A task-based directed acyclic graph implementation of additive AMG

Amani AlOnazi[1], George S. Markomanolis[2] and David Keyes[1]

[1]Extreme Computing Research Center, King Abdullah University of Science and Technology, Saudi Arabia
[2]KAUST Supercomputing Laboratory, King Abdullah University of Science and Technology, Saudi Arabia

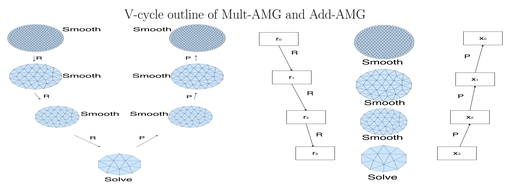
Summary of main contributions

- We present a parallel approach based on data-driven execution to speed up the on-node performance of AMG.
- We extend the Vassilevski and Yang "mult-add" AMG algorithm with a new implementation as an asynchronous directed acyclic graph, where nodes represent tasks, and edges represent dependencies among them.
- We implement a tiling approach for decomposing the grid hierarchy into parallel units within task containers. The distribution of tiles is left to the compiler and runtime to expose task-level parallelism.

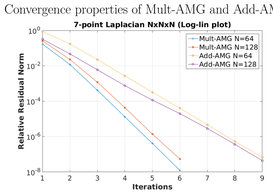
Introduction

Algebraic multigrid is the solver of choice in many and growing applications in today's petascale environments and scales well in a weak, distributed-memory sense. Exascale architectures exacerbate the challenges of increased strong SIMT concurrency, reduced memory capacity and bandwidth per core, and vulnerability to global synchronization. The recent Vassilevski-Yang (2014) "mult-additive" AMG nearly preserves the convergence rate of the multiplicative form of AMG, while reducing communication and synchronization frequency and controlling memory growth. However, it remains bulk-synchronous. We extend the algorithm further towards exascale environments with a task-based directed acyclic graph implementation.

V-cycle outline of Mult-AMG and Add-AMG



Convergence properties of Mult-AMG and Add-AMG



Main Ideas

The additive-AMG offers an extra factor of parallelism between levels of the grid hierarchy:

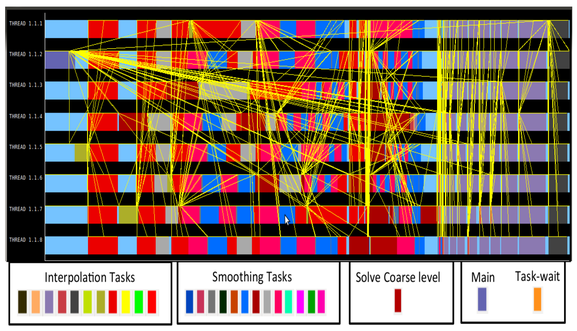
- Fine-grained task-based implementation to deal with intra-node concurrency
- Asynchronous execution model
- Hiding communication latency

Implementation Details:

1. BoomerAMG in *hypre* v-2.11.1 library (LLNL)
2. OmpSs task-based programming model (BSC)
3. Hybrid distributed-shared memory parallelism
4. MPI exploits inter-node parallelism while a task-based directed acyclic graph exploits node parallelism

Task Dependencies Graph Illustration

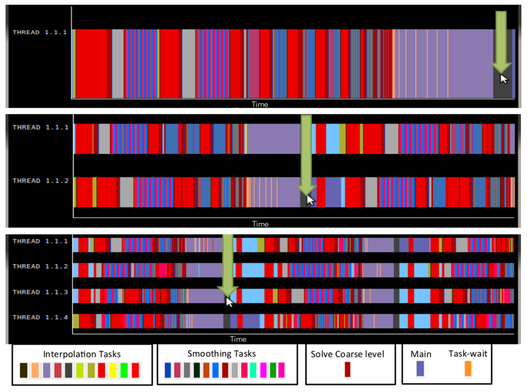
The following is the task dependencies graph of executing one iteration of additive AMG as a preconditioner for conjugate gradient solving a 7-point Laplacian 128×128×128 using 8 threads:



Generated by Extrae and Paraver, which are tools to profile and visualize trace-files for runtime analysis

Concurrency Analysis

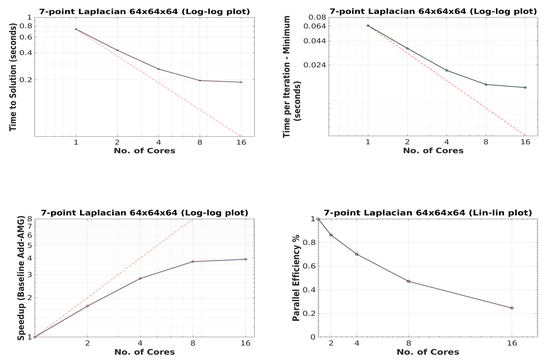
Execution trace of the additive AMG using 1, 2, 4 threads



Results 1

SHAHEEN at KAUST:
Cray XC40, Intel Haswell 2.3GHz, 2 CPU sockets per node, 16 processor cores per CPU.

Strong scalability of the solution of a 7-point Laplacian on a Cartesian grid 64×64×64

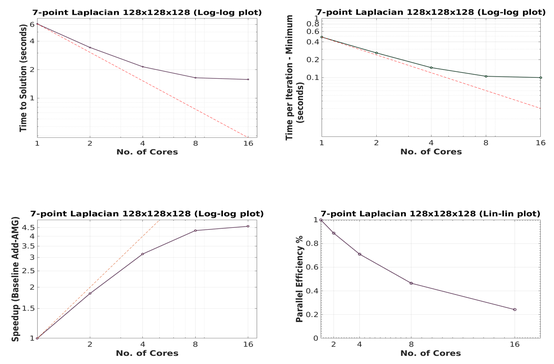


Speedup

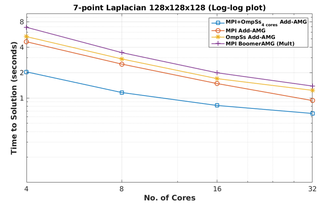
The results show more than 4× speedup on-node performance compared to single core execution. As we increase the number of cores, cache misses increase and for this reason the performance begins to degrade after 4 cores.

Results 2

Strong scalability of the solution of a 7-point Laplacian on a Cartesian grid 128×128×128



Strong scaling of BoomerAMG both multiplicative and additive forms (MPI), Add-AMG (OmpSs), and Add-AMG (MPI+OmpSs)



Ongoing Work

- Taskifying communication (multithreaded-MPI)
- Hierarchical data structure: expose data locality

Acknowledgements

The OmpSs and Tools (Paraver, Extrae) groups of Barcelona Supercomputing Center, and the *hypre* group of Lawrence Livermore National Laboratory are gratefully acknowledged for their software support.

- E-mail: {amani.alonazi, david.keyes, georgios.markomanolis}@kaust.edu.sa

KAUST SC16 Salt Lake City | hpc Union | matters.