Extreme Scale Computing

William Gropp
University of Illinois at Urbana-Champaign

Overview

Research Interests:
- Programming models for parallel computing
- Scalable numerical methods
- Extreme scale computing

Current projects
- MPI for petascale
- Scalable Algorithms
- Exascale Computing
- "Blue Waters", the NSF Track 1 Petascale System

Software Projects
- MPICH2 – A High Performance, Scalable, Portable Implementation of MPI
- pnetCDF – A parallel version of the netCDF 3 file format and utilities
- PETSc – A Scalable numerical library for solving large systems of linear and nonlinear equations

Two examples follow

Enhancing Performance of MPI Collective Operations

Paul Sack

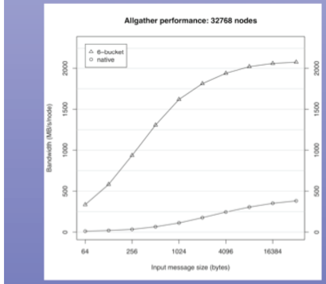
Collective communication and computation are important operations

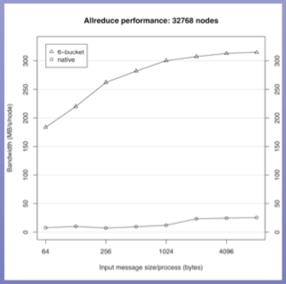
Implementations have often minimized the data moved and are optimal in this sense

Interconnection networks, particularly at extreme scale, are complex and have limited bisection bandwidth compared with a complete graph

Goal: Minimum *time* collective operations

Idea: Additional Data Motion can *reduce* contention in network, by avoiding contention





Addressing Scalability Challenges of Algebraic Multigrid

Illinois: **Hormozd Gahvari** and Luke Olson
LLNL: Martin Schulz and Ulrike Meier Yang
IBM: Kirk Jordan

Goal: Access and enhance the scalability of an "optimal" numerical method for extreme scale systems

Algebraic Multigrid

- **Algebraic multigrid (AMG) allows for the fast solution of large problems by using coarse-grid approximations that involve far fewer points:**
- **Work per unknown remains constant, which is great for supercomputing**

Scalability Challenges

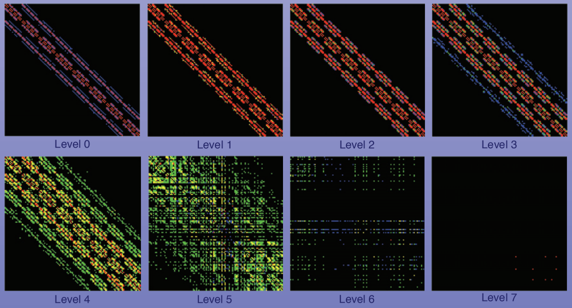
- **AMG scales well on IBM Blue Gene machines, but has difficulties on multicore clusters**
- **Quick comparison between two machines, Hera (multicore Linux cluster at LLNL) and Intrepid (IBM Blue Gene/P at ANL) is highly illustrative**
- **Results here are for one V-cycle on a 3D 7-point Laplace problem on 128 processors with 62,500 points and one MPI process per core:**

Hera:
- **Slowdown on levels 3 → 5.**
- **Level 5, with only 212 points, is nearly as slow as level 0!**

Level	Unknowns	Hera	Intrepid
0	8,000,000	2.30×10^{-2} s	5.19×10^{-2} s
1	614,521	8.28×10^{-3} s	1.42×10^{-2} s
2	120,607	6.52×10^{-3} s	4.38×10^{-3} s
3	13,643	1.24×10^{-2} s	1.53×10^{-3} s
4	1,509	1.87×10^{-2} s	1.27×10^{-3} s
5	212	2.02×10^{-2} s	1.73×10^{-3} s
6	23	2.75×10^{-3} s	5.36×10^{-4} s
7	3	5.44×10^{-5} s	2.22×10^{-5} s
total		9.18×10^{-2} s	7.55×10^{-2} s

Intrepid:
- **Slowdown on level 5.**

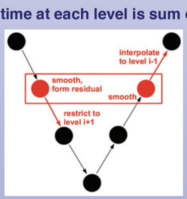
- **This is driven by increasing amounts of interprocessor communication:**



This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Performance Model

- **To help us understand what we are seeing, develop performance model for AMG solve cycle**
- **Baseline model: α-β (latency-bandwidth) with parameters**
  - P – number of processes
  - C_i – number of grid points in level i
  - L – maximum grid index
  - s_i, $\underline{s}_i$ – average number of nonzeros per row in solve and interpolation operators, respectively
  - p_i, $\underline{p}_i$ – maximum number of sends per active process in solve and interpolation operators, respectively
  - n_i, $\underline{n}_i$ – maximum number of elements sent per active process in solve and interpolation operators, respectively
  - t_i – time per floating-point operation on level i
- **Runtime at each level is sum of:**

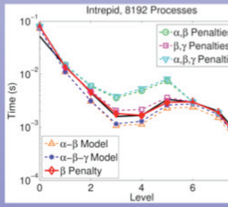


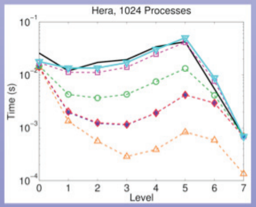
$$T^i_{smooth} = 6(C_i/P)s_i t_i + 3(p_i\alpha + n_i\beta)$$

$$T^i_{coarsen} = \begin{cases} 2(C_{i+1}/P)\underline{s}_i t_i + \underline{p}_i\alpha + \underline{n}_i\beta & \text{if } i < L \\ 0 & \text{if } i = L \end{cases}$$

$$T^i_{prolong} = \begin{cases} 0 & \text{if } i = 0 \\ 2(C_{i-1}/P)\underline{s}_{i-1} t_i + \underline{p}_{i-1}\alpha + \underline{n}_{i-1}\beta & \text{if } i > 0 \end{cases}$$

- **Take architectural features into account with penalties:**
  - Distance of communication: add γ term (charge distance = network diameter to each message)
  - Lower effective bandwidth: multiply β by Hardware Bandwidth/MPI Bandwidth
  - Multicore latency penalty: multiply α by $\sqrt{cP_i/P}$ (c = cores per node, P_i = no. active processes at level i)
  - Multicore distance penalty: multiply γ by $\sqrt{cP_i/P}$
- **Results spotlight impact of architecture on performance:**





Intrepid:
- Not too much degradation on coarse grids
- Only limited bandwidth penalty applies

Hera:
- Massive degradation on coarse grids
- All penalties apply – lots of network contention and switching delays from multicore nodes (16 cores/node)

- **Future machines will be more multicore…**