

Been There, Done That:

Lessons Learned from SMP Computing

Michael A. Heroux
Sandia National Laboratories



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.



Outline

- A brief (personal) history of SMP computing.
- Can we use shared memory parallel (SMP) only?
- Can we use distributed memory parallel (DMP) only?
- Possibilities for using SMP within DMP.
 - ◆ Performance characterization for preconditioned Krylov methods.
 - ◆ Possibilities for SMP with DMP for each Krylov operation.
- Insert: Brief Overview of Petra Object Model.
- Implications for multicore chips.
- About MPI.

Personal SMP History

Chapter 1: Cray YMP/C90/J90/T90

(Aside: Cray did *not* know how to name machines)

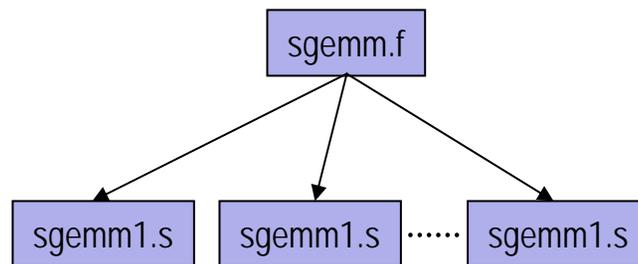
- Macrotasking/Microtasking/**Autotasking**.
- 7 out 8 speedup (or less) typical.
- Speed-down for 16 out of 16 or 32 out of 32.
- Only the “rich” could afford to use it: Job parallel was best.
- Complicated fast kernels programming.

Complicated Fast Kernels: Dense Matrix-Matrix Multiply (SGEMM)

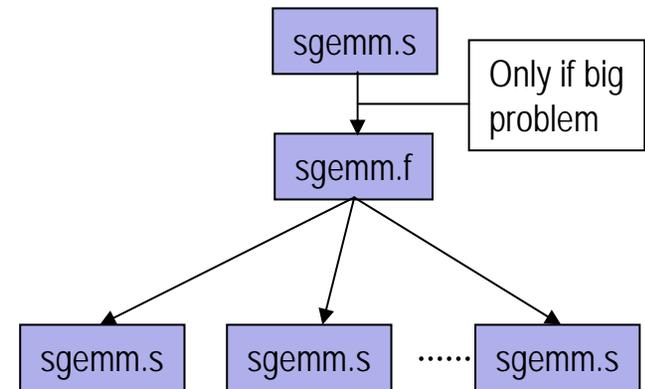
Original Serial



Original Parallel



Final Parallel



- Custom assembler
- Peak speed even at small sizes

- Fortran driver (asm too hard)
- Decompose problem
- Call one or more instances of renamed asm kernel.
- Huge hit for small sizes.
- Awful for FEM codes.

- Asm *with quick size check*.
- Fortran driver (only if big)
- Decompose problem
- Call one or more instances of asm kernel.
- asm kernel parallel-aware.
- Fast but complicated.

Personal SMP History

Chapter 2: SGI Origin

- Distributed-shared memory (DSM) architecture:
 - ◆ Physically distributed.
 - ◆ Logically shared.
- First efforts: Identify and parallelize “hotspots” (OpenMP).
 - ◆ “First-touch” page placement.
 - ◆ Modest gains for some codes.
 - ◆ Disastrous for many.
- Second efforts: Parallel OpenMP throughout.
 - ◆ LOTS of work.
 - ◆ Parallelism limited to largest SMP machine.
 - ◆ Still issues of task placement/migration: Complex runtime env.
- Best efforts: MPI using good MPI for DSM.
 - ◆ Scalable and portable.
 - ◆ Shared memory is great for large-memory read-only startup tasks.

Personal SMP History

Chapter 3: Multi-node with SMP nodes

- One or more SMP nodes connected by dedicated network.
- SMP only? MPI only? Hybrid?

SMP-only Possibilities

- Question: Is it possible to develop a *scalable* parallel application using only shared memory parallel programming?

SMP-only Observations

- Developing a scalable SMP application requires as much work as DMP:
 - ◆ Still must determine ownership of work and data.
 - ◆ Inability to assert placement of data on DSM architectures is big problem, not easily fixed.
 - ◆ Study after study illustrates this point.
- SMP application requires SMP machine:
 - ◆ Much more expensive per processor than DMP machine.
 - ◆ Poorer fault-tolerance properties.
- Number of processor usable by SMP application is limited by minimum of:
 - Operating System and Programming Environment support.
 - Global Address Space.
 - Physical processor count.
- ◆ Of these, the OS/Programming Model is the real limiting factor.

SMP-only Possibilities

- Question: Is it possible to develop a *scalable* parallel application using only shared memory parallel programming?
- Answer: No.

DMP-only Possibilities

- Question: Is it possible to develop a scalable parallel application using only distributed memory parallel programming?
- Answer: Don't need to ask. Scalable DMP applications are clearly possible to $O(100K-1M)$ processors.
- Thus:
 - ◆ DMP is required for scalable parallel applications.
 - ◆ Question: Is there still a role for SMP within a DMP application?

SMP-Under-DMP Possibilities

- Can we benefit from using SMP within DMP?
 - ◆ Example: OpenMP within an MPI process.

Case Study: Linear Equation Solvers

- Sandia has many engineering applications.
- A large fraction of newer apps are *implicit* in nature:
 - ◆ Requires solution of many large nonlinear systems.
 - ◆ Boils down to many sparse linear systems.
- Linear system solves are large fraction of total time.
 - ◆ Small as 30%.
 - ◆ Large as 90+%.
- Iterative solvers most commonly used.
- Iterative solvers have small handful of important kernels.
- We focus on performance issues for these kernels.
 - ◆ Caveat: These parts do not make the whole, but are a good chunk of it...

Problem Definition

- A frequent requirement for scientific and engineering computing is to solve:

$$Ax = b$$

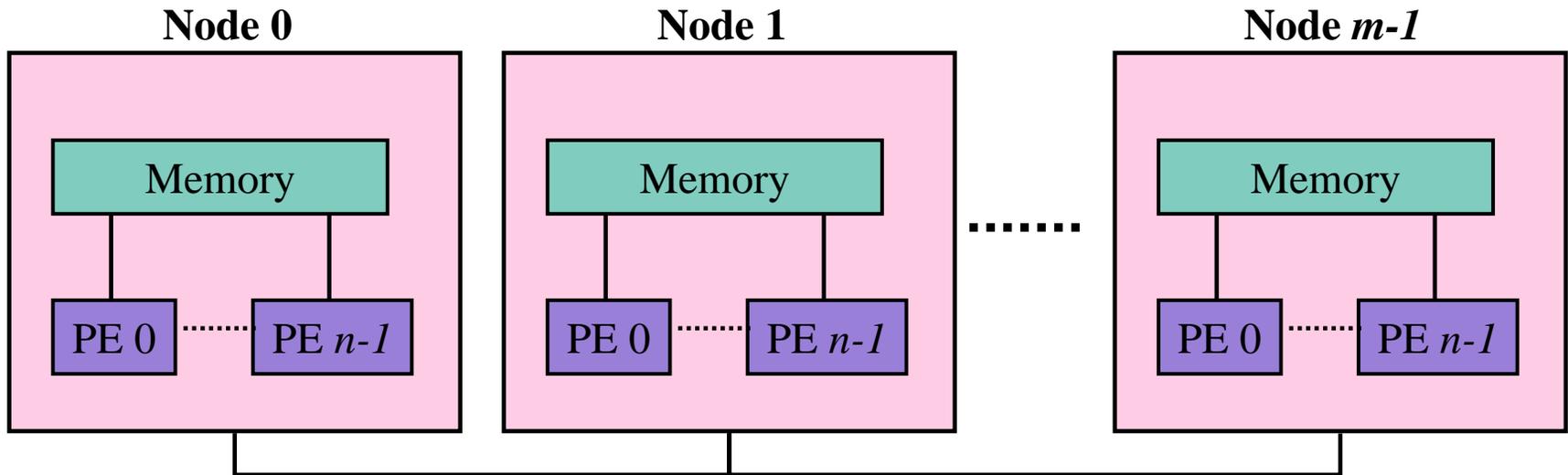
where A is a known large (sparse) matrix,
 b is a known vector,
 x is an unknown vector.

- Goal: Find x .
- Method:
 - ◆ Use Preconditioned Conjugate Gradient (PCG) method,
 - ◆ Or one of many variants, e.g., Preconditioned GMRES.
 - ◆ Called Krylov Solvers.

Performance Characteristics of Preconditioned Krylov Solvers

- The performance of a parallel preconditioned Krylov solver on any given machine can be characterized by the performance of the following operations:
 - ◆ Vector updates: $y = \alpha x + y$
 - ◆ Dot Products: $\delta = x^T y$
 - ◆ Matrix multiplication: $y = Ax$
 - ◆ Preconditioner application: $y = M^{-1}x$
- What can SMP within DMP do to improve performance for these operations?

Machine Block Diagram



- ◆ Parallel machine with $p = m * n$ processors,
 - m = number of nodes.
 - n = number of shared memory processors per node.
- ◆ Consider
 - p MPI processes vs.
 - m MPI processes with n threads per MPI process (nested data-parallel).

Vector Update Performance

- Vector computations are not (positively) impacted using nested parallelism.
 - ◆ These calculations are unaware that they are being done in parallel.
 - ◆ Problems of data locality and false cache line sharing can actually degrade performance for nested approach.
 - Example: What happens if
 - PE 0 must update $x[j]$.
 - PE 1 must update $x[j+1]$ and
 - $x[j]$ and $x[j+1]$ are in the same cache line?
- Note: These same observations hold for FEM/FVM calculations and many other common data parallel computations.

Dot Product Performance

- Global dot product performance can be improved using nested parallelism:
 - ◆ Compute the partial dot product on each node before going to binary reduction algorithm:
 - $O(\log(m))$ global synchronization steps vs. $O(\log(p))$ for DMP-only.
 - ◆ However, same can be accomplished using “SMP-aware” message passing library like LIBSM.
- Notes:
 - ◆ An SMP-aware message passing library addresses many of the initial performance problems when porting an MPI code to SMP nodes.
 - ◆ Reason? Not lower latency of intra-node message but reduced off-node network demand.

Matrix Multiplication Performance

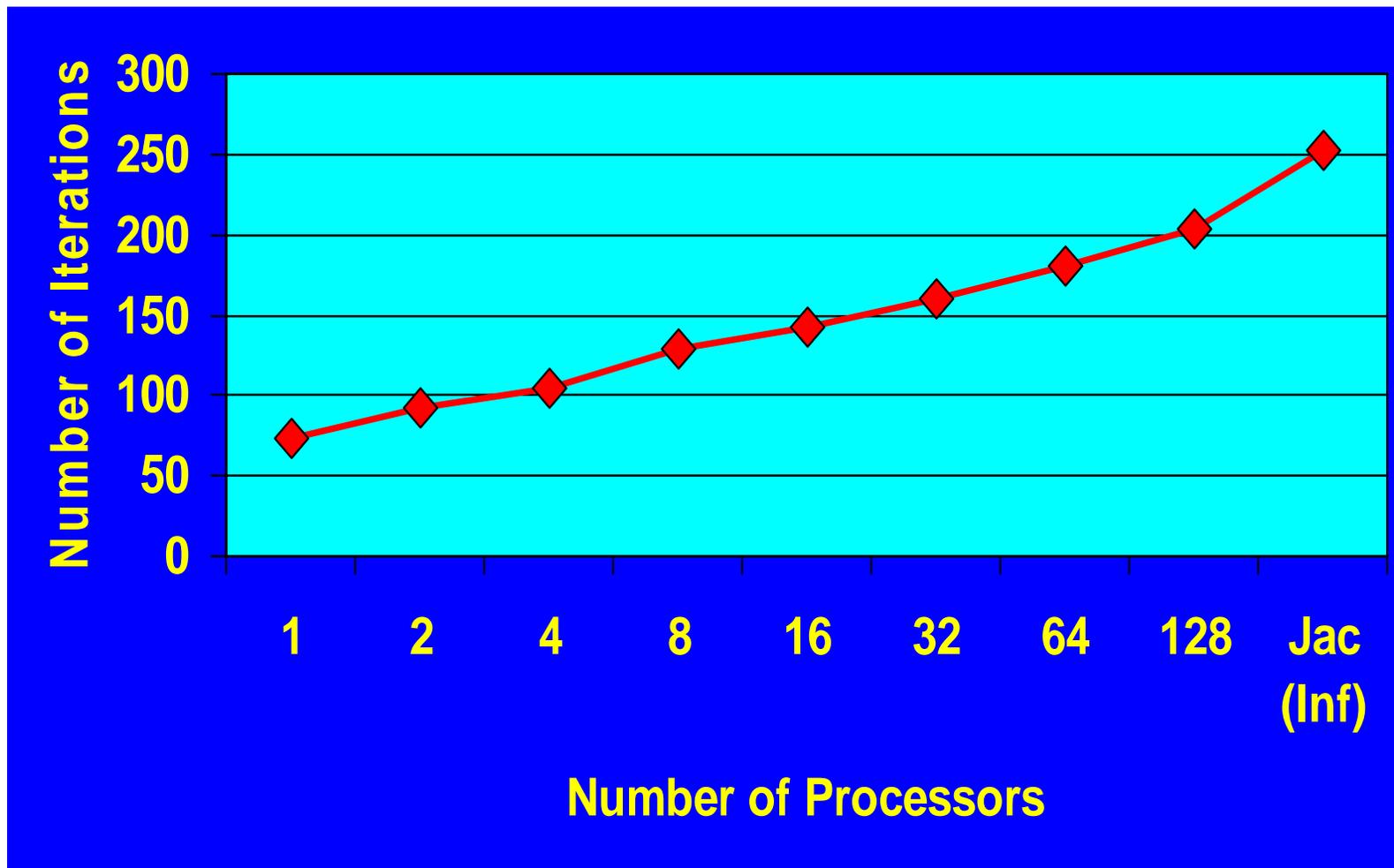
- Typical distributed sparse matrix multiplication requires “boundary exchange” before computing.
- Time for exchange is determined by longest latency remote access.
- Using SMP within a node does not reduce this latency.
- SMP matrix multiply has same cache performance issues as vector updates.
- Thus SMP within DMP for matrix multiplication is not attractive.

Batting Average So Far: 0 for 3

- So far there is no compelling reason to consider SMP within a DMP application.
- Problem: Nothing we have proposed so far provides an essential improvement in algorithms.
- Must search for situations where SMP provides a capability that DMP cannot.
- One possibility: Addressing iteration inflation in (Overlapping) Schwarz domain decomposition preconditioning.

Iteration Inflation

Overlapping Schwarz Domain Decomposition (Local ILU(0) with GMRES)



Using Level Scheduling SMP

- As the number of subdomains increases, iteration counts go up.
- Asymptotically, (non-overlapping) Schwarz becomes diagonal scaling.
- But note: ILU has parallelism due to sparsity of matrix.
- We can use parallelism within ILU to reduce the inflation effect.

Defining Levels

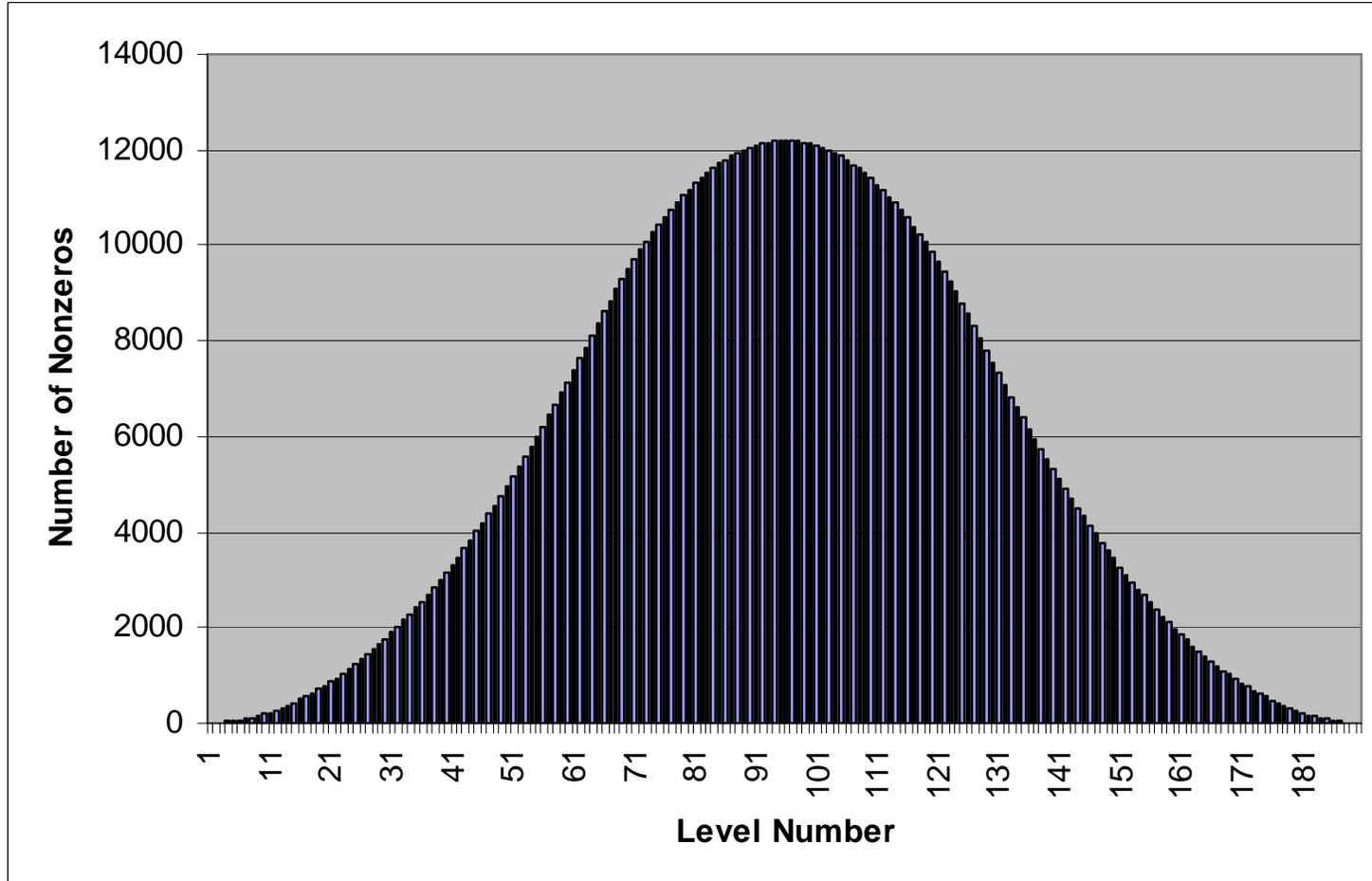
$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & 0 & 1 & 0 & 0 \\ 0 & 0 & l_{43} & 1 & 0 \\ l_{51} & 0 & l_{53} & 0 & 1 \end{bmatrix}$$

Solve $Ly = x$.

Some Sample Level Schedule Stats

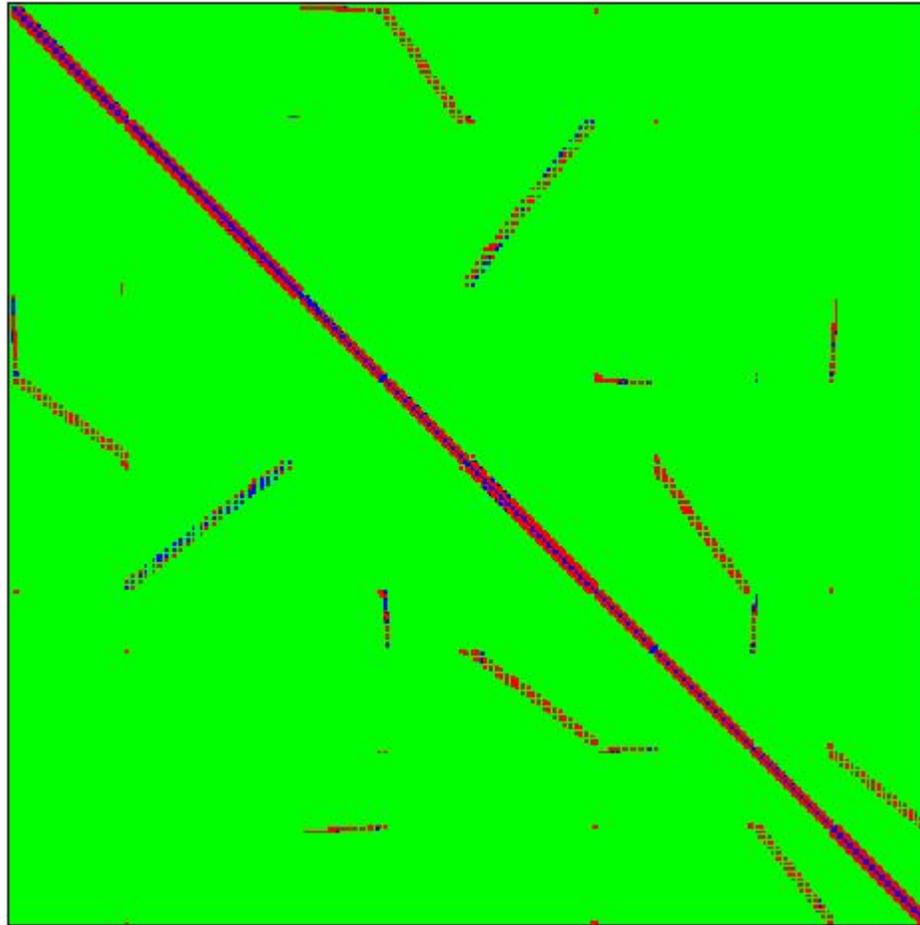
Linear FE basis fns on 3D grid

Avg nnz/level = 5500, Avg rows/level = 173.



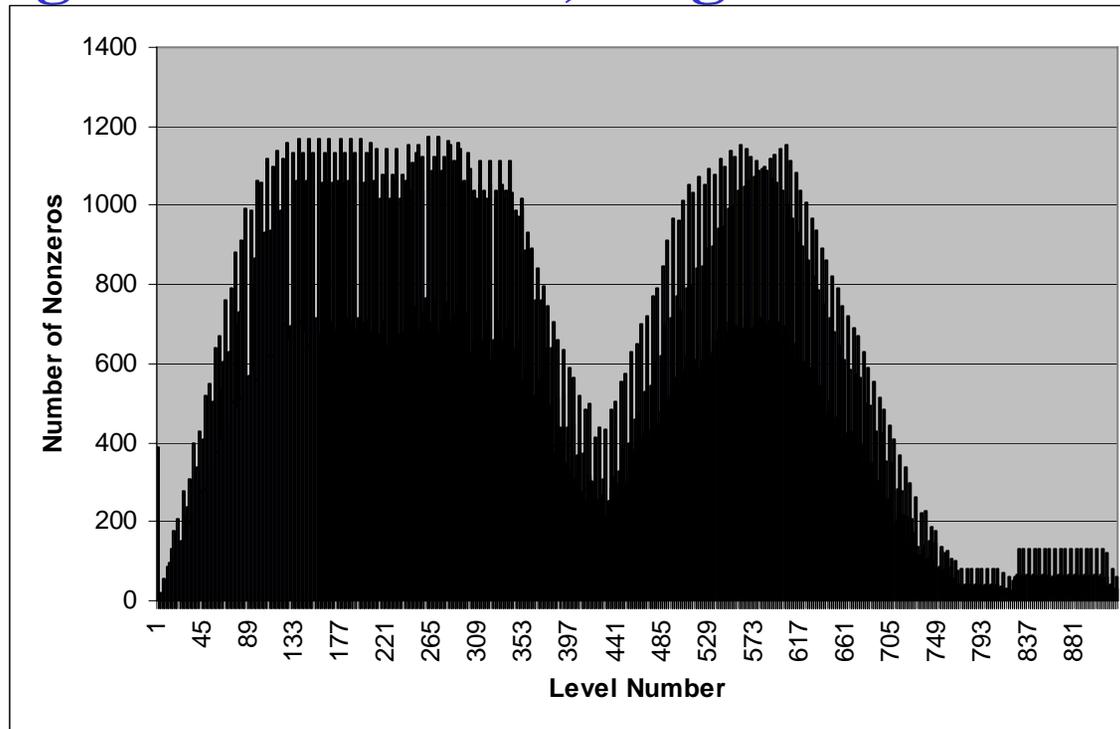
Linear Stability Analysis Problem

Unstructured domain, 21K eq, 923K nnz



Some Sample Level Schedule Stats

Unstructured linear stability analysis problem
Avg nnz/level = 520, Avg rows/level = 23.



Improvement Limits

- Assume number of PEs per node = n .
- Assume speedup for level scheduled F/B solve matches speedup of n MPI solves on same node.
- Then performance improvement is

$$\frac{\text{Number of iterations for } p \text{ domains}}{\text{Number of iterations for } m \text{ domains}}$$

- For previous graph and $n = 8$, $p = 128$ ($m = 16$), ratio = $203/142 = 1.43$ or 43%.

Practical Limitations

- Level scheduling speedup is largely determined by the cost of synchronization on a node.
 - ◆ F/B solve requires a synchronization after each level.
 - ◆ On machines with good hardware barrier, this is not a problem and excellent speed up can be expected.
 - ◆ On other machines, this can be a problem.

Reducing Synchronization Restrictions

- Use a flexible iterative method such as FGMRES.
 - ◆ Preconditioner at each iteration need not be the same, thus no need for sync'ing after each level.
 - ◆ Level updates will still be approximately obeyed.
 - ◆ Computational and communication complexity is identical to DMP-only F/B solve.
 - ◆ Iteration counts and cost per iteration go up.
- Multi-color reordering:
 - ◆ Reorder equations to increase level-set sizes.
 - ◆ Severe increase in iteration counts.
- Our motto:

The best parallel algorithm is the best parallel implementation of the best serial algorithm.

SMP-Under-DMP Possibilities

- Can we benefit from using SMP within DMP?
- Yes, but:
 - ◆ Must be able to take advantage of fine-grain shared memory data access.
 - ◆ In a way not feasible for MPI-alone.
- Even so: Nested SMP-Under-DMP is *very* complex to program.
- Most people answer, “It’s not worth it.”

Summary So Far

- SMP alone is insufficient for scalable parallelism.
- DMP alone is certainly sufficient, but can we improve by selective use of SMP within DMP?
- Analysis of key preconditioned Krylov kernels gives insight into possibilities of using SMP with DMP, and results can be extended to other algorithms.
- Most of the straight-forward techniques for introducing SMP into DMP will not work.
- Level scheduled ILU is one possible example of effectively using SMP within DMP (not always satisfactory).
- Most fruitful use of SMP within DMP seems to have a common theme of allowing multiple processes to have dynamic asynchronous access to large (read-only) data sets.

Implications for Multicore Chips

- MPI-only use of multicore is a respectable option.
 - ◆ May be the ultimate right answer for scalability and ease of programming.
 - ◆ Assumption: MPI is multicore-aware. Not completely true right now.
 - ◆ Helpful: High task affinity. Single program image per chip.
- Flexible, robust multicore kernels will be complicated.
 - ◆ Task parallelism is preferred if available (Media Player/Outlook).
 - ◆ Similar issues as Cray SMP programming:
 - How many cores can (available) or should (problem size) be used?
 - Illustrates difference between hetero/homo-geneous multicore.
- Data placement issues similar to Origin (if # CMP > 1).
 - ◆ Task affinity important.
 - ◆ Mitigating factor:
 - On-chip data movement is at processor speeds.
 - Shared cache should help?

About MPI

- Uncomfortable defending MPI: But...
- Can Parallel Programming be Easy?
 - ◆ Memory management is key.
 - ◆ Is it bad that parallel programming hard? Isn't programming hard?
- La-Z-Boy principle* impacts MPI adoption also.
- MPI is not that hard, does not impact majority of code.
- Real problem: Serial to MPI transition is not gradual.
- Can the mass market produce new parallel language quickly? Not convinced.
- Can develop MPI-based code that is portable, today!
- Still hope for better.

* Don't need to write parallel code because uni-processors are getting faster (No longer applies to next-gen processors).