Implementing Distributed Shared Memory

CS 451 Lecture 5
August 12, 2003

Prof. R.K. Joshi
Dept of CSE
IIT Bombay
Distributed Shared memory

- Data passing model vs. shared memory
- Shared memory: access data through read and write operations
- In absence of physically shared memory in loosely couple machines

Implement a software abstraction layer for supporting shared memory

The layer may itself be fully distributed
Two basic abstractions of shared memory

Data = read (address)
Write (data, address)
<table>
<thead>
<tr>
<th>No migration</th>
<th>Central server</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>No replication</td>
<td>Full-replication</td>
<td>Read-replication</td>
</tr>
<tr>
<td>Migration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Central Server Algorithm

Server::
- receive data request
- perform data access
- send response

Client::
- Send data request
- Receive response
Central Server Algorithm

- Only 1 copy of shared data
- Simple request-response protocol
- 2 messages for each data access

- Central server is a bottleneck
- How to improve performance?
  - Partition data into multiple servers
Migration Algorithm

Remote host:
- receive data request
- send data block

Client:
- If data block not local, determine location
- send request
- receive response
- Access data

Migration request
Migrate data block
Migration Algorithm

- Data always migrated to site where it is accessed
- With no replication, there is only one copy of a data block

  Single reader, single writer style
  No communication costs when block is held locally
  Location can be found by multicasting location request
  Good distribution of data blocks must be achieved
Read Replication Algorithm: write operation

Remote host::
- receive data request
- send data block

Write Client::
- receive invalidate
- invalidate block
- If block not local:
  - determine location
  - send request
  - receive block
  - multicast invalidate
  - Access data

write request

Invalidate requests

(case: block not local)

Migrate data block
Read Replication Algorithm: read operation

Remote host::
receive data request
send copy of data block

read Client::
If block not local, determine location
send request receive block mark it read only
Access data
Read Replication

- Multiple readers and single writer
- Either 1 site has R/W right on the block or all have read rights
- Migration occurs if block is not local: Data always available at site where it is accessed
- Allows concurrent read access as opposed to central and migration algorithms
Read Replication

- We considered following cases in class:
  - Read request, data block not available locally
  - Read request, data block local
  - Write request, data block not available locally
  - Write request, data block is available locally

\[
\begin{align*}
\text{Simultaneous } R, R \text{ from 2 sites} & \quad \text{When block available locally} \\
\text{Simultaneous } R, W \text{ from 2 sites} & \quad \text{And} \\
\text{Simultaneous } W, W \text{ from 2 sites} & \quad \text{When it’s not available locally}
\end{align*}
\]

You may not find this analysis in the paper as the paper mainly discusses the conceptual framework.
Full Replication Algorithm: A possible implementation

Sequencer::
receive write data
generate sequence number
multicast

hosts::
receive write data
update local memory

write Client::
Send data to sequencer
Receive ACK
Update local memory

Write request

write

update

sequencer
Full Replication

- Data available on all sites locally
- Consistency of write operations?

Globally sequence only the write operations

Locally sequence the local reads and writes

(e.g. in write update cache consistency in multiprocessors: read is done by an ordered broadcast over a bus)
Full Replication

Think of distributed implementation for full replication
Reference Reading