2.5D algorithms for parallel dense linear algebra

Edgar Solomonik and James Demmel

UC Berkeley

June, 2012
Outline

Introduction
  Strong scaling

Matrix multiplication
  2D and 3D algorithms
  2.5D matrix multiplication

LU factorization
  2.5D LU without pivoting
  2.5D LU with pivoting

QR factorization
  2.5D QR using Givens rotations
  2.5D QR using Householder transformations

Conclusion
Solving science problems faster

Parallel computers can solve bigger problems

▶ weak scaling

Parallel computers can also solve a fixed problem faster

▶ strong scaling

Obstacles to strong scaling

▶ may increase relative cost of communication
▶ may hurt load balance

How to reduce communication and maintain load balance?

▶ reduce (minimize) communication along the critical path
▶ exploit the network topology
Blocking matrix multiplication
2D matrix multiplication

[Cannon 69],
[Van De Geijn and Watts 97]

$O(n^3/p)$ flops
$O(n^2/\sqrt{p})$ words moved
$O(\sqrt{p})$ messages
$O(n^2/p)$ bytes of memory
3D matrix multiplication

[Agarwal et al 95],
[Aggarwal, Chandra, and Snir 90],
[Bernsten 89], [McColl and Tiskin 99]

\[
O(n^3/p) \text{ flops}
\]

\[
O(n^2/p^{2/3}) \text{ words moved}
\]

\[
O(1) \text{ messages}
\]

\[
O(n^2/p^{2/3}) \text{ bytes of memory}
\]
2.5D matrix multiplication

[McColl and Tiskin 99]

$$O(n^3/p)$$ flops
$$O(n^2/\sqrt{c \cdot p})$$ words moved
$$O(\sqrt{p/c^3})$$ messages
$$O(c \cdot n^2/p)$$ bytes of memory
Strong scaling matrix multiplication

2.5D MM on BG/P (n=65,536)

Percentage of machine peak

2.5D MM
2D MM
ScaLAPACK PDGEMM
2.5D recursive LU

\[ A = L \cdot U \text{ where } L \text{ is lower-triangular and } U \text{ is upper-triangular} \]

- A 2.5D recursive algorithm with no pivoting [A. Tiskin 2002]
- Tiskin gives algorithm under the BSP model
  - Bulk Synchronous Parallel
  - considers communication and synchronization
- We give an alternative distributed-memory adaptation and implementation
- Also, we lower-bound the latency cost
2D blocked LU factorization

A
2D blocked LU factorization

\[
\begin{array}{cccc}
L_{00} & & & \\
& U_{00} & & \\
& & & \\
\end{array}
\]
2D blocked LU factorization
2D blocked LU factorization

\[ S = A - LU \]
2D block-cyclic decomposition
2D block-cyclic LU factorization
2D block-cyclic LU factorization
2D block-cyclic LU factorization

\[ S = A - LU \]
A new latency lower bound for LU

- Relate volume to surface area to diameter
- For block size $n/d$ LU does
  - $\Omega(n^3/d^2)$ flops
  - $\Omega(n^2/d)$ words
  - $\Omega(d)$ msgs
- Now pick $d$ (latency cost)
  - $d = \Omega(\sqrt{p})$ to minimize flops
  - $d = \Omega(\sqrt{c \cdot p})$ to minimize words
- More generally, latency $\cdot$ bandwidth $= n^2$
2.5D LU factorization
2.5D LU factorization

(A)

(B)
2.5D LU factorization

2.5D LU without pivoting
2.5D LU with pivoting
2.5D LU strong scaling (without pivoting)

2.5D LU on BG/P (n=65,536)
2.5D LU with pivoting

\[
A = P \cdot L \cdot U, \text{ where } P \text{ is a permutation matrix}
\]

- 2.5D generic pairwise elimination (neighbor/pairwise pivoting or Givens rotations (QR)) [A. Tiskin 2007]
  - pairwise pivoting does not produce an explicit \( L \)
  - pairwise pivoting may have stability issues for large matrices
- Our approach uses tournament pivoting, which is more stable than pairwise pivoting and gives \( L \) explicitly
  - pass up rows of \( A \) instead of \( U \) to avoid error accumulation
Tournament pivoting

Partial pivoting is not communication-optimal on a blocked matrix
- requires message/synchronization for each column
- \( O(n) \) messages needed

Tournament pivoting is communication-optimal
- performs a tournament to determine best pivot row candidates
- passes up 'best rows' of \( A \)
2.5D LU factorization with tournament pivoting
2.5D LU factorization with tournament pivoting
2.5D LU factorization with tournament pivoting
2.5D LU factorization with tournament pivoting

![Diagram of 2.5D LU factorization with tournament pivoting]
Strong scaling of 2.5D LU with tournament pivoting

2.5D LU on BG/P (n=65,536)
2.5D QR factorization

\[ A = Q \cdot R \] where \( Q \) is orthogonal and \( R \) is upper-triangular

- 2.5D QR using Givens rotations (generic pairwise elimination) is given by [A. Tiskin 2007]
- Tiskin minimizes latency and bandwidth by working on slanted panels
- 2.5D QR cannot be done with right-looking updates as 2.5D LU due to non-commutativity of orthogonalization updates
2.5D QR factorization using the YT representation

The $YT$ representation of Householder QR factorization is more work efficient when computing only $R$

- We give an algorithm that performs 2.5D QR using the $YT$ representation
- The algorithm performs left-looking updates on $Y$
- Householder with $YT$ needs fewer computation (roughly 2x) than Givens rotations
- Our approach achieves optimal bandwidth cost, but has $O(n)$ latency
2.5D QR using YT representation
Conclusion

Our contributions:

- 2.5D mapping of matrix multiplication
  - Optimal according to lower bounds [Irony, Tiskin, Toledo 04] and [Aggarwal, Chandra, and Snir 90]
- A new latency lower bound for LU
- Communication-optimal 2.5D LU and QR
  - Both are bandwidth-optimal according to general lower bound [Ballard, Demmel, Holtz, Schwartz 10]
  - LU is latency-optimal according to new lower bound

Reflections:

- Replication allows better strong scaling
- Topology-aware mapping cuts communication costs
Backup slides
### Performance of multicast (BG/P vs Cray)

#### 1 MB multicast on BG/P, Cray XT5, and Cray XE6

<table>
<thead>
<tr>
<th>#nodes</th>
<th>8</th>
<th>64</th>
<th>512</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG/P</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XE6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XT5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Bandwidth (MB/sec)**

**#nodes**

*Edgar Solomonik and James Demmel*
Why the performance discrepancy in multicasts?

- Cray machines use **binomial multicasts**
  - Form spanning tree from a list of nodes
  - Route copies of message down each branch
  - Network contention degrades utilization on a 3D torus

- BG/P uses **rectangular multicasts**
  - Require network topology to be a \( k \)-ary \( n \)-cube
  - Form \( 2n \) edge-disjoint spanning trees
    - Route in different dimensional order
    - Use both directions of bidirectional network
Rectangular collectives

2D rectangular multicasts trees

2D 4X4 Torus

Spanning tree 1

Spanning tree 2

Spanning tree 3

Spanning tree 4

All 4 trees combined
Cost breakdown of MM on 65,536 cores

Matrix multiplication on 16,384 nodes of BG/P

95% reduction in comm

Execution time normalized by 2D

Edgar Solomonik and James Demmel
2.5D LU on 65,536 cores

LU on 16,384 nodes of BG/P (n=131,072)

- NO-pivot 2D
- NO-pivot 2.5D
- CA-pivot 2D
- CA-pivot 2.5D

Time (sec)

- Communication
- Idle
- Compute