Efficient Reliable Broadcast for Commodity Clusters

Raymond Wong and Cho-Li Wang
Department of Computer Science and Information Systems
The University of Hong Kong
What is a cluster?

A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource.

-- IEEE TFCC
Efficient Reliable Broadcast

- Efficient clustering requires efficient networking for tightly coupling all resources.
- Improving network performance helps improving performance in cluster computation.
- Efficient Reliable Broadcast: Let’s do all together − the most basic synchronization and data movement operation.
Main objectives

- To achieve the fastest broadcast in a commodity SMP cluster connected by a network with hardware broadcast.

- Reduce resource consumptions:
  - **Computation**: e.g., CPU cycles – low-latency
  - **Memory**: e.g., send/receive buffers
  - **Network**: e.g., avoid redundant traffic

- You can consider the collective operation as a “fat” communication command.
Outline

- Background
- Push-Pull Messaging
- Hardware Broadcast
- Performance Evaluation
- Conclusions
Tackling the Problem...

- Theoretical broadcast studies have focused on the delivery strategy of packets based on some abstract model
  - **LogP** (1993), Karp, [9] (Also, Subramonian’s multiple-item broadcast in LogP model)
  - **Star Graph** (1997): Y.C. Tseng, et. al., [14].
  - **Hypercube, Mesh, Tori**: Survey paper [McKinley: 1995]

- Efficient in terms of complexity.
- Could not be practically implemented.
Tackling the Problem...

- Broadcast algorithms in “message” level:
  - **IBM SP2 (MPL):** Abandah (U. of Michigan), [IPPS’ 96]
  - **InterCom Project** (iCC library, INTEL Paragon, 1995): Mitra, et. al. (short, long, hybrid)
  - **MPI CH:** Gropp, et. Al, (linear, tree-based) [6]

- To high level – good portability
- Cannot take advantage of underlying system features
Tackling the Problem...

- Hardware broadcast is efficient. We adopt it.
- But research issues are,
  - How to utilize the hardware broadcast operation in user-level for efficient data movement?
  - Transferring broadcast packets is not reliable. How to make it reliable?
  - Single "fat" packet for multiple nodes. What is the delivery strategy?
Push-Pull Messaging [17] : Concept

Sender

Push Phase

��

Allow better buffer management

Allow better buffer management

Can avoid long latency by applying various latency hiding techniques.

Receiver

Send BTP

Acknowledged

Send Rest

BTP: Bytes-to-Push (short message)

Now, prepare the buffer

Pull Phase

(send the rest of message)
Main Data Structures in DP

- **(1) send queue** stores pending send requests. **send buffer** stores the data.

- **(2) receive queue** stores pending receive requests. Packets received from the NIC are stored in **receive buffer**.

- **(3) buffer queue** and **pushed buffer** store pending incoming packets where their destinations in memory are not determined.

- Three queues can be accessed by both user and kernel threads.
Push-Pull Messaging: Architecture

Sender

- Source Buffer
- Buffer Queue and Pushed Buffer
- Receive Queue
- Send Queue
- Send Process
- Queues used by other processes
- Reception Handler

Receiver

- Incoming FIFO Buffer Queue
- Outgoing FIFO Buffer Queue
- Outgoing FIFO Buffer Queue
- Ack
- Reception Handler
- Queues used by other processes
- Send Process
- Kernel Space
- User Space
Architecture (Receive)

Sender

Kernel Space

Incoming FIFO Buffer Queue

Outgoing FIFO Buffer Queue

Reception Handler

Receiver

Kernel Space

Outgoing FIFO Buffer Queue

Incoming FIFO Buffer Queue

Buffer Queue and Pushed Buffer

Receive Queue

Send Queue

Ack

Reception Handler

Ack

Reception Handler

Destination Buffer

Receive Process

Queues used by other processes

DP Process

DP Proc.

DP Proc.
Performance Optimization : DP-SMP, 1999 ICPP [17]

- Cross-Space Zero Buffer
- Address Translation Overhead Masking
- Push-and-Acknowledge Overlapping
DP-SMP Performance

- **Internode**: (machine-to-machine)
  - **Single-trip latency** (ALR 4-way Pentium Pro. 200 MHz SMP, 66 MHz system bus, back-to-back) : 30.1 microseconds (8-byte message)
  - **Bandwidth**: 12.1 MB/s (Digital DEC 21140A Fast Ethernet) at 40KB message

- **Intranode**: (process-to-process within the same node)
  - **Single-trip latency** : 7.5 microseconds (8-byte message)
  - **Bandwidth**: 350.9 MB/s at 40KB message
Directed Point abstraction model [10]

NID: node ID
DPID: endpoint ID
**Broadcast: traditional high-level implementation**

- Use a sequence of point-to-point communication.
- Simple, but it cannot be fully optimized for the performance.
  - Reliable channels are maintained independently. Each channel may keep transmission and reception buffers.
  - Poor scalability: the number of transmission and reception buffers in the root node increases as the size of the cluster increases.
  - Extra synchronization overheads incur while switching from channel to channel.
New Data Structures

- **Enhanced Queue Architecture (EQA)**
  - Allows multiple senders to share one single queue and buffer properly.
  - An entry in a queue and buffer could be retrieved by many senders which linked to the queue and buffer.

- **Light-weight Directed Point (LDP)**
  - LDP = DP without buffers.
  - LDP stores pointers which point to appropriate BUF in a DP.
Enhanced Queue Architecture

- DP
- HMS
- LMS
- send buffer (BUF)

- NID1
- NID2
- NID3
- NID4

- Enhanced Queue Architecture
Broadcast with EQA

**Root node**

**Hardware broadcast**

**Leaf node**

**Home management structure**

**Local management structure**

HMS  
LMS
Two Hardware-based Broadcast Algorithms

(1) Simple Broadcast.

- Packets are (H/W) broadcast one by one.
- **Flow control**: go-back-n protocol -- controlled by DP (HMS)
- Packets may be lost if the destination buffers are not allocated due to the late receive operation.
- **Retransmission**: Lost packets will be re-sent (use point-to-point operation) according to transmission records stored in LDP (LMS)
(2) Push-Pull Broadcast.

- **Push phase:** a portion of the broadcast message is pushed to all the leaf nodes.
  - ONLY one DP would send acknowledge packets after finishing the push phase.

- **Pull phase:** the source DP broadcasts the remaining packets to all DPs one by one.
  - Point-to-point communication is used to re-send the lost packets during the pull phase based on a go-back-n protocol.
Performance Evaluation

- **Cluster configuration:**
  - 8 x Intel MP1.4-complaint SMP machines.
  - Each consisted of 2 Intel Celeron 450 MHz processors with 128 Mbytes memory.
  - Connected by Fast Ethernet.
  - OS: Linux 2.2.1

- **Broadcast algorithms tested:**
  - Simple Broadcast (SBCAST)
  - Push-Pull Broadcast (PPBCAST)
Broadcast Latency Test

![Graph showing latency vs size for PPBCAST/4, SBCAST/4, and MPICH/4.]
Broadcast Latency Comparison

Short Messages (256 bytes)

Latency (us)

Cluster Size

PPBCAST
MPICH

p=2
p=4
p=8

142μs

Long Messages (4096 bytes)

Latency (us)

Cluster Size

PPBCAST
MPICH

p=2
p=4
p=8
Parallel Ray Tracing

- Using MPIPOV by ParMa² with MPI/DP-SMP
Conclusions

- Using **hardware broadcast feature**, single “fat” packet could be received by a number of attached hosts at the switch.

- Compare to multiple “unicast” packet
  - Larger bandwidth
  - Shorter latency

- With EQA, the **computation**, **memory** and **network** resources can be utilized more efficiently.
Future Works

- Develop more efficient reliable protocols on larger cluster sizes.
- Incorporation of the hardware broadcast facility with other parallel applications:
  - Software DSM : JUMP-DP
  - N-Body simulation
  - Cluster-based Web Caching : fast lookup
  - Search Engine: broadcast queries
  - Performance benchmarking software
The End

The Systems Research Group
http://www.srg.csis.hku.hk/

Department of Computer Science
and Information Systems
The University of Hong Kong