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

H. Naitou
Yamaguchi University

Acknowledgements:
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™ 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”,
Ref.2 V. K. Decyk, T. V. Singh
“Particle-in-Cell algorithms for emerging computer architectures”
#### Several New Architectures

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

<table>
<thead>
<tr>
<th>cell</th>
<th>tile</th>
</tr>
</thead>
<tbody>
<tr>
<td>no random access</td>
<td>limited random access (effective use of cash)</td>
</tr>
<tr>
<td></td>
<td>reduced load for reordering of particles (SORT)</td>
</tr>
</tbody>
</table>
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
Δt = 0.1
1 step for Cell/Tile-Parallel PIC Code

SOURCE1: calculate charge density for each cell from particle data
SOURCE2: calculate charge density for the total mesh

FIELD1: calculate electric field for the total mesh from charge density for the total mesh
FIELD2: calculate electric field for each cell

PUSH: accelerate particles by using electric field for each cell

SORT1: save particles moving out of the cell
SORT2: redistribute particles to new cells

Caution: join SORT1 to PUSH for eliminating multiple data access
Xeon Phi

- 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
$\Delta t = 0.1$

FFT for a single core is used.
Scaling for Host CPU (1)

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

```
m_x = 4
m_y = 4
```

Tile-Parallel

- 256×256 mesh
- 100 particles / mesh
- 1000 steps
- $\Delta t = 0.1$
Scaling for Host CPU (2)

256x256 mesh
100 particles / mesh
1000 steps
$\Delta t = 0.1$

$mx = 4$
$my = 4$
Scaling for Intel MIC (1)

- Tile-Parallel
  - $m_x = 4$
  - $m_y = 4$

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

Diagram showing wall clock time (in seconds) against the number of threads for different tasks: SOURCE, PUSH, SORT, FIELD.
Scaling for Intel MIC (2)

Tile-Parallel

mx = 4
my = 4

256×256 mesh
100 particles / mesh
1000 steps
\(\Delta t = 0.1\)
Host CPU vs. Intel MIC

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

**Tile-Parallel**

- $mx = 4$
- $my = 4$

**Graph:**
- **Host CPU Processor**
- **Intel MIC Coprocessor**
- Wall clock time [sec]
- Number of threads

- SOURCE
- PUSH
- SORT
CPU time (8 threads) / Intel MIC time (240 threads)

Tile-Parallel

- $m_x = 4$
- $m_y = 4$

- 256×256 mesh
- 100 particles / mesh
- 1000 steps
- $\Delta t = 0.1$
Cell vs. Tile

Intel MIC
240 threads

Cell-Parallel
Tile-Parallel
mx = 4, my = 4

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

- 1×1
- 2×2
- 4×4
- 8×8
- 16×16
- 32×32

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

FFT for a single core is used.
Scaling for SPARC64 XI fx (1)

Tile-Parallel

mx = 16
my = 16

256x256 mesh
100 particles / mesh
1000 steps
Δt = 0.1
Scaling for SPARC64 Xifx (2)

- 256x256 mesh
- 100 particles / mesh
- 1000 steps
- $\Delta t = 0.1$

Tile-Parallel

- $m_x = 16$
- $m_y = 16$
Xeon Phi vs. SPARC64 XIfx

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

Tile-Parallel

Xeon Phi

SPARC64 XIfx

source

push

sort

$\begin{align*}
 mx &= 4 \\
 my &= 4 \\
 mx &= 16 \\
 my &= 16
\end{align*}$
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)

- Single precision
- Double precision

Tile-Parallel

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

mx = 8
my = 8

Wall clock time [sec]
Xeon Phi vs. SPARC64 XIfx vs. TESLA K80

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

**Tile-Parallel**
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.
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.