21st NEXT (Numerical Experiment of Tokamak) Workshop
2016/Mar./10-11
Kyoto Terrsa, Kyoto, Japan

Acceleration of PIC Code with Xeon Phi, GPU, and SPARK64 XIfx

H. Naitou Yamaguchi University

Acknowlegements:

V.K. Decyk	UCLA
M. Yagi, Y. Kagei, S. Miyamoto	JAEA
Y. Tauchi	Yamaguchi University
R. Toyota	Yamaguchi University (Student)

Outline

- Introduction
- Acceleration of PIC code by INTEL Xeon Phi 5110P
- Acceleration of PIC code by FUJITSU SPARC64^{TM} Xifx
- Acceleration of PIC code by NVIDIA GPU: K80
- Conclusions

Ref.1 V. K. Decyk, T. V. Singh,
"Adaptable Particle-in-Cell algorithms for graphical processing units",
Computer Physics Communications 182 (2011) 641-648.
Ref.2 V. K. Decyk, T. V. Singh
"Particle-in-Cell algorithms for emerging computer architectures"
Computer Physics Communications 185 (2014) 708-719.

Several New Architectures

	Xeon Phi 5110P (many cores)	SPARC64™ XIfx (multi-cores)	TESLA K80 (GPU)
Host / Accelerator	HOST + Coprocessor native mode (stand alone) offline mode	HOST	HOST + GPU
Performance of one core	medium	high	quite low
Number of cores	60 240 (hyper threading)	32+2	Dual GK210 GK210: 2496 cores 13 SMX 1 SMX = 192 cores
FLOPS	1.011 TFLOPS	0.908 TFLOPS	1.87 TFLOPS (DP) 5.6 TFLOPS (SP)
Programing model	OpenMP	OpenMP	CUDA OpenACC, OpenCL

Cell/Tile Parallel Method

One fine grain (thread) treats one cell/tile
and particles inside the cell/tile.

PUSH: Particle pushing

SOURCE: Charge Assignment

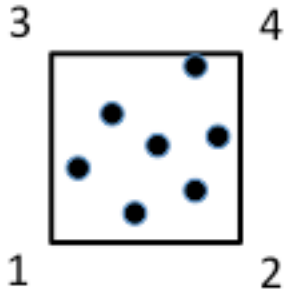
Additional computation is needed:

SORT: after every pushing, particles must move to proper cells/tiles.

- Eliminate random access from/to memory from PIC code (cell).
  Effective use of cash (tile)
- fine grain parallelism (cell / tile)
  = streaming algorithm

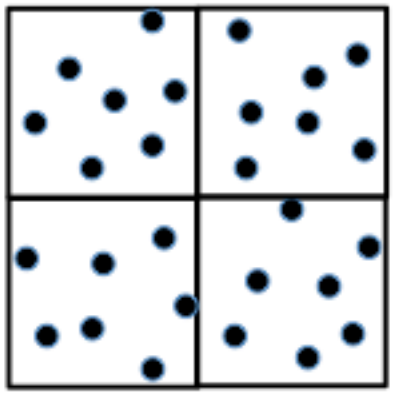
Cell or Tile

no dependency between cells/tiles



cell

no random access



mx = 2
my = 2

tile

limited random access (effective use of cash)

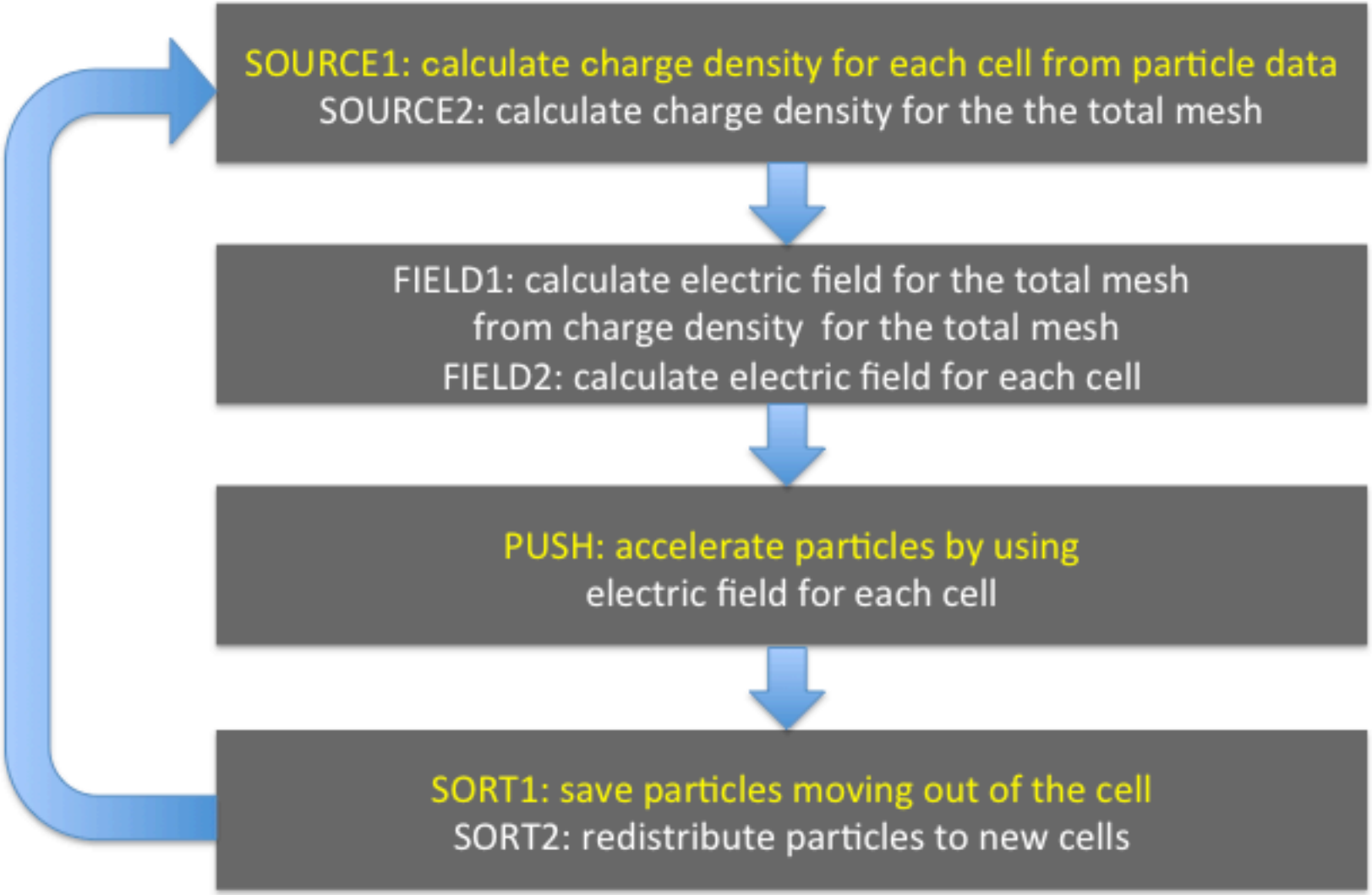
reduced load for reordering of particles (SORT)

Benchmark PIC Code

- 2D electrostatic PIC code.
- Double floating-point precision.
- Follow only electron dynamics.
- Linear interpolations for charge assignment and particle acceleration.
- No external magnetic field.

256×256 mesh
100 particles / mesh
1000 steps
$\Delta t = 0.1$

1 step for Cell/Tile-Parallel PIC Code



Caution : join SORT1 to PUSH for eliminating multiple data access

Xeon Phi

Ref:Wikipedia

- Intel
- MIC (Many Integrated Core) architecture
- coprocessor

Xeon Phi 5110P

November 12, 2012

22nm 60 cores 1.053GHz

double precision 1.011 TFLOPS

TOP500	November 2013	world fastest supercomputer	
Tianhe-2		Intel Ivy Bridge Xeon + Xeon Phi	33.86 PetaFLOPS

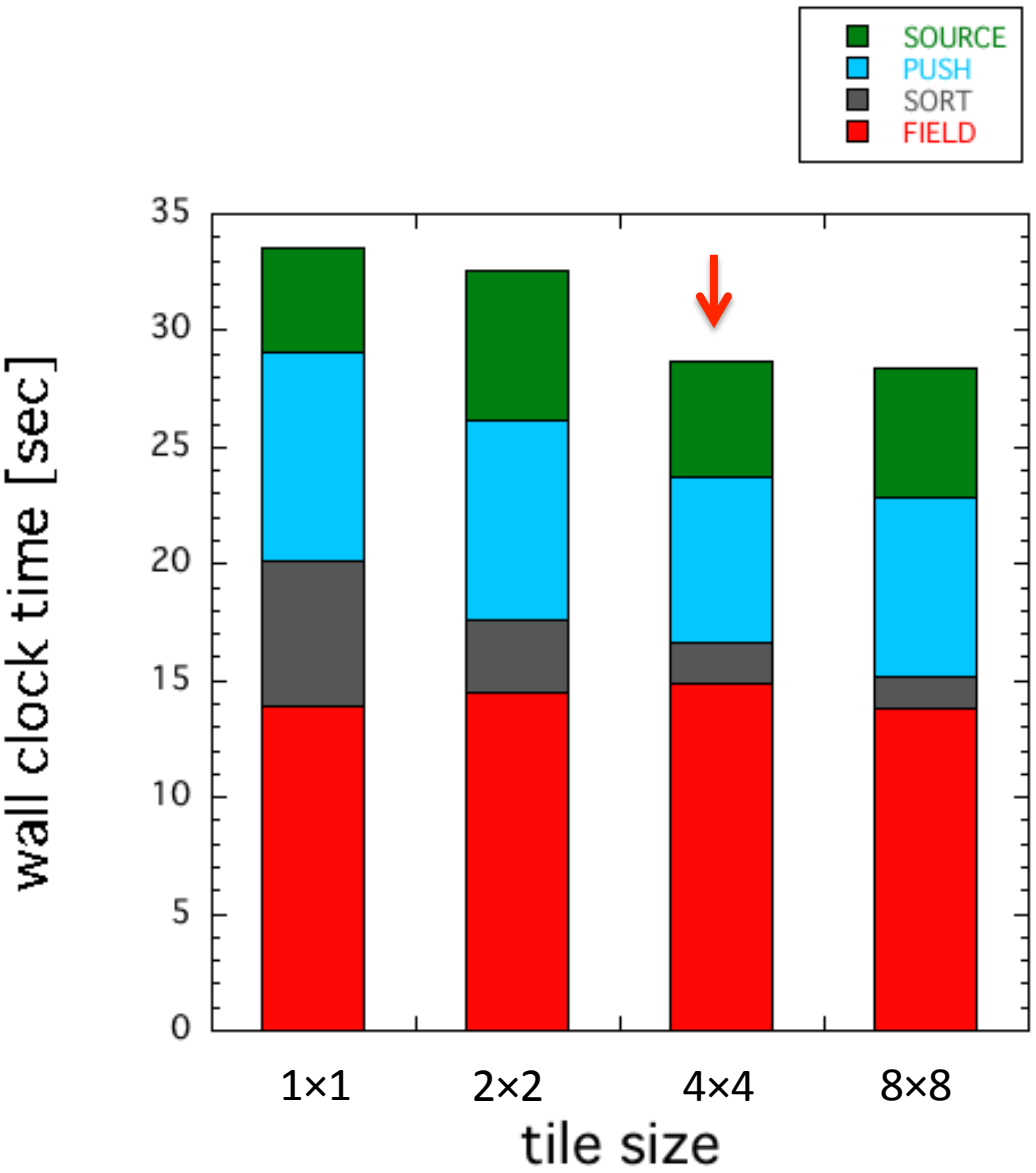
180 MIC nodes was added to Helios

One node consists of:

- Host CPU
  - Xeon processor E5 2450 x 2
    - 8 cores, 24 GB
- Coprocessor
  - Xeon Phi 5110P x 2
    - 60 cores, 8 GB

Offload execution mode	CPU -> MIC
Coprocessor native execution mode	MIC (ssh)
Symmetric execution mode	CPU+MIC (mpi)

Effect of Tile Size: Xeon Phi



256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

SOURCE

PUSH

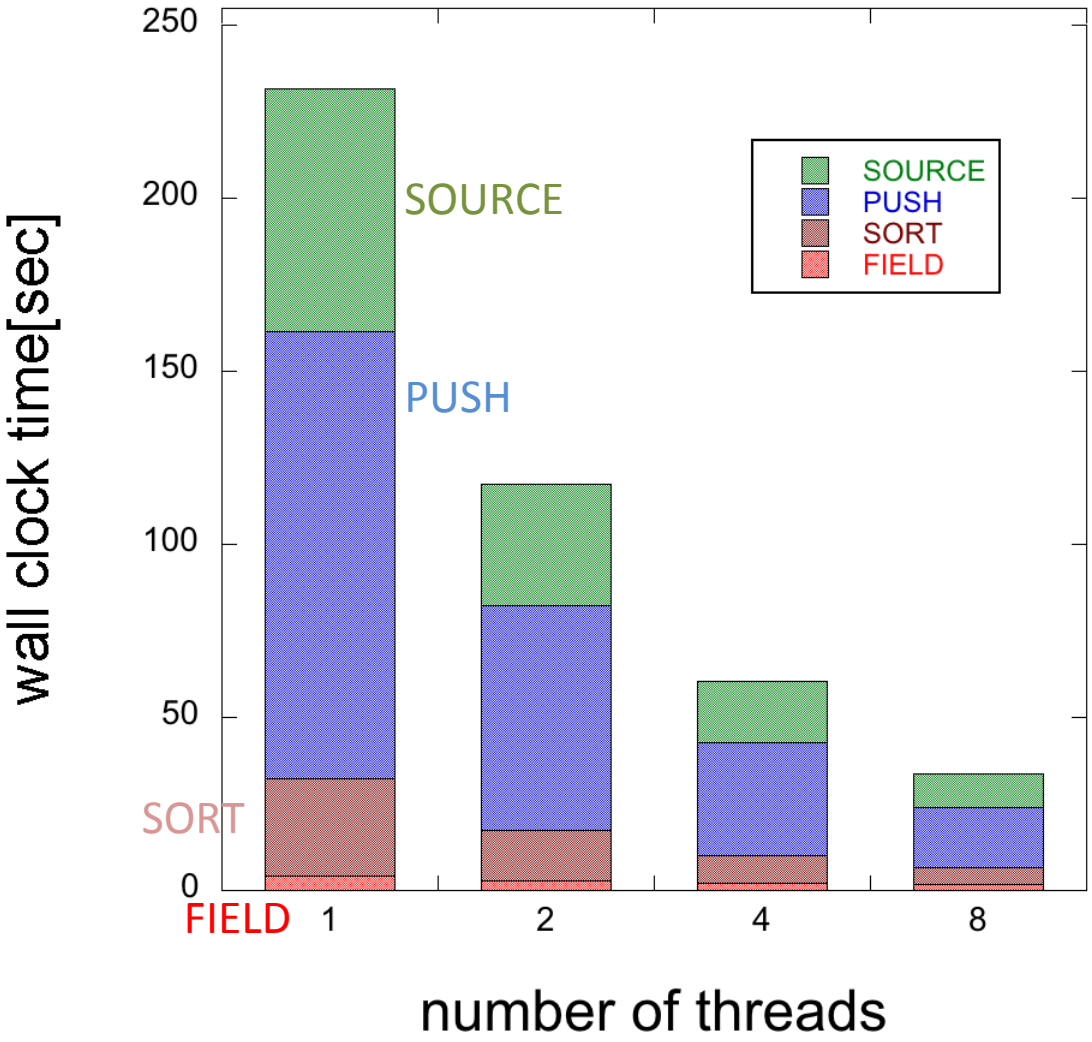
SORT

FIELD

FFT for a single core is used.

Scaling for Host CPU (1)

Tile-Parallel

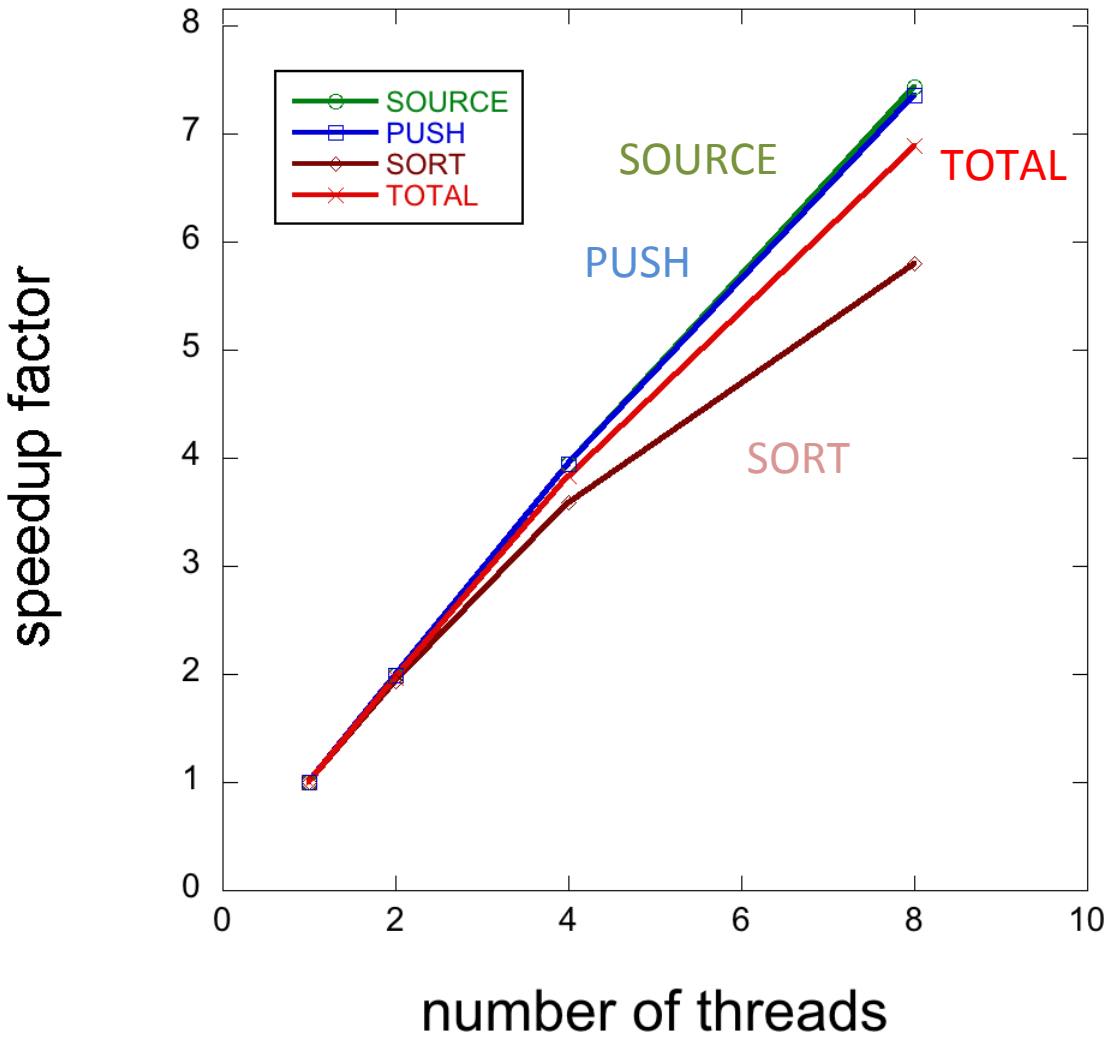


mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Scaling for Host CPU (2)

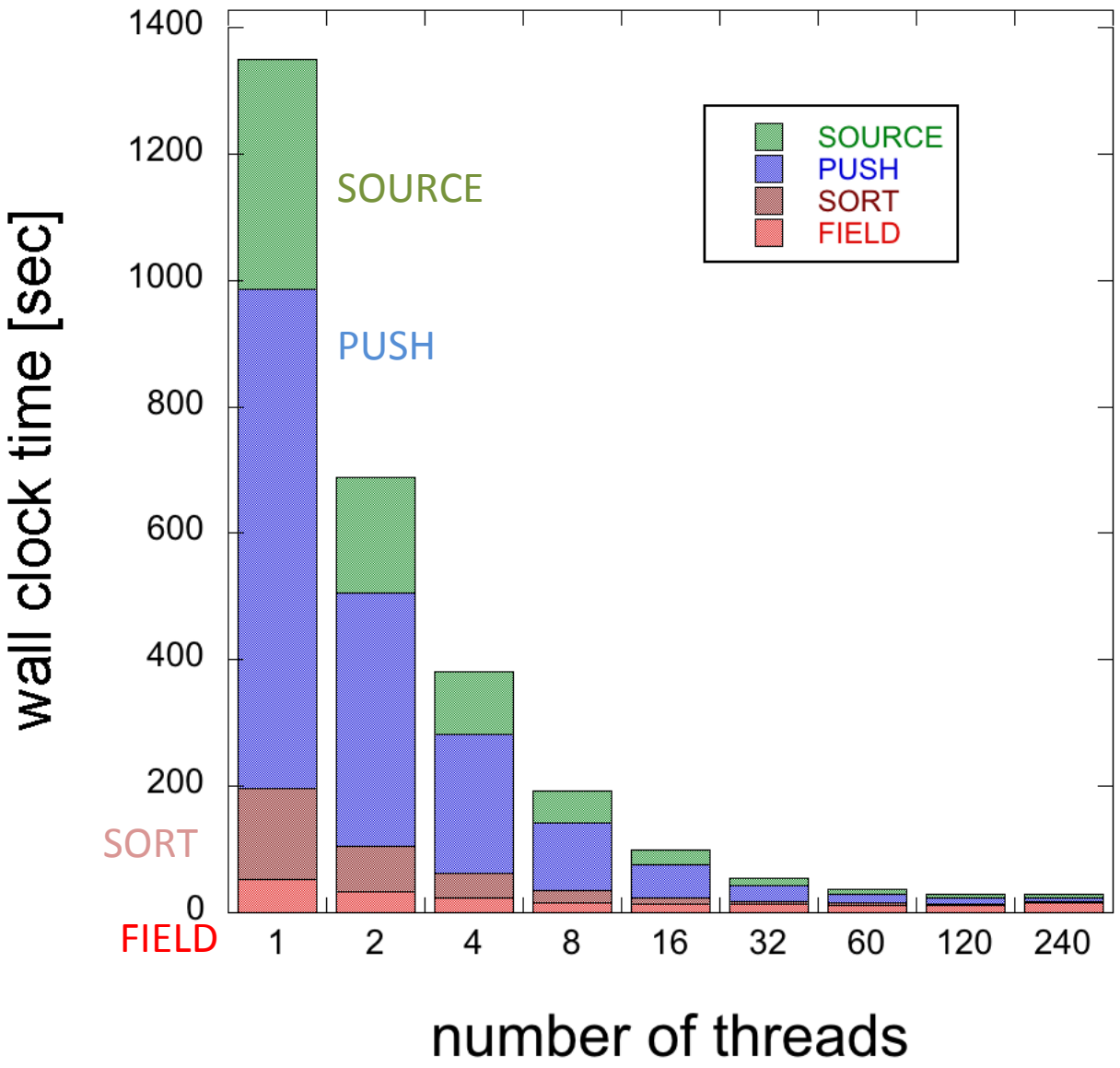
Tile-Parallel



mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Scaling for Intel MIC (1)

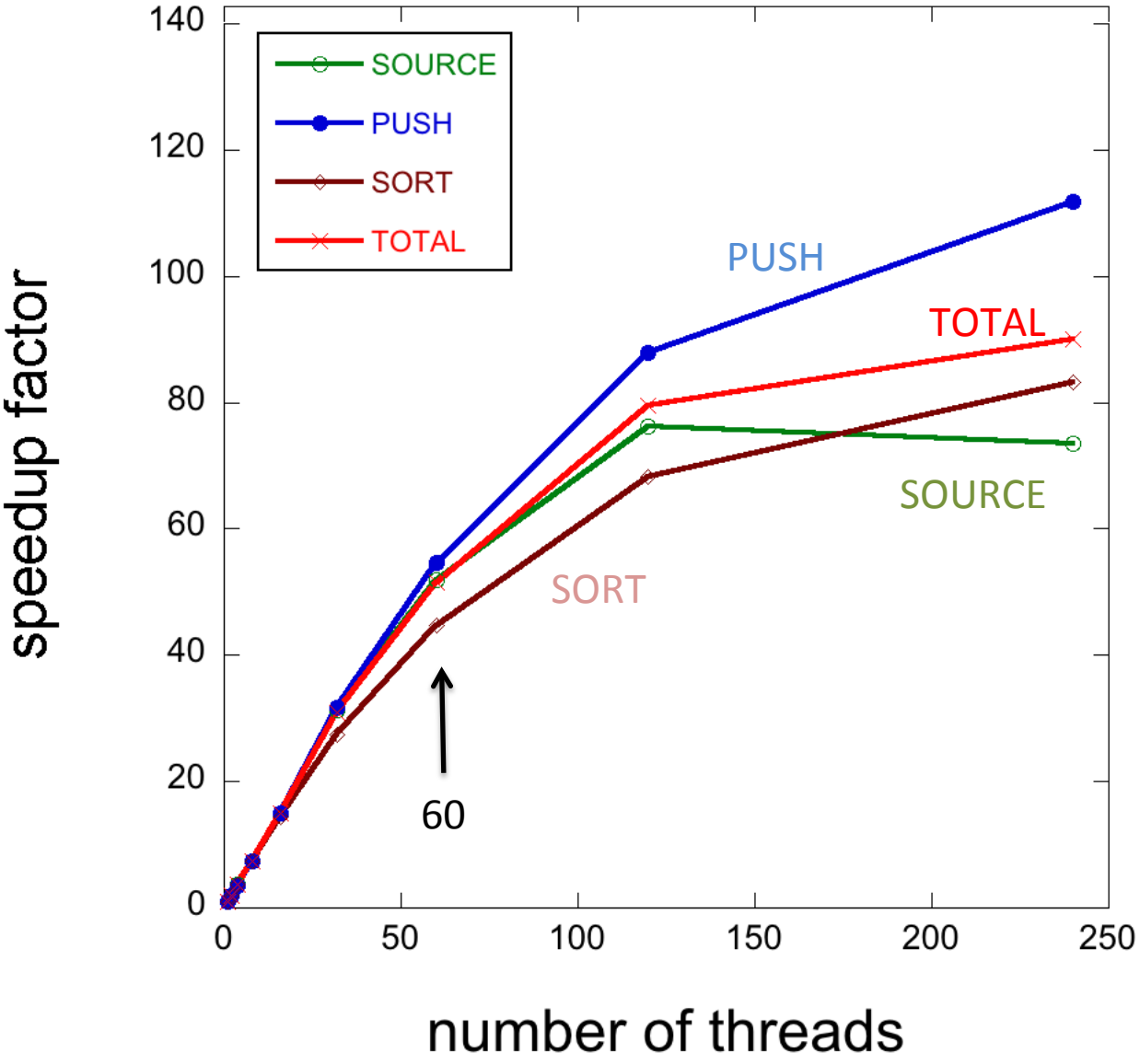


Tile-Parallel

mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Scaling for Intel MIC (2)

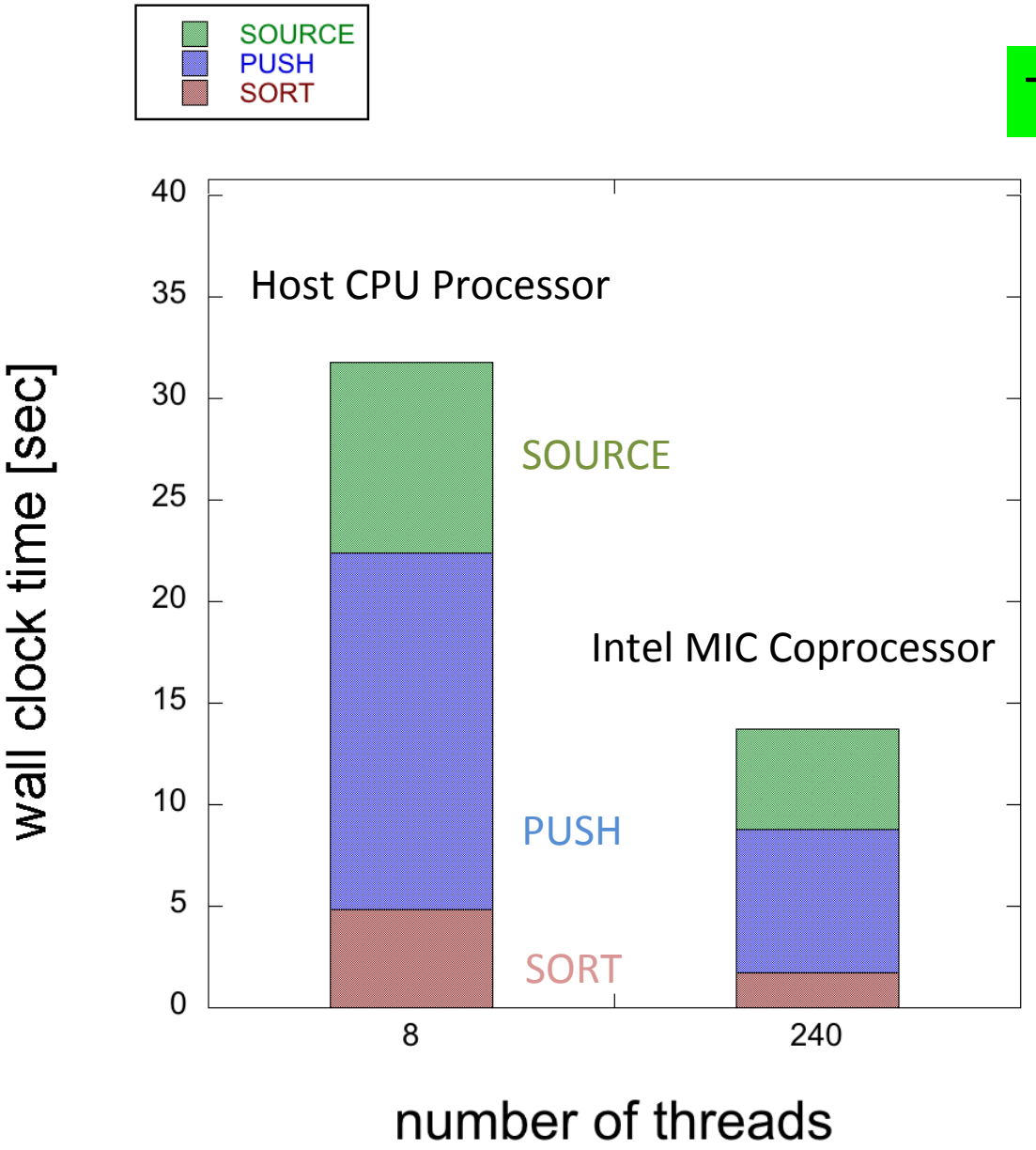


Tile-Parallel

mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Host CPU vs. Intel MIC



mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Speedup Factor

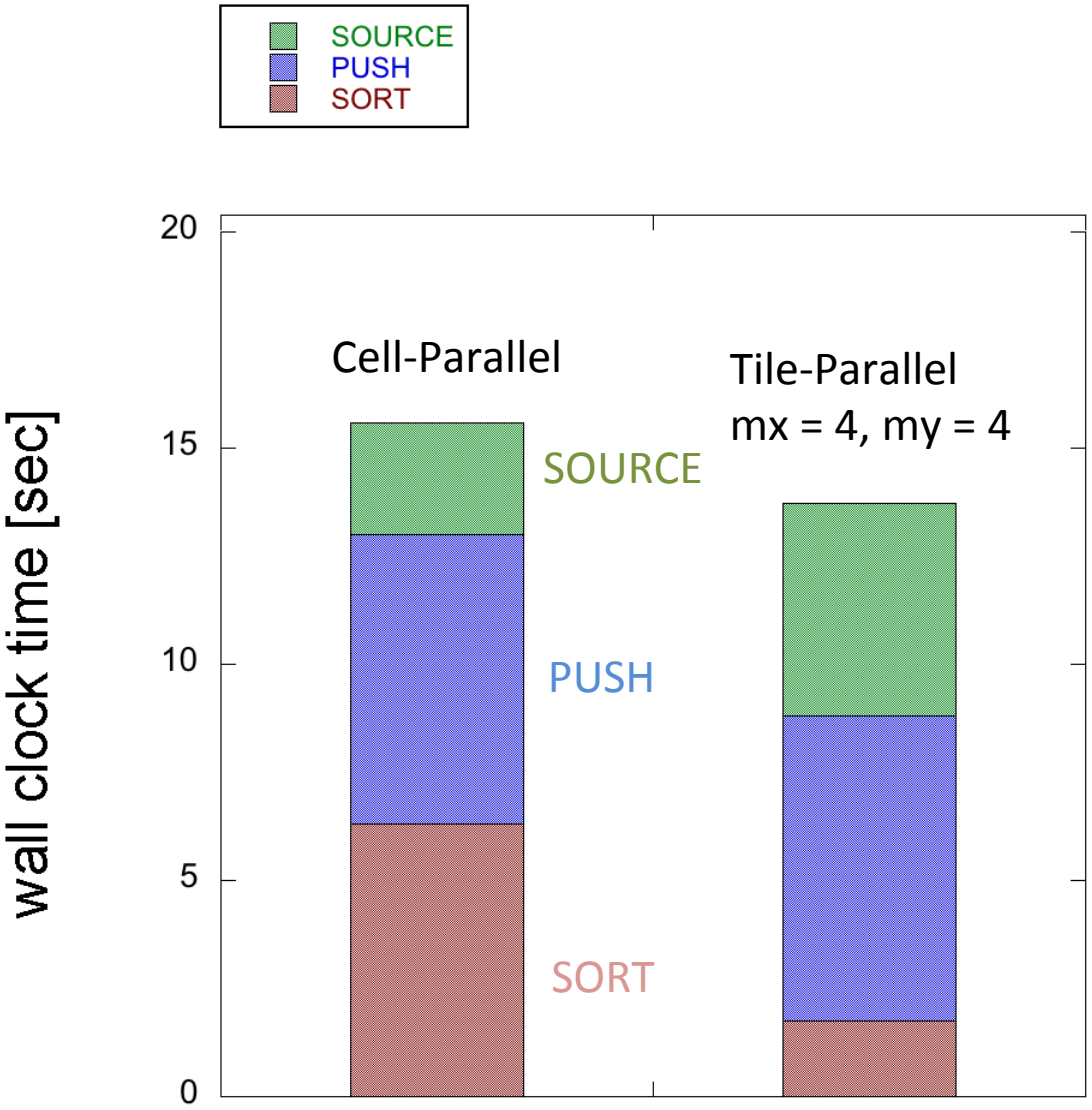
CPU time (8 threads) / Intel MIC time (240 threads)

Tile-Parallel

mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Cell vs. Tile



Intel MIC
240 threads

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

New Plasma Simulator

- PRIMEHPC FX100

  2592 nodes

  2.58 PFlops

  one node:

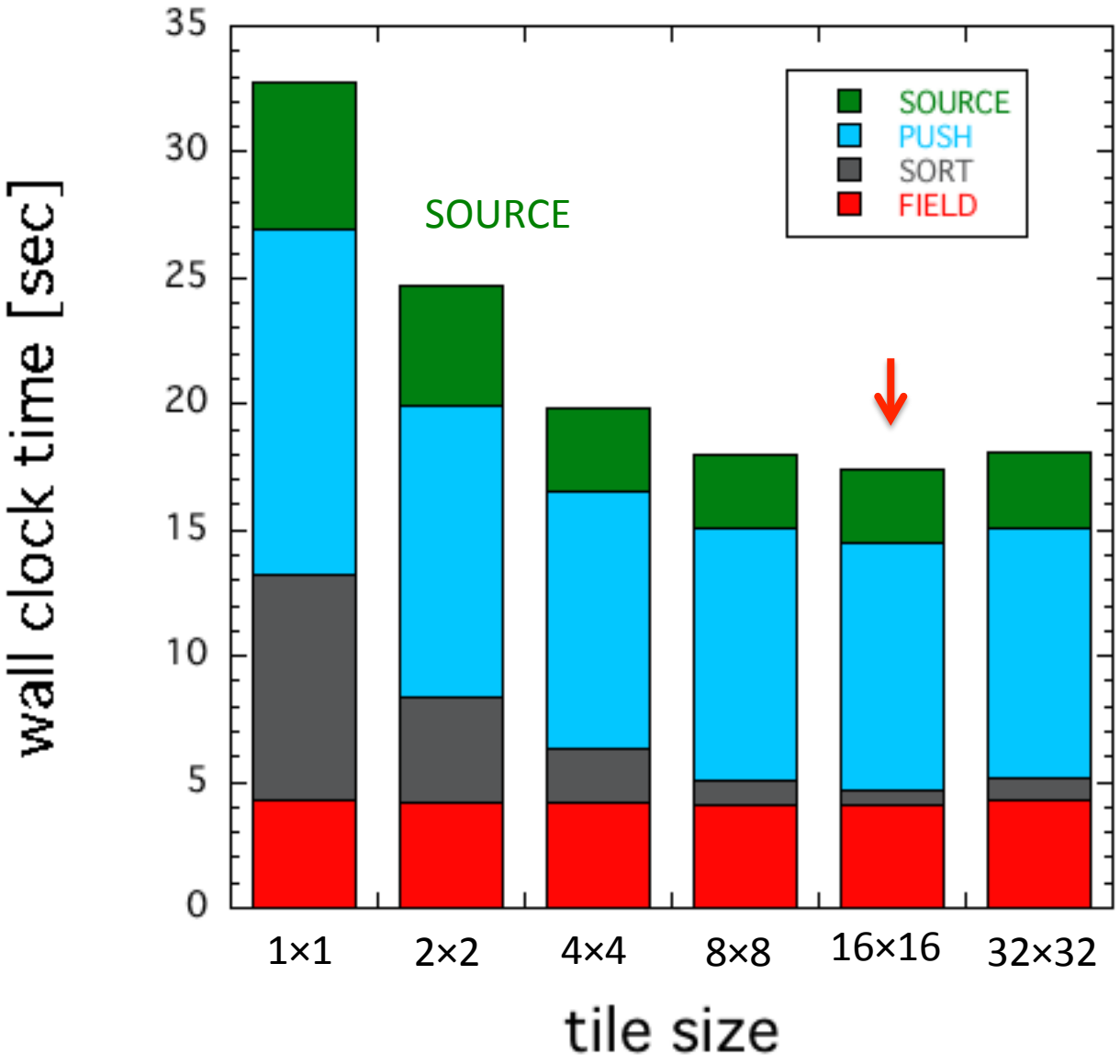
  SPARK64™ XIfx

  32+2 cores

  0.908 TFlops

  32 GB

Effect of Tile Size: SPARC64



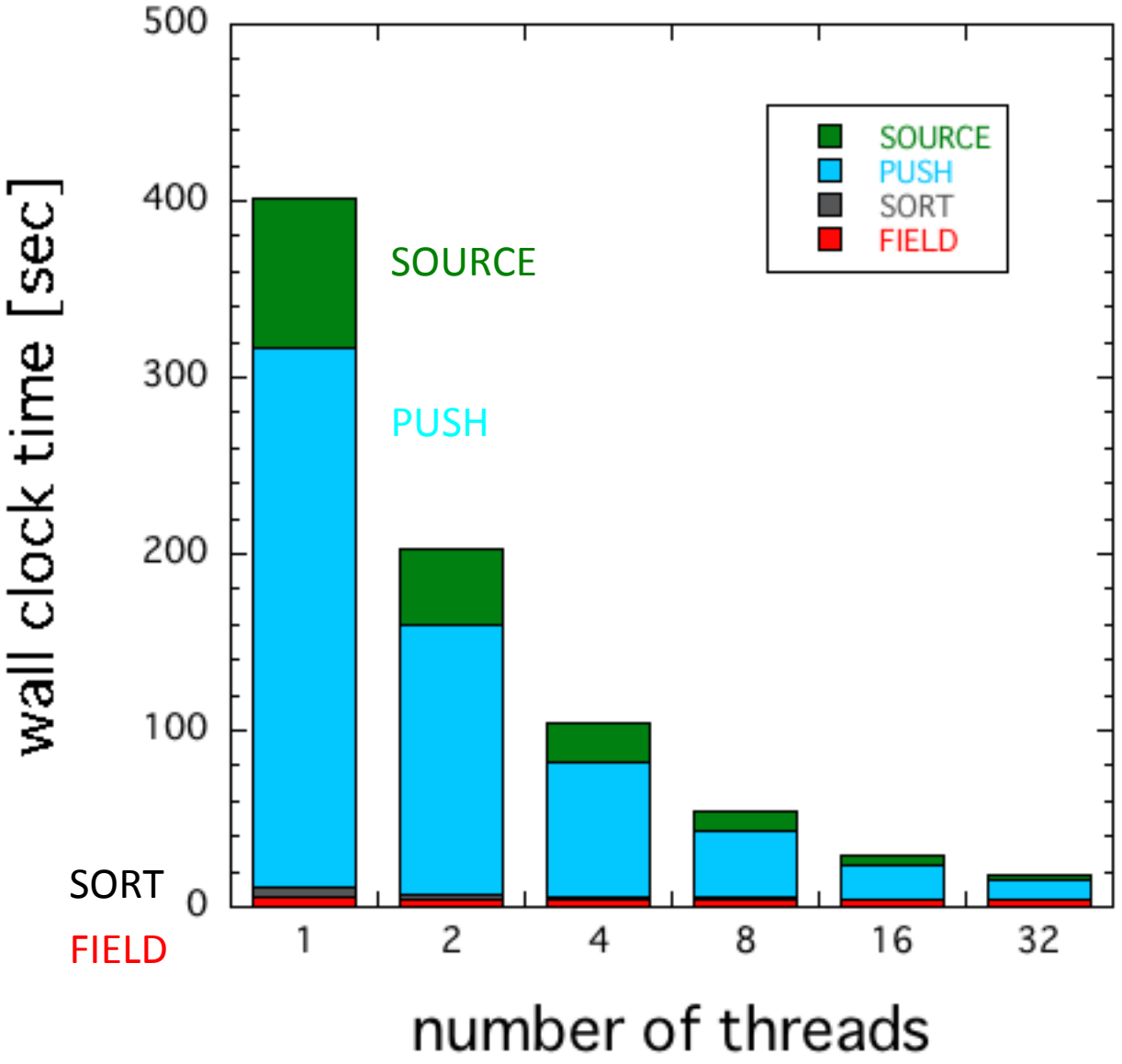
256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

PUSH

FIELD

FFT for a single core is used.

Scaling for SPARC64 XIfx (1)

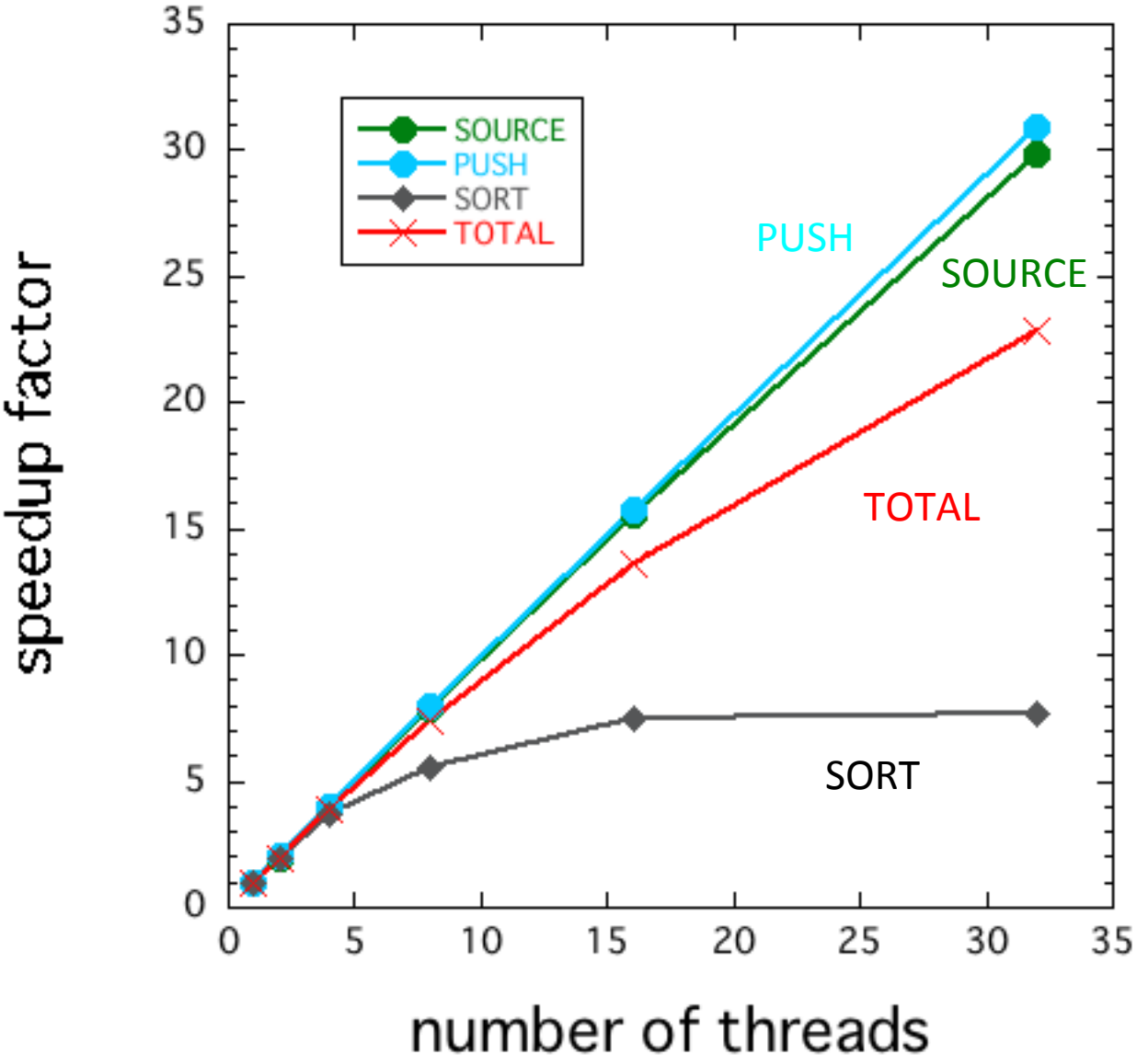


Tile-Parallel

mx = 16
my = 16

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Scaling for SPARC64 XIfx (2)

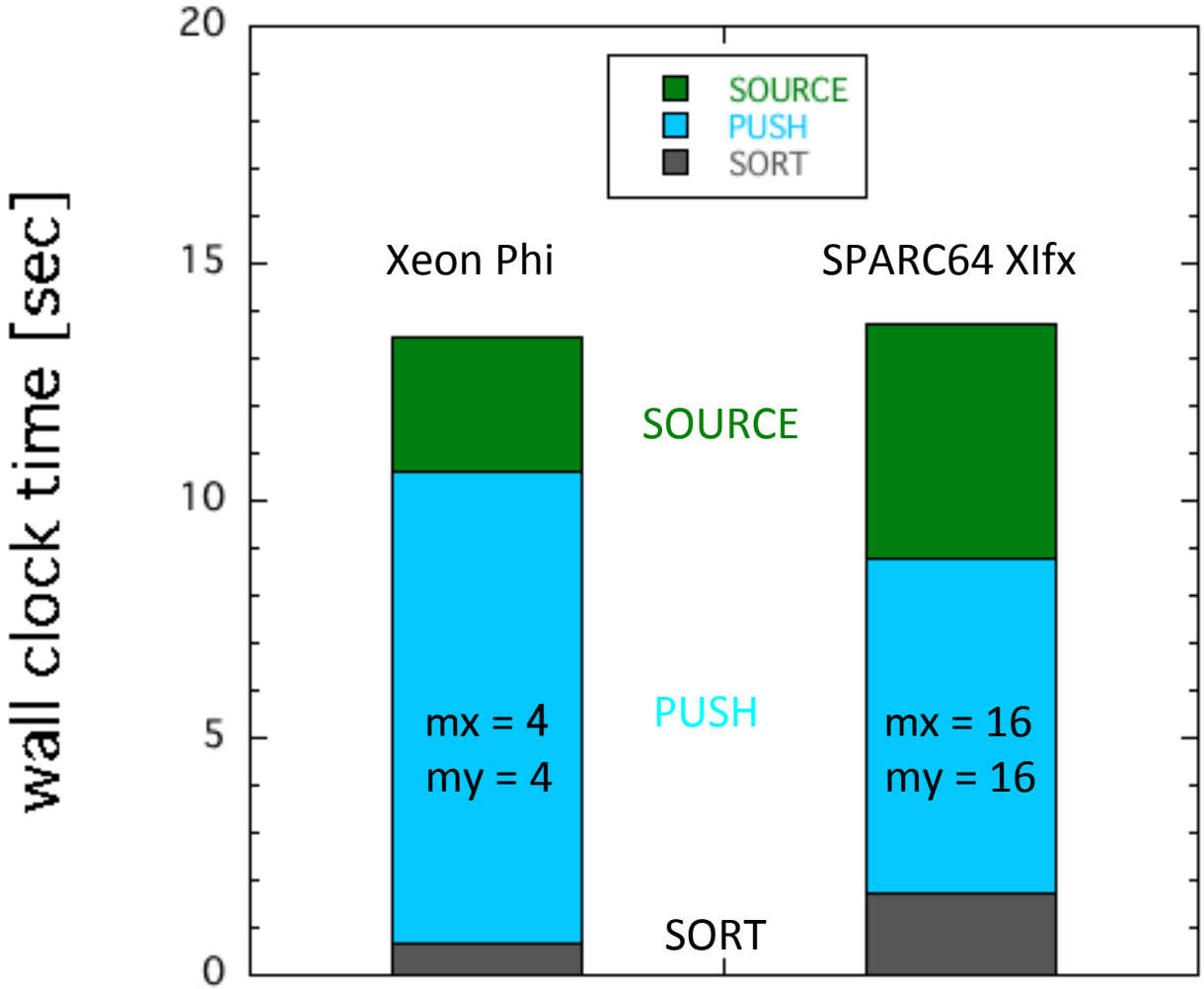


Tile-Parallel

mx = 16
my = 16

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Xeon Phi vs. SPARC64 XIfx

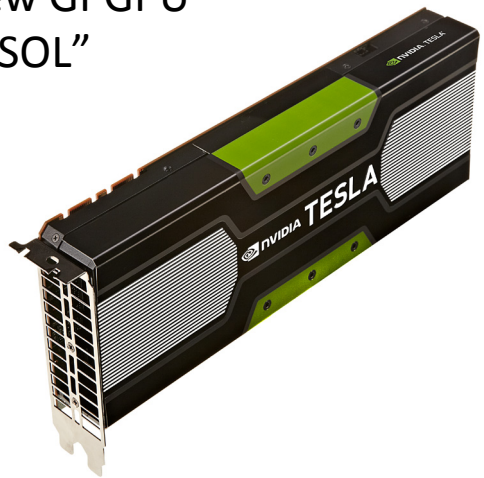


Tile-Parallel

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

GPU Computing Environment

Helios new GPGPU cluster “SOL”



Kepler architecture

NVIDIA TESLA K80

Dual GPU (GK210)

GK210:

13 SMX

2496 streaming processors

PGI CUDA Fortran

ver. 7.0

NVIDIA GeForce GTX TITAN

14 SMX

2688 streaming processors

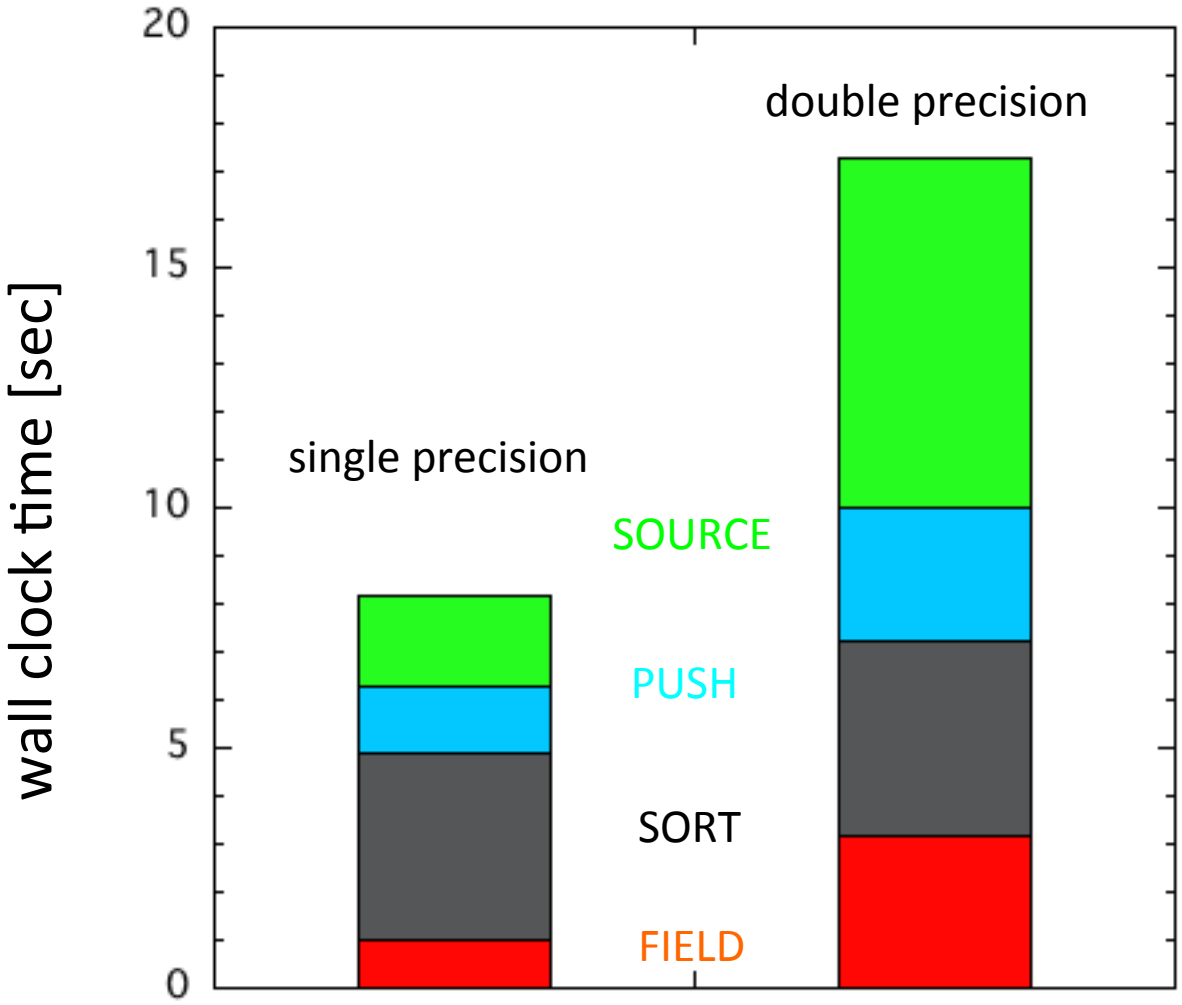
Naitou lab.

collision free vs. collision resolving

- Cell-Parallel
  collision free algorithm
- Tile-Parallel
  one thread treats one tile :
  collision free algorithm
  limited to small tile
  one block treats one tile:
  one block = one SMX
  collision resolving algorithm
  use shared memory
  atomicAdd is used for SOURCE

Performance of NVIDIA TESLA K80

One GPU (GK210)

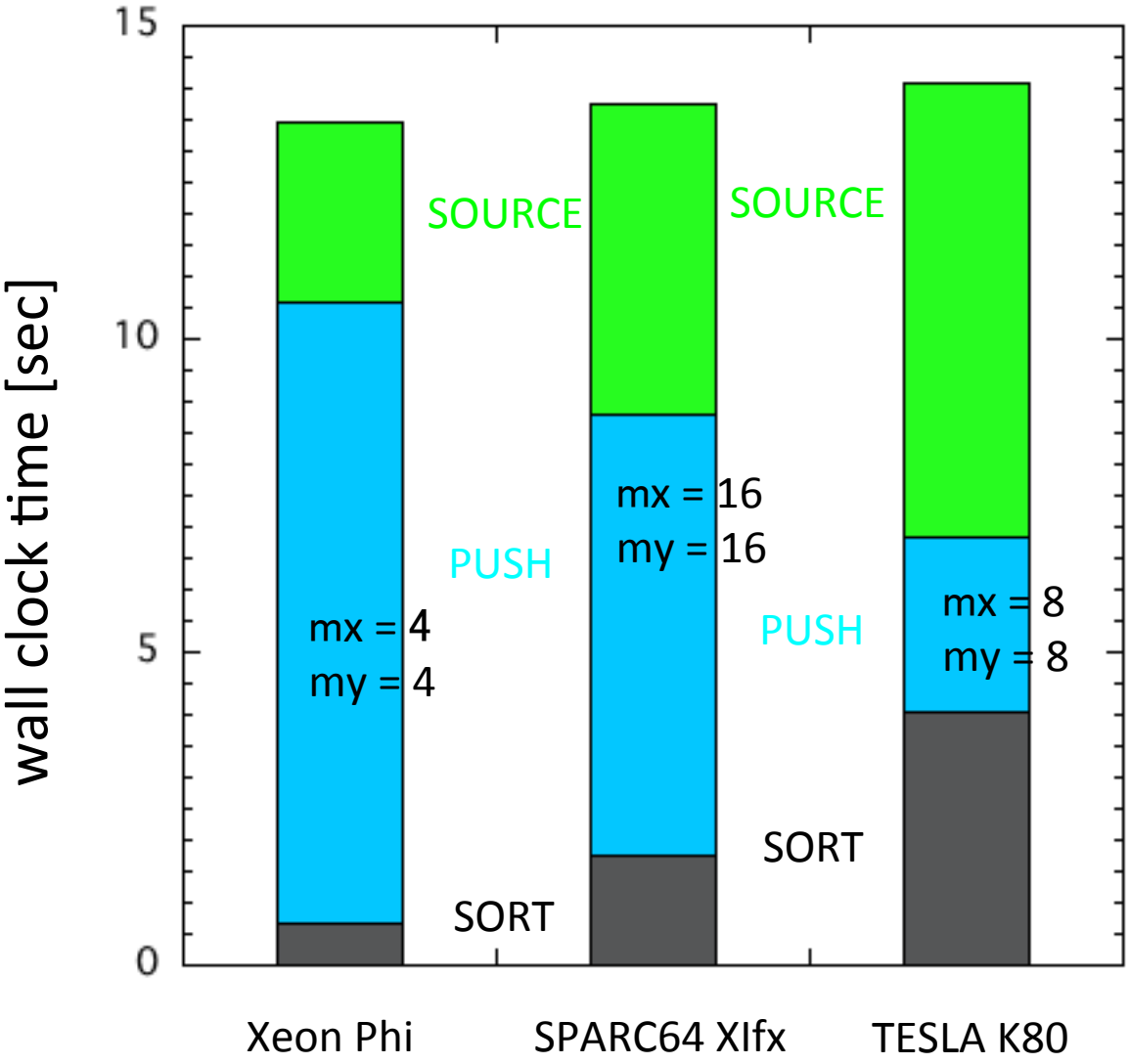


Tile-Parallel

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

mx = 8
my = 8

Xeon Phi vs. SPARC64 Xifx vs. TESLA K80



Tile-Parallel

256×256 mesh
100 particles / mesh
1000 steps
Δt = 0.1

Conclusions (1)

Basic code is parallelized for multi-core CPU by OpenMP.

Performance of Intel MIC Coprocessor

- Tile-parlallel PIC code can run with intel MIC coprocessor Xeon Phi 5110P without any modification.
- Native mode (stand alone mode) is used.
- As the number of threads increases, excellent scaling is obtained.
- Speedup factor for total particle time is 2.3.
- Cell-parallel code is comparable to tile-parallel code.

Performance of SPARC64 Xifx

- OpenMP is used
- Tile-parallel PIC code is tested for multi-core system of SPARC64 XIfx.
- Excellent parallel scaling is obtained.
- Performance of PIC code for Xeon Phi and SPARK64 XIfx is almost similar.

Conclusions (2)

Performance of GPU

- Tile-parallel PIC code for CPU-GPU system is tested for NVIDIA TESLA K80.
- CUDA FORTRAN is used.
- CUDA FFT is used.
- The performance of the 2D PIC code is similar to Intel Xeon Phi and SPARC Xifx.
- Atomic-add is slow for double precision floating point operations.