Effective use of XEON PHI accelerators for the PSC (Plasma Simulation Code)

Karl-Ulrich Bamberg and Hartmut Ruhl
Inhaltsverzeichnis

1. Implementation
2. Scaling - Hardware measurements
3. Large scale tests
4. Code adaptions

All Logos are trademarks of their respective companies. Picture see [1, Intel presentation LRZ]
Offloading: Copy particles up and down - Pipelining

Pro
- Easy to implement: No adaptions or special libraries needed (almost)
- Circumvents Memory-Limit

Contra
- PCIe-Bandwidth is Limiting Factor!
- Special memory limit requirements: Multi-rank access to one co-processor

Status: Implemented, improvements in memory awareness and pipelining possible.
Bandwidth to flop calculation

- Calculation speed: 1 TFLOP/s Double Precision
- 1 Particle: 80 Byte
- Assume pipe lining bandwidth full duplex: 5 GB/s peak → 62.500.000 Particle/s
- \( \approx 16.000 \) FLOP/Particle necessary

Real life bandwidth usually: 3 GB/s → 26.000 FLOP/Particle necessary for full utilization.

Boris pusher utilizes around 2.000 (TODO) FLOP/particles

Q.E.D. event generators issue massive FLOP/particles
Native mode: Run totally on MIC, communicate from MIC to MIC

Contra
- Libraries (hdf5, szip, mpi, fortran) need to be available for MIC
- Minor adaption to buildsystem
- Harder load balancing
- Wastes CPU resources

Pro
- Only Necessary Particles are transferred - circumvent limited Bandwidth, everything on Card (Fields and things)
- Less code adaption

Status:
Working on Stampede, due to necessary MPI library.

Karl-Ulrich Bamberg and Hartmut Ruhl
Effective use of XEON PHI accelerators for the PSC (Plasma Simulation Code)
Native heterogeneous mode

- Use Xeon-Ranks for memory intensive tasks: E. g. wide grid area with thin plasma.
- Use MIC-Ranks with many threads for heavy calculations: E.g. dense plasma (up to the memory limit), or Q.E.D. event generators

- Overcome memory limit
- Bypass PCIe bottleneck
- Utilize dense CPU performance
Native strong scaling MIC

15-480 cores on different nr. of accelerators

Logical cores/Hyperthreading yield up to 30% speed up. No memory bandwidth bottleneck → no speed up on multiple cards with same total nr. of cores.
Native strong scaling MIC

15-480 cores on different nr. of accelerators

Communication no problem: Different nr. of co-processors with same total core count, does not affect total calculation time. With communication, no speedup by hyperthreading.
MIC-Scaling: 4 to 128 accelerators (240 to 8000 Cores)

Scaling matrix for our Intel XEON PHI adaption, running in hybrid mode (full native support for MIC architecture).

(512+ and more were not requestable, 256 only in hybrid mode till now)
Hybrid MIC-Scaling up to 256 accelerators (8192 Cores)

PSC scaling

Homogenous microsphere - Stampede - 60(MIC) + 4 (Host) ranks per node

Scaling matrix for our Intel XEON PHI adaption, running in hybrid mode (4 cores on host and 60 cores on MIC).

(512+ and more were not requestable)

Karl-Ulrich Bamberg and Hartmut Ruhl
Effective use of XEON PHI accelerators for the PSC (Plasma Simulation Code)
Host-Scaling from 32 to 4096 cores

PSC scaling

Stampede Host processors

Scaling matrix for the host processors on Stampede.

Karl-Ulrich Bamberg and Hartmut Ruhl
Effective use of XEON PHI accelerators for the PSC (Plasma Simulation Code)
In-Patch-Parallelization (Status: 80%)

Code adaption for Compiler-Autovectorization (Status: 50%)
→ Also AVX and maintainability benefit from this.

Load-Balancing:
- Memory awareness for MICs
  (Status: Memory limits already there, adaption trivial)
- Hybrid MPI-openMP parallelization
  → more threads for heavy patches.
  (Status: Already implemented, combine with load-balancer)
  Also possible for CPU acceleration (Status: Planned)

Ultimate Solution

Additional Parallelization-Level: Shadow-Patches
(Status: Possible as post-doc project)
Thank you for your attention!