Improving communication performance in dense linear algebra via topology-aware collectives

Edgar Solomonik*, Abhinav Bhave†, and James Demmel*

* University of California, Berkeley
† Lawrence Livermore National Laboratory

Supercomputing, November 2011
Outline

Collective communication
   Rectangular collectives

2.5D algorithms
   2.5D Matrix Multiplication
   2.5D LU factorization

Modelling exascale
   Multicast performance
   MM and LU performance
Performance of multicast (BG/P vs Cray)

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

Bandwidth (MB/sec) vs #nodes

- BG/P
- XE6
- XT5
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
2D rectangular multicasts trees

[Watts and Van De Geijn 95]
Another look at that first plot

How much better are rectangular algorithms on $P = 4096$ nodes?

- Binomial collectives on XE6
  - 1/30th of link bandwidth
- Rectangular collectives on BG/P
  - 4X the link bandwidth
- 120X improvement in efficiency!

How can we apply this?
Matrix multiplication
2D matrix multiplication

[Cannon 69], [Van De Geijn and Watts 97]
3D matrix multiplication

[Agarwal et al 95], [Aggarwal, Chandra, and Snir 90], [Bernsten 89]
2.5D matrix multiplication

32 CPUs (4x4x2)

2 copies of matrices

Edgar Solomonik
Mapping dense linear algebra 10/29
Strong scaling matrix multiplication

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

- 2.5D SUMMA
- 2D SUMMA
- ScaLAPACK PDGEMM

Percentage of machine peak

#nodes
2.5D MM on 65,536 cores

Matrix multiplication on 16,384 nodes of BG/P

- 2.5D MM: 12X faster
- 2D MM: 2.7X faster

Using 16 matrix copies
Cost breakdown of MM on 65,536 cores
2.5D LU factorization

\[
\begin{align*}
L_{00} & & U_{00} & & U_{03} \\
L_{00} & & U_{00} & & U_{03} \\
L_{00} & & U_{00} & & U_{03} \\
\end{align*}
\]

(A)
2.5D LU factorization
2.5D LU factorization

Look at how this update is distributed.

Same 3D update in multiplication
2.5D LU factorization

[Solomonik and Demmel, EuroPar ’11, Distinguished Paper]
2.5D LU on 65,536 cores
Rectangular (RCT) vs binomial (BNM) collectives

Binomial vs rectangular collectives on BG/P (n=131,072, p=16,384)
A model for rectangular multicasts

\[ t_{mcast} = \frac{m}{B_n} + 2(d + 1) \cdot o + 3L + d \cdot P^{1/d} \cdot (2o + L) \]

Our multicast model consists of 3 terms

1. \( \frac{m}{B_n} \), the bandwidth cost
2. \( 2(d + 1) \cdot o + 3L \), the multicast start-up overhead
3. \( d \cdot P^{1/d} \cdot (2o + L) \), the path overhead
A model for binomial multicasts

\[ t_{bnm} = \log_2(P) \cdot (m/B_n + 2o + L) \]

- The root of the binomial tree sends \( \log_2(P) \) copies of message
- The setup overhead is overlapped with the path overhead
- **We assume no contention**
Model verification: one dimension

DCMF Broadcast on a ring of 8 nodes of BG/P

Bandwidth (MB/sec) vs. msg size (KB)

- $t_{\text{rect}}$ model
- DCMF rectangle dput
- $t_{\text{bnm}}$ model
- DCMF binominal
Model verification: two dimensions

DCMF Broadcast on 64 (8x8) nodes of BG/P

Bandwidth (MB/sec) vs msg size (KB)
Model verification: three dimensions

DCMF Broadcast on 512 (8x8x8) nodes of BG/P

Bandwidth (MB/sec)

msg size (KB)

DCMF Broadcast on 512 (8x8x8) nodes of BG/P

Bandwidth (MB/sec)

msg size (KB)
Modelling collectives at exascale ($p = 262, 144$)

Exascale broadcast performance
Modelling matrix multiplication at exascale

MM strong scaling at exascale (xy plane to full xyz torus)

Parallel efficiency vs. z dimension of partition:
- 2.5D with rectangular (c=z)
- 2.5D with binomial (c=z)
- 2D with binomial

Edgar Solomonik
Modelling LU factorization at exascale

LU strong scaling at exascale (xy plane to full xyz torus)

Parallel efficiency vs z dimension of partition

- 2.5D with rectangular (c=z)
- 2.5D with binomial (c=z)
- 2D LU with binomial
Conclusion

- Topology-aware scheduling
  - Present in IBM BG but not in Cray supercomputers
  - Avoids network contention/congestion
  - Enables optimized communication collectives
  - Leads to simple communication performance models

- Future work
  - An automated framework for topology-aware mapping
  - Tensor computations mapping
  - Better models for network contention
Acknowledgements

- Krell CSGF DOE fellowship (DE-AC02-06CH11357)
- Resources at Argonne National Lab and Lawrence Berkeley National Lab
  - DE-SC0003959
  - DE-SC0004938
  - DE-FC02-06-ER25786
  - DE-SC0001845
  - DE-AC02-06CH11357
- Berkeley ParLab funding
  - Microsoft (Award #024263) and Intel (Award #024894)
  - U.C. Discovery (Award #DIG07-10227)
- Released by Lawrence Livermore National Laboratory as LLNL-PRES-514231
Backup slides
A model for rectangular reductions

\[ t_{\text{red}} = \max[m/(8\gamma), 3m/\beta, m/B_n] + 2(d+1) \cdot o + 3L + d \cdot P^{1/d} \cdot (2o + L) \]

- Any multicast tree can be inverted to produce a reduction tree
- The reduction operator must be applied at each node
  - each node operates on \(2m\) data
  - both the memory bandwidth and computation cost can be overlapped
Rectangular reduction performance on BG/P

BG/P rectangular reduction performs significantly worse than multicast
Performance of custom line reduction
A new LU latency lower bound

flops lower bound requires \( d = \Omega(\sqrt{p}) \) blocks/messages

bandwidth lower bound required \( d = \Omega(\sqrt{cp}) \) blocks/messages
2.5D LU strong scaling (without pivoting)

LU without pivoting on BG/P (n=65,536)

Percentage of machine peak vs. #nodes for 2.5D LU strong scaling (without pivoting) on a BG/P system with n=65,536. The graph shows the performance in percentage of the machine peak for different numbers of nodes, illustrating the scaling behavior of the 2.5D LU algorithm.
Strong scaling of 2.5D LU with tournament pivoting

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

Percentage of machine peak vs. #nodes for different LU methods:
- Ideal scaling
- 2.5D LU
- 2D LU
- ScaLAPACK PDGETRF