A Fast Delivery Protocol for Total Order Broadcasting

Li Ou
Xubin (Ben) He

Tennessee Tech University

Christian Engelmann
Stephen L. Scott

Oak Ridge National Lab

Outline

- Introduction
  - Total ordered broadcast algorithms
  - Pros and Cons
- A fast delivery protocol
  - Fast delivery criterion
  - Smart acknowledgement
  - Dynamic systems
- Prototyping and measurement results
- Conclusions and future work

Totally Ordered Broadcast Algorithms

- **Sequencer:**
  - One machine is responsible for ordering the messages on behalf of other processes.
- **Privilege-based:**
  - One machine can only send messages after it get a token.
- **Communication history**
  - Delay the delivery of messages, until enough information has been gathered from other machines.

Advantages and Disadvantages

- **Sequencer and privilege-based:**
  - Good performance when system is idle.
  - High latency when multiple machines are active.

- **Communication history:**
  - Post-transmission delay in idle system.
  - Regular messages continuously form a total order when communication is heavy.

Total ordering in parallel computing systems

- In a parallel computing system, multiple concurrent requests are expected to arrive simultaneously.
- A communication history algorithm is preferred, since such algorithm performs well under heavy communication loads with concurrent requests.
- The post-transmission delay is high for relatively light load scenarios.

Post-transmission Delay

- Delay is most apparent when the system is relatively idle.
- The length of the delay is set by the slowest machine to respond.
- Worst case: latency = heartbeat interval.

Fast Delivery Protocol

- Reduce Post-transmission delay when system is idle.
- Maintain performance when communication is heavy.
- Implemented on top of Transis.

Total order broadcasting services

- **Basic broadcasting services**
  - responsible for the reliable delivery of messages.
  - causal delivery order is guaranteed.

- **Total order broadcasting**
  - built on top of the basic broadcasting service.
  - extends the underlying causal order to a total order for concurrent messages.

Notation and Definition

- **Mp:** set of candidate messages ready to be delivered.

- **sender(m):** sender machine of message m.
- **sender(Mp):** senders of all messages in the Mp.

- **id(p):** id of machine p.
- **Prefix(p):** all machines in the group whose ids are smaller than id(p).
- **Suffix(p):** all machines in the group whose ids are greater than id(p).

Fast Delivery Criterion

- **Server p deliver message m when:**
  - Messages from a subset of machines in the group are received

$$prefix(sender(m)) \in senders(Mp)$$

  - The messages are delivered in order of id of sender machines

$$id(sender(m_i)) < id(sender(m_j)) \rightarrow deliver(m_i) < deliver(m_j)$$

- **To speed up message delivery for a idle system, Server p fast acknowledge message m when:**

$$sender(m) \in suffix(p)$$

Smart acknowledgment (to prevent unnecessary acknowledgements for a busy system)

- **Server p only fast acknowledge message m when:**

1. Message m is a total order message.

2. There is no message waiting to be sent from the server p.

3. In the set of candidate messages (Mp), no messages are sent from p

$$/\exists m_j \in Mp, id(sender(m_j)) = p$$

Fast delivery protocol for dynamic systems

- A membership service maintains a view of current membership set (CMS)
- After a new agreement of CMS, the membership service notifies fast delivery protocol with a special machine set, Pf, which belongs to previous configuration, but is not included in the new configuration.
- Recalculate prefix(p) and suffix(p) for each machine in the new CMS:

$$1)\ prefix(p)_{new} = prefix(p)_{old} - P_f$$

$$2)\ suffix(p)_{new} = suffix(p)_{old} - P_f$$

Experimental Setup

- Transis v1.03 with fast delivery protocol for group communication
- Benchmarking on ORNL XTORC Cluster
  - Dual Intel Pentium 2GHz nodes with 768MB memory and 40 GB hard disk space
  - Fast Ethernet (100MBit/s full duplex)
  - OS: Linux FC5
- MPI-based benchmark on multiple clients to send concurrent requests

Request latency comparison for a idle system with a single active machine

# of Machines	Fast delivery protocol (us)	Communication history algorithm (Transis)
1	235	Random variable [230us, p]
2	952	Random variable [940us, p]
3	1031	Random variable [1020us, p]
4	1089	Random variable [1070us, p]
5	1158	Random variable [1150us, p]
6	1213	Random variable [1200us, p]
7	1290	Random variable [1280us, p]
8	1348	Random variable [1340us, p]

1. In idle system, the post-transmission of Transis is apparent. The latency is a random variable, and in the worst case, it is may equal to the interval of heart beat messages, p.

2. p is the gratitude of hundred milliseconds. The default value of Transis is 500ms.

Request latency comparison of a busy system where all machines are active

# of Machines	Fast delivery protocol (us)	Communication history algorithm (Transis)
1	235	227
2	971	966
3	1461	1458
4	1845	1842
5	2236	2231
6	2646	2639
7	3073	3068
8	3514	3509

In a busy system, the latency of fast delivery protocol is almost the same as the traditional communication history algorithm, because the protocol holds unnecessary acknowledgments

Scalability of fast delivery protocol: a client-server application with fast delivery protocol

Servers \ Clients	1	2	4	8	16	32
1	1	2.1	4.1	8.4	18	36.5
2	1.1	2.2	4.3	8.6	18.2	36.9
4	1.3	2.5	5.1	10	19.7	37.7

1. All latencies are formalized to latency of one server with one client.
2. The latency includes the overhead of receiving requests from clients, broadcasting requests with totally ordering, and sending back reply to clients.
3. Multiple servers form an total ordering group.
4. Each request received by one server is broadcasted to all other servers using fast delivery protocol.
5. The server used to receive a request from a client is randomly chosen.

Conclusions

- Fast delivery protocol reduces the latency of message ordering for idle systems and keep comparable performances with communication history algorithms for busy systems.
- The protocol optimizes the total ordering process by waiting for messages only from a subset of the machines in the group.
- The fast acknowledgment aggressively acknowledges total order messages to reduce the latency for idle system, and it is smart enough to hold the acknowledgments when the network communication is heavy.

Future Work

- Conducting comprehensive experiments.
- Tuning the fast acknowledgements .
- Refining the protocol for large scale systems.

Acknowledgements

- Laboratory Directed Research and Development Program of Oak Ridge National Laboratory.

- Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy.

- U.S. National Science Foundation under Grant Nos. CNS-0617528 and SCI-0453438.

Questions and Comments?

More questions, please contact Li Ou: lou@tntech.edu