Optimization of Geometric Multigrid for Emerging Multi- and Manycore Processors

Samuel Williams[1], Dhiraj Kalamkar[2], Amik Singh[3],
Anand Deshpande[2], Brian Van Straalen[1], Mikhail Smelyanskiy[2],
Ann Almgren[1], Pradeep Dubey[2], John Shalf[1], Leonid Oliker[1]

[1]Lawrence Berkeley National Laboratory
[2]Intel Corporation
[3]University of California at Berkeley

SWWilliams@lbl.gov

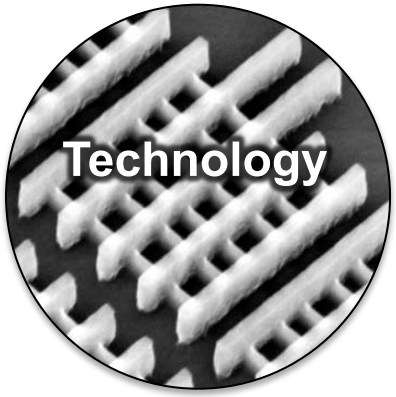
Notice and Disclaimers

- Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information. Contact your local Intel sales office or your distributor to obtain the latest specification before placing your product order.
- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications. Intel may make changes to specifications, product descriptions, and plans at any time, without notice.
- All products, dates, and figures are preliminary for planning purposes and are subject to change without notice.
- Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined.“ Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
- Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.
- The Intel products discussed herein may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's website at http://www.intel.com.
- Intel® Itanium®, Intel® Xeon®, Xeon Phi™, Pentium®, Intel SpeedStep® and Intel NetBurst® , Intel®, and VTune are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- *Other names and brands may be claimed as the property of others..

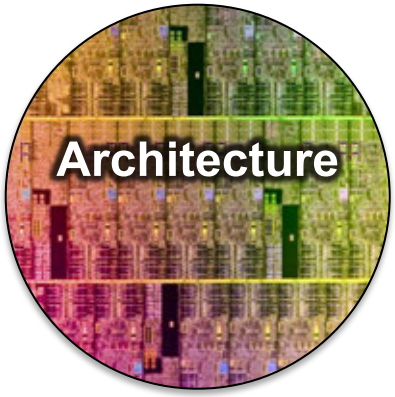
Notice and Disclaimers Continued …

- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance
- Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

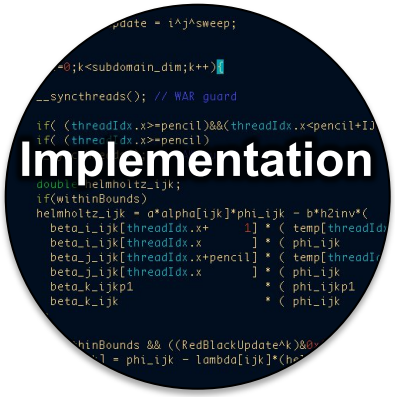
Pathways to Performance



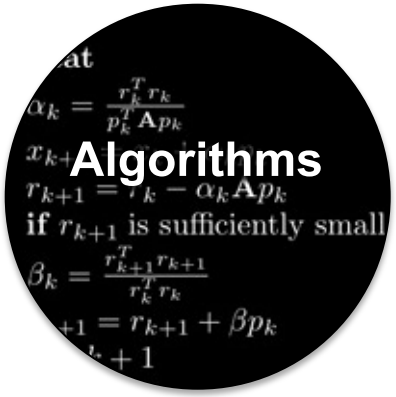
- *Transistor density, frequency, energy*
- *DRAM technology DDR3 vs. GDDR5*



- *coherent caches vs. local stores*
- *Prefetching vs. Multithreading*



- *cache blocking, loop fusion, SIMDization*
- *programming models*

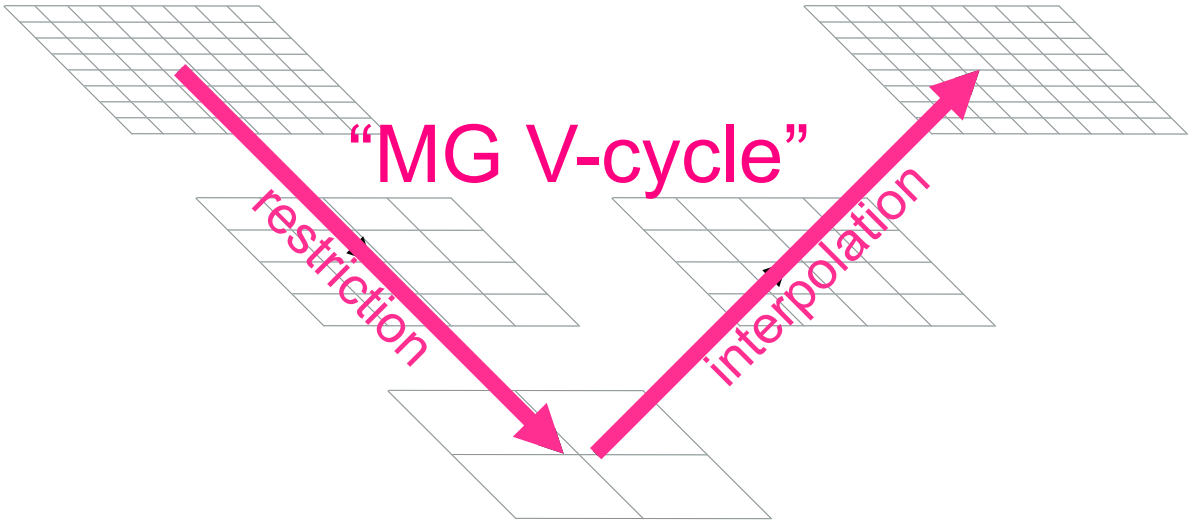


- *high-order discretizations*
- *Linear Solvers*
- *comm.-avoiding*

- In this talk, we…
  - Examine a variety of techniques from these areas to maximize performance of the **geometric multigrid v-cycle**.
  - Evaluate Intel's new **Xeon Phi™ (Knights Corner)** coprocessor
  - Differentiate the benefits of technology (GDDR) and architecture (coherent caches)
  - Quantify the benefits of optimization and communication-avoiding.

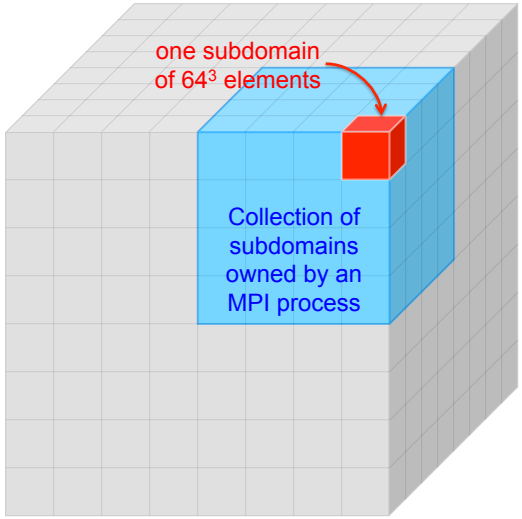
Multigrid Introduction

- Linear Solvers (Ax=b) are ubiquitous in scientific computing...
  - Combustion, Climate, Astrophysics, Cosmology, etc…
- Multigrid exploits the nature of elliptic PDEs to provide a hierarchical approach with O(N) computational complexity.
  - Geometric Multigrid is specialization in which the linear operator (A) is simply a stencil on a structured grid (i.e. *matrix-free*)
  - Applicable to small (yet very important) range of linear systems



Benchmark: Specification

- 256^3 grid/node at the finest resolution…
  - fits into KNC or GPU device DRAM =
  - **no PCIe transfers are required.**
- subdivided into 64^3 subdomains (“boxes”)
  - designed to mimic some of the challenges associated with AMR MG
  - **explicit communication of subdomain boundaries (ghost zones).**
- Truncated V-cycle (stops at 4^3 subdomains)
- Gauss Seidel, Red-Black (“GSRB”) relaxation…
  - 2 GSRB’s (4 stencil sweeps) at each level down the v-cycle
  - 24 GSRB’s at the bottom (sufficient for single-node convergence)
  - 2 GSRB’s at each level back up the v-cycle
- Fixed 10 V-cycles



Benchmark: GSRB Operator

variable coefficient

```
helmholtz = a*alpha[i,j,k]*phi[i,j,k] - b*h2inv*(
  beta_i[i+1,j,k] * ( phi[i+1,j,k] - phi[i  ,j,k] ) -
  beta_i[i  ,j,k] * ( phi[i  ,j,k] - phi[i-1,j,k] ) +
  beta_j[i,j+1,k] * ( phi[i,j+1,k] - phi[i,j  ,k] ) -
  beta_j[i,j  ,k] * ( phi[i,j  ,k] - phi[i,j-1,k] ) +
  beta_k[i,j,k+1] * ( phi[i,j,k+1] - phi[i,j,k  ] ) -
  beta_k[i,j,k  ] * ( phi[i,j,k  ] - phi[i,j,k-1] )
)

phi[i,j,k] = phi[i,j,k] -
  lambda[i,j,k] * ( helmholtz - rhs[i,j,k] )
```

Precomputed from α & β's

- **High-performance baseline implementation…**
  - construction of the Laplacian, Helmholtz, and GSRB relaxation have been fused to minimize data movement (=2x performance gain).
  - Hybrid MPI+OpenMP implementation where everything is threaded

Performance Challenges

- Code has little reuse and large working sets:
  - VC 2nd order => **DRAM bandwidth-bound** (flop:byte ~ 0.2)
  - Cache working set **~350KB per subdomain**
  - Only 1 stencil sweep per MPI communication step
  - Small subdomains = **significant % in communication (poor surface:volume)**
- Quadruply nested parallelism:
  - no individual loop longer than 64 iterations **(bad for OpenMP)**
  - worse, 3 loops see exponentially decreasing parallelism

Evaluation Platforms

Evaluation Platforms

- 2P Cray XE6 node (“Opteron”)
- 2P Intel® Xeon® E5-2670 (“SNBe”)
- Intel® Xeon Phi™ coprocessor (“KNC”)

	Opteron	SNBe	KNC[1]
cache per core	64+512KB	32+256KB	32+512KB
cores (threads) per chip	6	8	60 (240)
SIMD-width (DP)	2	4	8
L3 per chip	6MB	20MB	-
chips per node	4	2	1
DP GFlop/s per node	201.6	332.8	1248
STREAM GB/s per node	49.4	70	150
Programming Model	MPI+OMP	MPI+OMP	OpenMP

[1]Evaluation card is not necessarily reflective of production card specifications

Evaluation Platforms

- 2P Cray XE6 node ("Opteron")
- 2P Intel® Xeon® E5-2670 ("SNBe")
- Intel® Xeon Phi™ coprocessor ("KNC")

	Opteron	SNBe	KNC[1]
cache per core	64+512KB	32+256KB	32+512KB
cores (threads) per chip	8	8	60 (240)
SIMD width (DP)	2	4	8
L3 per chip	6MB	20MB	-
chips per node	4	2	1
DP GFlop/s per node	201.6	332.8	1248
STREAM GB/s per node	49.4	70	150
Programming Model	MPI+OMP	MPI+OMP	OpenMP

30MB of coherent L2 Cache
= easy to exploit locality
= easy to share data on chip

[1]Evaluation card is not necessarily reflective of production card specifications

Evaluation Platforms

- 2P Cray XE6 node ("Opteron")
- 2P Intel® Xeon® E5-2670 ("SNBe")
- Intel® Xeon Phi™ coprocessor ("KNC")

	Opteron	SNBe	KNC[1]
cache per core	64+512KB	32+256KB	32+512KB
cores (threads) per chip	6	8	60 (240)
SIMD-width (DP)	2	4	8
L3 per chip	6MB	20MB	-
chips per node	4	2	1
DP GFlop/s per node	201.6	332.8	1248
STREAM GB/s per node	49.4	70	150
Programming Model	MPI+OMP	MPI+OMP	OpenMP

KNC has 3x the XE6 bandwidth

[1]Evaluation card is not necessarily reflective of production card specifications

Evaluation Platforms

- 2P Cray XE6 node ("Opteron")
- 2P Intel® Xeon® E5-2670 ("SNBe")
- Intel® Xeon Phi™ coprocessor ("KNC")

	Opteron	SNBe	KNC[1]
cache per core	64+512KB	32+256KB	32+512KB
cores (threads) per chip	6	8	60 (240)
SIMD-width (DP)	2	4	8
L3 per chip	6MB	20MB	-
chips per node	4	2	1
DP GFlop/s per node	201.6	332.8	1018
STREAM GB/s per node	49.4	70	150
Programming Model	MPI+OMP	MPI+OMP	OpenMP

NUMA
(sidestep with MPI)

[1]Evaluation card is not necessarily reflective of production card specifications

Evaluation Platforms

- 2P Cray XE6 node ("Opteron")
- 2P Intel® Xeon® E5-2670 ("SNBe")
- Intel® Xeon Phi™ coprocessor ("KNC")

	Opteron	SNBe	KNC[1]
cache per core	64+512KB	32+256KB	32+512KB
cores (threads) per chip	6	8	60 (240)
SIMD-width (DP)	2	4	8
DP GFlop/s per node	201.6	332.8	1248
STREAM GB/s per node	49.4	70	150
Programming Model	MPI+OMP	MPI+OMP	OpenMP

Good news = you can run OpenMP

Challenge = ubiquitous 240-way TLP

[1]Evaluation card is not necessarily reflective of production card specifications

Evaluation Platforms

- 2P Cray XE6 node ("Opteron")
- 2P Intel® Xeon® E5-2670 ("SNBe")
- Intel® Xeon Phi™ coprocessor ("KNC")

	Opteron	SNBe	KNC[1]
cache per core	64+512KB	32+256KB	32+512KB
cores (threads) per chip	6	8	60 (240)
SIMD-width (DP)	2	4	8
			-
			1
DP GFlop/s per node	201.6	332.8	1248
STREAM GB/s per node	49.4	70	150
Programming Model	MPI+OMP	MPI+OMP	OpenMP

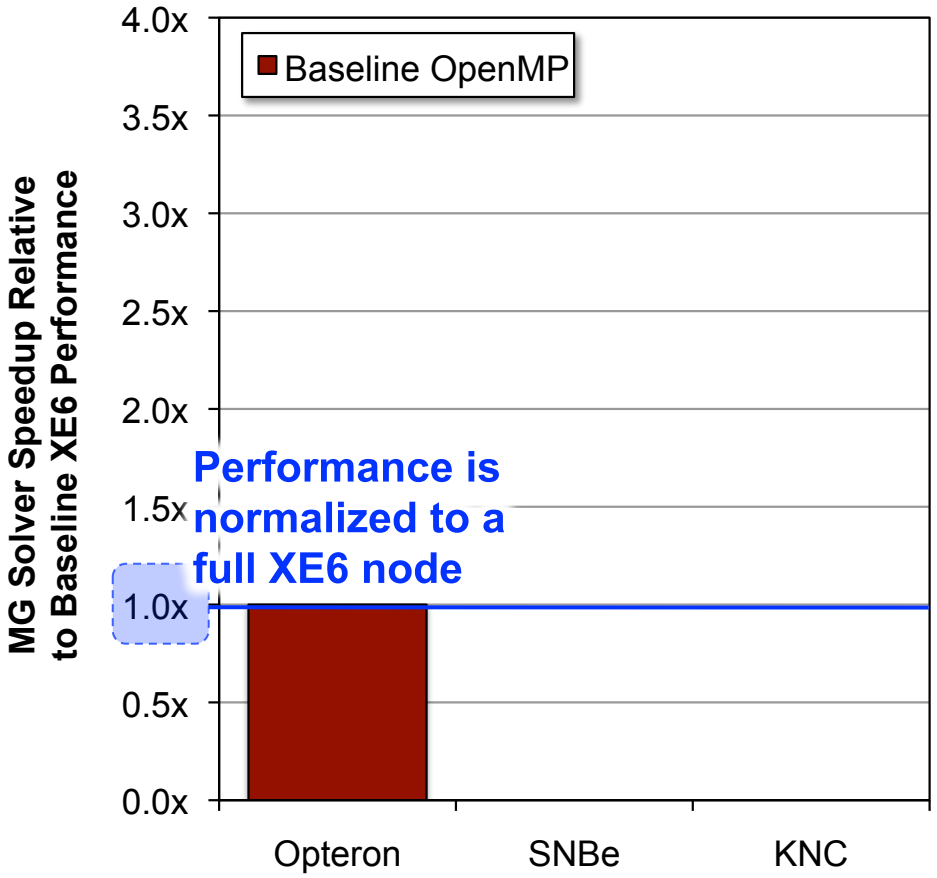
Must generate 8-way SIMD to attain peak performance

[1]Evaluation card is not necessarily reflective of production card specifications

Optimization and Single Node Results

Baseline OpenMP Performance

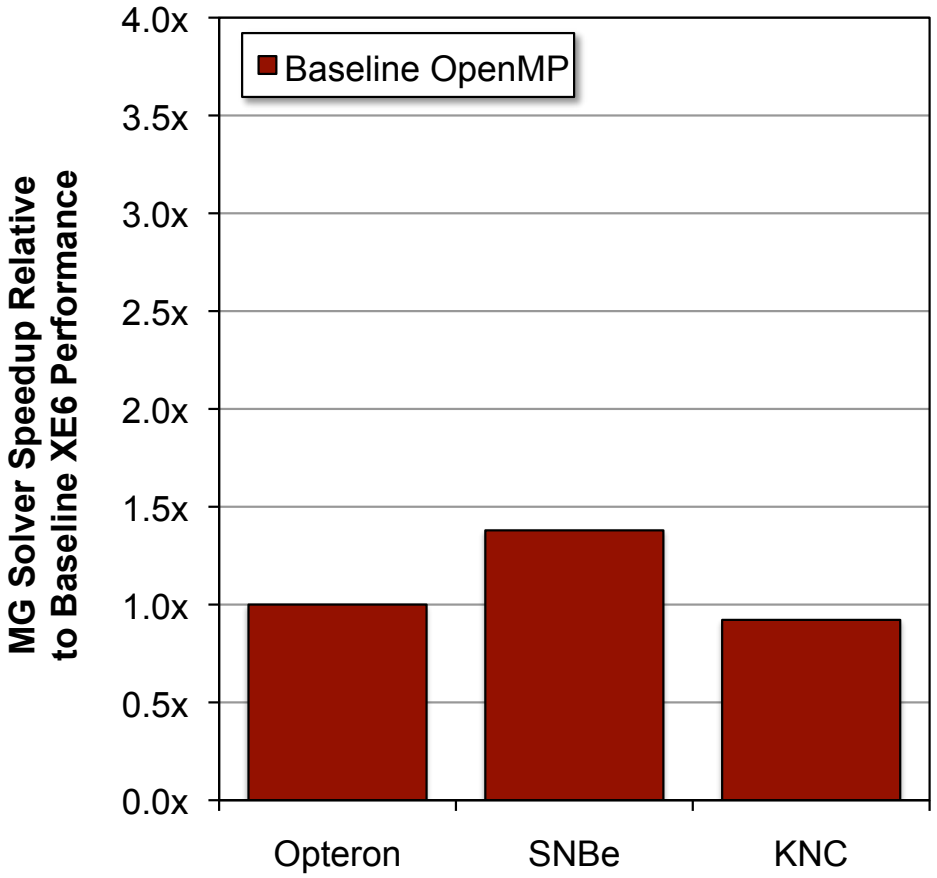
- We normalize performance (higher is better) to that of a full XE6 (Opteron) node.



Baseline OpenMP Performance

- Using ***native mode*** on KNC, we can run exactly the same OpenMP source code on all three platforms (**huge productivity win**)

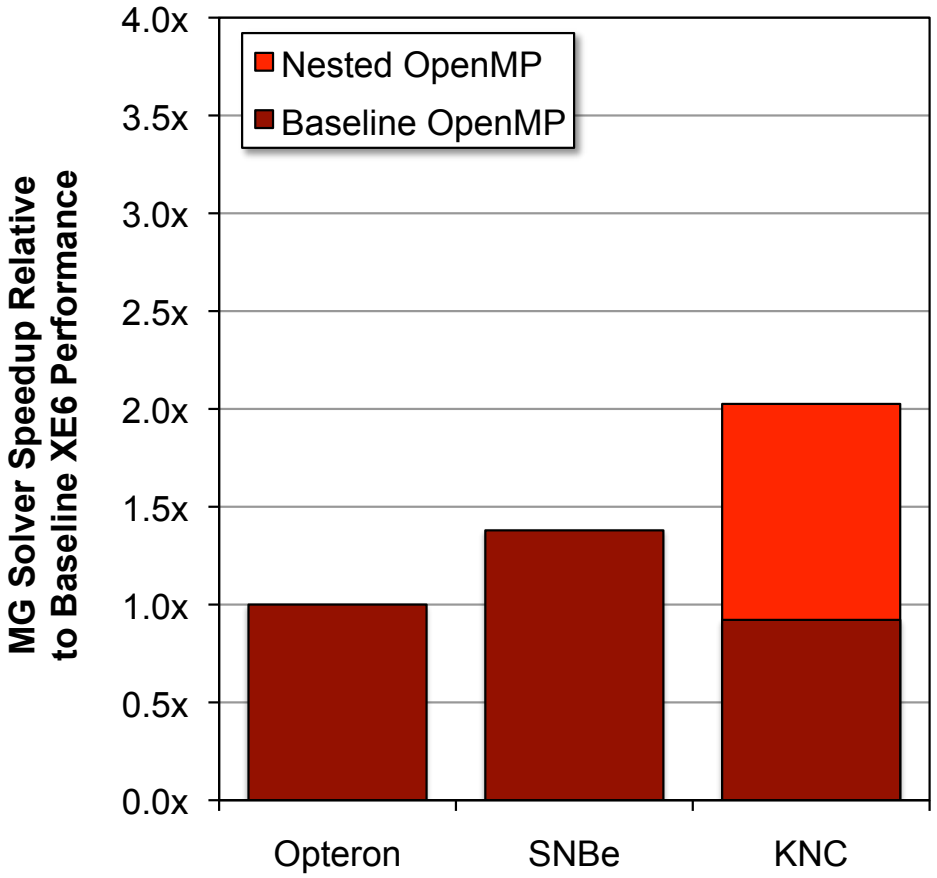
- out-of-the-box KNC **performance was less than expected**.
- Recall, each loop is no more than 64 iterations (box, i, j, k) and the latter decrease exponentially.
- Baseline code applies a single `#pragma omp parallel for` to either the box- or k-loop.
- Perfectly acceptable for 8 threads.
- Disastrous approach when there are 240. (**73% of the cores on KNC were idle**).



Nested OpenMP Performance

- Remedy is to apply parallelism hierarchically:
  - some number of threads per box
  - multiple concurrent boxes
- The optimal ratio varies as one descends through the v-cycle.
- **Optimal values were determined empirically (auto-tuning)**
- Realized within a single OpenMP parallel region with **no other optimizations**.

- Theoretically, KNC should be twice as fast as SNBe (2x the bandwidth)
  - **lack of SIMD**
  - **lack of SW prefetching**



Breakdown of Time on SNBe

- An examination of time in conjunction with a performance bound will tell us where to focus our efforts…

operation	Level in the v-cycle 64^3	32^3	16^3	8^3	4^3	**Total**
smooth	63.5%	8.7%	0.4%	0.1%	0.1%	**72.8%**
residual	1.1%	1.1%	0.0%	0.0%	0.0%	**9.0%**
restriction	1.2%	0.1%	0.0%	0.0%	0.0%	**1.3%**
interpolation	1.8%	0.2%	0.0%	0.0%	0.0%	**2.0%**
communication	10.7%	2.1%	0.4%	0.2%	0.5%	**14.0%**
Total	**85.1%**	**12.2%**	**0.9%**	**0.3%**	**0.6%**	

Breakdown of Time on SNBe

operation	Level in the v-cycle 64^3	32^3	16^3	8^3	4^3	**Total**
smooth	63.5%	8.7%	0.4%	0.1%	0.1%	**72.8%**
residual	1.1%	1.1%	0.0%	0.0%	0.0%	**9.0%**
restriction	1.2%	0.1%	0.0%	0.0%	0.0%	**1.3%**
interpolation	1.8%	0.2%	0.0%	0.0%	0.0%	**2.0%**
communication	10.7%	2.1%	0.4%	0.2%	0.5%	**14.0%**
Total	**85.1%**	**12.2%**	**0.9%**	**0.3%**	**0.6%**	

82% of solver time is spent in smooth() and residual()

Breakdown of Time on SNBe

- If we construct a **Roofline model** for the solver…

operation	Level in the v-cycle 64^3	32^3	16^3	8^3	4^3	**Total**
smooth	63.5%	8.7%	0.4%	0.1%	0.1%	**72.8%**
residual	1.1%	1.1%	0.0%	0.0%	0.0%	**2.0%**
restriction	1.2%	0.1%	0.0%	0.0%	0.0%	**1.3%**
interpolation	1.8%	0.2%	0.0%	0.0%	0.0%	**2.0%**
communication	10.7%	2.1%	0.4%	0.2%	0.5%	**14.0%**
Total	**85.1%**	**12.2%**	**0.9%**	**0.3%**	**0.6%**	

Corresponds with 98% of STREAM bandwidth !!

- We're doing the best the SNBe architecture allows.
- Only option is to **change the algorithm**.

Communication-Aggregation

- The basic algorithm demands we nominally perform one operation on each box, then communicate (exchange data on/off nodes).
  - destroys any ability to exploit on-chip locality (reuse)
  - incurs repeated overheads
- Leverage the well-known technique of using **deeper ghost zones** and **performing redundant work**.

```
for(all sweeps){
  for(all boxes){
    // exchange boundary
  }
  for(all boxes){
    // apply operator
  }
}
```

Locality only if D$ >= Nsubdomains * GridSize (~ 1GB !!!)

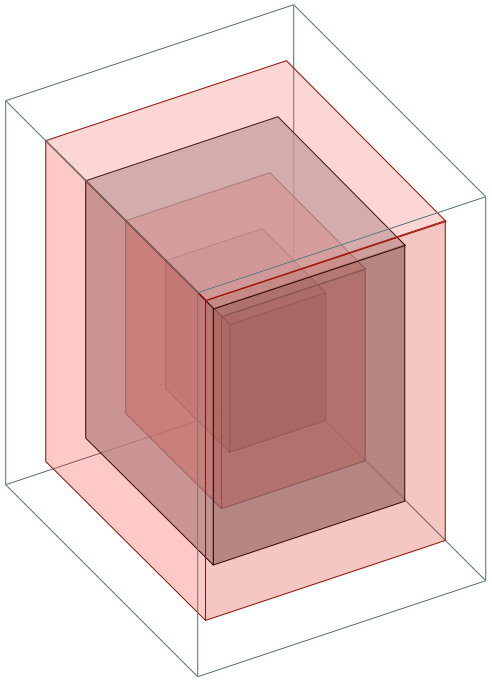
```
for(all boxes){
  // exchange deep boundary
}
for(all boxes){
  for(all sweeps){
    // apply operator
  }
}
```

Locality if D$ >= GridSize (~20MB)

(if OpenMP is used one box at a time)

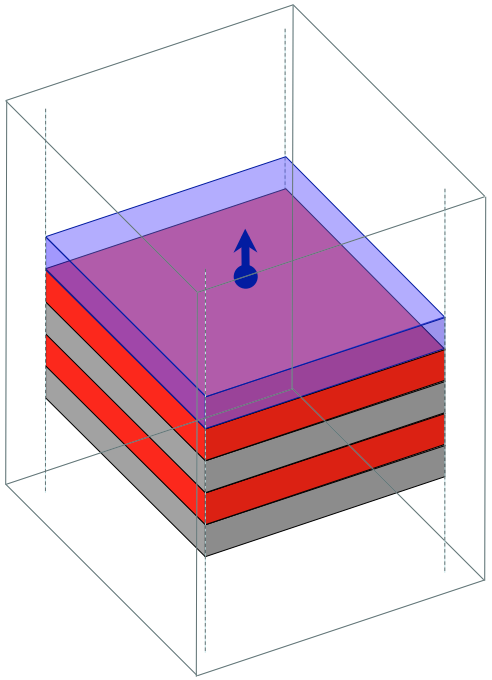
DRAM Communication-Avoiding

- In theory, a 4-deep ghost zone allows for **up to a 3.1x speedup on smooth** on the finest (64^3) grids.
- However, in a naïve implementation, one updates a color on the whole grid before the next
  - **First (red) sweep is very slow**
    (reading data from DRAM)
  - Next 3 sweeps are fast (reading from LLC)
    These **amortize** the first iteration



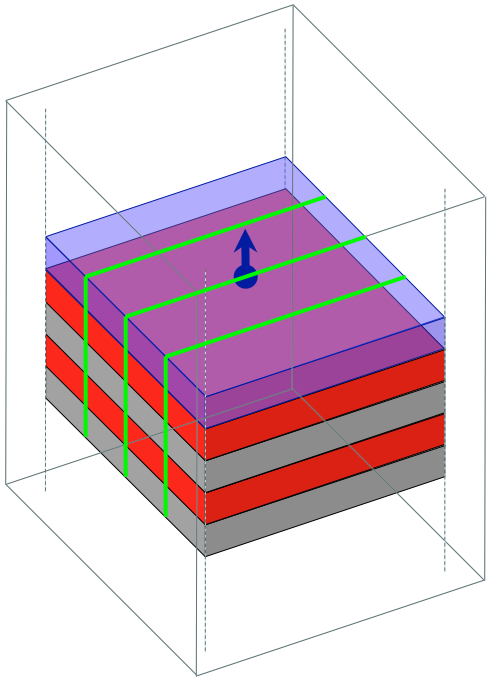
DRAM Communication-Avoiding

- In theory, a 4-deep ghost zone allows for **up to a 3.1x speedup on smooth** on the finest (64^3) grids.
- However, in a naïve implementation, one updates a color on the whole grid before the next
  - **First (red) sweep is very slow**
    (reading data from DRAM)
  - Next 3 sweeps are fast (reading from LLC)
    These **amortize** the first iteration
- A better solution is to apply it in a wavefront.
  - prefetches (reads) the next plane from DRAM
  - Sequentially apply the GSRB relax to the next four planes (red/black/red/black)
  - **Decoupling communication and computation hides the slow DRAM access**.



DRAM Communication-Avoiding

- In theory, a 4-deep ghost zone allows for **up to a 3.1x speedup on smooth** on the finest (64^3) grids.
- However, in a naïve implementation, one updates a color on the whole grid before the next
  - **First (red) sweep is very slow**
    (reading data from DRAM)
  - Next 3 sweeps are fast (reading from LLC)
    These **amortize** the first iteration



- A better solution is to apply it in a wavefront.
  - prefetches (reads) the next plane from DRAM
  - Sequentially apply the GSRB relax to the next four planes (red/black/red/black)
  - **Decoupling communication and computation hides the slow DRAM access**.
- Even faster if we thread the wavefront with OpenMP
  - requires **cache coherency** and **fine-grained synchronization**
    (had to write a custom pairwise barrier for performance)

Optimization and Auto-tuning

- Explicit SIMDization (via intrinsics)
  - wrote a Perl script to generate the inner AVX and SSE stencil kernels (including SW prefetching). **Auto-tuned to find best version.**
  - SIMDize GSRB via predication (e.g. `blendvpd`)

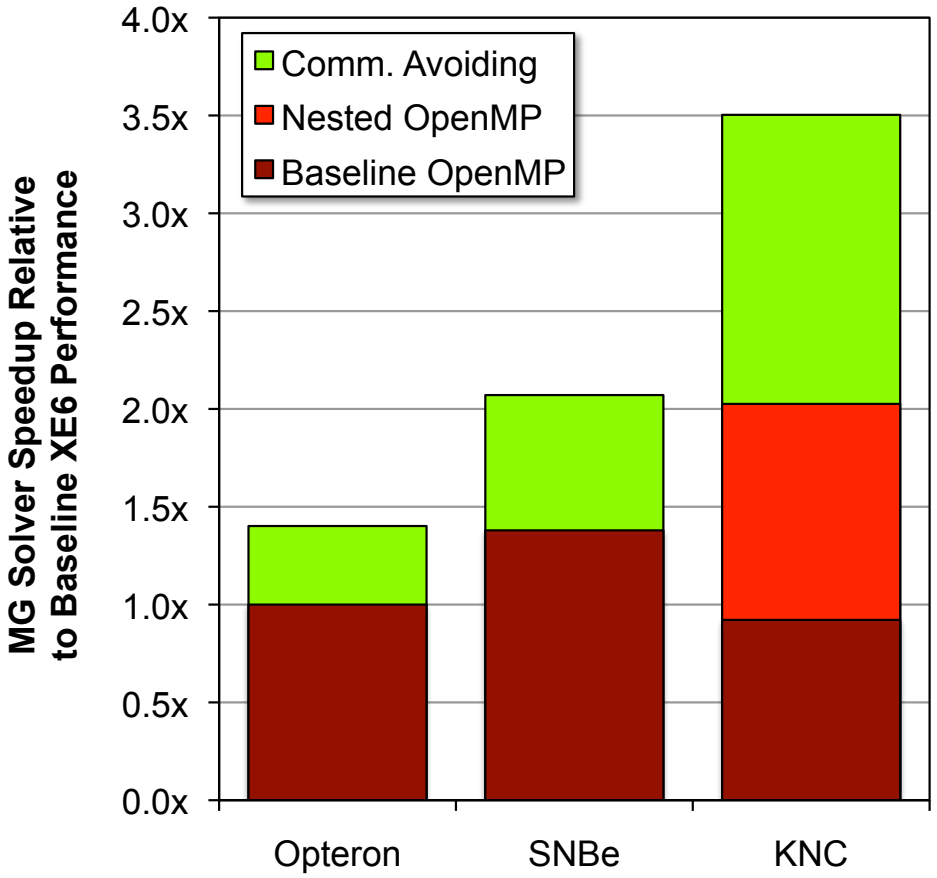
    i.e. all lanes within a SIMD register perform stencils on consecutive elements, but a merge restores the elements not to be modified.
- Fusion of residual and restriction.
  - The residual is immediately restricted (becoming the new RHS)
  - Eliminates the write and re-reading of the residual
- 2MB pages on KNC

- For background papers on auto-tuning, see

  http://crd.lbl.gov/about/staff/cds/ftg/samuel-williams

Communication-Avoiding, Fully-Optimized

- We observe a substantial speedup in **overall solver time** when both communication-avoiding AND low-level optimizations are used…
  - 1.5x on SNBe
  - 1.7x on KNC
- **speedup for smooth()…**
  - 2.1x on SNBe (out of 3.1x)
  - 2.2x on KNC
- Challenged by **large D$ working set** comparable to the L2$
- Challenged by memory access pattern in smooth (**few KB stanza**)
  - 66% of STREAM on SNBe
  - 56% of STREAM on KNC



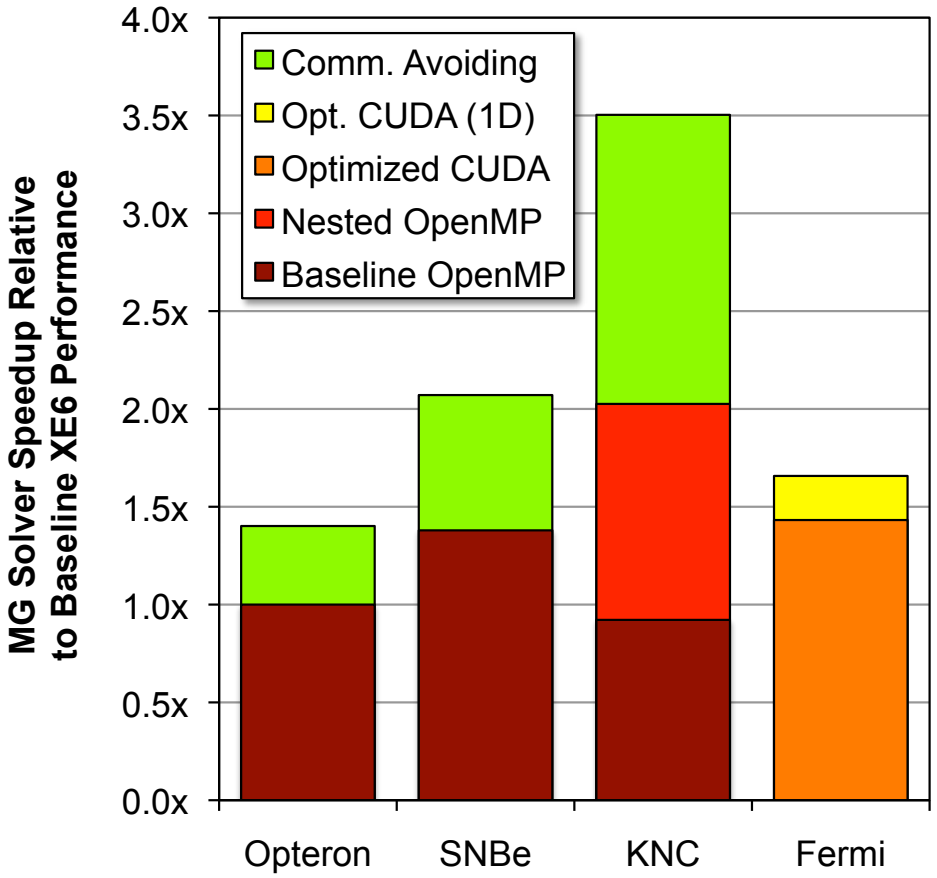
GPU Implementations

- In the time since the paper submission, we've created a **highly-optimized CUDA implementation**…
  - parameterized 2D and 1D(=perfect memory coalescing) thread block conceptualizations
  - tuned over subset of several million optimization combinations

- Single-node experiments evaluated using an NVIDIA M2090 (Fermi) GPU with ECC enabled.
- Machine has a streaming bandwidth of ~120GB/s.

GPU Comparison

- M2090 is slightly faster than the 2P SNBe on the baseline algorithm
  - M2090 has more bandwidth
  - SNBe likely moves less data
- The portable, nested OpenMP implementation running on KNC is faster than heavily-optimized CUDA on Fermi.
- The combined benefits of processor (KNC), communication-avoiding, and optimization resulted in a **3.5x speedup** on the solver time.



Communication-Avoiding on GPUs

- We have implemented several communication-avoiding versions for the GPU.

- However, they demand we either…
  - have thread blocks perform a great deal of **redundant work** within a subdomain (bound by SM capabilities)
  - use enormous thread blocks (**5K threads and 1.5MB of registers**)
  - abandon CUDA ethos and use **persistent threads, fine-grained synchronization**, and communication through the L2
- In reality…
  - The former has yet to result in better performance than the non-communication-avoiding implementation
  - The second is a non-starter (we can't change architecture or CUDA), however, the hope is Kepler will mitigate some issues.
  - We have not explored the third.

Summary

Summary

- Geometric multigrid solvers for 2nd order operators are **heavily bound by data movement**
- To improve performance, one must embrace:
  - Faster algorithms that **move less data**
  - Architectures and DRAM technologies that **move data faster**
- We quantified the benefits of:
  - communication-avoiding to address the former
  - Use of the Intel® Xeon Phi™ and NVIDIA GPUs to address the latter.

Summary

- Xeon Phi™'s use of a **coherent cache** allows for a simpler (**conventional OpenMP**) implementations that can minimize data movement whilst exploiting superior GDDR bandwidths
- CUDA/GPUs can constrain one's ability to minimize data movement.

- Manual expression of data parallelism (**SIMDization**) and data movement (**SW prefetching**) remained essential on all CPUs.
  - The GPU's SIMT architecture allows one to **rewrite code once (in CUDA), and then exploit increased parallelism in future GPUs**
  - CPUs must make similar advances in languages/models/compilers.
  - OpenMP needs better constructs/runtimes for **hierarchical parallelism**

- Even when provided with the opportunity for communication-avoiding (deep ghost zones), **compilers utterly fail to exploit it**.
  - manually implementing a wavefront is **painful and error prone**
  - need compilers/DSLs/tools to step up

Future Work

- Continued optimization and convergence of CPU, KNC, GPU and (eventually) BGQ implementations
- Distributed studies of KNC and GPUs
  - KNC and GPU were single node/card studies
  - MPI performance can become critical on coarse grids
- High-order relaxation schemes
  - require same deep ghost zones (no more communication)
  - one high arithmetic intensity stencil (instead of multiple passes)
  - no redundant work
- Communication-avoiding bottom solvers
  - scalable, fast bottom solvers demand an altogether different set of optimizations from the v-cycle

Acknowledgements

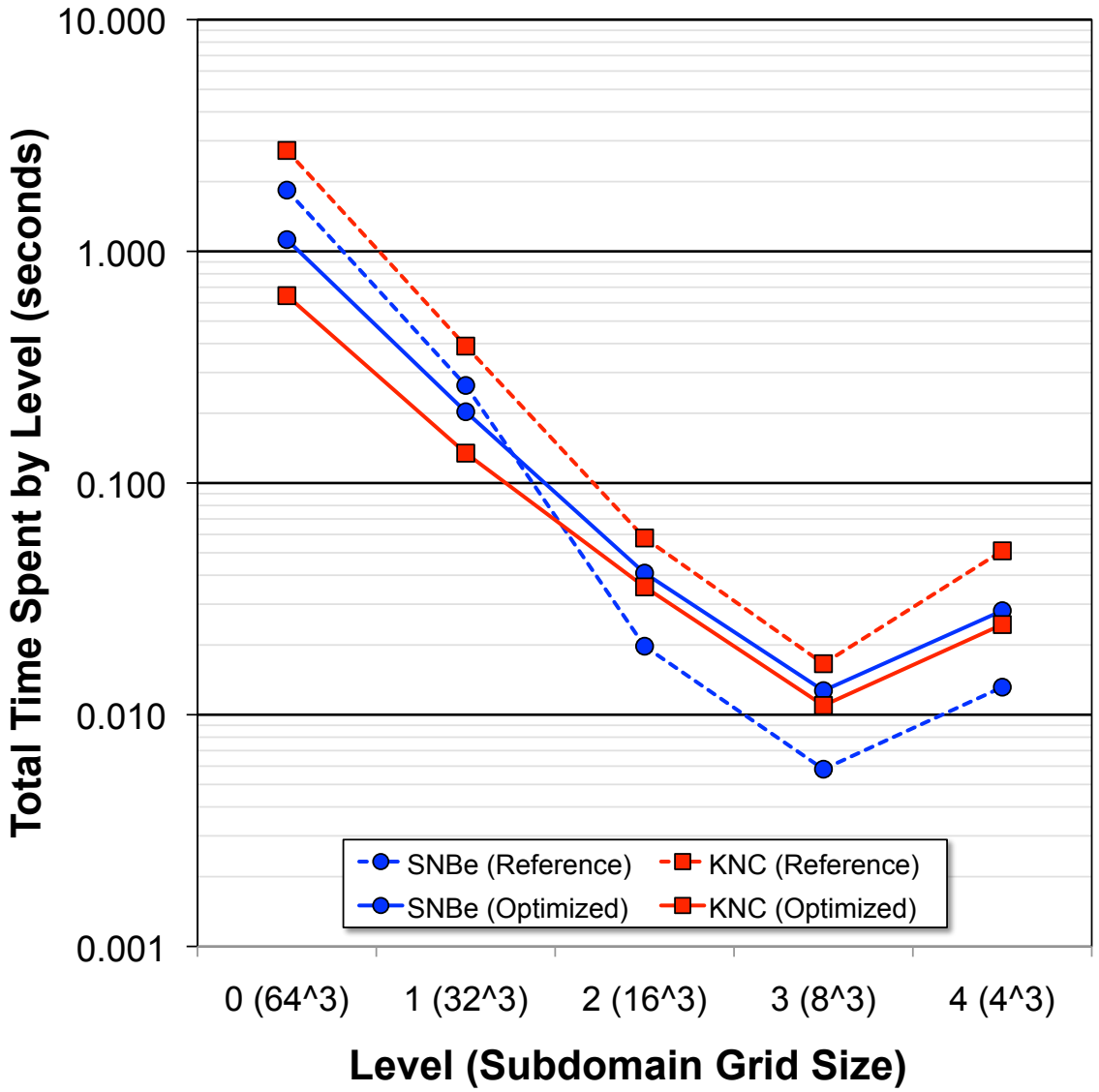
- All authors from Lawrence Berkeley National Laboratory were supported by the DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231.
- This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.
- This work was supported in part by National Science Foundation grants: OCI #0910847, Gordon: A Data Intensive Supercomputer; and OCI #1053575, Extreme Science and Engineering Discovery Environment (XSEDE).
- This research used resources of the Keeneland Computing Facility at the Georgia Institute of Technology, which is supported by the National Science Foundation under Contract OCI-0910735.

Questions?

Backup Slides

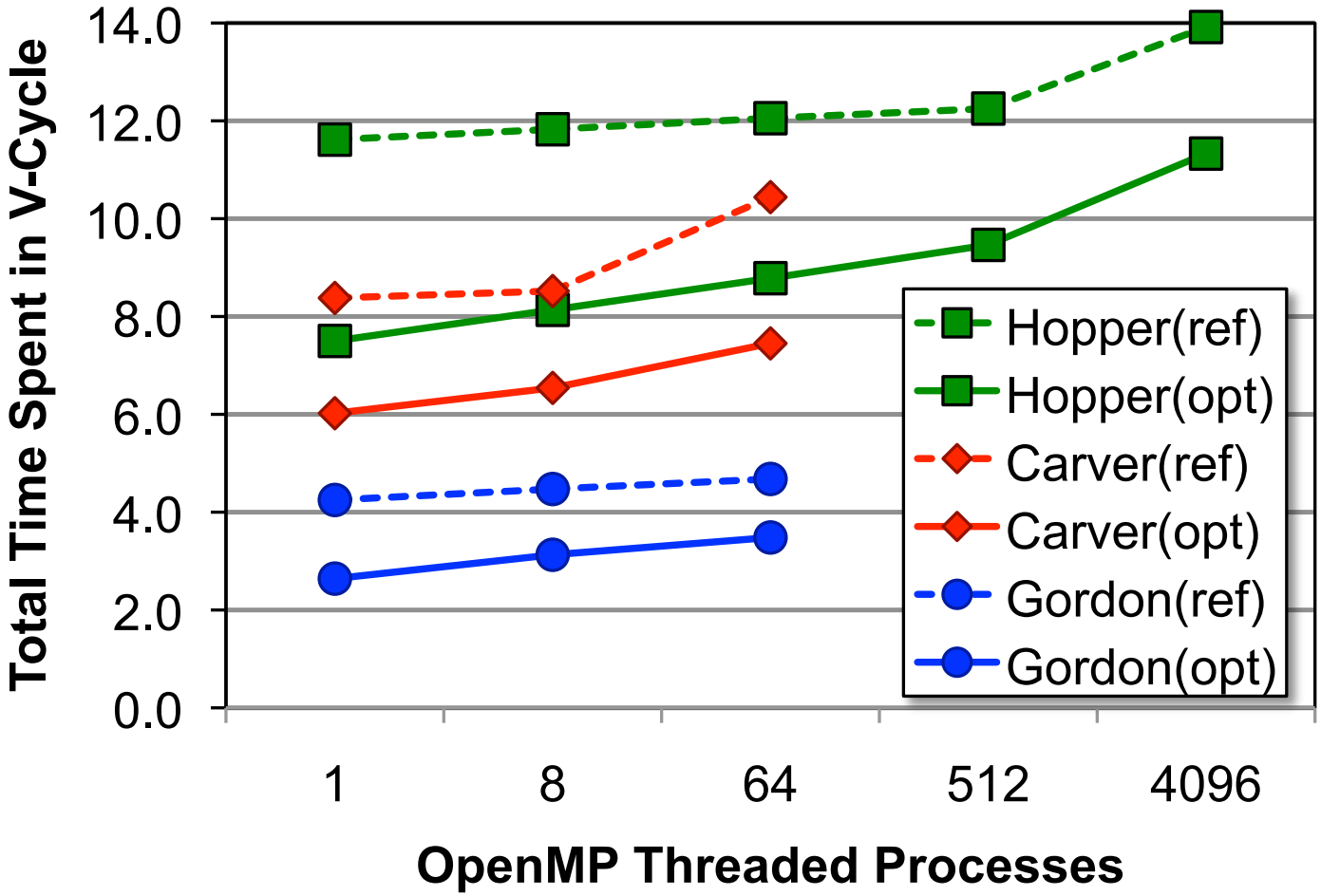
Time per Level

- Optimized KNC was fastest at the finest levels.
- Eventually, there is a crossover, and unoptimized (no CA) was fastest at the bottom (all subdomains fit in LLC)



Weak Scaling of the V-cycle

- 256^3 (64 x 64^3 subdomains) per NUMA node
- 1 'threaded process' = 1 chip



Evaluation Platforms

- …
- NVIDIA M2090 (“Fermi”)

	Opteron	SNBe	KNC[1]	Fermi[2]
cache per core	64+512KB	32+256KB	32+512KB	192KB
cores (threads) per chip	6	8	60 (240)	16
SIMD-width (DP)	2	4	8	32
LLC per chip	6MB	20MB	-	768KB
chips per node	4	2	1	1
DP GFlop/s per node	201.6	332.8	1248	665
STREAM GB/s per node	49.4	70	150	120
Programming Model	MPI+OMP	MPI+OMP	OpenMP	CUDA

[1]Evaluation card is not necessarily reflective of production card specifications
[2]We consider 1 SM as the equivalent of 1 CPU core

1D CUDA Thread Blocks

- Within a block, there are…
  - Threads that will perform the stencil
  - Threads to hold the halo (within the plane)
  - Threads to guarantee coalescing
- Thread block streams thru the k-dimension
- Can tune for any # of threads (restricted to multiples of 16 or 32)
- Optimal # of threads/block were 480, 512, or 992