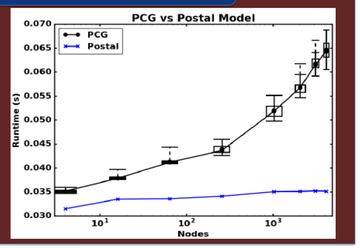
Scalable Non-blocking Krylov Solvers for Extreme-scale Computing

Paul R. Eller, William Gropp (Advisor)
University of Illinois at Urbana-Champaign

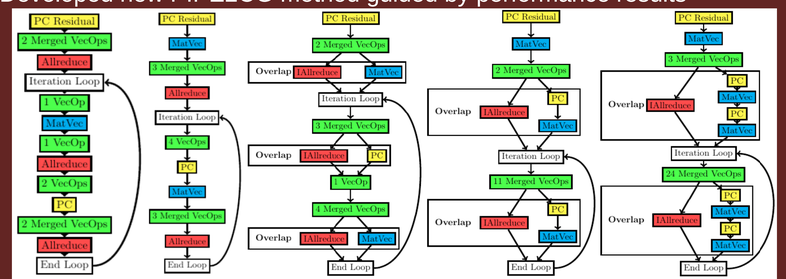
Preconditioned Conjugate Gradient Method

- Popular iterative method for solving sparse linear systems
- Blocking allreduce limits performance at scale
- Further slowdowns observed at scale
- Standard performance models do not explain performance decreases



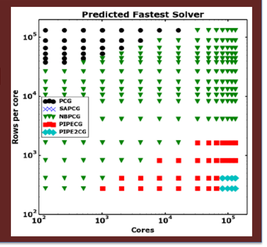
What are Scalable PCG Methods?

- Can rearrange PCG to combine multiple allreduces and/or overlap communication and computation
- Developed new PIPE2CG method guided by performance results



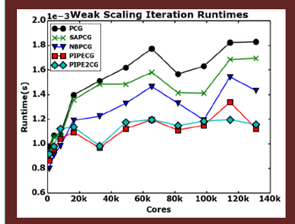
(a) PCG Hestenes and Stiefel 1952 (b) SAPCG D'Azevedo et al. 1993 (c) NBPCG Gropp 2010 (d) PIPECG Ghysels et al. 2014 (e) PIPE2CG Eller and Gropp 2016

- Rearranging methods adds initialization and vector operations overhead
- Compute vector ops element-wise to reduce overhead
- Call MPI_Test() or use progress threads to obtain effective overlap for non-blocking allreduce
- LogGOPS model suggests fastest solver transitions from blocking to overlapped methods as cores increase and rows per core decrease

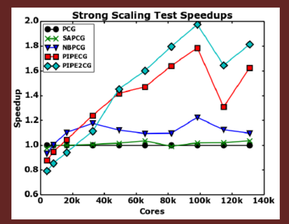


How do Scalable PCG Methods Perform?

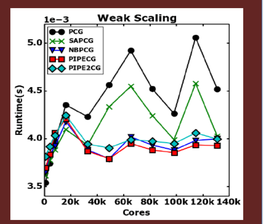
Ran tests for 27-point Poisson on Blue Waters on up to 128k cores



Weak scaling tests show improved performance and less performance variation across core counts for non-blocking methods



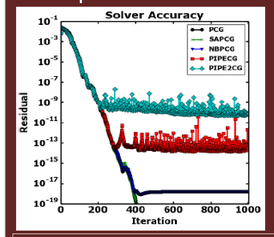
Non-blocking methods scale further than blocking methods for strong scaling tests



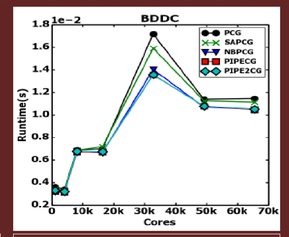
Network noise can cause significant variation in runtime for blocking methods

How Robust are Scalable PCG Methods?

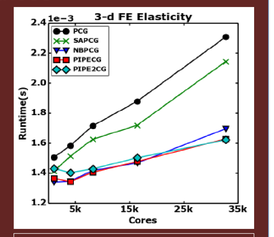
Studied solver accuracy, performance with more complex preconditioners, and performance for 2-d and 3-d finite element matrices



Further pipelined methods have reduced accuracy due to the accumulation of rounding error (Cools et al 2016)



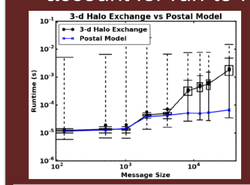
Obtained slight speedups for black box BDDC and multigrid preconditioners without modification



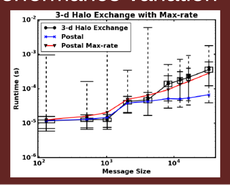
Significantly improved performance for 3-d finite element matrices with many non-zeros per row

Improving the Performance Model at Scale

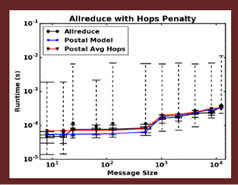
- Developed analytic performance models that read information from actual runs for network information and computation runtimes
- Produced accurate communication models on up to 1k nodes that can account for run-to-run performance variation



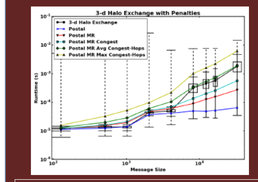
Postal model: $T = \alpha + \beta n$
α: Latency cost
B: Bandwidth cost
Simple approach to modeling parallel communication



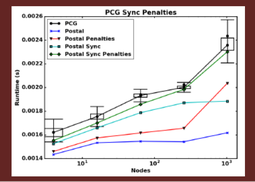
Max-rate (Gropp 2016):
$T = \alpha + n_{cores} \cdot n/\min(\beta_{node}, n_{cores}\beta_{core})$
Updates postal model for limited off-node bandwidth



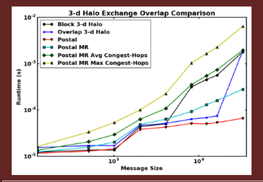
Hops penalty:
$T_{hops} = T_{hop\_lat} \cdot n \cdot n_{hops}$
Models cost of varying message distances



Congestion penalty:
$\gamma = n_{stalls}/n_{bytes}$
$T_{congest} = \gamma \cdot n \cdot n_{cores} \cdot T_{stall}$
$T_{congest\text{-}hops} = T_{congest} \cdot n_{hops}$
Models slowdown due to network contention



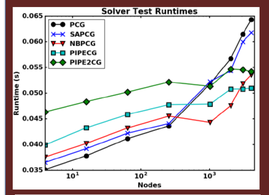
Synchronization penalty:
Computes difference between computation for median and max process since last sync and adds to allreduce cost



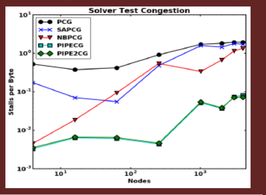
Overlap comparison:
Basic postal model provides reasonable model for effective overlap and postal with penalties models ineffective overlap

Analyzing Network Performance

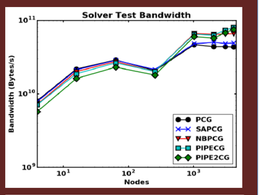
Developed analysis tools using PAPI network performance counters



Blocking solver performance decreases at higher node counts. Non-blocking solvers avoid large slowdowns but still have slight slowdowns.



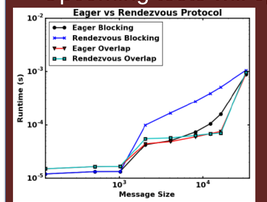
Non-blocking solvers generally experience less congestion, while blocking solvers reach peak congestion (1+ stalls per byte) at 1k+ nodes



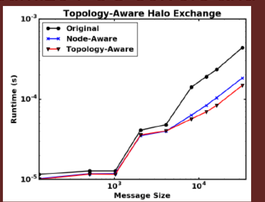
High bandwidth coincides with high congestion and a drop in performance, especially for blocking solvers

Improving PCG Communication

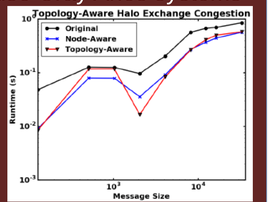
- Use techniques to reduce off-node messages, hop count, and congestion to improve 3-d halo exchange communication costs
- Upcoming tests will optimize PCG solvers and use Cray Aries systems



Using eager over rendezvous protocol can allow more effective overlap of sends and receives for blocking routines



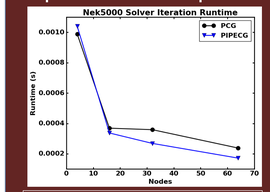
Node-aware communication (Gropp 2018) decreases off-node messages while topology-aware extends node-aware to reduce message distance



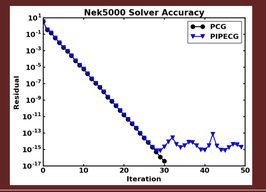
Topology-aware and node-aware methods reduce congestion due to reducing off-node communication

How do PCG Methods Perform in Applications?

- Investigating scalable Krylov solvers, performance analysis tools, and performance optimizations within Nek5000 and Quda applications



PIPECG scales better than PCG in initial Nek5000 tests



PIPECG has reduced accuracy but more than meets desired tolerance for initial Nek5000 tests

- Nek5000 is a computational fluid dynamics application with unstructured grids for CPU-based systems
- Quda is a quantum chromodynamics application with structured grids for GPU-based systems

Key Contributions

- Thesis provides detailed analysis of performance issues limiting Krylov solver performance at scale and provides guidance for improving performance focused around using a class of scalable Krylov solvers
- Provides detailed performance analysis of non-blocking pipelined PCG solvers and develops new PIPE2CG method
- Develops software tools to analyze performance variation and network performance at scale
- Develops performance models capable of accounting for decreased performance at scale and performance variation across multiple runs
- Will provide detailed study of wider range of non-blocking Krylov solvers within Nek5000 and Quda for improving application performance at scale

Publications: Paul Eller and William Gropp, Scalable non-blocking preconditioned conjugate gradient methods, SC '16.
Paul Eller and William Gropp, Performance Modeling Scalable Krylov Solvers for Structured Grid Problems, In Preparation.

Special thanks to Mark Hoemmen, the Exascale Computing Project QCD group led by Richard Brower, Paul Fischer, and Torsten Hoefler for help on parts of this project.

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy award DE-FG02-13ER26138/DE-SC0010049 and in part by the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois.