Trends of Scientific Computation

Jack Dongarra

University of Tennessee
and
Oak Ridge National Laboratory

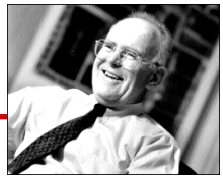
1

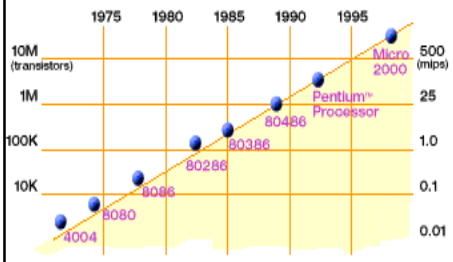
Numerical Libraries

- 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

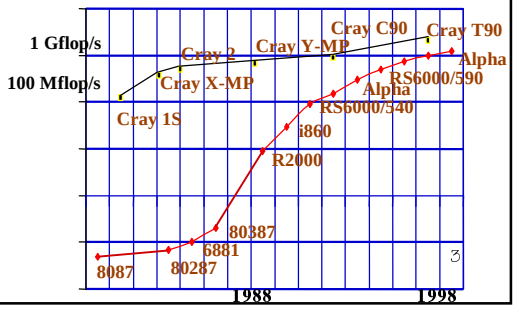




2X transistors/Chip Every 1.5 years
Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful.

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.



3

The Maturation of Highly Parallel Technology

- Affordable parallel systems with distributed memory are the architecture of choice for price performance.
- Reliability of systems is greatly improved.
- Third-party scientific and engineering applications are appearing.
- Business applications are appearing.
- Commercial customers, not just research labs, are acquiring systems.

4

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

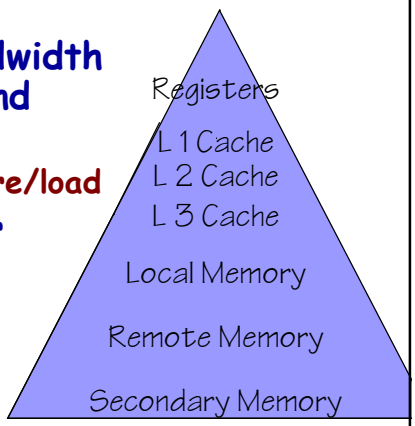
Challenges in Developing Distributed Memory Libraries

- 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

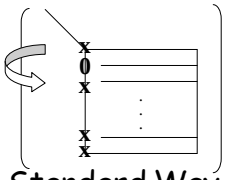
Blocked Partitioned Algorithms

- 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

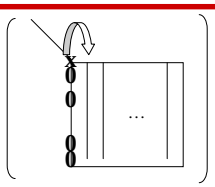
LAPACK and ScaLAPACK are build on these

9

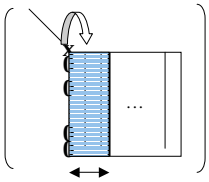
Gaussian Elimination



Standard Way
subtract a multiple of a row

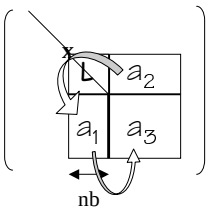


LINPACK
apply sequence to a column



apply sequence to nb

LAPACK



$a_2 = L^{-1} a_2$
$a_3 = a_3 - a_1 * a_2$

then apply nb to rest of matrix

10

Gaussian Elimination via a Recursive Algorithm

F. Gustavson and S. Toledo

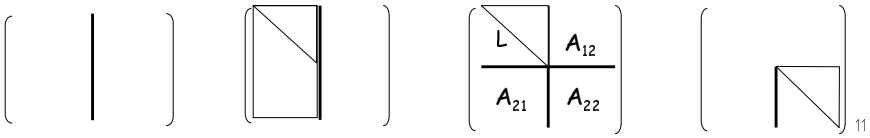
LU Algorithm:

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
$A_{21} \leftarrow A_{21} \bullet \mathbf{U}^{-1}(A_{11})$;	xTRSM() on upper triangular subma
$A_{12} \leftarrow L_1^{-1}(A_{11}) \bullet A_{12}$;	xTRSM() on lower triangular submat
$A_{22} \leftarrow A_{22}-A_{21} \bullet A_{12}$;	xGEMM()
rlu(A_{22});	recursive call

end.

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

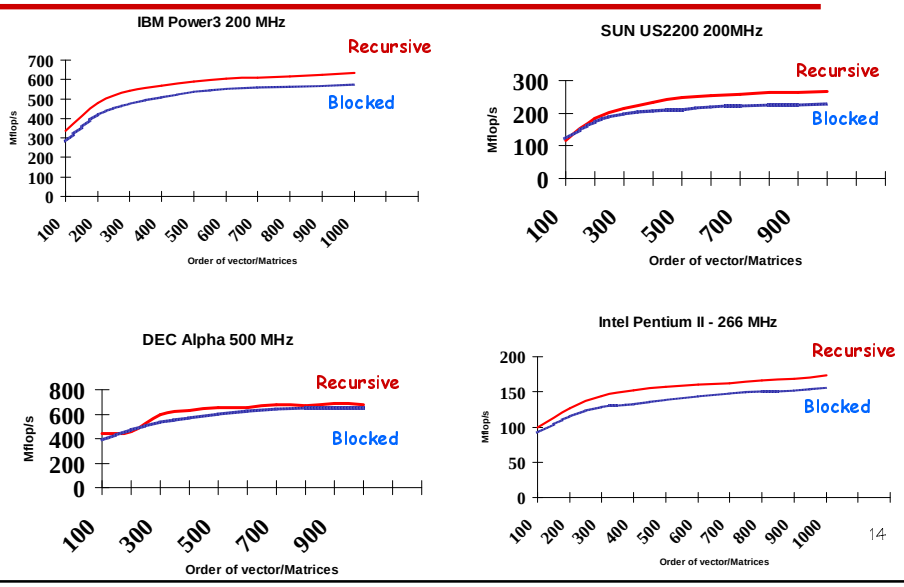
12

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

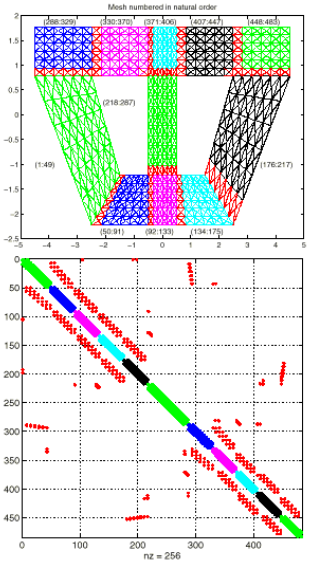
LU Performance Recursive/Blocked



14

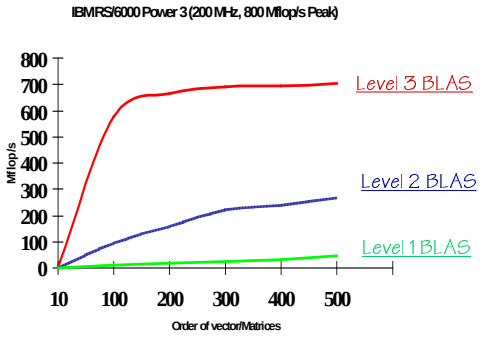
Sparse Gaussian Elimination

- 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

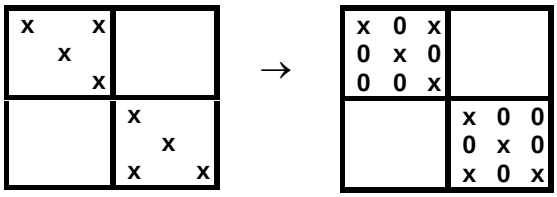
- Want to exploit block operations in sparse matrix
- Designed to exploit memory hierarchy
  - Serial Supernode-panel organization permits "BLAS 2.5" performance
  - Up to 40% of machine peak on large sparse matrices on IBM RS6000/590, MIPS R8000, 25% on Alpha 21164



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:



17

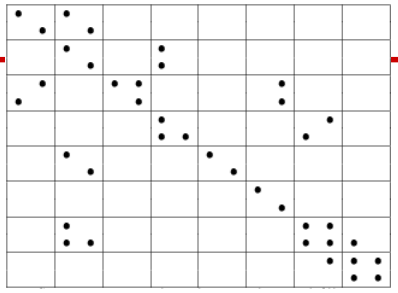
Sparse Recursive Factorization Algorithm

- 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

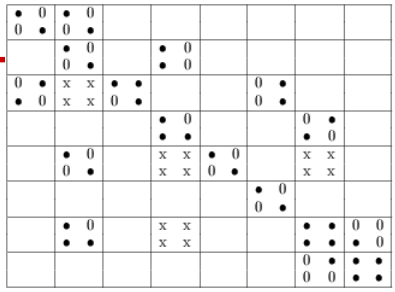
18

Recursive storage conversion steps

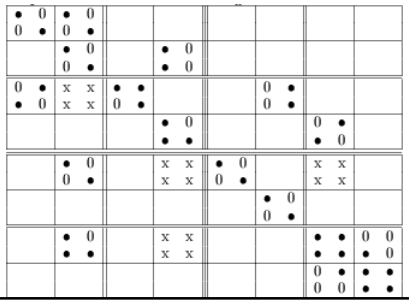
Matrix divided into 2x2 blocks



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

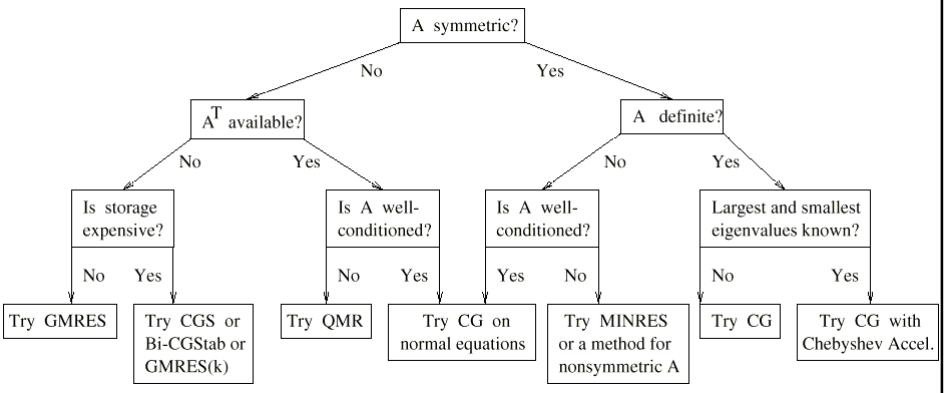
20

Iterative Solvers - Krylov Subspace Methods

- Iterative methods for
  - $Ax = b$ and $Ax = \lambda x$
- 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 $Ax = \lambda x$: 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

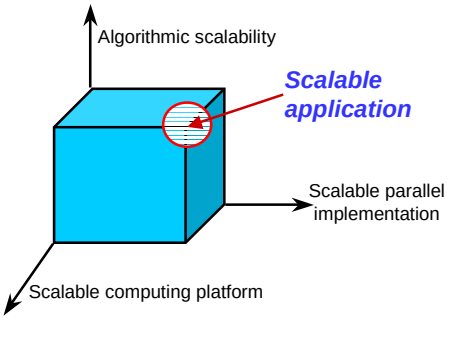
	Real	Complex	f77	c	c++	Seq	Dist		SPD	Gen	SPD	Gen
BILUM	X		X			X					X	X
BlockSolve95	X		X	X	X		M				X	X
BPKIT	X		X	X	X						P	P
IML++	X		X	X	X	X					X	X
ISIS++	X				X		M				X	X
ITPACK	X		X			X					X	X
LASPack	X			X		X					X	X
PARPRE	X			X			M				P	P
PCG	X		X	X	X		P				X	
PETSc	X	X	X	X		X	M				X	X
PIM	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

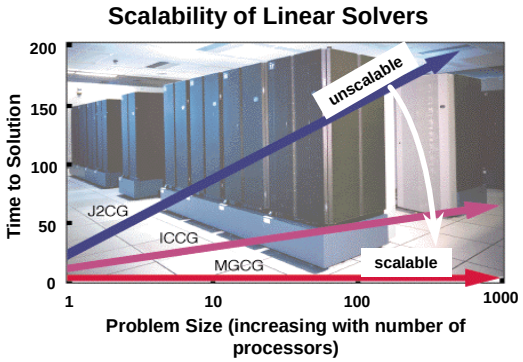
Today Developing Application Codes that Scale to Thousands of Processors

- 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

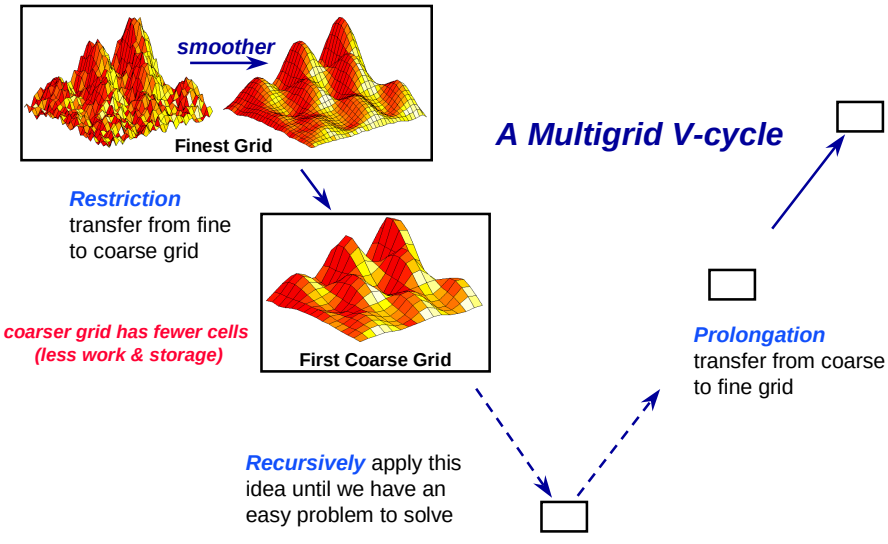


Scalable Algorithms are the Key to Terascale Simulation



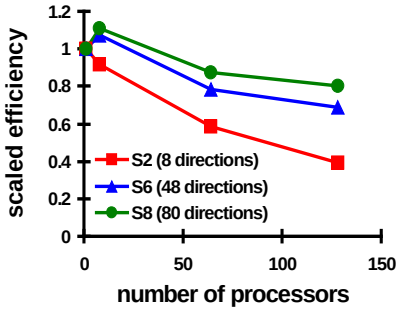
- Algorithmic scalability is *independent* of parallel scalability
- An iterative method is scalable if the number of iterations required for convergence does not depend on problem size
- An algorithm implementation is scalable if the time-to-solution is constant as problem size increases with machine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

Scalable Parallel Multigrid Methods



Multigrid-Preconditioned Speeds up source iteration by factor of 100

- Source (i.e., Richardson) iteration converges slowly in diffusive regimes
- MG can be an effective preconditioner



$\sigma_T = 100$, $\sigma_s = 99.9$ S_2 (8) \ 40^3	iters	secs
Source Iteration	12712	1864
MGDSA-Source Iter	14	19

$\sigma_T = 1000$, $\sigma_s = 999.99$ S_2 (8) \ 40^3	iters	secs
BICGSTAB	1173	317
MGDSA-BICGSTAB	14	33

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

28

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.

29

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

30

Heterogeneous Computing Issues

- Experience developing math software using clusters of workstations (PVM & MPI).
- Some users would like/want/expect/demand the same results they obtained yesterday the next time they run the application.
- How can we guarantee reproducibility?
- Issues to be considered:
  - Communication of floating point values between processors
  - Machine precision and other machine parameters
  - Checking global floating-point arguments
  - Algorithmic Integrity
  - Deadlock

31

Heterogeneous Computing

- Software intended to be used in cluster computing context
- Difficulties arise with the following issues:
  - Communication of ft. pt. numbers between processors
  - Repeatability and Coherency
  - Machine precision and other machine specific parameters
  - Different versions of compilers
  - Checking global floating-point arguments
  - Iterative convergence across clusters of processors

32

Heterogeneous Computing Issues

- Issues to be considered:
  - Communication of floating point values between processors
  - Machine precision and other machine parameters
  - Checking global floating-point arguments
  - Algorithmic Integrity
  - Deadlock

33

Heterogeneous Conclusions

- Defensive programming
- Machine parameters
- Communication and representation between processors
- Controlling processor
- Additional communication
- Testing strategies

34

Heterogeneous Computing

- Defensive programming required to address these issues.
- Details of ScaLAPACK experiences can be found in:
  - http://www.netlib.org/lapack/lawns/lawn112.ps

35

DOE ASCI Program: Stockpile Stewardship via Numerical Simulation and Non-nuclear Experiments

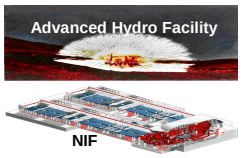
Motivation

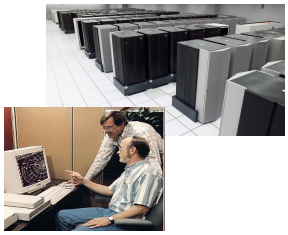
CTBT

Aging stockpile

Aging designers

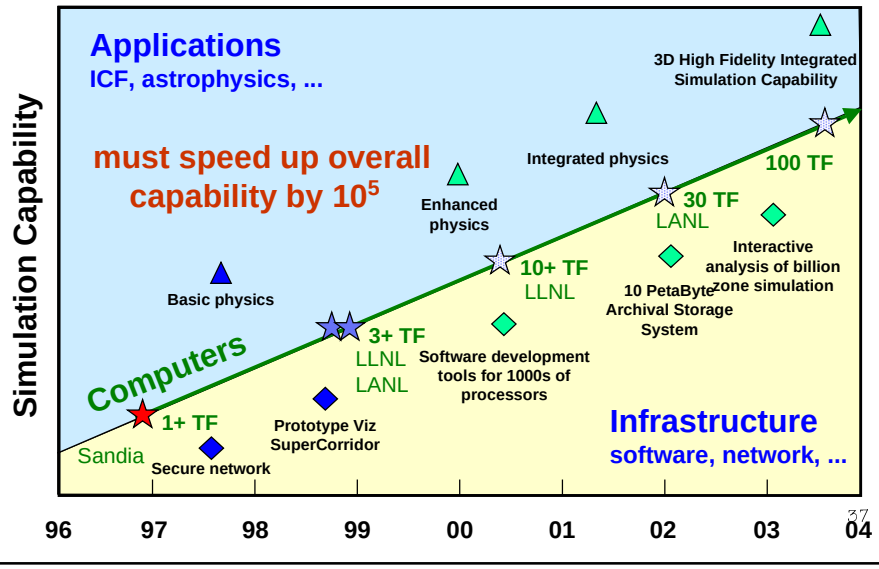
physics
engineering
materials science
scientific computing
computer science
mathematics





"Shift promptly from nuclear test-based methods to computation-based methods"

Application needs drive the platforms and software infrastructure development



37

ASCI Computing Power

Lawrence Livermore's
Terascale Simulation Facility
will have one acre of machine room floor space and 24MW of power!

Nonsymmetric Eigenvalue Problem

- QR Algorithm
  - Francis-Kublanovskaya, 1960-1961
- Let H be an nxn upper Hessenberg matrix

$H_1 = p_1(H)$ Multishift QR iteration

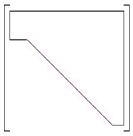
for $k = 1,2,\ldots$

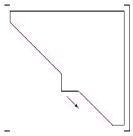
$p_k(H_k) = Q_k R_k$ (implicit QR factorization)

$H_{k+1} = Q_k^* H_k Q_k$

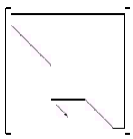
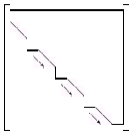
end

p_k: polynominal of degree k





Parallelation

- Hard to solve on distributed memory machines.
- Shifts are against parallelization
- Key idea: generate many shifts at once, rather than two or more at a time.
  - Pipel   orithm

40

Performance of PDLAHQR

			Schur Decomposition		Eigenvalue Only
Nb. of nodes (p)	N	Block size	Speed-up / DLAHQR	Efficiency	Efficiency
1	1800	100	1.17	1.00	1.00
4	3600	100	4.31	0.92	0.96
9	5400	100	9.28	0.88	0.90
16	7200	100	17.00	0.90	0.93
25	9000	100	26.43	0.89	0.92
36	10800	100	37.84	0.89	0.91
49	12600	100	51.19	0.89	0.91
64	14400	100	66.58	0.88	0.90

41

Symmetric Eigenvalue Problem

- Goal: compute all the eigenvalues and eigenvectors of a a symmetric matrix.
- Nice properties:
  - Eigenvalues are real
  - Eigenvectors are orthogonal
- Three steps:
  - Tridiagonalization:
  - Spectral decomposition
  - Back transform vectors
- Use D&C approach

$$T = Q^T A Q$$

$$T = U \Lambda U^T \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

U is orthogonal eigenvectors of matrix

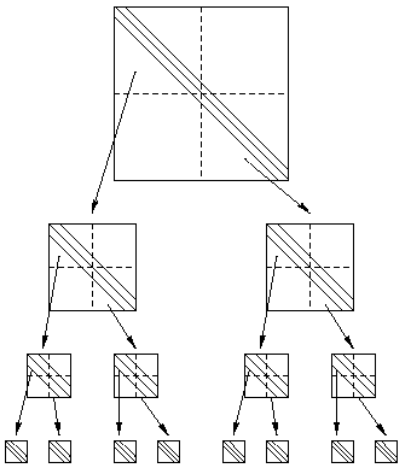
$$W = QU$$

W is orthogonal eigenvectors

42

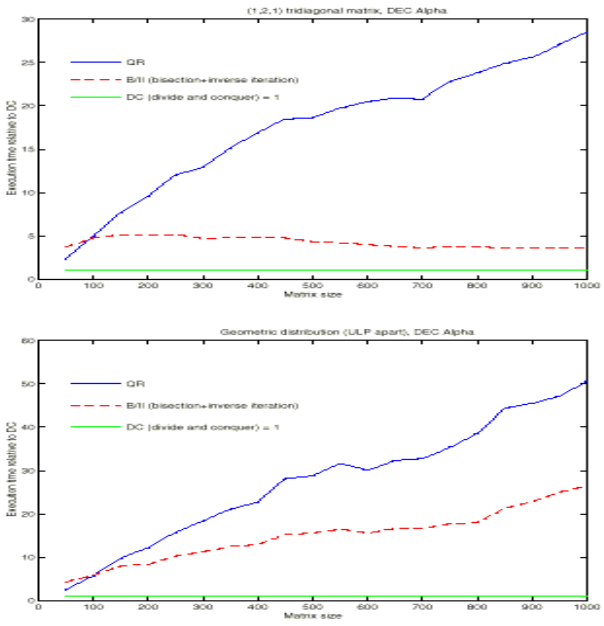
Basic Idea

- Use rank modifications to tear out selected off-diagonal elements of a tridiagonal problem.
- Yields a number of independent subproblems of smaller order
- Subproblems are solved a the lowest level
- Results are successively glued together



43

Execution time

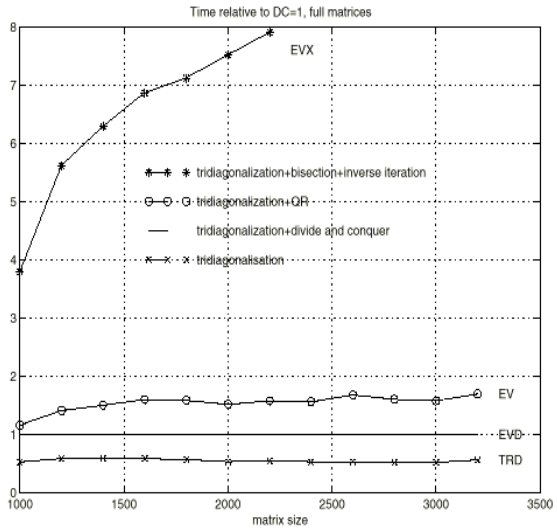


Parallelization

- Previous Implementations
  - D & Sorensen, 87, Shared Memory
  - Ispen & Jessup, 90 Hypercube
  - Gates & Arbenz, 94 Intel Paragon
- Aim: to have an efficient, scaable and portable parallel D&C for ScaLAPACK
- Issues:
  - How to distribute the data
  - How to split up the work among processors
  - How to maintain orthogonality
  - How to take advantage of deflation

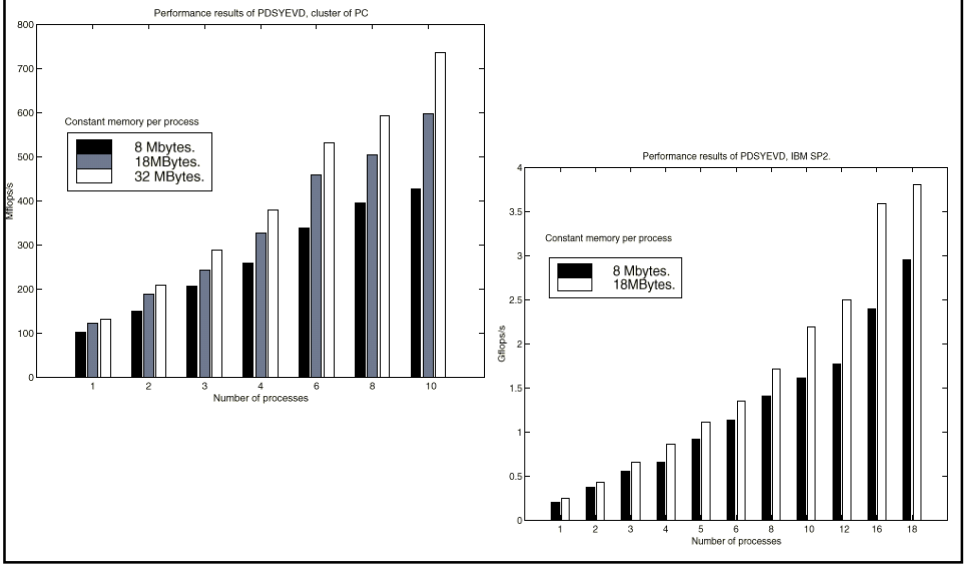
45

Execution Times IBM SP2 - 8 Procs



46

Scalability



Conclusions

- We have a stable parallel implementation for the symmetric and non-symmetric eigenvalue problem.
- Execution times are in most cases better than the classic approaches available in ScaLAPACK, QR and Parallel Bisection.
- Performance results demonstrate the scalability of the parallel software.
- No restrictions on power of 2 number of processors
- Requires $2n^2$ extra storage.

48

Summary

- Follow on to LAPACK - Designed for MPP's
- Object based design
- Software available today
- Portability across wide range of architectures
  - will work on a heterogeneous platform
- Already in use by SGI/Cray, IBM, HP-Convex, Fujitsu, NEC, NAG, & Visual Numerics (IMSL)
  - Tailor performance & provide support
- Software, examples, prebuilt available in netlib