Improving Communication-Phase Completion Times in HPC Clusters Through Congestion Mitigation

Vladimir Zdornov and Yitzhak Birk

Department of Electrical Engineering
Technion - Israel Institute of Technology

SYSTOR 2009

Outline

1. Introduction
   - Modern Clusters and Interconnects
   - Congestion Problem
   - Performance Metric and Goals
   - Existing Solutions
2. Adaptive Routing
   - Generic Scheme
   - Application in k-ary n-tree
3. Explicit Rate Calculation (Phase-Based Application)
   - Optimal Rate Assignment
   - Distributed Algorithm
4. Realization of Calculated Rates
5. Summary

Outline

Modern Clusters

- Massively parallel – hundreds to tens of thousands computing nodes
- Off-the-shelf computing hardware
- Standard or proprietary interconnect

Interconnect Standards

InfiniBand

- Inherently oriented for clusters
- Used in apprx. 28% of top-500

Ethernet

- Originally defined for general purpose communication
- Cluster versions gradually adopt InfiniBand-like properties
- Gigabit Ethernet used in apprx. 57% of top-500

- We used InfiniBand as the platform, but expect results to be applicable for cluster networks in general

InfiniBand Characteristics

Fabric properties

1. Small buffers
2. Virtual-output queueing
   - "Infinite speedup" was assumed in simulations
3. Lossless fabric
4. Oblivious, destination-based routing
   - Together with 3, guarantees in-order packet delivery

Network management

1. Managed environment – known behavior of network elements
2. Reliable communication – failures are exceptional

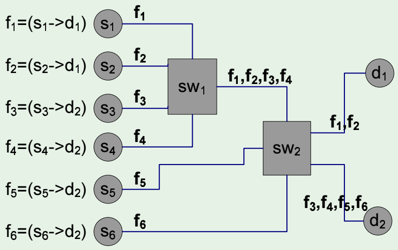
Direct Damage of Contention

- If flows didn't share links, all of them would be transmitted at line speed and we could go home...
- ... in practice, however, flows compete for resources, which reduces their transmission rates
- Apparently, the reduction of contention through load-balancing (adaptive routing) should improve the performance

Congestion Spreading

- Adaptive routing may be beneficial, but is not a panacea

Example



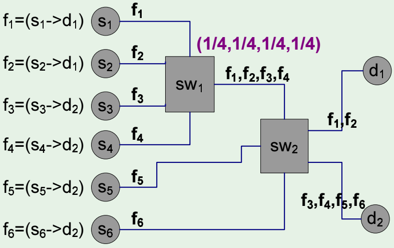
- Outputs serve different inputs in Round-Robin
- Regardless of buffer size
- For equisized flows, total completion time is 25% above the optimum

- An appropriate rate control can solve all above problems

Congestion Spreading

- Adaptive routing may be beneficial, but is not a panacea

Example



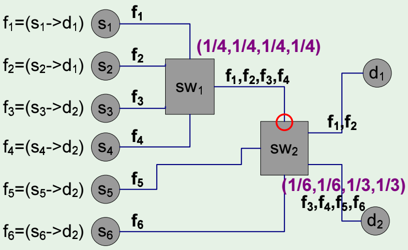
- Outputs serve different inputs in Round-Robin
- Regardless of buffer size
- For equisized flows, total completion time is 25% above the optimum

- An appropriate rate control can solve all above problems

Congestion Spreading

- Adaptive routing may be beneficial, but is not a panacea

Example



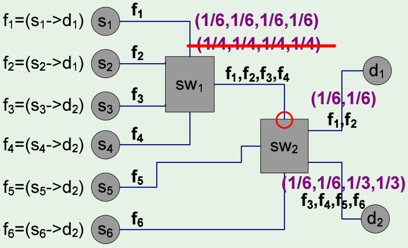
- Outputs serve different inputs in Round-Robin
- Regardless of buffer size
- For equisized flows, total completion time is 25% above the optimum

- An appropriate rate control can solve all above problems

Congestion Spreading

- Adaptive routing may be beneficial, but is not a panacea

Example



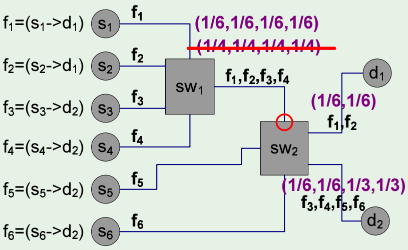
- Outputs serve different inputs in Round-Robin
- Regardless of buffer size
- For equisized flows, total completion time is 25% above the optimum

- An appropriate rate control can solve all above problems

Congestion Spreading

- Adaptive routing may be beneficial, but is not a panacea

Example



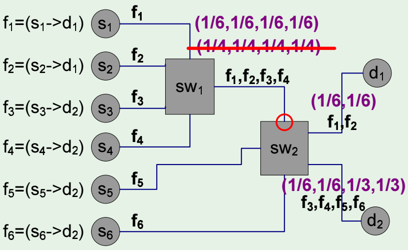
- Outputs serve different inputs in Round-Robin
- Regardless of buffer size
- For equisized flows, total completion time is 25% above the optimum

- An appropriate rate control can solve all above problems

Congestion Spreading

- Adaptive routing may be beneficial, but is not a panacea

Example



- Outputs serve different inputs in Round-Robin
- Regardless of buffer size
- For equisized flows, total completion time is 25% above the optimum

- An appropriate rate control can solve all above problems

A Single Phase-Based Application

Scenario

- The cluster is used by a single application that alternates between computation and communication phases, separated by a global (to the application's nodes) barrier.
- At the beginning of a communication phase, each source knows its destinations and the exact amount of data to be transferred.

Goals and Limitations

Goal

Use adaptive routing and rate control to minimize the length of the communication phase, which is defined by the maximum completion time among flows (total completion time).

Limitations

1. No per-flow state at switches
2. In-order packet delivery

Routing Problem

- Seeking optimal routing usually leads to a variant of the integral multi-commodity flow problem, which is NP-Complete [Even et al., 1975]

⇒ Known practical approaches to adaptive routing are heuristic

Flavors of Adaptive Routing

Approach

- Packet-level adaptation [Kim et al., 2006]
- Predefined alternate paths [Lin et al., 2004]
- Incorporation in virtual circuits (VC) [Dao et al., 1997]

Drawbacks

- Breaks the in-order delivery guarantee
- Limited adaptivity, reduced number of available addresses
- Limited number of connections per switch

Our Approach

VC routing for in-order delivery, while retaining scalability

Rate Control – TCP

Scheme

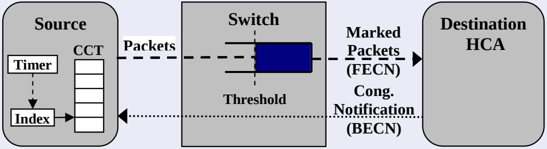
- TCP congestion control schemes use a congestion window to control the number of in-flight packets
- The size of the window is adjusted in response to collected feedback (RTT, packet loss)

Limitations

- Cluster networks have a small bandwidth-delay product
- The maximum window size should be a few MTU packets per flow
- Even with window size of 1, congestion spreading can occur!

Rate Control – InfiniBand CCA

Scheme



Parameter Choice?

- [Santos et al., 2003, Yan et al., 2006] – analytical models
- [Pfister et al., 2005] – extensive simulations
- Our observations – tuning for topology and traffic pattern is required

Outline

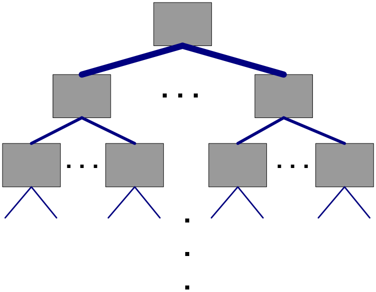
Our VC Routing

- Routing information is stored at each switch on the path
  - Default port number per destination (+alternate ports)
  - A fixed number of routing entries, each for a single flow (not all flows get an entry)
- Set up a path by adaptively routing the flow's first packet
  - Use local information for adaptation at switches
  - Use the default port when no free routing entries are present or when it is the best choice
  - In our simulations, the number of flows traversing each output link was used as the basis for the routing decision
- Route the rest of the flow's packets on the path in-order
- Tear down the path after flow's last packet

Fat Tree

Ideal fat tree

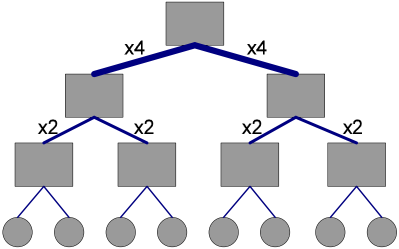
Practical 2-ary 3-tree



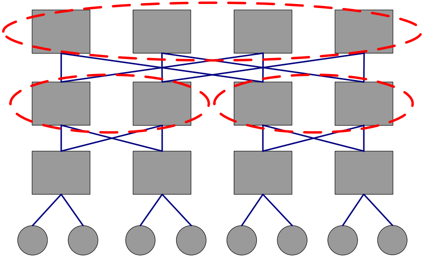
- We use *k-ary n-tree* topology in all our simulations

Fat Tree

Ideal binary, height-3 fat tree



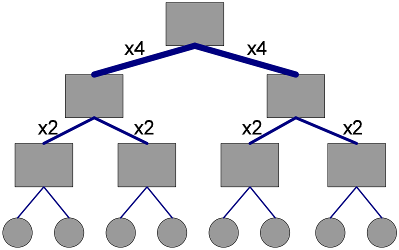
Practical 2-ary 3-tree



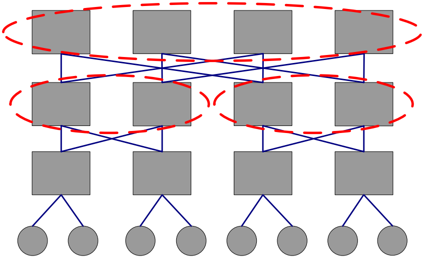
- We use *k-ary n-tree* topology in all our simulations

Fat Tree

Ideal binary, height-3 fat tree



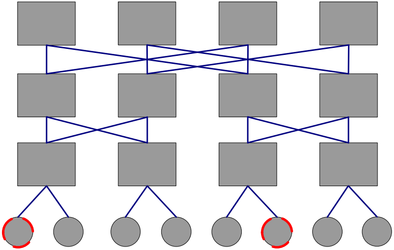
Practical 2-ary 3-tree



- We use *k-ary n-tree* topology in all our simulations

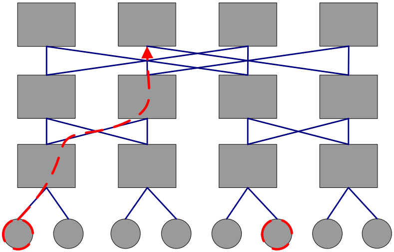
Routing in k-ary n-tree

- Up-down routing
- Arbitrary ascent
- The ascent path uniquely determines the descent path



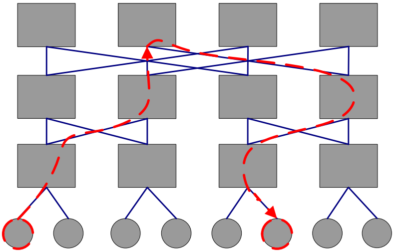
Routing in k-ary n-tree

- Up-down routing
- Arbitrary ascent
- The ascent path uniquely determines the descent path



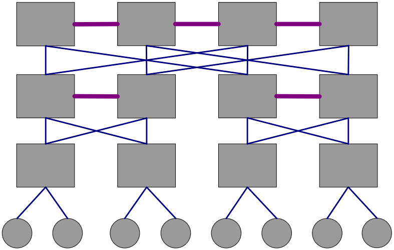
Routing in k-ary n-tree

- Up-down routing
- Arbitrary ascent
- The ascent path uniquely determines the descent path



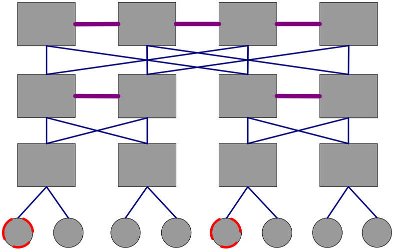
Modified k-ary n-tree

- Horizontal links are added between switches that represent the same ideal fat-tree node
- Routing in a consistent horizontal direction is enabled at every level during descent



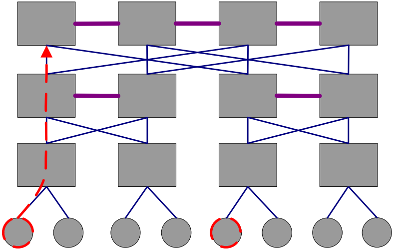
Modified k-ary n-tree

- Horizontal links are added between switches that represent the same ideal fat-tree node
- Routing in a consistent horizontal direction is enabled at every level during descent



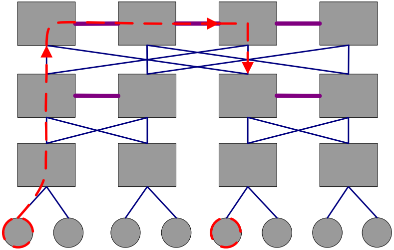
Modified k-ary n-tree

- Horizontal links are added between switches that represent the same ideal fat-tree node
- Routing in a consistent horizontal direction is enabled at every level during descent



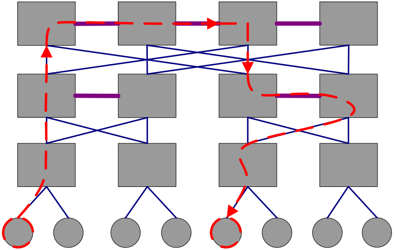
Modified k-ary n-tree

- Horizontal links are added between switches that represent the same ideal fat-tree node
- Routing in a consistent horizontal direction is enabled at every level during descent



Modified k-ary n-tree

- Horizontal links are added between switches that represent the same ideal fat-tree node
- Routing in a consistent horizontal direction is enabled at every level during descent



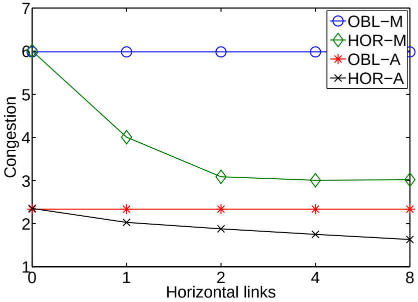
Adaptive Routing in Modified Tree – Simulation Setting

- Topology – modified 16-ary, 3-tree (4096 end nodes) with varying “width” of horizontal links
- Traffic – a single random permutation
- Routing – oblivious, adaptive in modified tree
- Metric – maximum and average (over flows) encountered congestion

- Results averaged over 1000 runs

Adaptive Routing in Modified Tree – Simulation Results

Results



Summary

Horizontal width of 2 gives the best tradeoff:

1. $\sim$ 10% additional ports
2. $\sim$ 50% reduction of max
3. $\sim$ 20% reduction of average

Adaptive Routing – Summary

- We presented a simple scheme for adaptive routing that
  1. Preserves order of delivery
  2. Is scalable (no limit on number of VCs)
- Our scheme in conjunction with a small enrichment of fat trees, offers a significant reduction in congestion with low overhead

Remarks

1. Our approach is heuristic. In some cases oblivious routing is optimal, and adaptation may actually harm.
2. From here on, we refer by "adaptive routing" to the combination of additional capacity (width=2) and our routing scheme.

Outline

Single Application Setup

Definitions

1. w_f – flow weight, equal to its size d_f
2. W_l – link weight, aggregate weight of its flows
3. W_f – maximum link weight encountered by flow f
4. W – maximum link weight in the network
5. $r(f)$ – rate of flow f
6. $\bar{r}(f)$ – normalized rate of f, $\frac{r(f)}{w_f}$

- Different flows (compute nodes) enter the communication phase independently, approximately at the same time
- We assume long flows, so all flows are considered to start "simultaneously"

Seeking Optimal Assignment

Goal

- Minimize the total completion time
- Achieved by maximization of $\min_f \{\bar{r}(f)\} = \min_f \left\{ \frac{r(f)}{d_f} \right\}$

Approach

- Initially, consider non-weighted, fixed size flows
- Let $W = N$; setting $\forall f : r(f) = \frac{1}{N}$ is optimal
- In fact, if $W_f = N_f$, $\forall f : r(f) = \frac{1}{N_f}$ is optimal as well
- For varying size flows, simply replace $r(f)$ with $\bar{r}(f)$

Single Application Assignment (SAA)

Theorem

Rate assignment r, for which $\forall f \in F : r(f) = \frac{w_f}{W_f}$ (or $\overline{r}(f) = \frac{1}{W_f}$), is feasible and guarantees the shortest completion in W units of time

Remarks

1. Reducing maximum link weight by means of adaptive routing directly improves the achievable completion time
2. SAA does not provide maximality, i.e., there is possibly usable (but not useful!) residual capacity

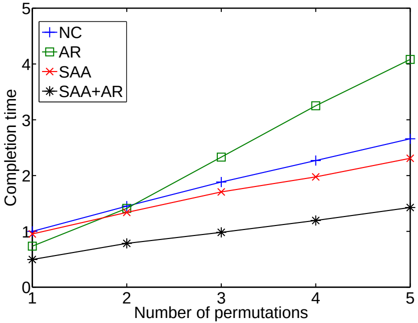
SAA – Simulation Setting

- Topology – modified 16-ary, 3-tree (4096 end nodes) with horizontal width 2
- Traffic – superposition of a varying number of random permutations, fixed length flows
- Routing – oblivious in regular tree; adaptive in modified tree
- Rate Control – no control; SAA
- Metric – total completion time

- Results averaged over 50 runs

SAA – Simulation Results

Results



Summary

- SAA ⇒ up to 13% improvement
- SAA+AR ⇒ up to 50% improvement
- AR without rate control can cause damage

Distributed Implementation of SAA

- Links store W_l values

Upon flow start at the beginning of a communication phase

- The flow sends a control packet to update relevant links about weight increase by w_f

Periodically (piggy-backed on data packets)

- The flow sends a control packet to collect W_f, and set $r(f) = \frac{w_f}{W_f}$

Upon flow end

- The flow sends a control packet to update relevant links about weight decrease by w_f

Distributed Implementation of SAA

- Links store W_l values

Upon flow start at the beginning of a communication phase

- The flow sends a control packet to update relevant links about weight increase by w_f

Periodically (piggy-backed on data packets)

- The flow sends a control packet to collect W_f, and set $r(f) = \frac{w_f}{W_f}$

Upon flow end

- The flow sends a control packet to update relevant links about weight decrease by w_f

Distributed Implementation of SAA

- Links store W_l values

Upon flow start at the beginning of a communication phase

- The flow sends a control packet to update relevant links about weight increase by w_f

Periodically (piggy-backed on data packets)

- The flow sends a control packet to collect W_f, and set $r(f) = \frac{w_f}{W_f}$

Upon flow end

- The flow sends a control packet to update relevant links about weight decrease by w_f

Distributed Implementation of SAA

- Links store W_l values

Upon flow start at the beginning of a communication phase

- The flow sends a control packet to update relevant links about weight increase by w_f

Periodically (piggy-backed on data packets)

- The flow sends a control packet to collect W_f, and set $r(f) = \frac{w_f}{W_f}$

Upon flow end

- The flow sends a control packet to update relevant links about weight decrease by w_f

Distributed SAA – Properties

- A flow gets an *initial* allocation within one round trip
- After all flows "announce" their start, it takes each flow a single probing to acquire the final rate (occurs very fast if piggy-backed on data packets)

Outline

Fluid Model – Problem

- The rate calculation considers link capacity only
  - Implicitly assuming that traffic has fluid nature
- In practice discrete data packets are used
  - We could rely on buffers for smoothing...
  - But buffers in InfiniBand are too small

Fluid Model – Problem

- The rate calculation considers link capacity only
  - Implicitly assuming that traffic has fluid nature
- In practice discrete data packets are used
  - We could rely on buffers for smoothing...
  - But buffers in InfiniBand are too small

Closing the Gap

- Three factors affect the gap between fluid and discrete models:
  1. Buffer size
  2. Injection scheme
  3. Packet service policy in switches
- We propose an injection scheme that:
  - Suppresses bursts (stronger than leaky bucket)
  - Empirically shown to realize calculated rates for realistic buffer size (under FCFS)

Outline

Conclusions

1. Generic adaptive routing with in-order guarantees
   - Application in modified k-ary n-trees
   - Up to 50% reduction in maximum contention for random permutations
2. Explicit rate calculation algorithm for single phase-based application scenario
   - We show that rate control is required to turn the reduced "topological" contention into an actual performance gain
   - Additional rate calculation algorithm to be published elsewhere (independent flows, multiple phase-based applications)
3. A practical injection scheme that effectively realizes the desired rates even with small buffers

Directions for Future Work

1. Adaptive routing
   - Application in other topologies
   - Generic framework for adaptation policies
2. Rate calculation
   - Deeper quantitative examination of dynamic properties of the algorithm
3. Testing on real-life benchmarks
4. Implementation in InfiniBand

References I

Dao, B. V., Yalamanchili, S., and Duato, J. (1997).
Architectural support for reducing communication overhead in multiprocessor interconnection networks.
In *Proc. 3rd IEEE Symp. on High-Performance Computer Architecture (HPCA)*, page 343.

Even, S., Itai, A., and Shamir, A. (1975).
On the complexity of time table and multi-commodity flow problems.
In *Proc. 16th Annual Symp. on Foundations of Computer Science (SFCS)*, pages 184–193.

Kim, J., Dally, W. J., and Abts, D. (2006).
Adaptive routing in high-radix clos network.
In *Proc. ACM/IEEE Conf. on Supercomputing (SC)*, page 92.

References II

Lin, X.-Y., Chung, Y.-C., and Huang, T.-Y. (2004).
A multiple lid routing scheme for fat-tree-based infiniband networks.
In *Proc. 18th Int'l Parallel and Distributed Processing Symp. (IPDPS)*.

Pfister, G., Gusat, M., and Craddock, D. (2005).
Solving hot spot contention using infiniband architecture congestion control.
In *Proc. High Performance Interconnects for Distributed Computing Workshop*.

Santos, J. R., Turner, Y., and Janakiraman, G. J. (2003).
End-to-end congestion control for infiniband.
In *Proc. IEEE INFOCOM*, volume 2, pages 1123–1133.

References III

Yan, S., Min, G., and Awan, I. (2006).
An enhanced congestion control mechanism in infiniband networks for high performance computing systems.
In *Proc. 20th Advanced Information Networking and Applications (AINA)*, volume 1, pages 845–850.