Matrix multiplication on multidimensional torus networks

Edgar Solomonik, and James Demmel

University of California, Berkeley

VECPAR, July 2012

Outline

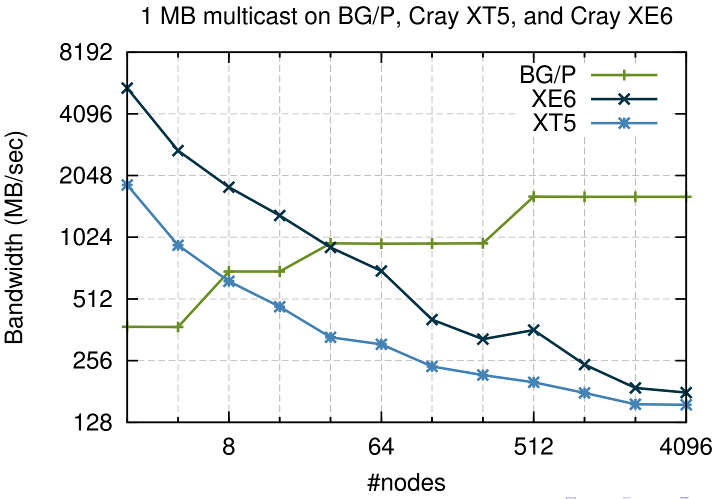
- Torus networks
  - BlueGene architecture
  - Collective communication
- Algorithms
  - SUMMA
  - Cannon's algorithm
  - SD-Cannon's algorithm
- Implementation
  - One-sided MPI communication
  - Charm++ virtualization
- Performance
- Conclusion

BlueGene/P and BlueGene/Q

Direct torus networks

- BG/P is 3D, BG/Q is 5D
- Both are bidirectional networks (6 and 10 links per node)
- Injection bandwidth sufficient to saturate all links
- Topology-aware partition allocation and collectives

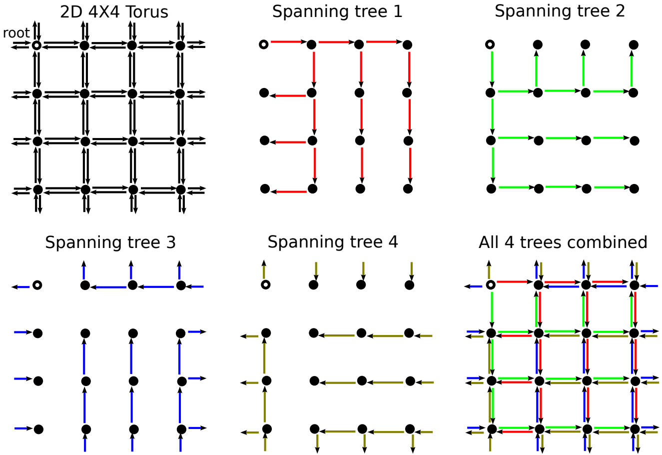
Performance of multicast (BG/P vs Cray)



Why the performance discrepancy in multicasts?

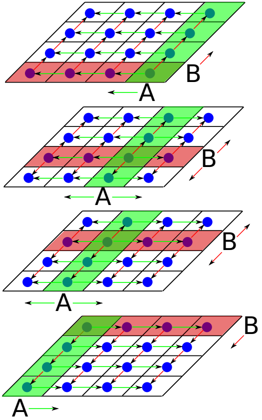
- Cray machines use **binomial multicasts**
  - Form spanning tree from a list of nodes
  - Route copies of message down each branch
  - Network contention degrades utilization on a 3D torus
- BG/P uses **rectangular (pipelined) multicasts**
  - Require network topology to be a k-ary n-cube
  - Form $2n$ edge-disjoint spanning trees
    - Route in different dimensional order
    - Use both directions of bidirectional network

2D rectangular pipelined multicast trees



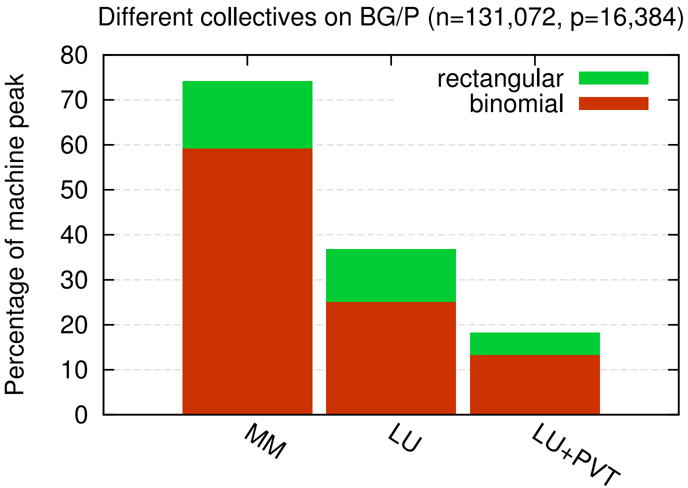
[Watts and Van De Geijn 95]

Matrix multiplication

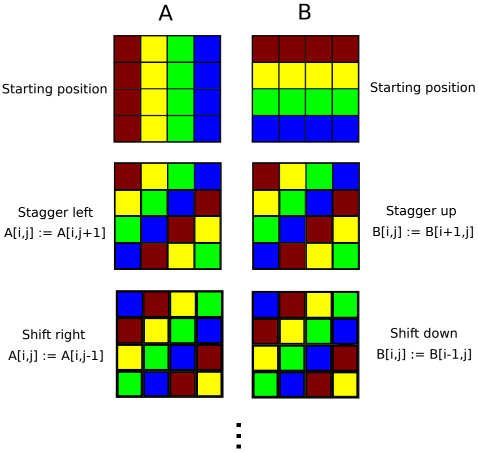


[Van De Geijn and Watts 97]

SUMMA and LU with rectangular vs binomial collectives



Cannon's algorithm



[Cannon 69]

Cannon's algorithm

Advantages over SUMMA

- Uses only near-neighbor sends rather than multicasts
  - **lower latency cost**
- Can be done in-place given near-neighbor data-swaps

Disadvantages with respect to SUMMA

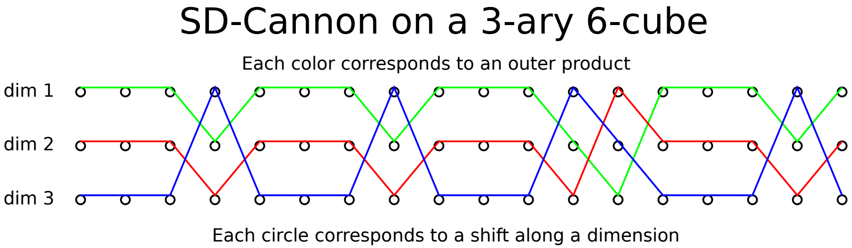
- Does not generalize well to non-square processor grids
- Cannot exploit multiple links via rectangular multicasts

Split-dimensional Cannon's algorithm

Improves over Cannon by saturating all network links

- Subdivide whole multiply into 2n block outer products
- Use bidirectional links by shifting half the outer products in opposite direction
- Perform each outer product in a different dimensional order
- Accumulation of outer-products into one buffer allows for same-sized local multiplications as pure Cannon's algorithm
- Does not require pipelined multicasts

Split-dimensional Cannon's algorithm



MPI implementation

SD-Cannon parallel implementation using MPI

- All communication done with near-neighbor one-sided puts
- Code is simple (200 lines)
- Limited to square (k-ary n-cube) processor grids

Charm++

Charm++ is an asynchronous dynamic runtime system

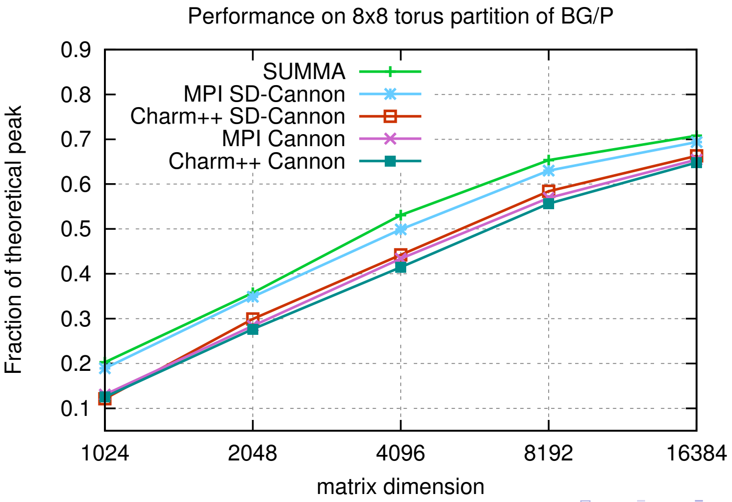
- Provides object-based (chares) virtualization (decouples from process grid)
- Message-directed task invocation
- Allows topology-aware task mapping
- Provides additional features such as dynamic load balancing and performance profiling

Virtual topology via Charm++

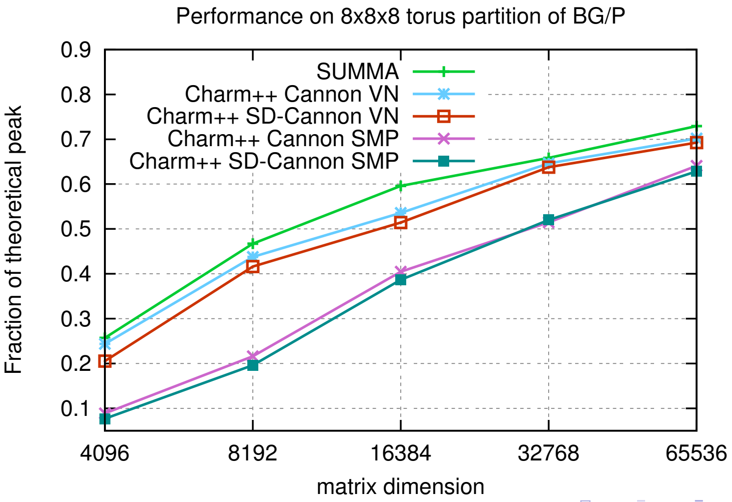
Implemented Cannon and SD-Cannon in Charm++

- Can map to any torus process topology
- Code more complex, but not significantly so
- Not using one-sided communication (though possible via CkDirect)
- Virtualization lowers task granularity

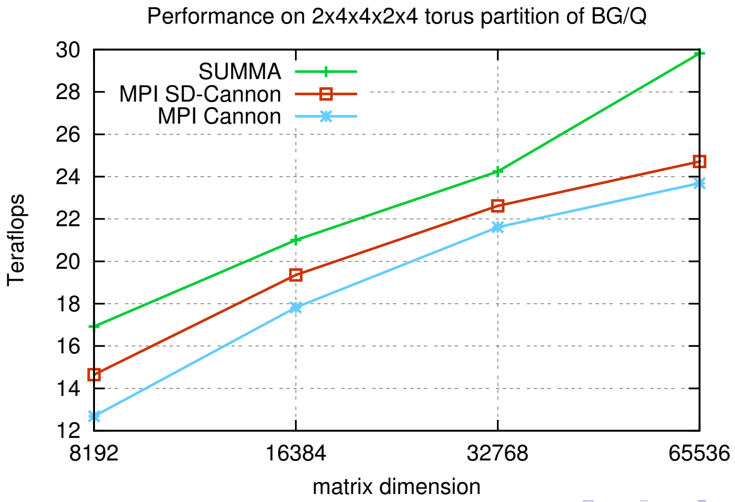
2D BlueGene/P performance



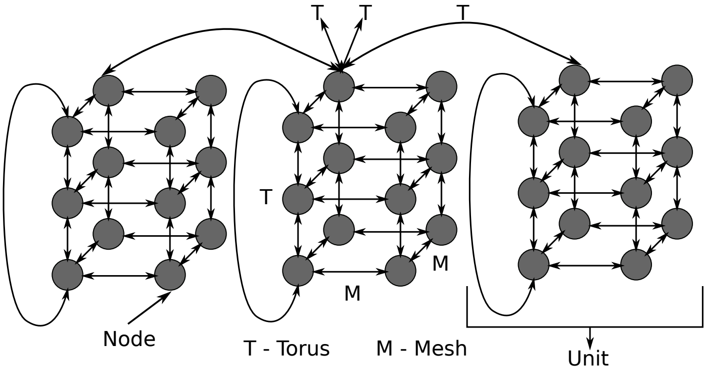
3D BlueGene/P performance



Preliminary 5D BlueGene/Q performance

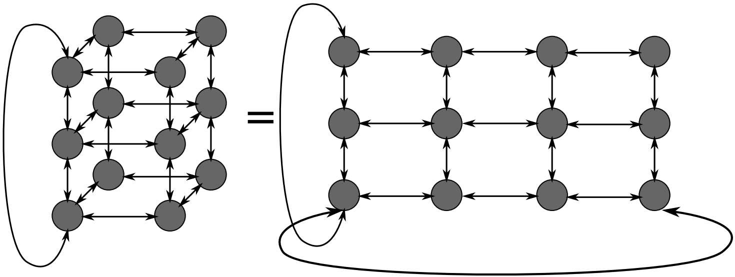


K computer Tofu network



10 links per node / 4 torus dimensions / 2 mesh dimensions of length 2

Tofu network is 5D not 6D



A two-by-two mesh is a 1D ring of length 4

SD-Cannon K computer potential

5D K computer torus different from 5D BG/Q

- K computer injection bandwidth can saturate only 4/10 links

SD-Cannon could still be beneficial

- Can saturate 4 links rather than 2 (up to 2X speed-up)
- Does not require pipelined broadcast implementation

Conclusion

SD-Cannon

- Breaches performance gap between Cannon and SUMMA
- Is uniquely asymptotically communication-optimal on a k-ary n-cube
- Virtualization allows general mapping support but incurs overhead

Topology-aware mapping and algorithm design

- Allows zero network contention
- Permits saturation of much more bandwidth on torus networks
- Pervasive for parallel scalability on high-end supercomputers

Finish

Acknowledgements:

- Krell CSGF DOE fellowship (DE-AC02-06CH11357)
- Resources at Argonne National Lab
- Anonymous VECPAR reviewers
- Discussions with collaborators
  - James Demmel (UC Berkeley)
  - Jeff Hammond (Argonne National Lab)
  - Grey Ballard (UC Berkeley)

Also, see my talk on 2.5D algorithms at University of Tokyo

- Tuesday, July 24, 10:00-12:00

Backup slides