collide $\approx$ 530 GFlops. $\approx$ 35% of raw peak.

LBM: Performance Comparison



Conclusions

In summary, based on our experience related to our LBM application:

- efficient use of modern HPC is not an easy task
- several level of parallelism to exploit
- appropriate data layout to use

It is useful to have a stable "program" of computing activities within INFN to explore upcoming generations of HPC systems and train users for an efficient use of them.

CIPE Project

- CIPE project funded in 2016
- $\sim$ 10*MEuro* used for several activities related to computing of INFN:
  - HPC systems for theoretical physics
  - update infrastructure for experimental physics
  - new architectures
- Up to now:
  - several grants "assegni di ricerca" have been funded both for theoretical and experimental activities
  - co-funded extension of Marconi system
  - agreement with CINECA for hosting systems for HTC

Some budget is still available for funding activities of techonology survey and procurement of new generations of HPC and HTC systems.

Existing HPC systems

- Systems are based on the model CPU+accelerators: the large fraction of computing will be delivered by accelerators.
- Our HPC codes are not fully optimized to exploit all parallelism levels of these hydrid systems.
- The same for HTC codes

It is necessary to make an efficient exploitation of these systems

This activity aims ...

- training activities for efficient programming of existing and up-coming systems
- procure two or more small/medium systems for training, test and benchmarking activities (NOT for production).

Status

- several codes have been selected for benchmarking
- started activity of technology survey
- plan to procure one or two systems for the beginning of 2018
- we have met several vendors: Intel, IBM and NVIDIA
- candidate systems: Xeon+FPGA, Power9+NVLINK2+Volta, Xeon+NVLINK2+Volta
- collaboration with vendors á la **OpenLAB** of CERN for early access to new systems
- dissemination, workshops, schools