Using miniGMG as a Testbed for Computer Science Research

Samuel Williams

Lawrence Berkeley National Laboratory

SWWilliams@lbl.gov

Exascale Challenges

- Compared to today's CPU processors, Exascale machines may see:
  - O(100x) increase in peak flops per chip
  - Massive (>>100x) increase in on-chip parallelism (<<GHz? for NTV)
  - O(<<100x) increase in (fast) DRAM bandwidth
  - systems with ~100,000 nodes (or more)
  - minimal reductions (or possible increases) in latencies and overheads
- A number of performance challenges will emerge or persist.
  - At 100K nodes, the performance of **collectives** can be poor.
  - Codes will become increasingly **memory-bound**
  - The massive increase in parallelism will challenge the straightforward application of OpenMP to loop nests (**finite parallelism in loop nests**)
  - **Communication/Synchronization overheads** may impede performance.
  - Serial components receive a triple whammy:
    - Amdhal's Law
    - in-order scalar cores
    - lower frequency

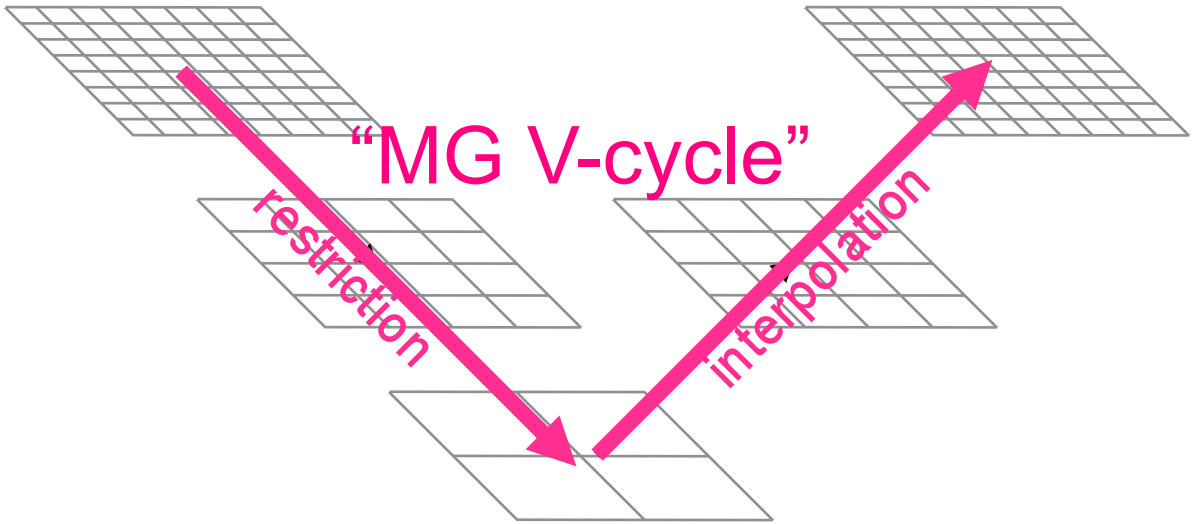
Benchmarks…

- Benchmarks/compact apps that have…
  - massive, regular, static parallelism within each loop nest,
  - poor surface:volume ratios,
  - regular communication patterns with large messages,
  - minimal use of collectives
- …will likely **run well on exascale machines**.
- Porting such codes to exascale tends to be a software engineering task, not a CS research challenge.

- We wish to focus on benchmarks that…
  - represent key/fundamental computational characteristics of applications
  - will be a challenge to run on exascale architectures (make for interesting CS, AM, and co-design research projects)
  - **We have thus focused on multigrid…**

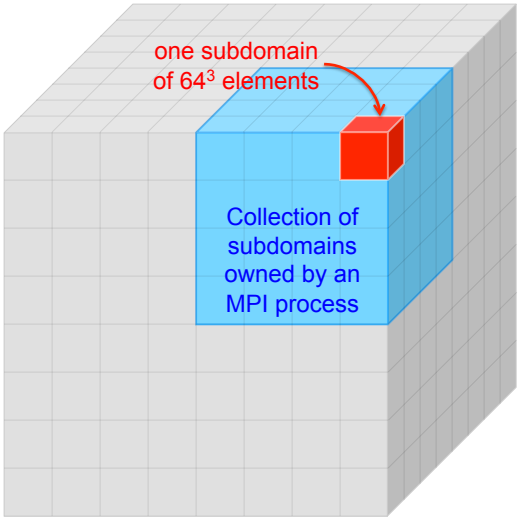
Multigrid Introduction

- 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



miniGMG: Specification

- Cubical domain decomposed into subdomains ("boxes") and distributed across a machine.
- Fine-grid box size is selectable.
  - smaller boxes mimic AMR MG challenges
  - fewer boxes per process can be used to mimic some AMR MG combustion codes.
- Nominally, problems are usually small enough to fit into Xeon Phi or GPU device DRAM => **no PCIe transfers are required.**



- Gauss Seidel, Red-Black ("GSRB") relaxation in the v-cycle
  - (other smoothers / stencils can be used)
- Configurable U-cycle (default stops at 4^3 subdomains)
- Selectable bottom solver
  - GSRB's, BiCGStab, CG, CA-BiCGStab, CA-CG, etc…
- Fixed 10 V-cycles

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

miniGMG Challenges

- Code has little reuse and large working sets:
  - VC 2nd order => **DRAM bandwidth-bound** (flop:byte ~ 0.2)
  - Cache working set **~350KB(planes) & >20MB(volume) per subdomain**
  - Only 1 stencil sweep per MPI communication step
  - Small subdomains = **significant time 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**
- Communication is variable and can become a bottleneck
  - P2P MPI is initially a small fraction of the time, but as one descends through the v-cycle, it becomes a bottleneck (message size doesn't decrease at the same rate as computation).
  - Worse, Krylov bottom solvers are **not O(N) and require collectives** (become bottlenecks when weak scaled)

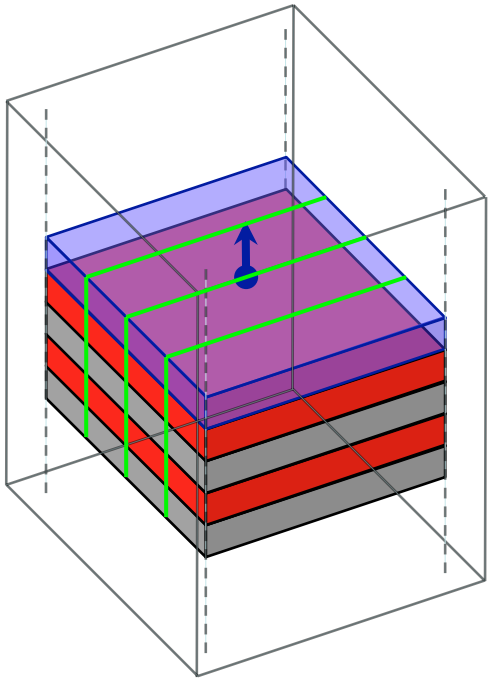
Research Projects using miniGMG

- Currently using miniGMG for research into…
  - Processor and network exploration/extrapolation — XTune/ExaCT
  - Communication-avoiding smoothers — CACHE
  - Compiler-based (CHiLL) smoother optimization — XTune
  - Communication-avoiding (bottom) Krylov solvers — CACHE/ExaCT
  - Numerical Challenges in CA Bottom Solvers — CORVETTE
  - Algorithmic exploration of FMG (F-Cycle) — XTune
  - High-order operators — XTune/ExaCT
  - Programming models (omp task/Habanero) — DEGAS
  - PGAS for P2P communication (UPC/CAF) — DEGAS
  - Resource management — DEGAS
  - Automatic overlap of communication/computation — BAMBOO at UCSD

- I will detail a few here…

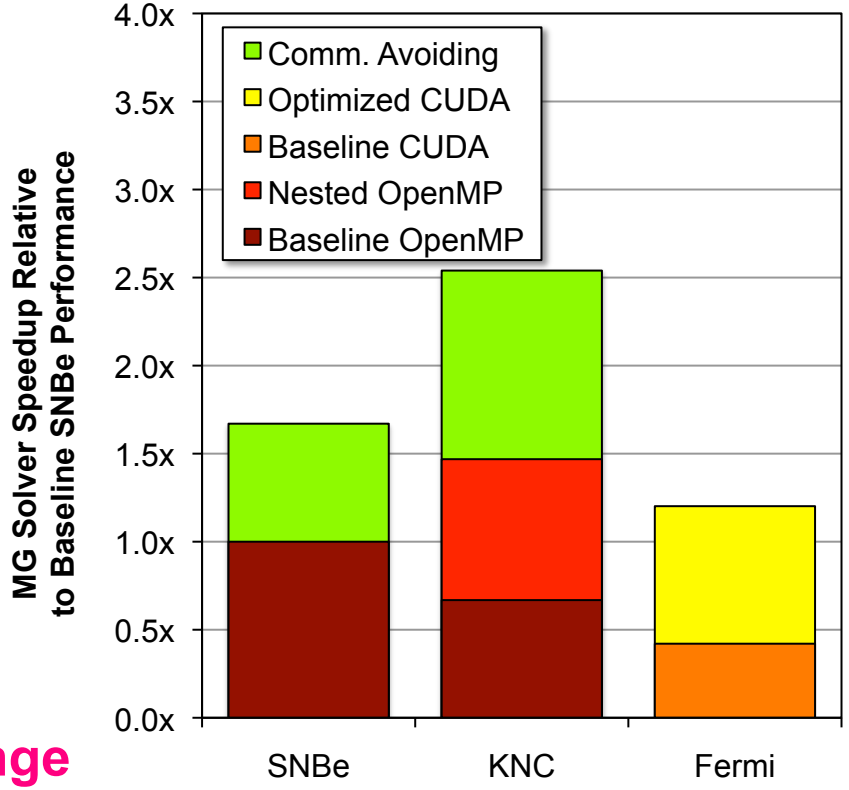
Communication-Avoiding Smoothers (CACHE/Xtune)

- Restructure the smooth( )'s in the V-cycle to:
  - load thick ghost zones
  - perform some (inter-process) redundant computation
  - and in effect, fuse multiple smooth( )'s together.
- High-performance wavefront implementations require
  - require fine-grained synch.
  - parallelization in 2D or 3D
  - Software prefetching to overlap communication and computation
  - SIMDization of GSRB kernels



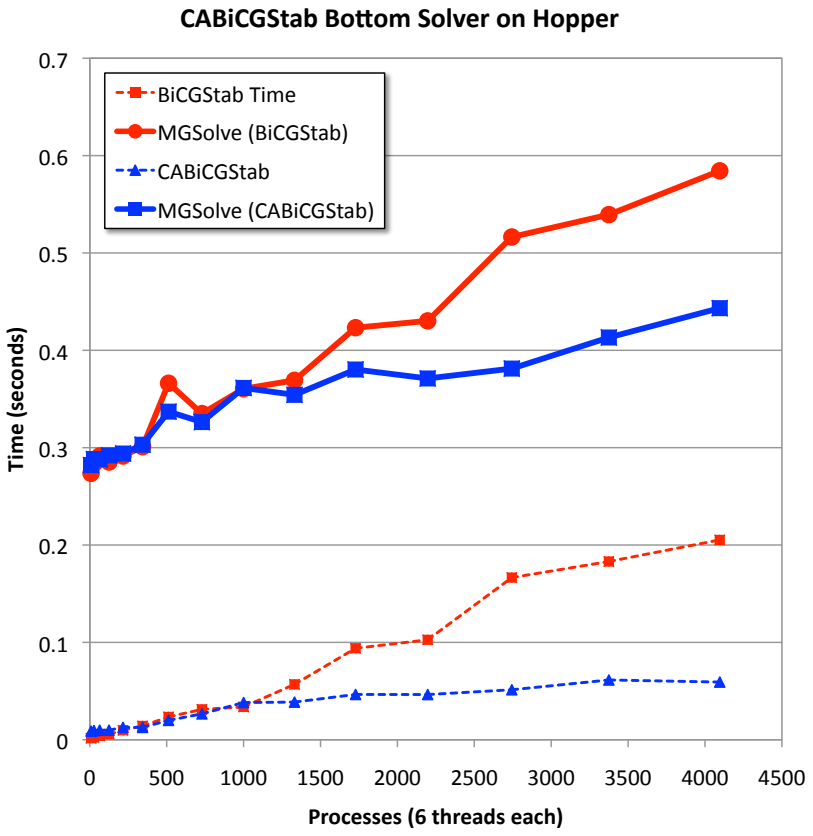
Communication-Avoiding Smoothers (CACHE/Xtune)

- Restructure the smooth( )'s in the V-cycle to:
  - load thick ghost zones
  - perform some (inter-process) redundant computation
  - and in effect, fuse multiple smooth( )'s together.
- High-performance wavefront implementations require
  - require fine-grained synch.
  - parallelization in 2D or 3D
  - Software prefetching to overlap communication and computation
  - SIMDization of GSRB kernels
- **Benefit can be significant** even on manycore architectures like Intel's Xeon Phi.
- **However, automation is a challenge**



Communication-Avoiding Bottom Solvers (CACHE/ExaCT)

- In geometric multigrid, one is often forced to switch to a Krylov solver when further restriction of the grid become infeasible.
- Unfortunately…
  - iterative Krylov methods are not O(N). Thus, weakly scaled problems require far more iterations to converge.
  - Worse, global dot products are required on every Iteration.
  - As the performance of collectives like MPI_AllReduce( ) do not scale well, these global dot products amplify the bottom solver challenge.
- Solution is to change the algorithm so that it aggregates these collectives together. (see poster by Erin Carson, Nick Knight, and Sam Williams)



Numerical Challenges in Comm.-Avoiding Krylov Solvers

- Communication-avoiding Krylov Methods are parameterized by ‘s’ (the number of steps in an s-step method).
- As ‘s’ increases, effects of finite precision are manifested.

- Increased precision or alternate formulations may address this.
  - Is simply changing the loop vectors to double-double sufficient? or must one change the entire computation?
  - Nominally, double-double is shunned due to cost.
  - However, bottom solvers are tiny and usually latency-limited.
  - Perhaps the additional cost can be amortized by MPI_WaitAll( ) or MPI_AllReduce( ) time

- See Corvette(Xstack) poster (Cindy Gonzalez, Costin Iancu, …)

Habanero (DEGAS)

- The best communication-avoiding implementations on BGQ and especially MIC are extremely complicated.
  - Multiple thread teams map to inter- and intra-box parallelism.
  - Teams of threads must be placed compactly. This is analogous to a CUDA thread block with the caveat that the team can grow as large as the chip (not just the SMX)
  - On MIC, one must manually orchestrate parallelism, locality, and synchronization via a omp parallel region.

- We are exploring Habanero as a productive alternative.
  - an architecturally-derived HPT can be constructed
  - async's can be inserted into specific nodes to balance inter- and intra-box parallelism

- Costin and Vivek can detail progress to date…

PGAS for Fast Communication (DEGAS)

- Progressive optimization of on-node computation (e.g. DRAM communication-avoiding, threading, SIMD, etc…) can result in P2P MPI communication becoming a performance bottleneck.
- We are exploring the use of UPC to accelerate this data exchange.

- Costin Iancu / Nick Vrvilo will explain tomorrow…

Questions?