Performance impacts of PBLAS interface and implementation decisions on LU decomposition and reduction to tridiagonal form

Ken Stanley
CS Division
UC Berkeley

James Demmel
CS Division and Math Dept.
UC Berkeley

Execution time is a function of:

- **Problem**
- **Algorithm**
- **Input Matrix**
  - n **- matrix size**
  - **- data layout**
- **Hardware and underlying software**
  - p **- Number of processors**
  - α **- Message latency (seconds per message)**
  - β **- 1/bandwith (seconds per word)**
  - γ **- 1/(BLAS3 flop rate) (seconds per flop)**
  - nb **- Block size required for efficient BLAS**

Major execution time costs of LU and reduction to tridiagonal form

- **Computation -** $O(\frac{n^3}{p}\gamma)$
- **Words sent -** $O(\frac{n^2}{\sqrt{p}}[log(p)]\alpha)$
- **Messages sent -** $O(nlog(p)\beta)$
- **Load imbalance costs:**
  - **Computation -** $O(\frac{n^2 \times nb}{\sqrt{p}}\gamma)$
  - **Words sent -** $O(n \times nb[log(p)]\beta)$
- **Algorithmic blocking costs -** $O(\frac{n^2 \times nb}{[\sqrt{p}]}\gamma)$
- **Data redistribution costs -** $O(\frac{n^2}{\sqrt{p}}[log(p)]\beta)$

Performance implications of PBLAS interface choices

Possible PBLAS extension	Large LU 256 proc SP2	Large LU 256 proc Paragon	Small LU 1000 proc SP2	Large Tridi 256 proc Paragon	Small Tridi 256 proc SP2
Asynchronous PBLAS	0-15%	0-5%	0-5%	0	0
PBMD layout	0-0%	1-5%	0-0%	0	0
Algorithmic blocking	0-2%	0-5%	0-2%	0-2%	0-5%
3D layout	XX	XX	0-6%	XX	XX
Virtual 2D torus wrap	0	0	0	??	??

- **Numbers direct from matlab code http://HTTP.CS.Berkeley.EDU/~ stanley/**
- **PBMD [Edwards, Gunnels, Chtchelkanova, van de Geijn]**
- **Algorithmic blocking [Hendrickson, Smith, Jessup] [van de Geijn]**
- **3D layout [Gao][Ashcraft][Gustavson]**
- **virtual 2D torus wrap (PRISM) [Bischof, Huss-Lederman, Jacobson, Sun, Tsao]**

Performance Impact of the PBLAS Implementation Choices

	Large LU 256 proc SP2	Large LU 256 proc Paragon	Small LU 1000 proc SP2	Large Tridi 256 proc Paragon	Small Tridi 256 proc SP2
Replicated data structures	0	0	0	5-10%	10-20%
PBLAS direct (no BLACS)	2-10%	0-2%	0-3%	5-15%	0-1%
scatter-collect broadcast	0-8%	0-2%	0-0%	0-0%	0-0%
optimal broadcast	2-8%	1-3%	20-55%	10-25%	30-70%
PBLAS error checks	0-1%	1-4%	3-15%	0-2%	10-25%
Two step band reduction	XX	XX	XX	0-0%	15-40%
Internal layout same as external layout	Up to a factor of 10 if the user chooses a poor data layout				

- **Scatter-collect collective communications (Intercom) [van de Geijn et al]**
- **Two step band reduction [Bischof, Lang, Sun]**

Performance impact of basing the BLACS on point to point sends and receives

- Hardware latencies $\sim$ 1 microsecond
- Software message latencies $\sim$ 100 microseconds
- Collective Communications time can be $>$ 50%
- Based on point to point sends and receives
  - incur multiple software latencies
  - Large latency $\rightarrow$ few messages $\rightarrow$ long messages $\rightarrow$ high bandwidth
  - Broadcasting k words to p procs costs: $log(p)(\alpha_{sw} + k\beta)$.
- Implemented at a lower level
  - incur one software latency
  - Small latency $\rightarrow$ more messages $\rightarrow$ short messages $\rightarrow$ low bandwidth
  - Broadcasting k words to p procs costs: $\alpha_{sw} + log(p)\alpha_{hw} + k\beta$.

Collective Communication projects:

- Intercom Collective Communication [van de Geijn et al]

- **GLACS - Generic active message Linear Algebra Communication Subroutines [Demmel & Ivory]**

Communication patterns for LU with Physically Based Matrix Distribution [van de Geijn et al]

A_{11}	A_{12}
A_{12}	A_{22}

$L_{11}U_{11} = A_{11}$
$L_{12} = A_{12}U_{11}^{-1}$
Spread across
$U_{12} = L_{11}^{-1}A_{21}$
Spread down
$A'_{22} = A_{22} - L_{12}U_{21}$

- "Spread Across" = independent broadcasts of k words to c processors
- Each broadcast can be accomplished as follows:
  - "Scatter" $\frac{k}{npcol}$ words each to c processors
  - "Collect" all k words to all c processors

$L_{11}U_{11} = A_{11}$
$L_{12} = A_{12}U_{11}^{-1}$
Scatter L_{11}, U_{11} and L_{12}
Collect L_{11}, U_{11} and L_{12}
$\vdots$

Communication patterns for LU with Physically Based Matrix Distribution (cont.)

$L_{11}U_{11} = A_{11}$
$L_{12} = A_{12}U_{11}^{-1}$
Scatter L_{11}, U_{11} and L_{12}
Collect L_{11}, U_{11} and L_{12}
$\vdots$

- **Perform the "Scatter" prior to the column block LU**

Scatter A_{11} and A_{12}
$L_{11}U_{11} = A_{11}$
$L_{12} = A_{12}U_{11}^{-1}$
Collect L_{11}, U_{11} and L_{12}
$\vdots$

- **Result: Column block LU is performed by all processors instead of only a few.**
- **Similar trick allows triangular solve to be performed by all processors.**

PBLAS Interface Choice: 2D block cyclic versus Physically Based Matrix Distribution Performance Impacts

- **Different communication costs for spread across and spread down**
  - **Implementation difference only**
- **Higher communication costs during block LU (twice the latency)**
- **Lower computation costs during block LU and triangular solve (factor of $\sqrt{p}$)**
- **Winner depends on relative importance of latency versus block LU and triangular solve costs.**
- **PBMD wins when large block size is required to obtain high performance from matrix matrix multiply.**

PBLAS Interface Choice: 2D block cyclic versus Algorithmic Blocking [Hendrickson, Jessup, Smith and van de Geijn]

- Algorithmic blocking factor need not equal data layout blocking factor
- Separating these blocking factors allows data layout blocking factor to be set to 1, i.e. cyclic data layout

Performance Impacts

- Better load balance
- More codes require $O(n)$ messages. Examples:
  - Cholesky
  - Two step band reduction
  - Applying block householder reflectors
- May impede improvement of the efficiency of the communication
  - Many small messages and a few large messages will be replaced
  - by many moderate sized messages.

Improvement depends on how large the blocking factor must be to allow efficient matrix multiplication.

PBLAS Interface Choice: PBLAS error checks

- **Can take up to 40% of total execution time for small problems, say n=400, p=16.**
- **Will limit performance even more when message latency costs are reduced.**
- **PBLAS error checks may cost as much as 10 times as much as the BLAS error checks - many users have already complained about the cost of the BLAS error checks.**
- **Therefore if these checks cannot be made dramatically more efficient, we should consider making them optional.**

PBLAS Implementation Choice: Replicated data structures

- **Example:** $w = Av + Bv$
- **Ideally we would like to:**
  - **Spread** v **downward**
  - **Compute temp =** Bv
  - **Compute temp =** $Av + temp$
  - **Perform reduction:** $w = sum(temp)$
- **Not possible with initial PBLAS implementation**
- **Not useful in LU decomposition**
- **Could be used in reduction to tridiagonal form**

PBLAS Interface Choice: Synchronous PBLAS

- Overlapping computation with communication in LU
  - Pipelining the spread across [PBLAS supports]
  - Overlapping communication and computation within a PBLAS call [PBLAS interface allows; implementation does not]
  - Decoupling communication and computation
    - Greatly increases the software complexity, potentially adding execution time costs
    - Bandwidth overlap is rarely perfect because of contention for resources (even on paragon)
    - Unlikely to be useful for anythign other than the LINPACK benchmark
- The PBLAS interface does not allow a complete decoupling of the communication and computation.

PBLAS Interface Choice: PBLAS built on top of the BLACS

- Overlapping computation with communication in LU
  - Pipelining the spread across [PBLAS supports]
  - Overlapping communication and computation within a PBLAS call
  - Decoupling communication and computation [PBLAS interface does not allow]
- Present PBLAS implementation does not overlap communication and computation within a PBLAS call.
- PBLAS would have to be written at some level that offers very low message latency, today that means the PBLAS would have to be written separately for each architecture.
- Active Messages might offer low enough message latency to allow portability.

PBLAS Interface Choice: 2D block cyclic versus 3D layout [Ashcraft]

- **Similar to the 3D matrix multiplication scheme [Fred Gustafson - session 6]**
- **Can reduce bandwidth costs from** $O(\frac{n^2}{\sqrt{p}})$ **to** $O(\frac{n^2}{p^{\frac{2}{3}}})$

PBLAS Interface Choice: 2D block cyclic versus Virtual 2D torus wrap (PRISM) [Bischof, Huss-Lederman, Jacobson, Sun and Tsao]

- Allows greater flexibility in assigning matrix rows and columns to processors.
- Assigns $\frac{lcm(r,c)}{gcd(r,c)}$ processes to each processor to allow a square process layout on a non-square processor layout.

Performance Impact:

- None for square processor layouts - allows all code to be written assuming a square process layout.
-

PBLAS Implementation Choice: Non square processor layouts

- **Square processor layouts could be special cased**

ScaLAPACK Implementation Choice: Internal data format same as External data format

- **Cost of data redistribution is $O(\frac{n^2}{p})$ which is substantially less that the other bandwdith requirements in LU and reduction to tridiagonal form, which are $O(\frac{n^2}{\sqrt{p}})$.**

Conclusions and Recommendations

Dense Linear Algebra
Performance Perspective Only

- Performance considerations need not be the primary decision criteria, because the competing options do not offer a compelling performance advantage.
- PBLAS error checks, if they cannot be made 5 times faster, should be optional.
- Collective communications implemented at a lower level will greatly iprove performance on some problem sizes and machines.
- Special casing square processor layouts offers the potential for improved performance.