Practical Experience in the Dangers of Heterogeneous Computing

L. Susan Blackford[1]
University of Tennessee, Knoxville, USA
Email: susan@cs.utk.edu

[1]formerly L. Susan Ostrouchov

Thanks to Coauthors:

Andrew Cleary (UT Knoxville),
Jim Demmel (UC Berkeley),
Inderjit Dhillon (UC Berkeley),
Jack Dongarra (UT Knoxville & ORNL),
Sven Hammarling (UT Knoxville & NAG Ltd, Oxford),
Antoine Petitet (UT Knoxville),
Huan Ren (UC Berkeley),
Ken Stanley (UC Berkeley),
Clint Whaley (UT Knoxville)

And to:

The rest of the ScaLAPACK team and to colleagues at NAG Ltd.

Outline of Talk

- Motivation and Background
- Defining Homogeneity/Heterogeneity
- Machine Parameters
- Global Arguments and Floating Point Values
- IEEE Machines
- Algorithmic Integrity
- Closing Remarks

ScaLAPACK

Scalable Linear Algebra PACKage

Aim: To port LAPACK to distributed memory environments

- Scalability and performance
- Portability
- Interfaces resemble LAPACK as much as possible
- Use LAPACK algorithms wherever possible
- Additional consideration is the data distribution

As with LAPACK, ScaLAPACK is in the public domain.

Target Machines for ScaLAPACK

- Dedicated distributed memory machines (MIMD)
  - IBM SP series
  - Intel series (Gamma, Delta, Paragon)
  - Cray T3 series
- Clusters of workstations
- Networks of workstations
- Any system for which PVM or MPI is available

Homogeneous Versus Heterogeneous

ISSUE # 1: Hardware

What is a homogeneous machine?

A *homogeneous machine* is one which satisfies (1), and guarantees that the result of a particular sequence of floating point operations is the same no matter which processor is used.

Is this sufficient?

Homogeneous Versus Heterogeneous

ISSUE # 2: Communication layer

What is a homogeneous network?

A *homogeneous network* is collection of machines which satisfy (1) and (2). Floating point numbers are communicated exactly between homogeneous machines.

Is this sufficient?

Homogeneous Versus Heterogeneous

ISSUE # 3: Software (operating system, compiler, ...)

What is a homogeneous computing environment?

A *homogeneous computing environment* is a network which satisfies (1), (2), and (3). The operating system, compiler, and compiler options do not alter the representation of floating point values.

If any of these requirements is not met, the system is *heterogeneous*.

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

Communicating on IEEE Machines

The IEEE standard specifies how machines should represent floating point numbers. XDR (External Data Representation) uses the IEEE representation.

PVM uses XDR and thus will communicate numbers without change on IEEE machines. (Denormalized numbers?)

MPI suggests, but does not mandate, the use of XDR.

Machine Precision

Smallest floating point number u such that

$$1 + u > 1.$$

Used in:

- Convergence tests – has an iteration converged?
- Tolerance tests – is a matrix numerically singular?
- Error analysis – how does floating point arithmetic affect results?

Heterogeneous Machine Precision

Return the maximum relative machine precision over all the processors

LAPACK $\longrightarrow$ DLAMCH
ScaLAPACK $\rightarrow$ PDLAMCH

Note that even on IEEE machines, the machine precision can vary slightly. For double precision:

$$\epsilon = 2^{-53} + q$$

where, $q = 2^{-105}$ or $q = 2^{-64} + 2^{-105}$.

Other Machine Parameters

A number of other parameters are used to characterize a machine. E.g.

- underflow threshold
- overflow threshold
- smallest number that can be safely reciprocated
- ...

For small numbers – use the largest over all the processors.

For large numbers - use the smallest over all the processors.

Heterogeneous Networks – Arithmetic issues

Even on IEEE machines results may differ between machines, compilers and compiler switches

- Global variables may have different values
- Eigenproblem for a tridiagonal matrix – run QR on each processor and each processor finds k eigenvectors. But each processor may compute a different QR sequence
- Stopping criteria for iterative methods – may not be satisfied on each processor simultaneously

Algorithmic Integrity – Examples

- Convergence of iterative methods
- Scaling vectors and matrices
- Bisection and inverse iteration for a symmetric tridiagonal matrix
- Solving the eigenproblem for a symmetric tridiagonal matrix by the QR algorithm

Convergence of Iterative Methods

`If` $\|e_r\|_2/\|x_r\|_2 < f(\epsilon)$ `then converged`

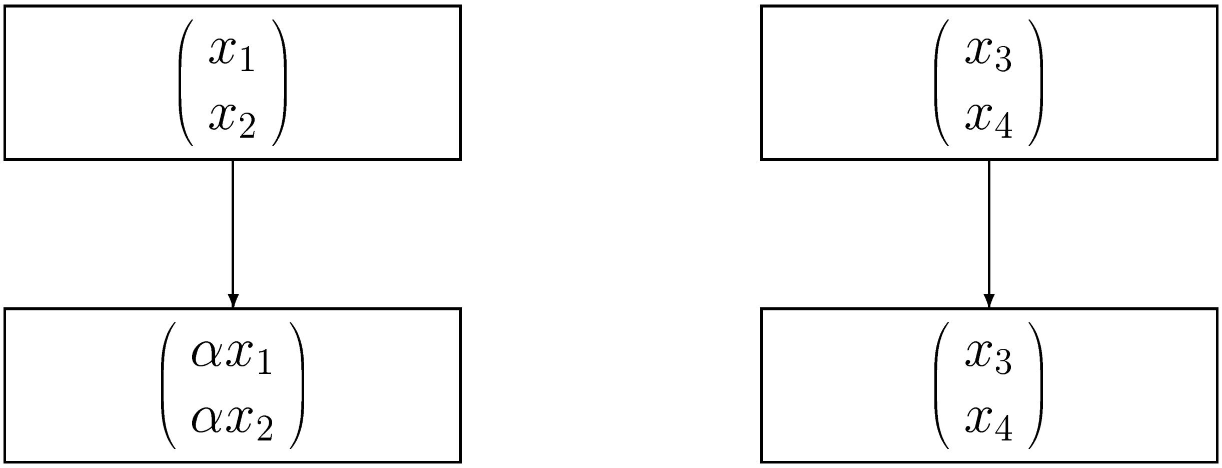
- f different on different processors
- e_r and x_r may depend upon data of different accuracies
- One or more processors may converge before the others

Have one processor make the convergence decision and broadcast that decision

Scaling a Vector

$$x^T = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$$

$$\texttt{if } \|x\|_2 < \delta \texttt{ then } x \leftarrow \alpha x$$



Have one processor make the decision about scaling

QR Factorization Example

$$A = \delta \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & & \vdots \\ 1 & \ldots & 1 \end{pmatrix}, \quad \delta \text{ small}$$

`sfmin` $\approx 10^{-38}$ on an IEEE machine (returned by PDLAMCH)

`sfmin`	`2*sfmin`

The QR factorization of A hangs (deadlocks)

QR Algorithm for a Tridiagonal Matrix

Computing the eigenvalues is $O(n^2)$ arithmetic, but computing the eigenvectors is $O(n^3)$

- Run the QR algorithm on each processor
- Save the plane rotations on each processor
- Let each processor compute k of the eigenvectors

QR is backward stable, but not forward stable, so sequence of plane rotations may be different

Have one processor run the QR algorithm and broadcast the plane rotations

Closing Remarks

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