Trends of Scientific Computation

Jack Dongarra

University of Tennessee
and
Oak Ridge National Laboratory

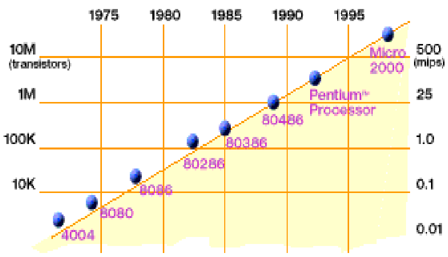
1

Numerical Libraries

- **20 years ago**
  - 1 Mflop/s - Scalar based
    - » Linpack, Level 1 BLAS, loop unrolling
- **10 years ago**
  - 1 Gflop/s - Vector & SMP computing, cache aware
    - » LAPACK, Level 2 & 3 BLAS, block partitioned, latency tolerant
- **Today**
  - 1 Tflop/s - Highly parallel, network based, message passing
    - » ScaLAPACK, data decomposition, communication/computation
- **10 years away**
  - 1 Pflop/s - Many more levels MH, combination/grids&HPC
    - » More adaptive, LT and bandwidth aware, fault tolerant, extended precision, attention to SMP nodes

2

Technology Trends: Microprocessor Capacity

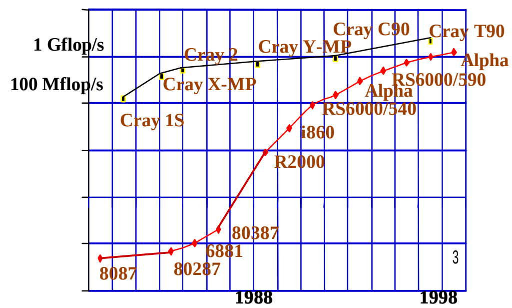


2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful.

Gordon Moore (co-founder of



Directions

- Move toward clusters
  - Distributed Memory
  - deep memory hierarchy
- Efficiency of message passing and data parallel programming
  - Helped by standards efforts such as MPI, PVM, and OpenMP

5

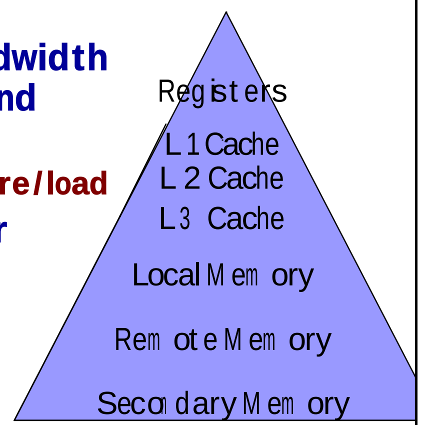
Challenges in Developing Distributed Memory Libraries

- How to integrate software?
  - Until recently no standards
  - Many parallel languages
  - Various parallel programming models
  - Assumptions about the parallel environment
    - granularity
    - topology
    - overlapping of communication/computation
    - development tools
- Where is the data
  - Who owns it?
  - Opt data distribution
- Who determines data layout
  - Determined by user?
  - Determined by library developer?
  - Allow dynamic data dist.
  - Load balancing

6

Performance Issues - Cache &Bandwidth

- Performance instability
  - Small changes may cause dramatic changes in delivered performance.
- Latency tolerant and bandwidth parsimonious algorithms and software are critical
  - Recompute rather than store/load
- Need to help the compiler
- Have a hard time today getting performance
  - Only going to get harder



History of Block Partitioned Algorithms

- Early algorithms involved use of small main memory using tapes as secondary storage.
- Recent work centers on use of vector registers, level 1 and 2 cache, main memory, and “out of core” memory.

8

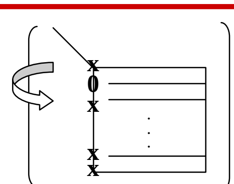
Blocked Partitioned Algorithms

- LU Factorization
- Cholesky factorization
- Symmetric indefinite factorization
- Matrix inversion
- QR, QL, RQ, LQ factorizations
- Form Q or Q^TC
- Orthogonal reduction to:
  - (upper) Hessenberg form
  - symmetric tridiagonal form
  - bidiagonal form
- Block QR iteration for nonsymmetric eigenvalue problems

LAPACK and ScaLAPACK are build on these

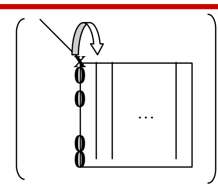
9

Gaussian Elimination



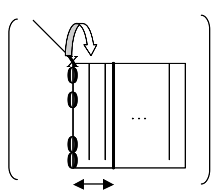
Standard Way

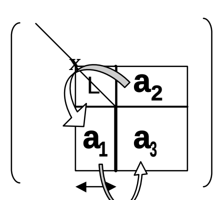
subtract a multiple of a row



LINPACK

apply sequence to a column





$a_2 = L^{-1} a_2$

$a_3 = a_3 - a_1 * a_2$

10

LAPACK

apply sequence to nb then apply nb to rest of matrix

Gaussian Elimination via a Recursive Algorithm

F. Gustavson and S. Toledo

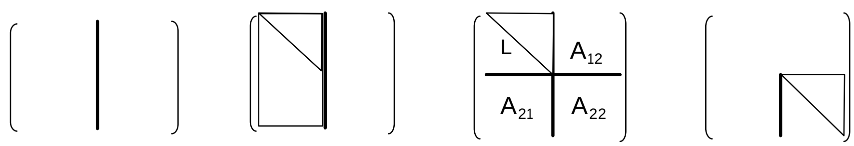
LU Algorithm:

1: Split matrix into two rectangles (m x n/2)
   if only 1 column, scale by reciprocal of pivot & return

2: Apply LU Algorithm to the left part

3: Apply transformations to right part
   (triangular solve $A_{12} = L^{-1}A_{12}$ and
   matrix multiplication $A_{22}=A_{22} - A_{21}*A_{12}$)

4: Apply LU Algorithm to right part



11

Dense recursive factorization

? **The algorithm:**

function rlu(A)

begin

rlu(A_{11});	recursive call
A_{21}? $A_{21} \cdot \mathbf{U}^{-1}(A_{11})$;	xTRSM() on upper triangular submatrix
A_{12} ? $L_1^{-1}(A_{11}) \cdot A_{12}$;	xTRSM() on lower triangular submatrix
A_{22} ? A_{22}-$A_{21} \cdot A_{12}$;	xGEMM()
rlu(A_{22});	recursive call

end.

? **Performs BLAS Level 3 operations on large matrix blocks ($n^2/2$, $n^2/4$, $n^2/8$, …)**

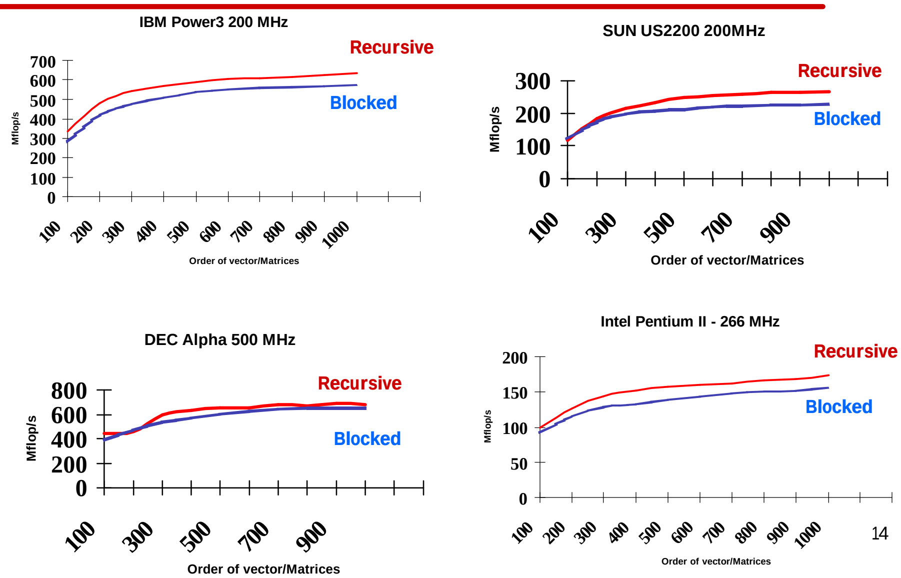
12

Recursive Factorizations

- Larger matrix-matrix operations performed
- Automatic Blocking
  - Drive the recursion down to 1
  - No blocksize needed
- Applies to LU, Cholesky and QR factorization
  - QR needs some care because of triangular matrix in block Householder
- Unclear if ideas applies to 2-sided algorithms (reduction for eigenvalue problem)

13

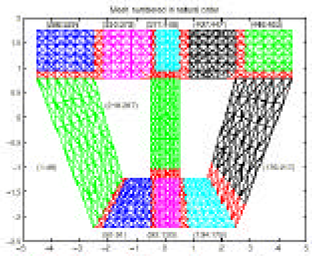
LU Performance Recursive/Blocked

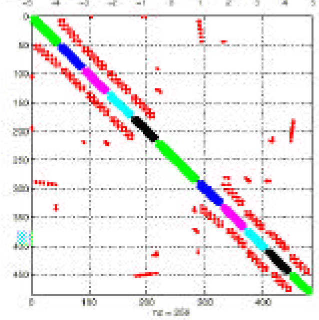


14

Sparse Gaussian Elimination

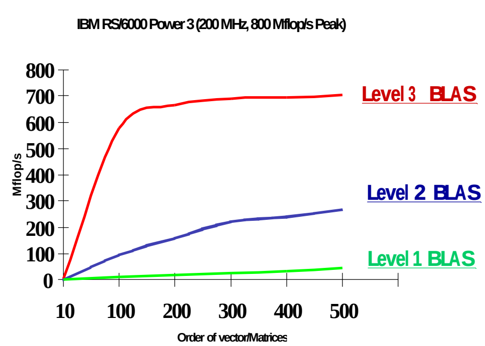
- Sparse Elimination much more complicated than dense
  - Only store and compute with nonzero entries in A
  - More nonzero “fill-in” as elimination proceeds
  - Hard to organize the algorithm so that most time not spent in traversing data structures, rather than floating point.
  - Fill-in, parallelizability, cost depends strongly on order of equations.
  - Many parallelization strategies, depending on structure.





Sparse Gaussian Elimination

- Want to exploit block operations in sparse matrix
- Designed to exploit memory hierarchy
  - Serial Supernode-panel organization permits "BLAS 2.5" performance
  - Up to 40% of machine peak on large sparse matrices on IBM RS6000/590, MIPS R8000, 25% on Alpha 21164



16

Sparse recursive factorization algorithm

? **Main problems:**

- no large blocks that could be passed to xGEMM()
- no fast xGEMM() counterpart for sparse data structures
- fill reducing ordering required
- pivoting is harder on sparse data structures

? **Solutions**

- create dense blocks for dense xGEMM() by storing some zero values explicitly:

$$\left[\begin{array}{ccc|ccc} x & & x & & & \\ & x & & & & \\ & & x & & & \\ \hline & & & x & & \\ & & & & x & \\ & & & x & & x \end{array}\right] \quad ? \quad \left[\begin{array}{ccc|ccc} x & 0 & x & & & \\ 0 & x & 0 & & & \\ 0 & 0 & x & & & \\ \hline & & & x & 0 & 0 \\ & & & 0 & x & 0 \\ & & & x & 0 & x \end{array}\right]$$

17

Sparse Recursive Factorization Algorithm

? **Solutions - continued**

- fast sparse xGEMM() is two-level algorithm
  - » recursive operation on sparse data structures
  - » dense xGEMM() call when recursion reaches single block

? **fill reducing ordering can be applied before recursive algorithm**

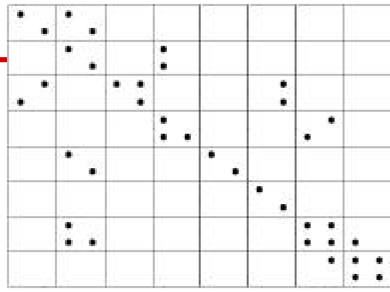
? **no partial pivoting**

- use iterative improvement or
- pivot only within blocks

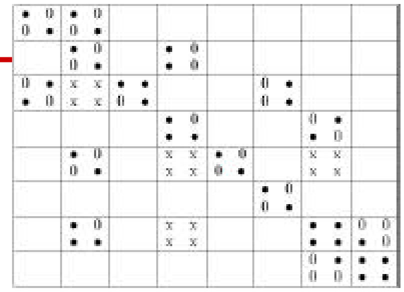
18

Recursive storage conversion steps

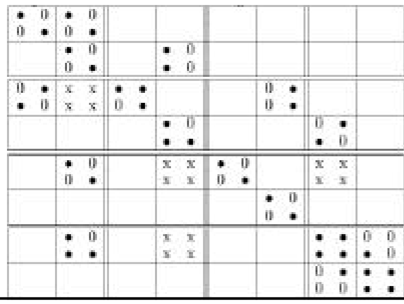
Matrix divided into 2x2 blocks



Matrix with explicit 0's and fill-in



Recursive algorithm division lines



? - original nonzero value
0 - zero value introduced due to blocking
x - zero value introduced due to fill-in

19

Many Sparse Direct Solvers

	Real	Complex	f77	c	c++	Seq	Dist		SPD	Gen
MA28	X		X			X			X	X
MFACT	X			X		X			X	
MP_SOLVE	X	X	X				M			X
PSPASES	X		X	X			M		X	
SPARSE	X	X		X		X			X	X
SPARSEQR	X			X	X	X			X	X
SPOOLS	X	X		X		X	M		X	X
SuperLU	X	X	X	X		X			X	X
UMFPACK	X	X	X			X			X	X
Y12M	X		X			X			X	X

www.netlib.org/utk/people/JackDongarra/la-sw.html

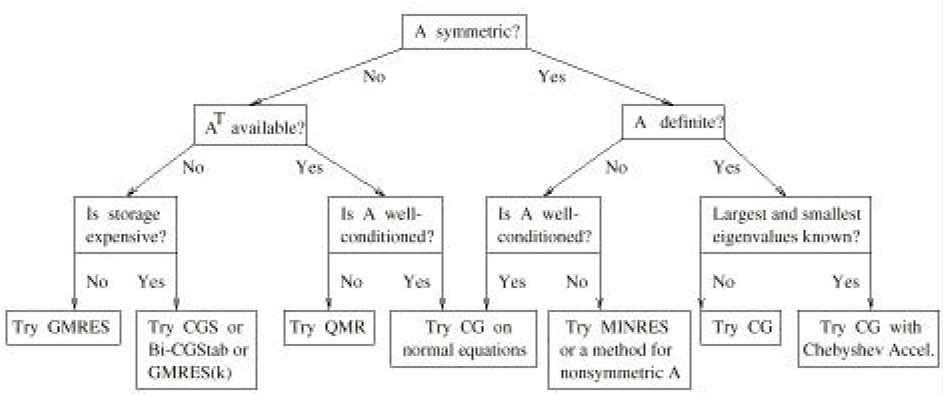
20

Iterative Solvers - Krylov Subspace Methods

- ? **Iterative methods for**
  - **Ax = b and Ax = ? x**
- ? **Ingredients**
  - **Matrix vector multiplication Ax, possibly $A^t z$**
  - **Dot products, Saxpy's, Scaling vectors**
  - **"Preconditioner", M, such that MA is better conditioned than A**
- ? **Many methods with different properties**
  - **For Ax = b: CG, GMRES, QMR, …**
  - **For Ax = ? x: Lanczos, Arnoldi, …**
- ? **Many preconditioners: Jacobi, SOR, ILU, Domain Decomposition, Multigrid, …**
- ? **Best choice problem dependent, no easy answer**

21

Decision Tree



- ? **"Templates for the Solution of Linear Systems"**
  - **Book with short description of each method & advice**
  - **Pseudocode + Matlab, Fortran, and C**
  - **http://www.netlib.org/templates/Templates.html**

22

www.netlib.org/utk/people/JackDongarra/la-sw.html

Iterative Solvers

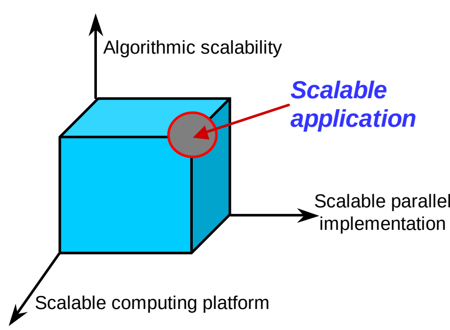
	Real	Complex	f77	c	c++	Seq	Dist		SPD	Gen	SPD	Gen
BILUM	X		X			X					X	X
BlockSolve95	X		X	X	X		M				X	X
BPKIT	X		X	X	X						P	P
DML++	X		X	X	X	X					X	X
ISIS++	X				X		M				X	X
ITPACK	X		X			X					X	X
LASPack	X			X		X					X	X
PARPRE	X			X			M				P	P
PCG	X		X	X	X		P				X	
PETSc	X	X	X	X		X	M				X	X
PIM	X	X	X			X	M/P				X	X
P-SparsLIB	X		X				M					X
QMRPACK	X	X	X			X					X	X
SPLIB	X		X			X					X	X
SPOOLES	X	X		X		X	M		X	X	X	X
Templates	X		X	X		X					X	X

23

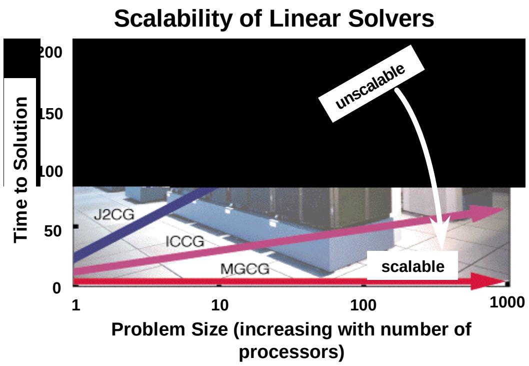
Today Developing Application Codes that Scale to Thousands of Processors

- **Certain applications needs drive development of new codes**
  - **3D, high-resolution (1000x)**
  - **better physics (100x)**
- **Math & CS research is essential**
  - **need 10-100X speedup from smarter algorithms**
  - **using modern software development & engineering practices**
  - **investment in algorithm research leverages huge hardware investment**

Overall scalability requires algorithmic, parallel, and platform scalability

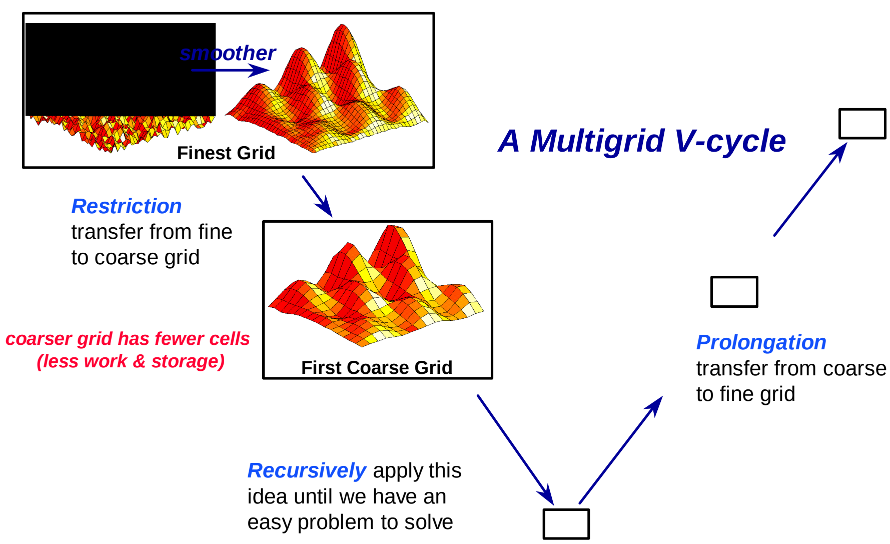


Scalable Algorithms are the Key to Terascale Simulation



- ? Algorithmic scalability is *independent* of parallel scalability
- ? An iterative method is scalable if the number of iterations required for convergence does not depend on problem size
- ? An algorithm implementation is scalable if the time-to-solution is constant as problem size increases with machine size

Scalable Parallel Multigrid Methods



Numerical Algorithms and Software

- Numerical computing will be adaptive, iterative, exploratory, and intelligent.
- Determinism in numerical computing will be gone.
  - After all, its not reasonable to ask for exactness in numerical computations.
  - Audibility of the computation, reproducibility at a cost
- Importance of floating point arithmetic will be undiminished.
  - 16, 32, 64, 128 bits and beyond.
  - Standards being developed
- New methods, multipole methods and their descendants will be ubiquitous.
- Standards are critical, need to evolve

27

Major Challenge - Adaptivity

- These characteristics have major implications for applications that require performance guarantees.
- Adaptivity is a key so applications can function appropriately...
  - as resource utilization and availability change,
  - as processors and networks fail,
  - as old components are retired,
  - as new systems are added, and
  - as both software and hardware on existing systems are updated and modified.

28

Conclusions

- Numerical linear algebra major activity in applied mathematics
- Uses tools from CS, Applied Math, and Pure Math.
- Constant stream of new applications demands new algorithms.
- More information see:

www.netlib.org/utk/people/JackDongarra/la-sw.html

29