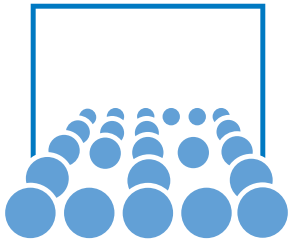
Advanced Scientific Computing on Intel Xeon Phi

PARCO'13, Garching (Germany)

Alexander Heinecke
Technische Universität München, Chair of Scientific Computing

Sep 13th 2013



Acknowledgment

For input and fruitful discussions I want to thank:

Intel DE: Michael Klemm, Hans Pabst, Christopher Dahnken

Intel Labs US/IN, PCL: Mikhail Smelyanskiy, Karthikeyan Vaidyanathan, Pradeep Dubey, Victor Lee, Christopher Hughes

Intel US: Mike Julier, Susan Meredith, Catherine Djunaedi, Michael Hebenstreit

Intel RU: Dmitry Budnikov, Maxim Shevtsov, Alexander Batushin

Intel IL: Ayal Zaks, Arik Narkis, Arnon Peleg

And a very special thank you to Joe Curley and the KNC Team for providing the SE10P cards for my test cluster!

Outline

Introduction to Intel Xeon Phi

Codes, which are not fully optimized for Xeon Phi, yet!

Molecular Dynamics

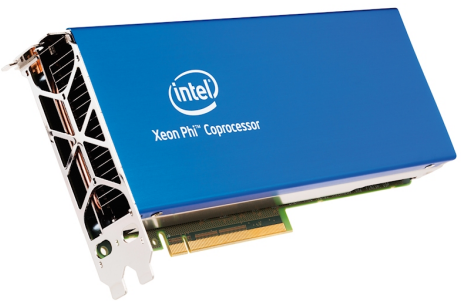
Earthquake Simulation - SeisSol, SuperMUC bid workload

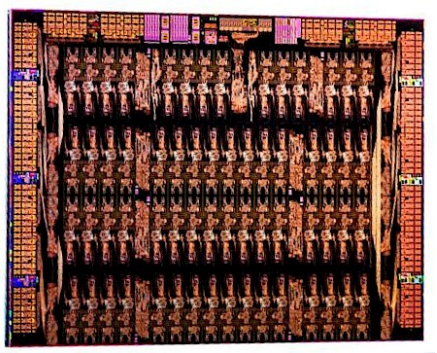
Hybrid HPL

Data Mining with Sparse Grids

Conclusions

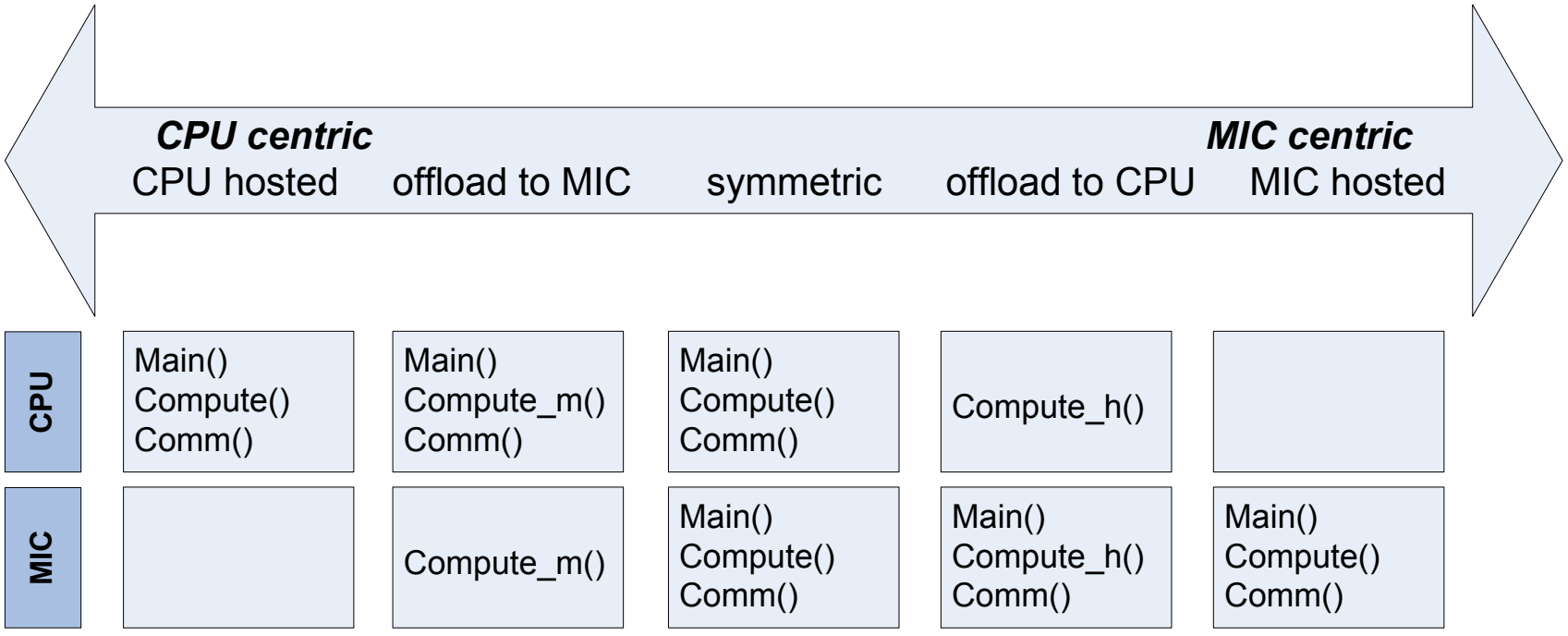
The Intel Xeon Phi SE10P (5110P)





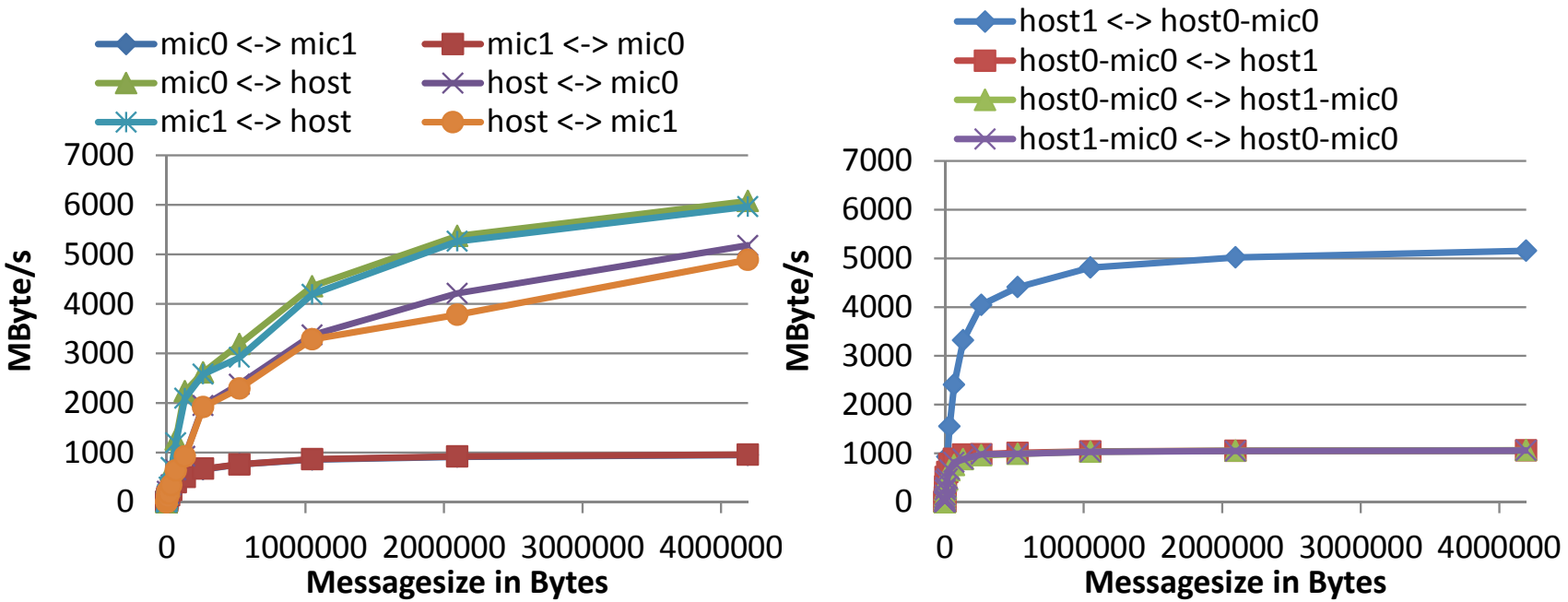
- 61(60) cores with 4 threads each at 1.1(1.053) GHz
- 32 KB L1$, 512 KB L2$, pre core
- L2$ are kept coherent through a fast on-chip ring bus network
- 512 bit wide vector instructions, 2 cycle throughput
- 1074(1010) GFLOPS Peak DP
- 2148(2020) GFLOPS Peak SP
- 8 GB GDDR5, 512 bit interface, 352(330) GB/s
- PCIe 2.0 Host-Interface

The Intel Xeon Phi Usage Model



- scalar: CPU hosted
- highly parallel: MIC offload
- offload through MPI: symmetric
- scalar phases: reverse offload
- highly parallel: MIC hosted

Infiniband-Connectivity of Xeon Phi



⇒ Xeon E5 has only 16 PCIe read buffers but 64 PCIe write buffers!

⇒ Start-up latency is 9us (host ↔ host: 1.5us, 1 hop), with 3 hops!

⇒ Using hosts as proxy like in GPU diret results in up to 6GB/s!

Outline

Introduction to Intel Xeon Phi

Codes, which are not fully optimized for Xeon Phi, yet!

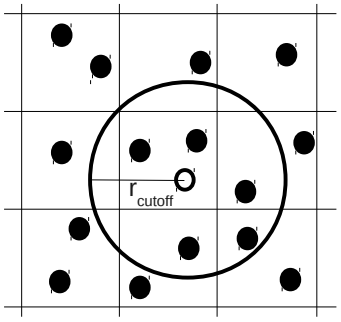
- Molecular Dynamics
- Earthquake Simulation - SeisSol, SuperMUC bid workload

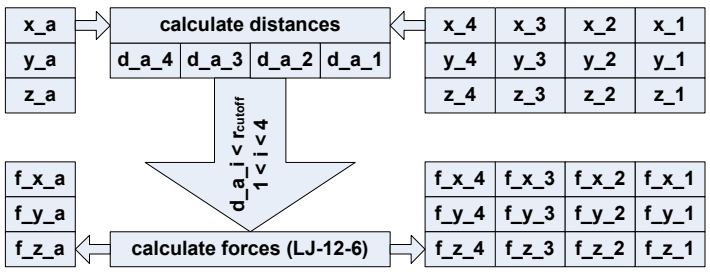
Hybrid HPL

Data Mining with Sparse Grids

Conclusions

Linked Cell Particle Simulations with ls1 mardyn*





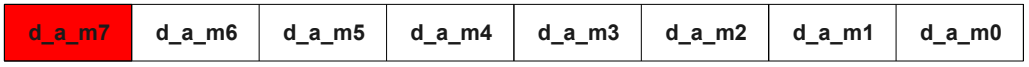
- with increasing vectorlength net-vector usage goes down (holes)
- Xeon E5 has too low bandwidth to L1 (32 byte read, 16 byte write)
- gather/scatter can be used to achieve 100% vectorregister load

⇒ Porting `ls1 mardyn` to MIC hosted mode is ongoing research

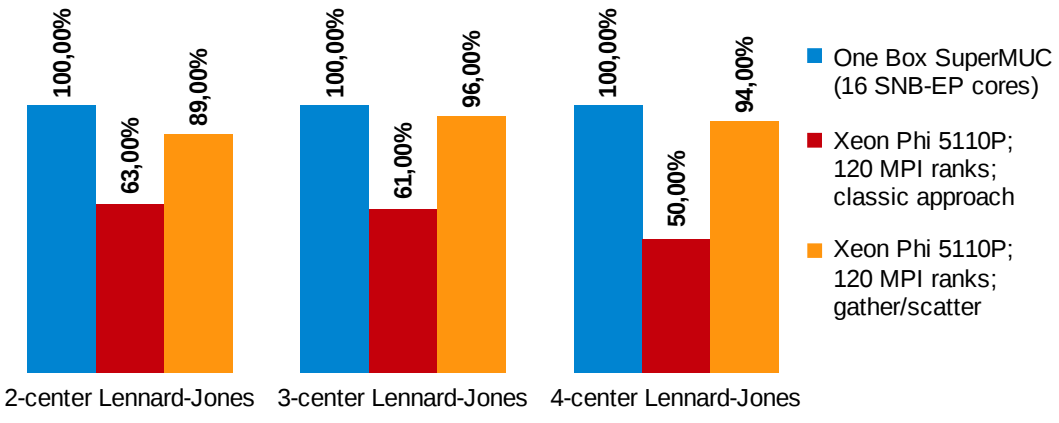
*Eckhardt, et.al.: 591 TFLOPS Multi-Trillion Particles Simulation on SuperMUC, ISC'13

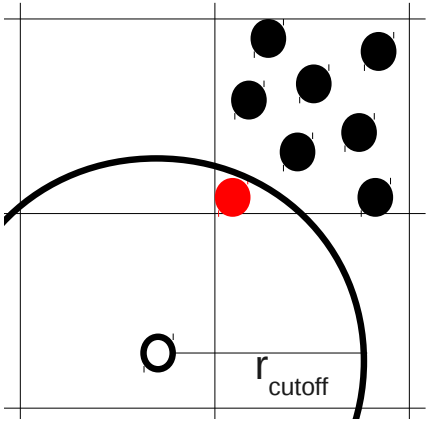
Single Card Performance Xeon Phi*

- Compute distances



- Execute force computation



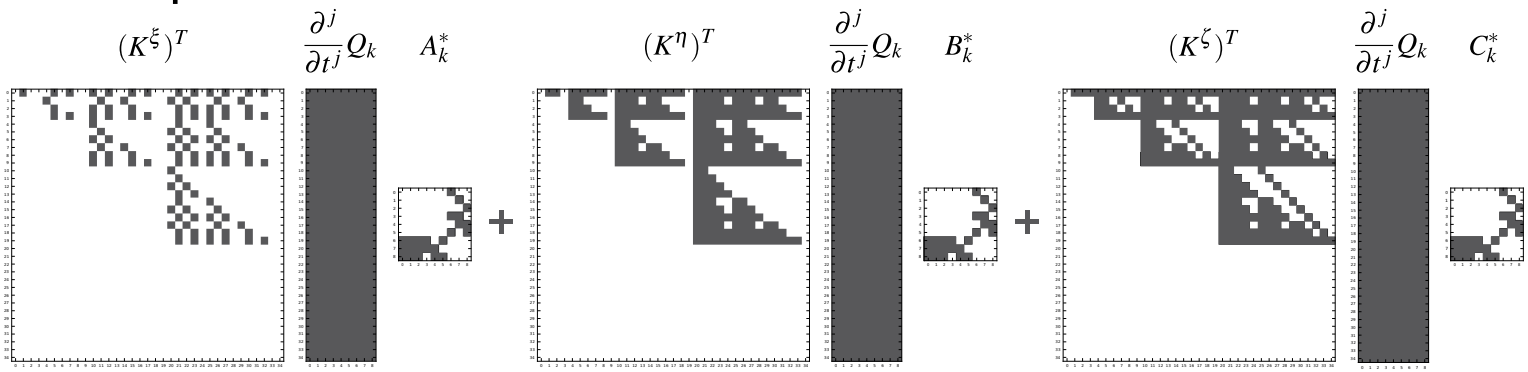


*Eckhardt, et.al.: Vectorization of Multi-Center, Highly-Parallel Rigid-Body Molecular Dynamics Simulations, Poster SC'13, accepted

SeisSol*

SeisSol is a software-package for the simulation of seismic wave phenomena on unstructured grids, based on the discontinuous Galerkin method com- bined with ADER time discretization.

On key-routine is the ADER time integration working on following matrix patterns:



$\Rightarrow$ we are currently at 1.3X for compute kernels comparing 1 Xeon Phi to 1 dual socket SNB-EP!

*Breuer, et.al.: Accelerating SeisSol by Generating Vectorcode for Sparse Matrix Operators, PARCO'13, accepted

*Heinecke, et.al.: Optimized Kernels for large scale earthquake simulations with SeisSol, an unstructured ADER-DG code, Poster SC'13, accepted

Outline

Introduction to Intel Xeon Phi

Codes, which are not fully optimized for Xeon Phi, yet!

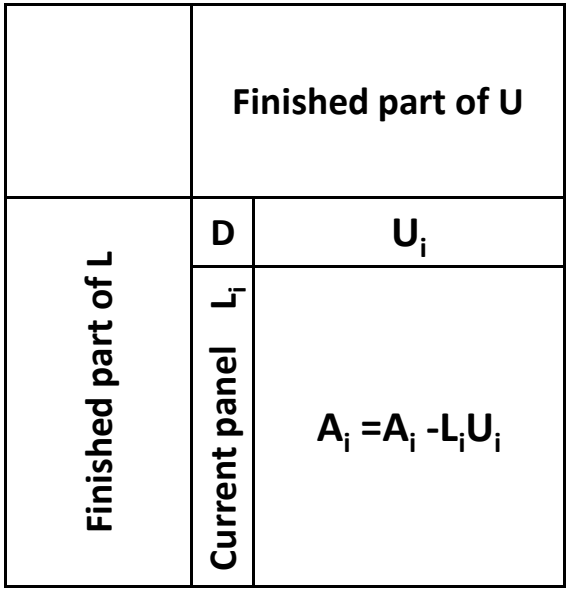
- Molecular Dynamics
- Earthquake Simulation - SeisSol, SuperMUC bid workload

Hybrid HPL

Data Mining with Sparse Grids

Conclusions

The HPL Benchmark*

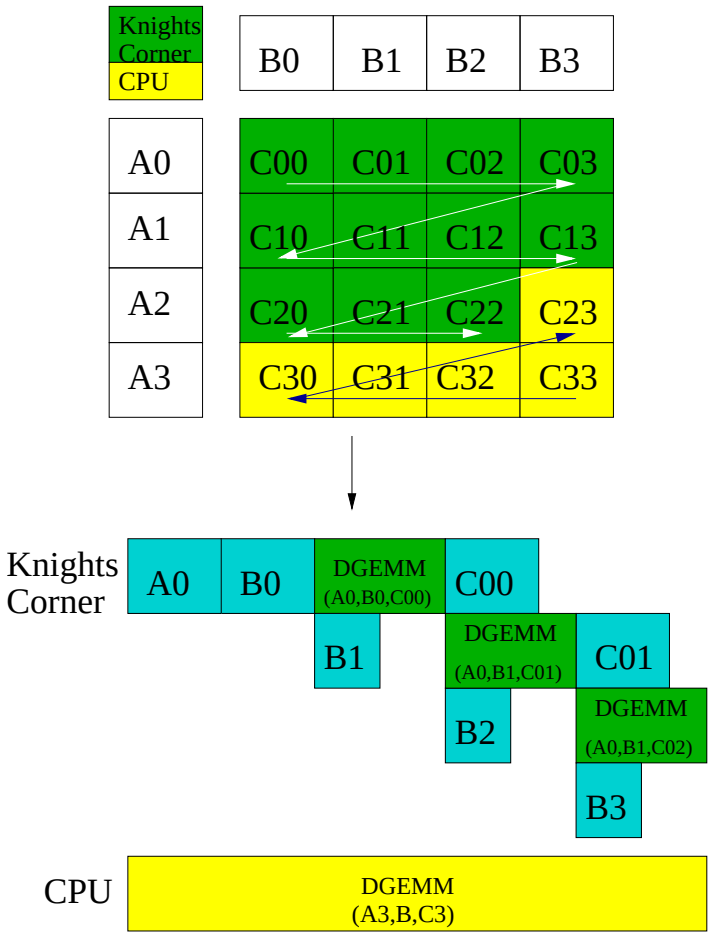


- Matrix is stored in block-cyclic layout
- Matrix is decomposed into *P*-process-rows and *Q*-process-columns
- LU is most critical component
- DGEMM fraction grows with increasing system size

$\Rightarrow$ solve largest possible problem for best performance
$\Rightarrow$ keep MIC busy all the time (doing DGEMM)

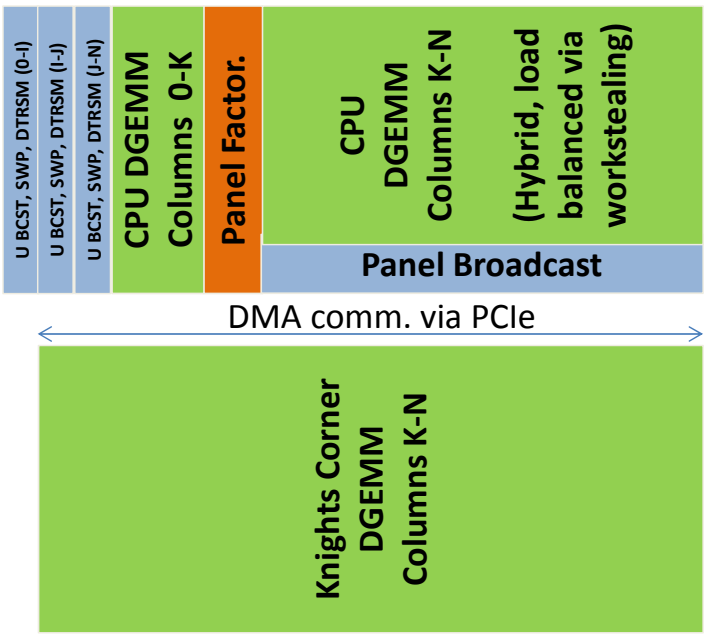
*: Heinecke, et.al.: Design and Implementation of the Linpack Benchmark for Single and Multi-Node Systems Based on Intel(R) Xeon Phi(TM) Coprocessor, IPDPS'13

Designing Offload-DGEMM with work-stealing



- $T_c = \frac{2MNK}{\text{Peak}}$
- $T_t = \frac{8MN}{\text{PCIe}} < T_c$
- $K > 4\frac{\text{Peak}}{\text{PCIe}} \approx 1000$
- C,A,B are split into tiles
- MIC starts to process tiles from $C(0,0)$
- CPU starts to process tiles from $C(X,X)$
- dynamic load balancing assigns work to MIC or CPU

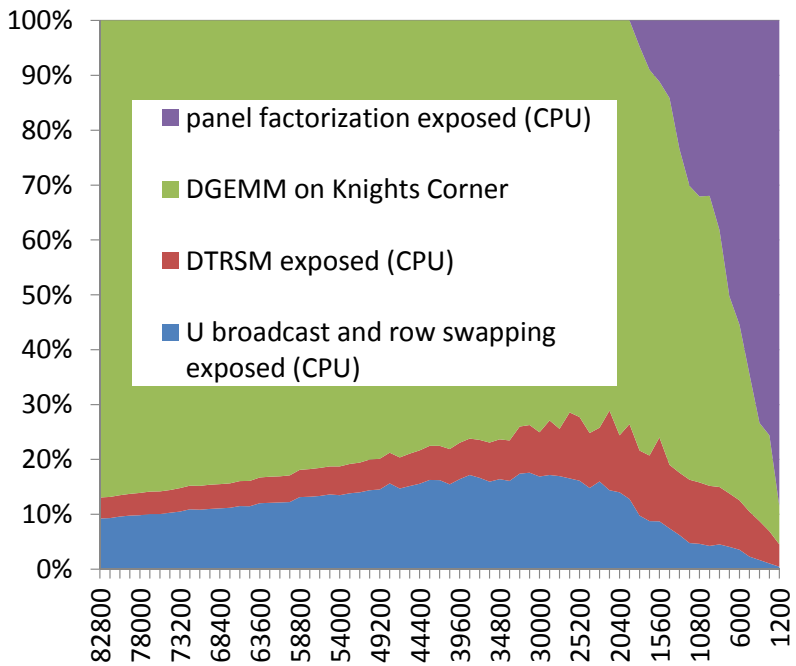
Hybrid HPL using Pipelined Lookahead



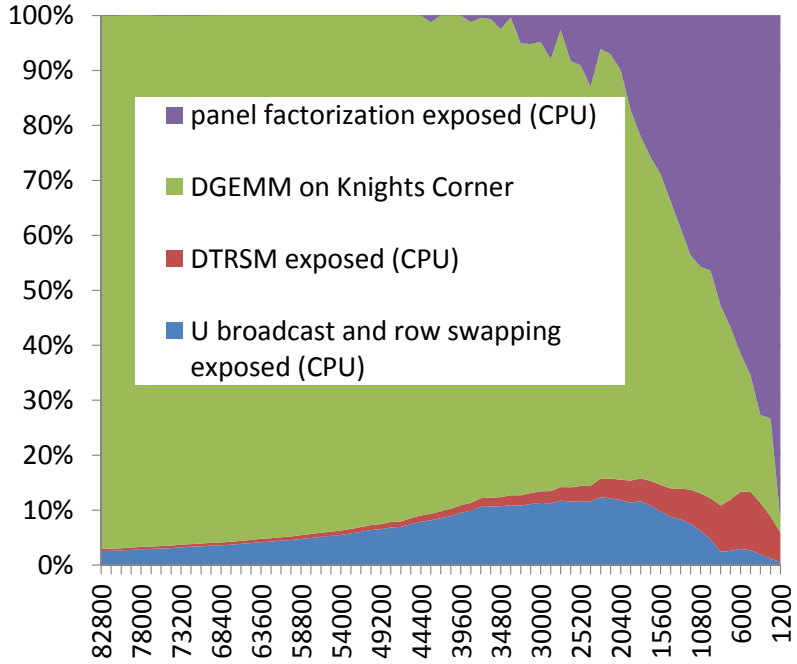
- A panel was already swapped in last iteration
- swap, U broadcast and DTRSM on one B-tile
- panel free-up on CPU is merged with swap, U broadcast and DTRSM
- afterwards start pipelined execution of swap, U broadcast, DTRSM and trailing update DGEMM

$\Rightarrow$ Xeon Phi is executing DGEMM basically all the time!

Lookahead Comparison



Standard Lookahead



Pipelined Lookahead

Comparison of Lookahead variants on a four node cluster with 2 Xeon Phi per node and 64 GB RAM per node, local matrix size is 82.8K, $P = Q = 2$.

Performance Results

System	N	P	Q	TFLOPS	Eff.
CPU only, 64GB	84K	1	1	0.29	86.4
CPU only, 64GB	168K	2	2	1.10	82.8
no pipeline, 1 card, 64GB	84K	1	1	0.99	71.0
pipeline, 1 card, 64GB	84K	1	1	1.12	79.8
no pipeline, 1 card, 64GB	168K	2	2	3.88	69.1
pipeline, 1 card, 64GB	168K	2	2	4.36	77.6
no pipeline, 1 card, 64GB	825K	10	10	95.2	67.7
pipeline, 1 card, 64GB	825K	10	10	107.0	76.1
no pipeline, 2 cards, 64GB	84K	1	1	1.66	68.2
pipeline, 2 cards, 64GB	84K	1	1	1.87	76.6
no pipeline, 2 cards, 64GB	166K	2	2	6.36	65.0
pipeline, 2 cards, 64GB	166K	2	2	7.15	73.1
no pipeline, 2 cards, 64GB	822K	10	10	156.5	64.0
pipeline, 2 cards, 64GB	822K	10	10	175.8	71.9
pipeline, 1 card, 128GB	242K	2	2	4.42	79.6

Outline

Introduction to Intel Xeon Phi

Codes, which are not fully optimized for Xeon Phi, yet!

- Molecular Dynamics
- Earthquake Simulation - SeisSol, SuperMUC bid workload

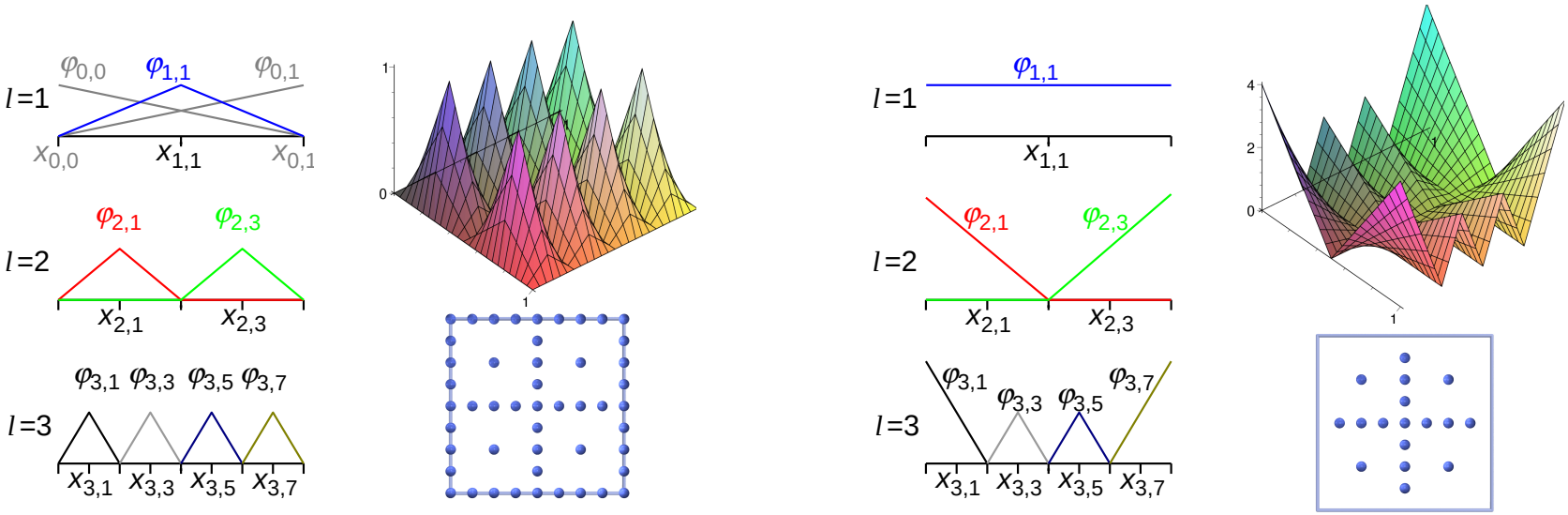
Hybrid HPL

Data Mining with Sparse Grids

Conclusions

Sparse Grids in a nutshell*

- Curse of Dimensionality: $O(N^d)$ grid points
- Therefore: Sparse Grids
  - Reduce cost to $O(N\log(N)^{d-1})$
  - Similar accuracy, if problem sufficiently smooth
- Basic idea: hierarchical basis in 1d (here: piecewise linear)



*: Heinecke, et.al.: Data Mining on Vast Dataset as a Cluster System Benchmark, PMBS'13, submitted

Data Mining with SG - In a nutshell

d-dimensional instances and ansatz-functions $f_N(\vec{x})$:

$$S = \left\{ (\vec{x}_m, y_m) \in \mathbb{R}^d \times K \right\}_{m=1,\ldots,M} \qquad f_N(\vec{x}) = \sum_{j=1}^{N} \vec{u}_j \varphi_j(\vec{x})$$

problem to be solved:

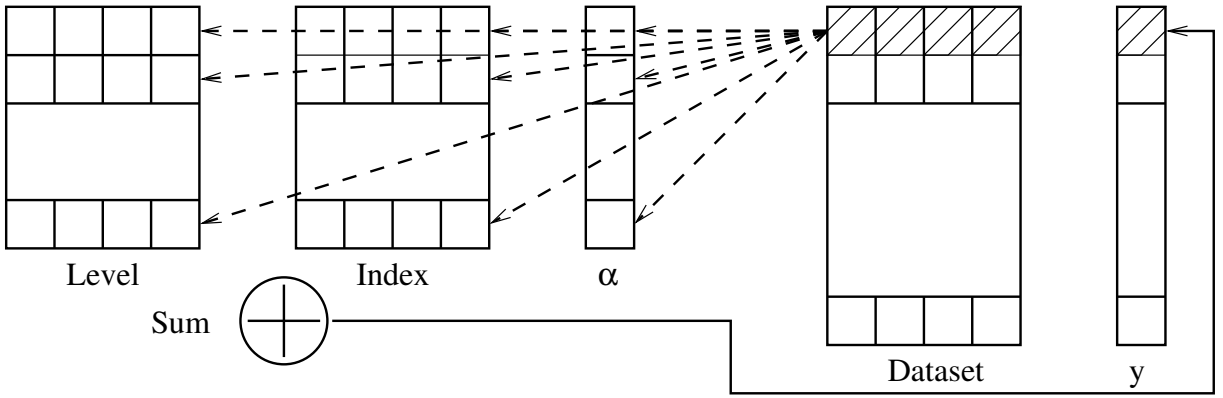
$$f_N \stackrel{!}{=} \underset{f_N \in V_N}{\arg\min} \left(\frac{1}{M} \sum_{m=1}^{M} \left(y_m - f_N(\vec{x}_m) \right)^2 + \lambda ||\nabla f_N||_{L^2}^2 \right)$$

results into SLE:

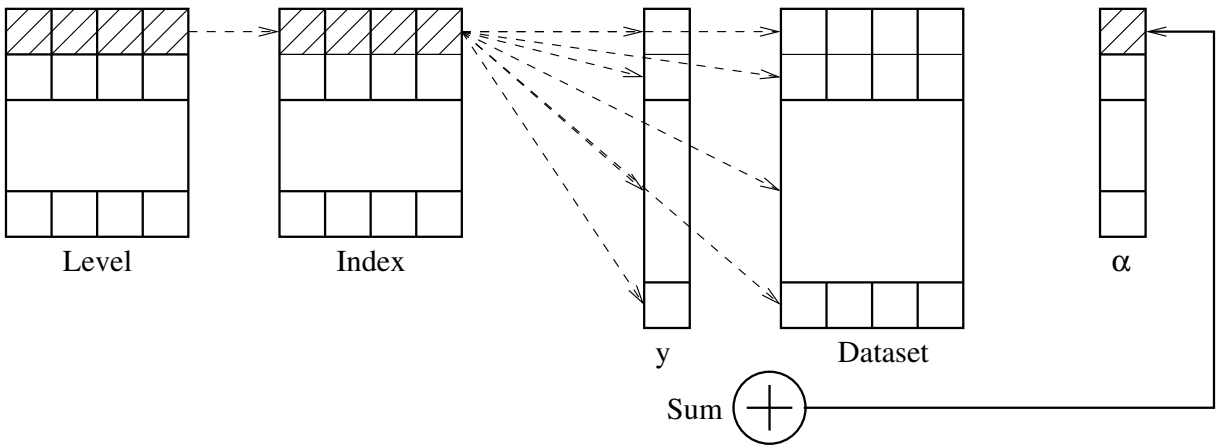
$$\left(\frac{1}{M} B B^T + \lambda C \right) \vec{u} = \frac{1}{M} B \vec{y} \qquad \text{here } C = I, b_{ij} = \varphi_i(\vec{x}_j)$$

Implementation of B^T and B

B^T: Parallelization over dataset instances:



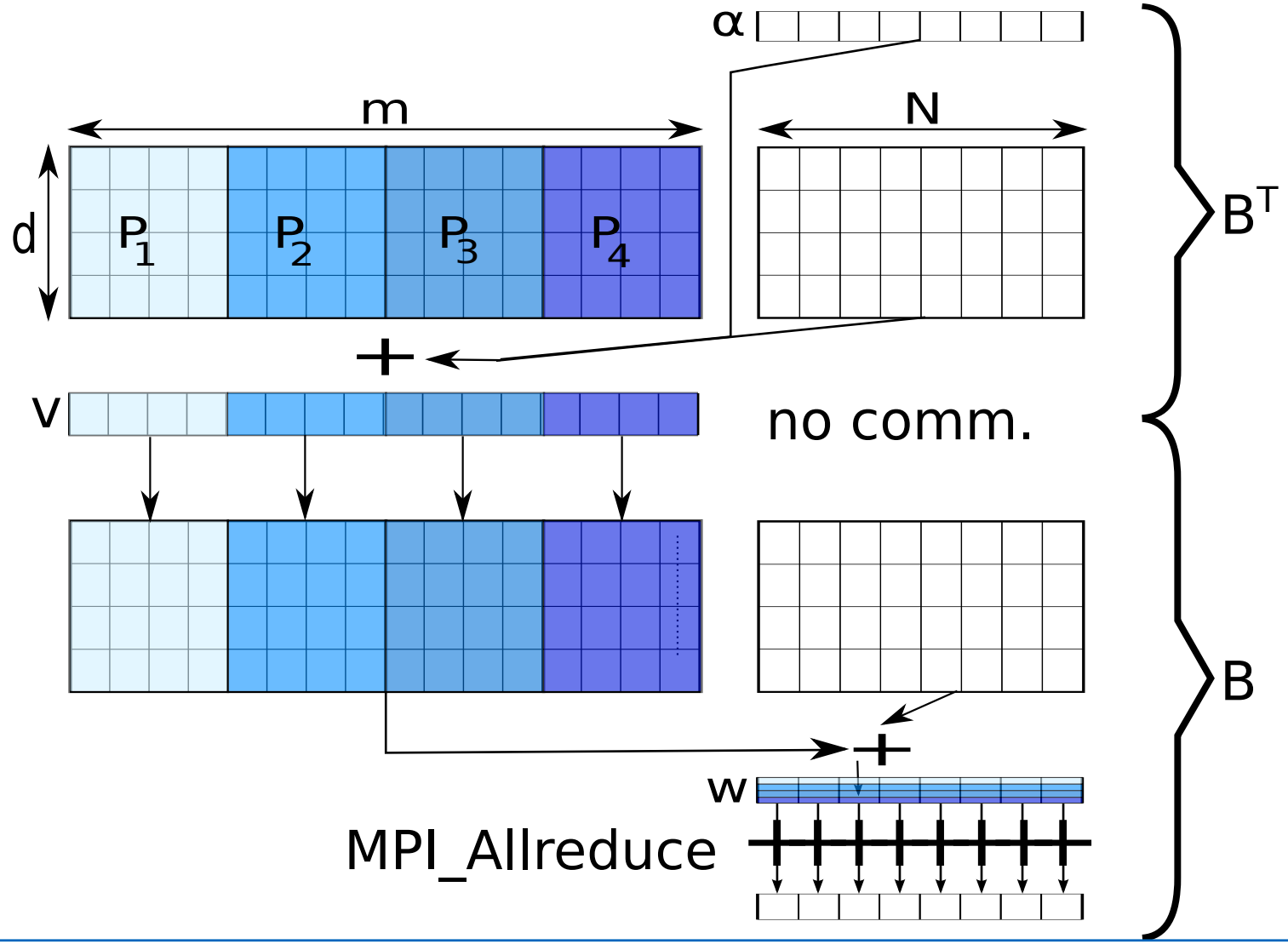
B: Parallelization over grid points:



Non-uniform Basis Functions - B^T Operator

for $\vec{x}_m \in S$ with $|S| = M$, $\vec{y}_m \leftarrow 0$ **do**
  for all $g \in$ grid with $s \leftarrow \vec{\alpha}_g$ **do**
    for $k = 1$ to d **do**
      if $g_{l_k} = 1$ **then**
        $s \leftarrow s \cdot 1$
      else if $g_{i_k} = 1$ **then**
        $s \leftarrow s \cdot \max(2 - g_{l_k} \cdot \vec{x}_{m_k}; 0)$
      else if $g_{i_k} = 2^{g_{l_k}} - 1$ **then**
        $s \leftarrow s \cdot \max(g_{l_k} \cdot \vec{x}_{m_k} - g_{i_k} + 1; 0)$
      else
        $s \leftarrow s \cdot \max(1 - |g_{l_k} \cdot \vec{x}_{m_k} - g_{i_k}|; 0)$
      end if
    end for
    $\vec{y}_m \leftarrow \vec{y}_m + s$
  end for
end for

Extending B^T and B to Clusters



Performance Model for `MPI_Allreduce`

Total learning time on p nodes:

$$T_{\text{learn}}(p,N) = T_{\text{seq}}(N) + T_B(p,N) + T_{B^T}(p,N) + T_{\text{comm}}(p,N),$$

with

$$T_{\text{seq}} = T_{\text{total}_{\text{singlenode}}} - T_{B_{\text{singlenode}}} - T_{B^T_{\text{singlenode}}}$$

and

$$T_B(p) = \frac{T_{B_{\text{singlenode}}}}{p} \quad \text{and} \quad T_{B^T}(p) = \frac{T_{B^T_{\text{singlenode}}}}{p}.$$

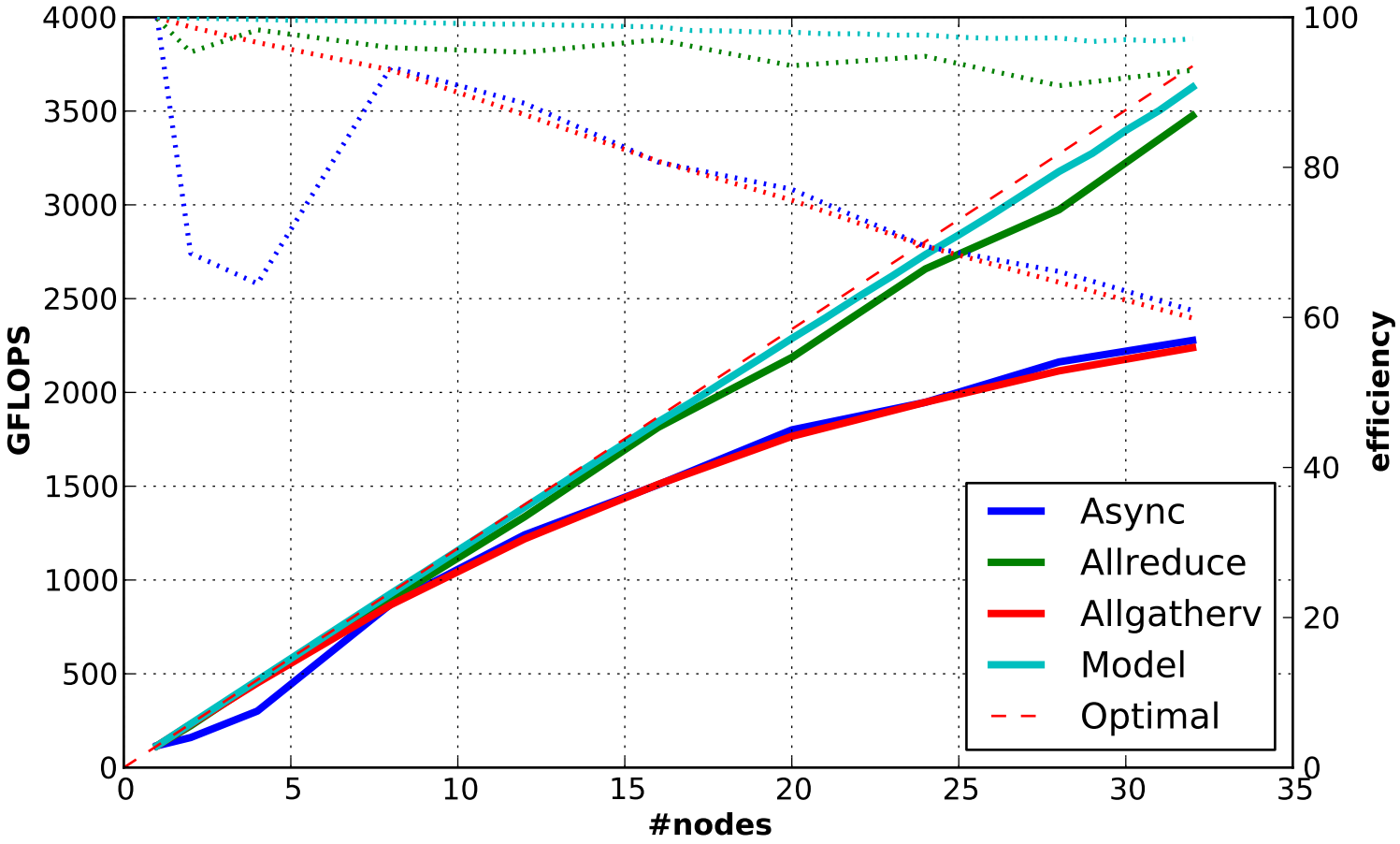
The communication time can be estimated by:

$$T_{\text{comm}}(p,N) = \sum_{r \in \text{Refinements}} T_{\text{commCG}}(p,N),$$

with employing the OSU MVAPICH benchmark suite:

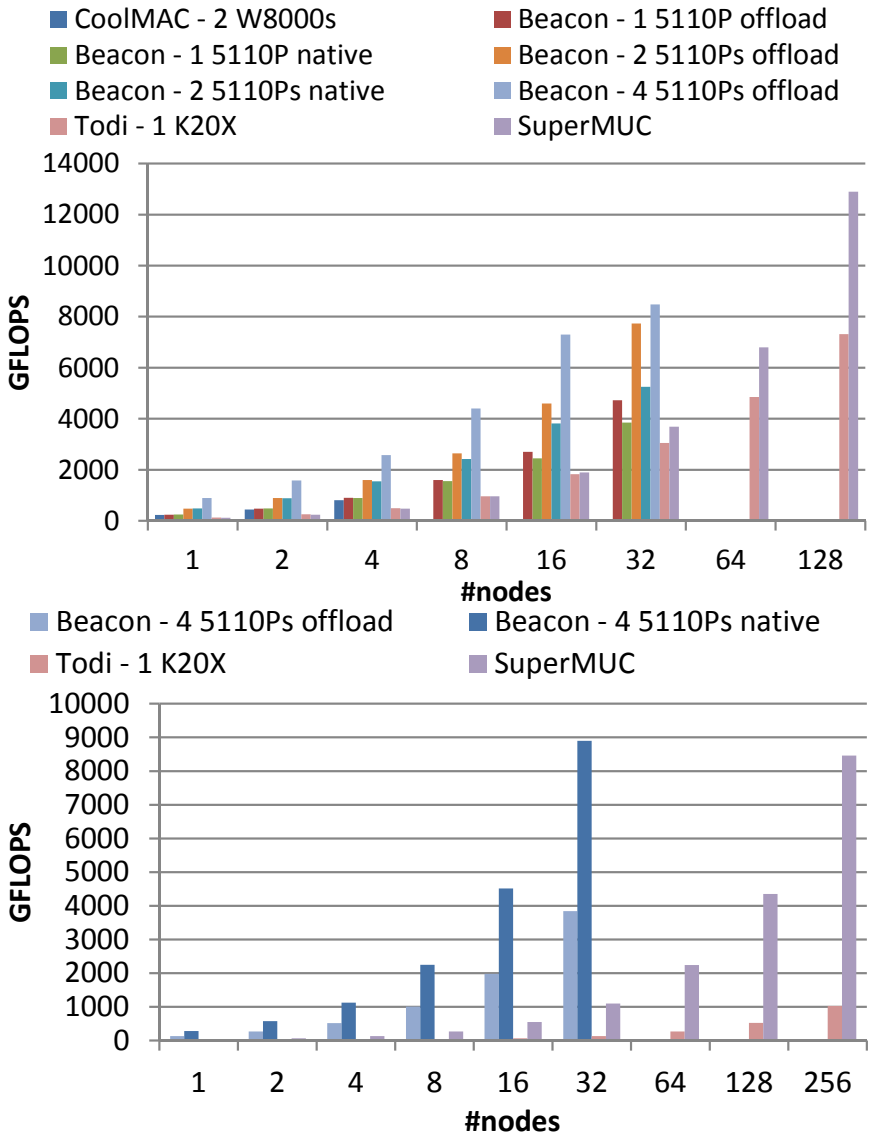
$$T_{\text{commCG}}(p,N) = \text{microbenchmark(N, p)} \cdot (\#\text{iterations} + 1).$$

Performance Model Evaluation



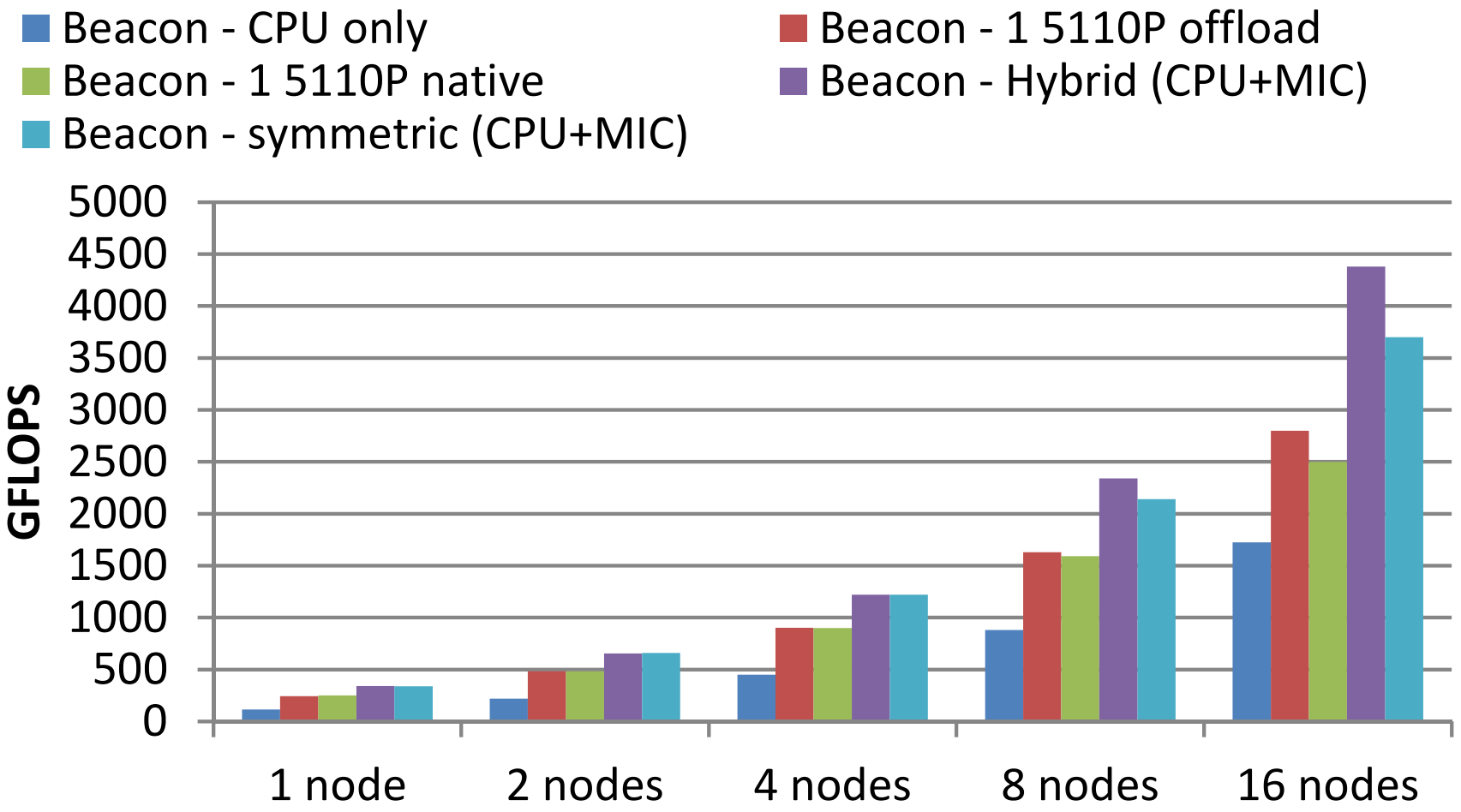
- test dataset DR5: 400K instances, 5 dimensions

Xeon Phi Offload vs. Xeon Phi Native



- DR5 dataset, 400K instances, 5 dimensions (upper)
- Friedman1 dataset, 10M instances per node, 10 dimensions (lower)
- We're running out of parallel tasks
- offloading is a limitation in case of small grids

Using MIC native implementation on Cluster



- test dataset DR5: 400K instances, 5 dimensions
- symmetric mode: 2 ranks per Xeon Phi, 1 per Xeon E5

Conclusions

Data Parallelism in x86 delivers performance comparable to GPUs, but at a higher programming productivity!

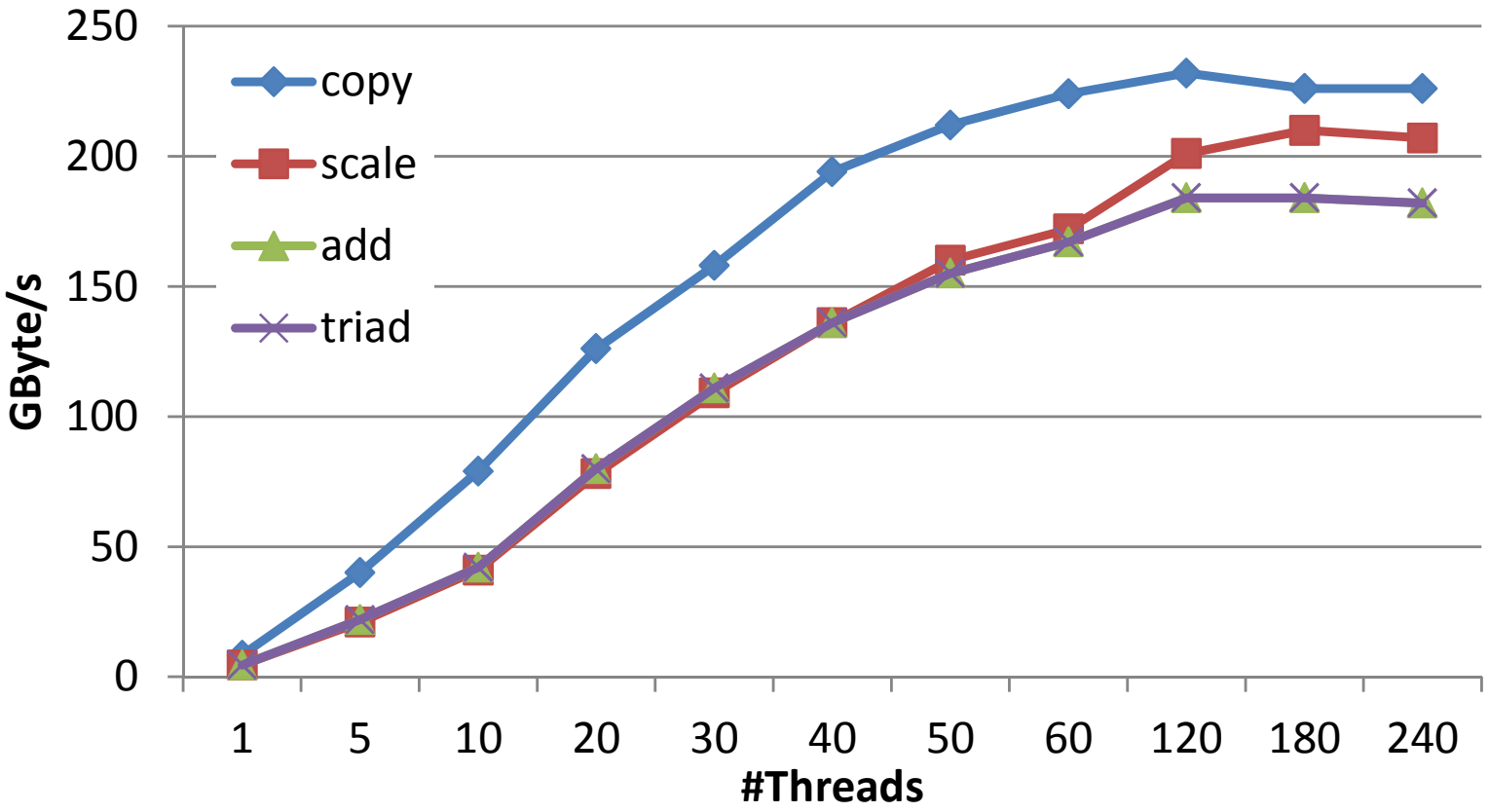
Key Findings:

- we are flexible to use different programming models
- we can just re-compile and code runs directly on Xeon Phi
- Intrinsics can be easily transferred
- Optimizations on Xeon Phi are optimization on Xeon E5, too!
- tuning for Intel MIC Architecture does not differ from tuning for a standard CPU which is also required when using GPUs (since CPU FLOPS should not be neglected)
- but: running 60+ MPI ranks on Xeon Phi seems to be suboptimal
- but: ccNUMA like L2 cache needs to be respected

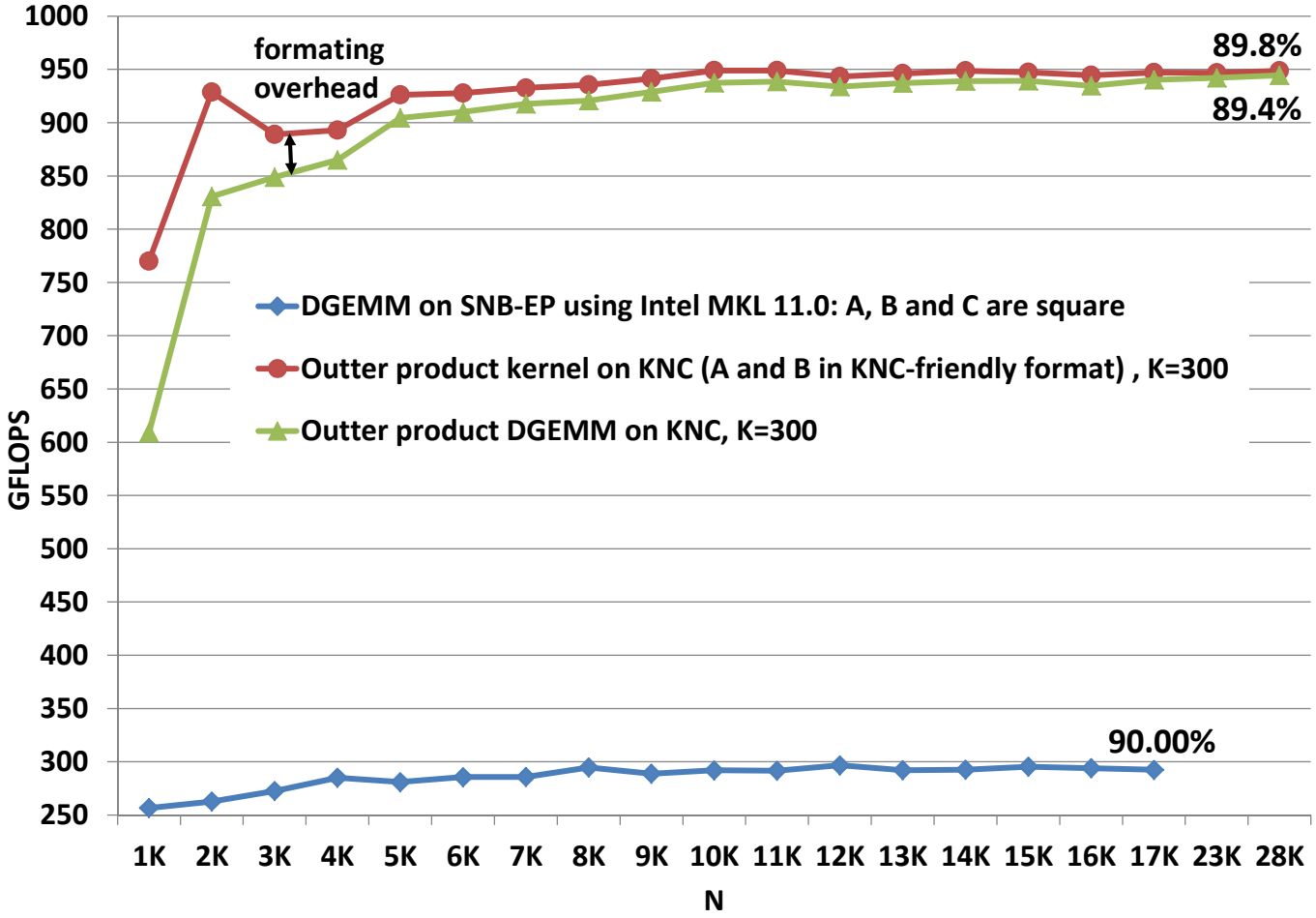
BACKUP

BACKUP

STREAM on Xeon Phi

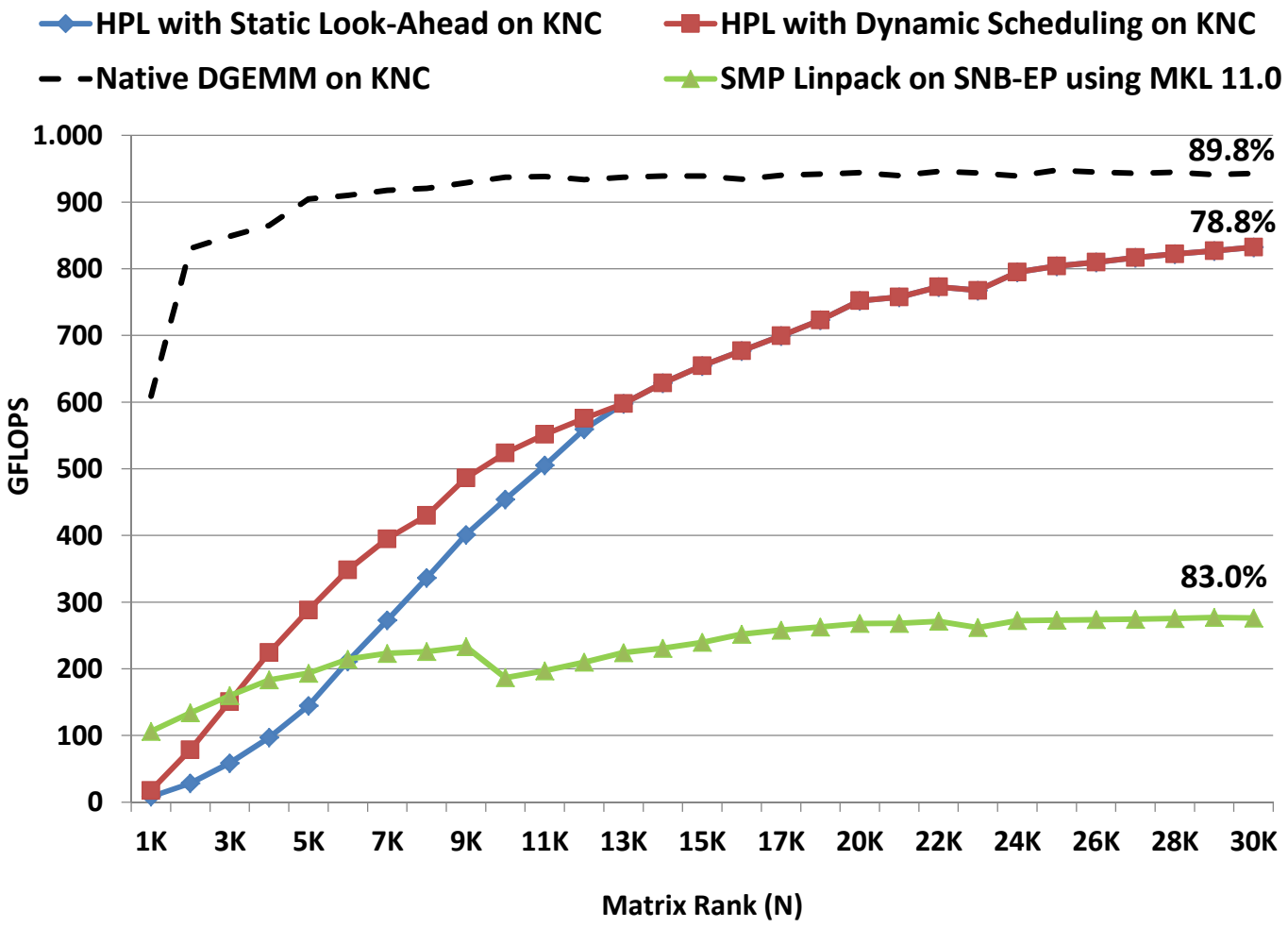


DGEMM on Xeon Phi



Heinecke, et.al.: Design and Implementation of the Linpack Benchmark for Single and Multi-Node Systems Based on Intel(R) Xeon Phi(TM) Coprocessor, at IPDPS'13

DGETRF on Xeon Phi



Heinecke, et.al.: IPDPS'13

The Intel Xeon Phi Usage Model - Offloading

low-level SCI Low-level MPI-like interface (DMA-windows, mem-fences)

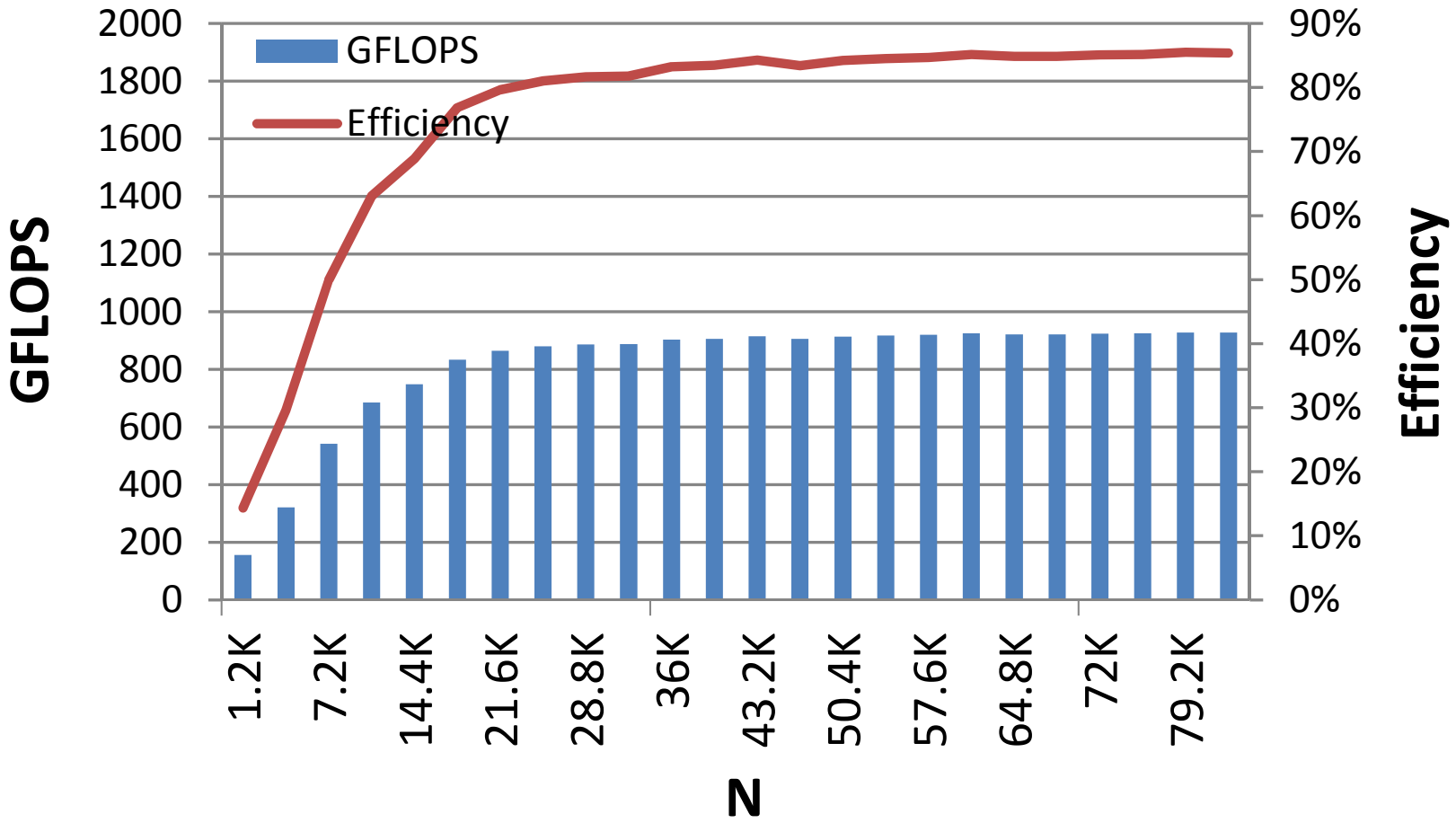
med-level COI OpenCL-like, handles, buffers, etc.

high-level #pragma annotations for driving offload to MIC

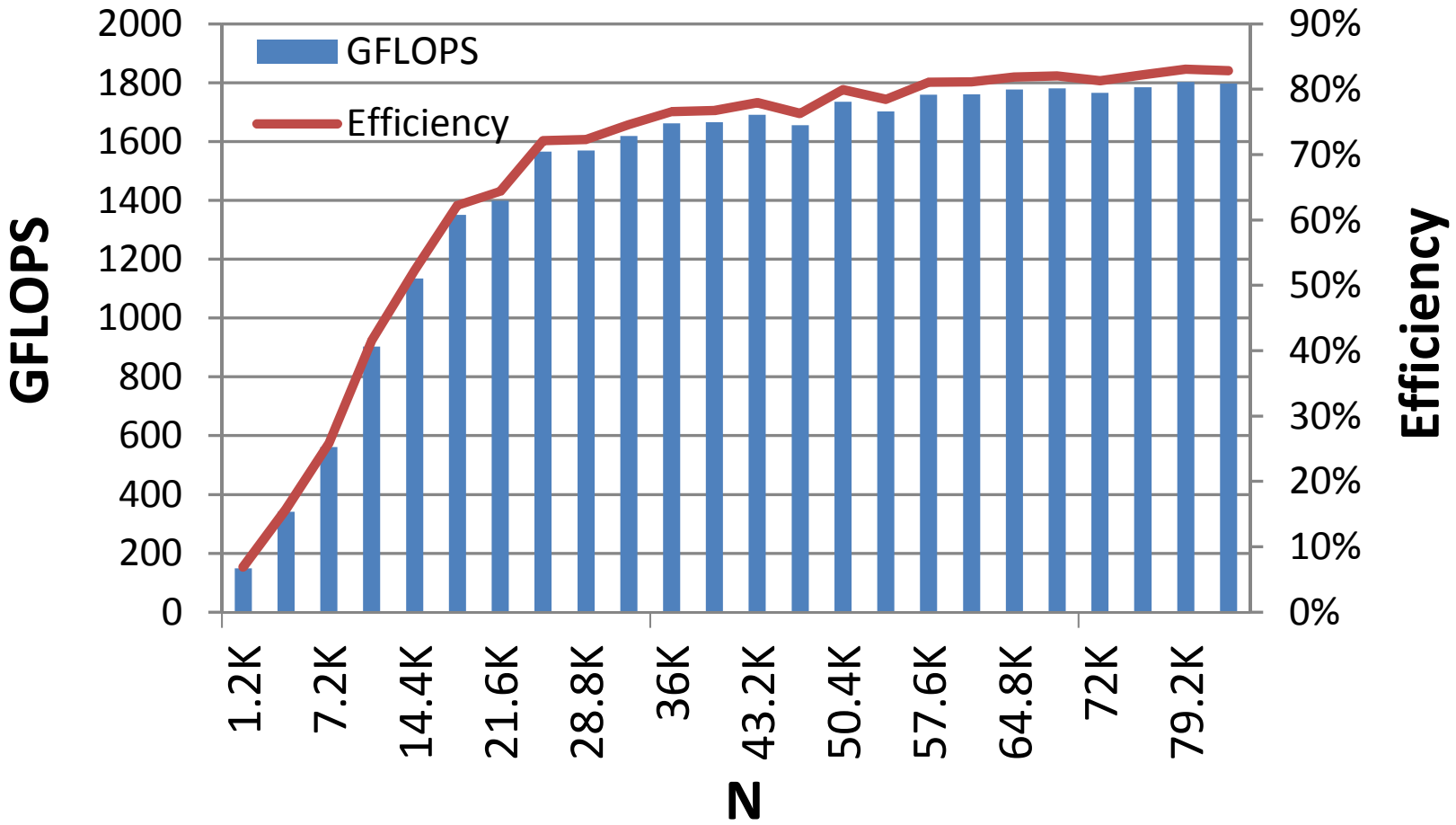
Example:

```
#pragma offload target(mic:d) nocopy(ptrData)
in(ptrCoef:length(size) alloc_if(0) free_if(0) align(64))
out(ptrCoef[start:mychunk] :  into
(ptrCoef[start:mychunk]) alloc_if(0) free_if(0) align(64))
in(size)
{
..
}
```

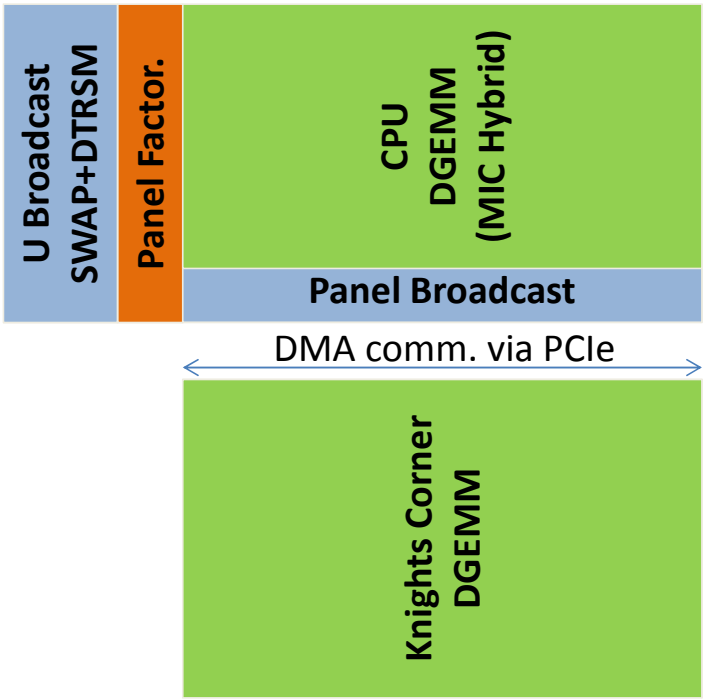
Offload-DGEMM with one Xeon Phi



Offload-DGEMM with two Xeon Phis



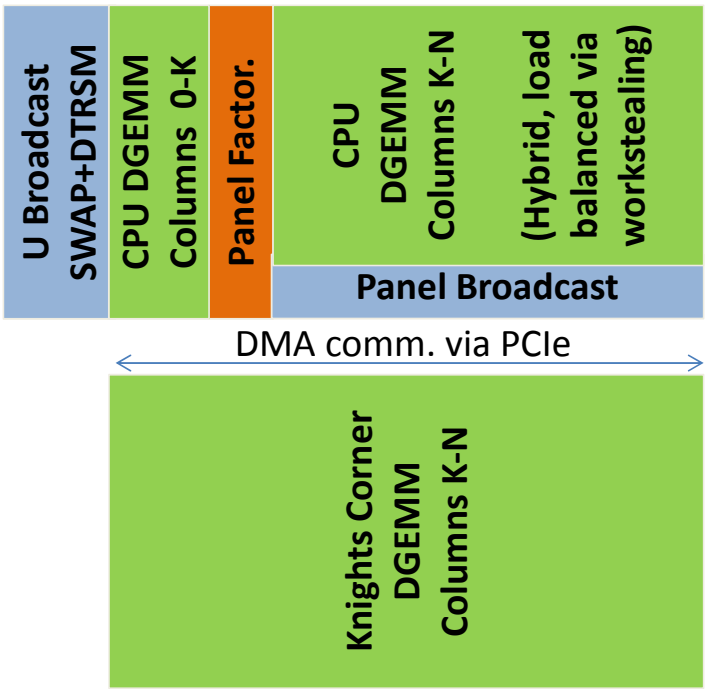
Naive Hybrid HPL



- very simple
- Least change to HPL, plug and play
- no-lookahead is even inefficient on CPU
- Performance is limited by swapping and panel fact. running on CPU
- As MIC is 3-4$\times$ faster than CPU, MIC idle time is more expensive

$\Rightarrow$ Keeping Xeon Phi as busy as possible is major goal!

Hybrid HPL using Standard Lookahead



- little changes to HPL
- DGEMM is split into tow parts: panel free-up and trailing update
- trailing update DGEMM is load-balanced via work-stealing between MIC and CPU
- panel fact. is entirely hidden

⇒ U broadcast, swapping of trailing matrix and DTRSM are still exposed!

Outline

Introduction to Intel Xeon Phi

Codes, which are not fully optimized for Xeon Phi, yet!

- Molecular Dynamics
- Earthquake Simulation - SeisSol, SuperMUC bid workload

Hybrid HPL

Data Mining with Sparse Grids

Conclusions

THE UNIVERSITY of TENNESSEE
NATIONAL LEADERSHIP IN HPC

SC12
Salt Lake City, Utah

Beacon:
A Path to Energy-Efficient HPC

R. Glenn Brook

Director, Application Acceleration Center of Excellence
National Institute for Computational Sciences
glenn-brook@tennessee.edu

Green500 Hardware

- Appro Xtreme-X Supercomputer powered by ACE

Beacon Green500 Cluster	
Nodes	36 compute
CPU model	Intel® Xeon® E5-2670
CPUs per node	2x 8-core, 2.6GHz
RAM per node	256 GB
SSD per node	960 GB (RAID 0)
Intel® Xeon Phi™ Coprocessors 5110P per node	4
Cores per Intel® Xeon Phi™ coprocessor 5110P	60 @ 1.053GHz
RAM per Intel® Xeon Phi™ coprocessor 5110P	8 GB GDDR5

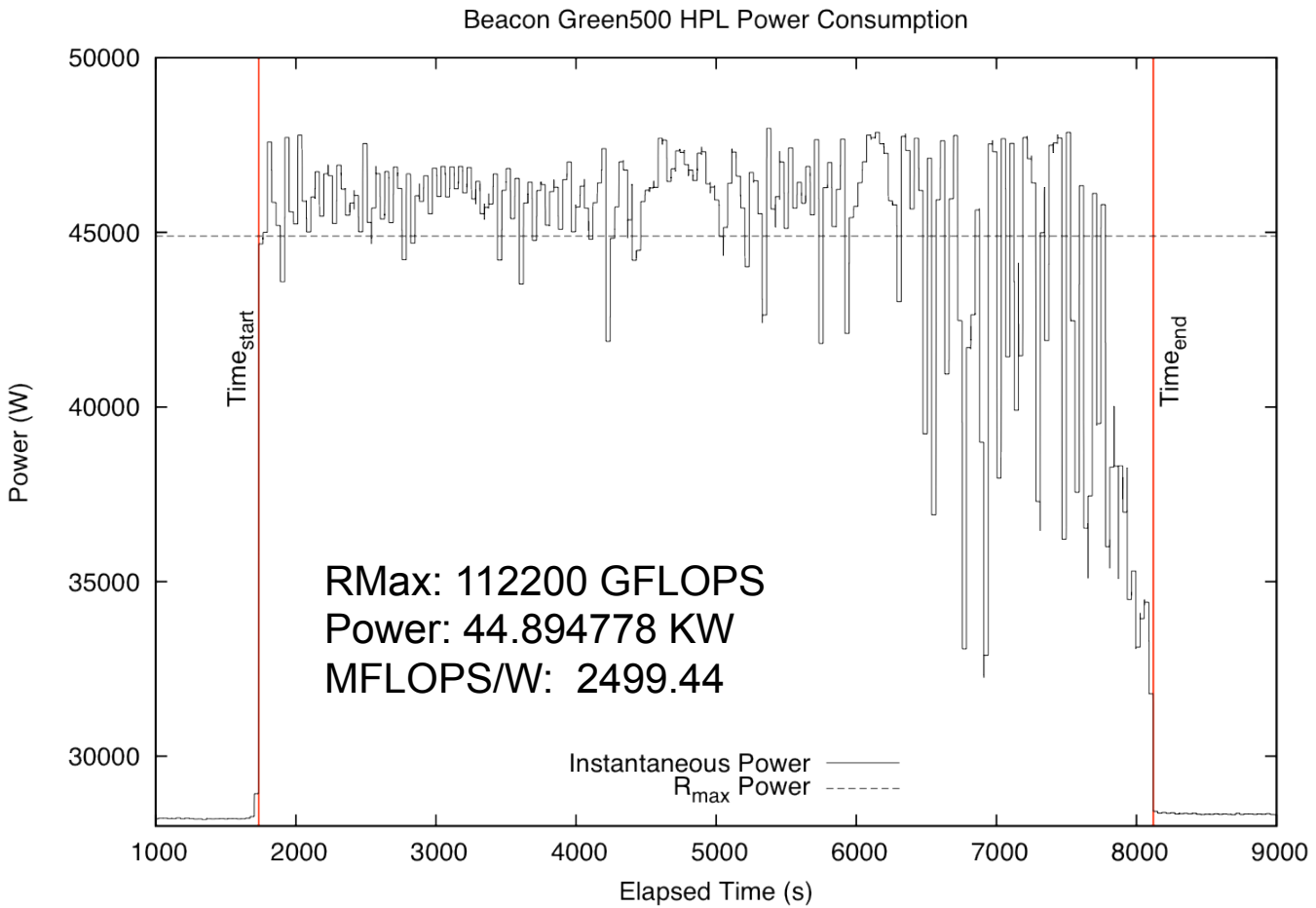
THE UNIVERSITY of TENNESSEE NATIONAL LEADERSHIP IN HPC

Assembled Team

- Intel:
  - Mikhail Smelyanskiy – software lead
  - Karthikeyan Vaidyanathan – MIC HPL
  - Ian Steiner – host HPL optimization
  - Many, many others: Joe Curley, Jim Jeffers, Pradeep Dubey, Susan Meredith, Rajesh Agny, Russ Fromkin, and many dedicated and passionate team members
- Technische Universität München:
  - Alexander Heinecke – MIC HPL
- Appro:
  - John Lee – hardware lead
  - David Parks – HPL and system software
  - Edgardo Evangelista, Danny Tran, others – system deployment and support
- NICS:
  - Glenn Brook – project lead
  - Ryan Braby, Troy Baer – system support

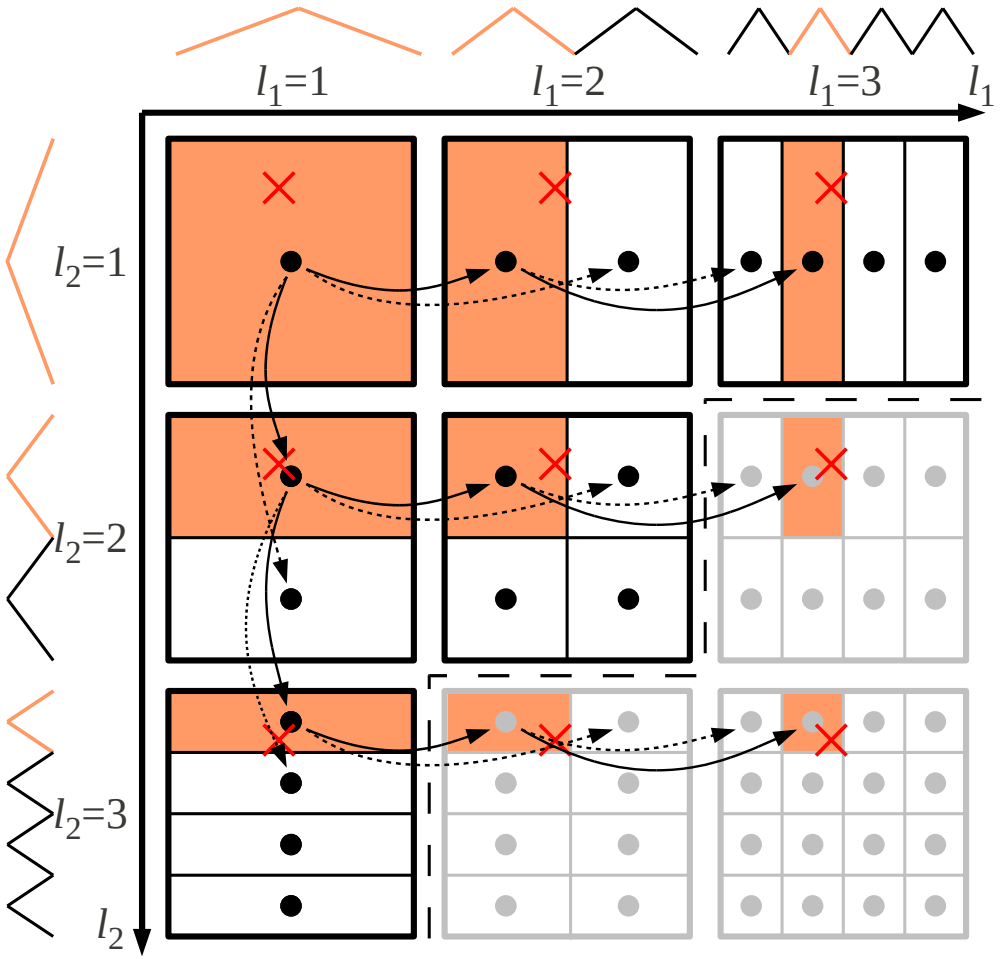
THE UNIVERSITY of TENNESSEE NATIONAL LEADERSHIP IN HPC

Power Consumption



THE UNIVERSITY of TENNESSEE NATIONAL LEADERSHIP IN HPC

classical, recursive function evaluation

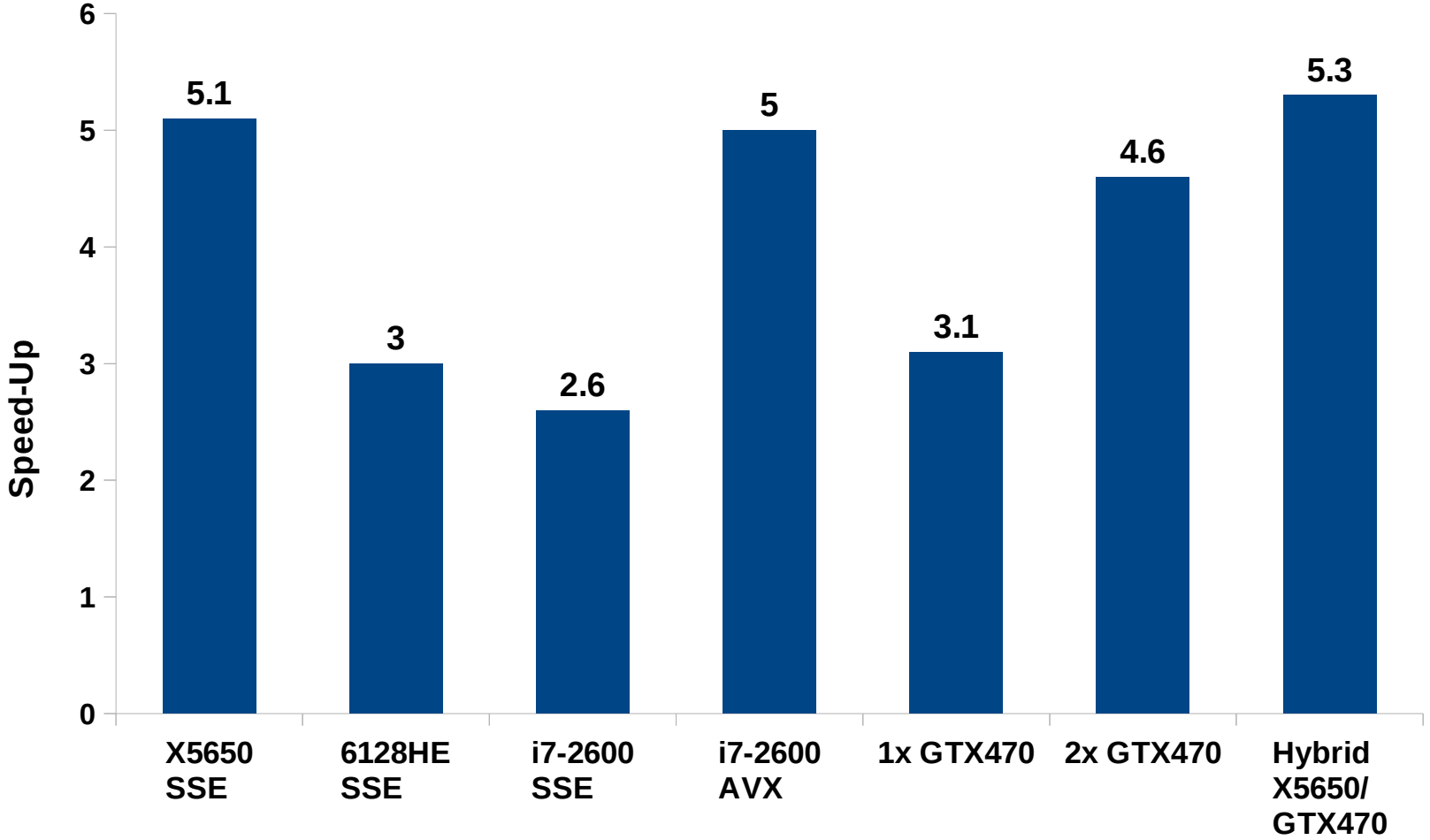


Algorithm: Learning data sets with adaptive sparse grids

1: $D := \textit{data set}$
2: $G := \textit{start grid}$
3: $\vec{u} := \vec{0}$
4: **while** TRUE **do**
5: $\quad \textit{trainGrid}(G, \vec{u}, D)$
6: $\quad \textit{acc} = \textit{getAccuracy}(G, \vec{u}, D)$
7: $\quad$ **if** $\textit{acc} \geq \textit{acc}_{\textit{needed}}$ **then**
8: $\quad\quad$ **return** $\vec{u}, G$
9: $\quad$ **end if**
10: $\quad \textit{refineGrid}(G, \vec{u})$
11: **end while**

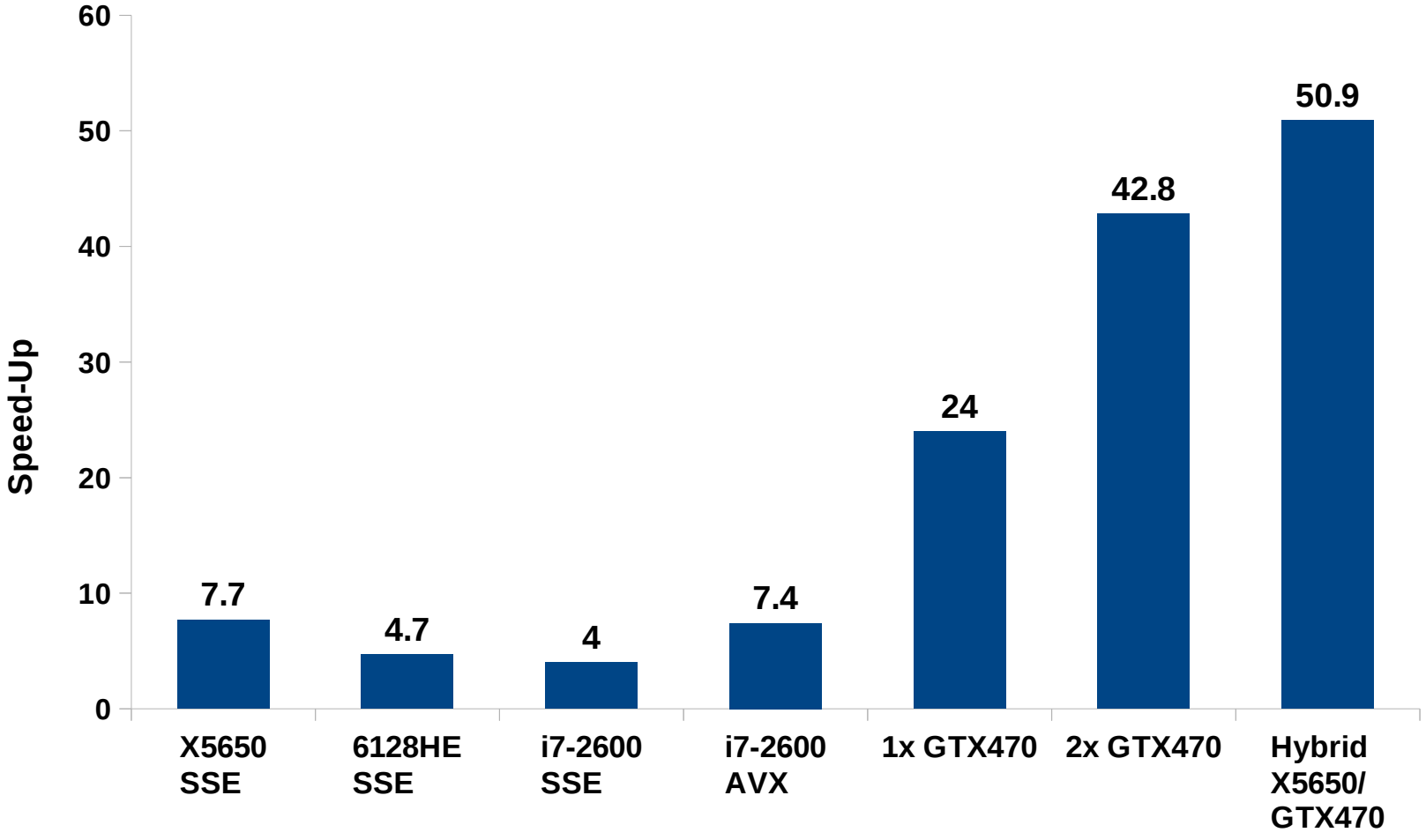
First results (modlinear) - Baseline: classic algorithms

DR5 data set (370K instances):

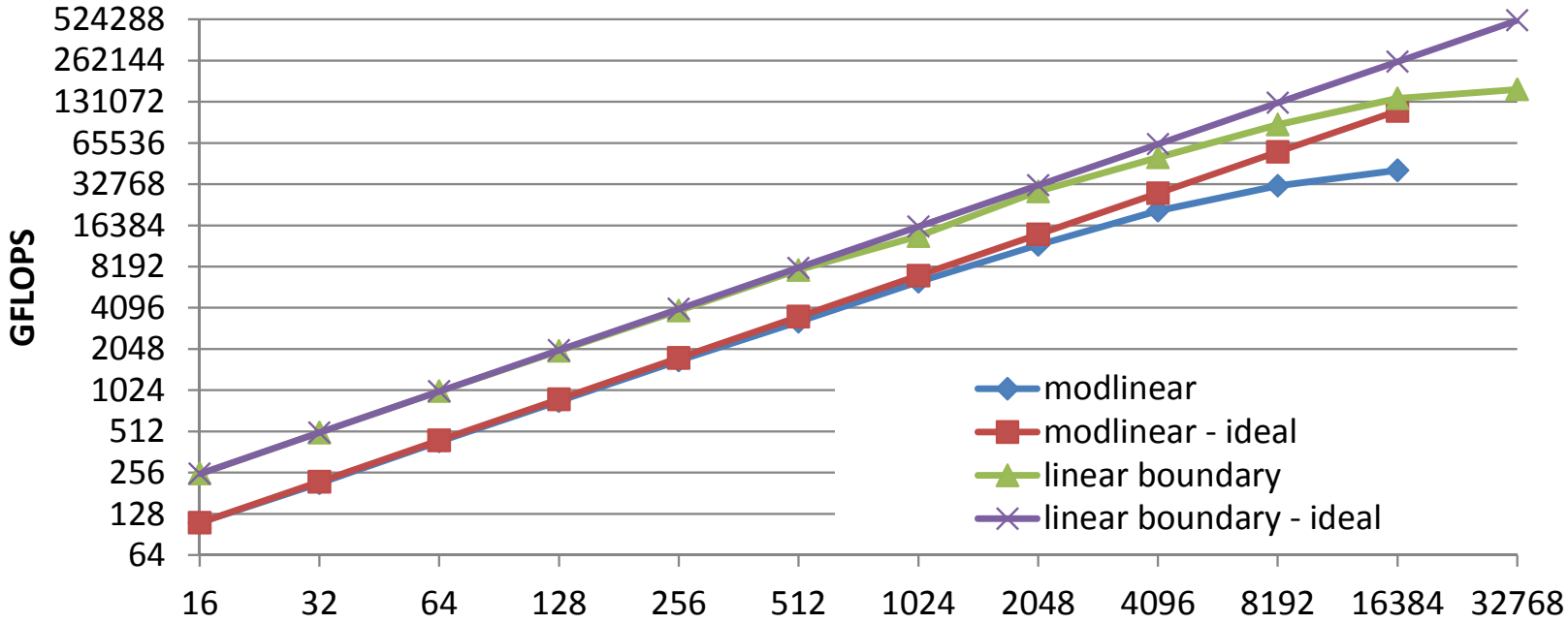


First results (regular) - Baseline: classic algorithms

DR5 data set (370K instances):



Scaling on SuperMUC



modlinear: spatially adaptive grids (5K - 21K gridpoints in 7 refinements), modlinear basis, MSE: 4.8*10-4, runtime 16384: 4.4s

linear boundary: regular grid (100K gridpoints), regular linear basis, MSE: 5.4*10-4, runtime 32768: 3.6s

Data Mining with Sparse Grids @SC12