CS 555: DISTRIBUTED SYSTEMS
[MESSAGING SYSTEMS]

Shrideep Pallickara
Computer Science
Colorado State University

October 6, 2015

Frequently asked questions from the previous class survey

- Daisy chain MapReduce job? Multiple mappers or reducers in a row
- Mapper has produced intermediate results and then fails before data is fetched by reducer. What happens?
- If we run the same job multiple times are the combiner groups the same every time?
- Problem domains where MapReduce is not useful?
- Alternatives to Hadoop?
- If source code contains while(true) will Hadoop skip it?
- How do static variables work in Mapper/Reducer?
- Can we configure skip bad records in MapReduce?
- org.apache.hadoop.mapred.SkipBadRecords
- Setting number of mappers: conf.setNumMapTasks(int num)
- Can we define a combiner that is different from the reducer?
- If mapper produces (k1, v1) can combiner produce (k2, v2)?

October 6, 2015

Topics covered in this lecture

- Messaging Systems
  - IP Multicast
  - Indirect communications
  - Group-based communications

Several distributed systems built on top of the service offered by transport layer

- Sockets
- Message Passing Interface (MPI)

Interfaces to the transport layer

- Sockets interface
  - Introduced in the 1970s in Berkley Unix
  - X/Open Transport Interface (XTI) by AT&T
  - Sockets and XTI are very similar

Socket primitives for TCP/IP

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket</td>
<td>Create a new communication endpoint</td>
</tr>
<tr>
<td>Bind</td>
<td>Attach a local address to a socket</td>
</tr>
<tr>
<td>Listen</td>
<td>Announce willingness to accept connections</td>
</tr>
<tr>
<td>Accept</td>
<td>Block caller until a request arrives</td>
</tr>
<tr>
<td>Connect</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>Send</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>Receive</td>
<td>Receive some data over the connection</td>
</tr>
<tr>
<td>Close</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>
The communication pattern using sockets

Server
- socket
- bind
- listen
- accept
- read
- write
- close

Synchronization point
Communications

Client
- socket
- connect
- write
- read
- close

Message Passing Interface (MPI)
- High performance computers need highly efficient communications
- Primitives must be:
  - At the right level of abstraction
  - Implementation must have minimal overhead

Sockets were deemed inefficient ...
- Wrong level of abstraction
  - Only simple {send, receive} primitives
- Performance issues
  - Designed for network communications
  - Uses general-purpose stacks
  - Not suitable for proprietary protocols in HPC

Result?
- Most HPC systems were shipped with proprietary communication libraries
  - High-level and efficient communication primitives
- Of course they were all mutually incompatible
  - Portability became an issue

Need for hardware and platform independence led to a standard
- Message Passing Interface (MPI)
- Designed for parallel applications
- Tailored for transient communications
- Assumes process crashes/partitions are fatal
  - No automatic recovery

MPI assumes communications takes place within a group of processes
- Each group is assigned an identifier
- Each process within a group has an identifier
  - (groupId, processId) pair
    - Uniquely identifies endpoints
    - Instead of transport-level addresses
Some of the message passing primitives in MPI

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_bsend</td>
<td>Append outgoing message to local send buffer</td>
</tr>
<tr>
<td>MPI_send</td>
<td>Send message and wait until it is copied to local or remote buffer</td>
</tr>
<tr>
<td>MPI_ssend</td>
<td>Send message and wait until receipt starts</td>
</tr>
<tr>
<td>MPI_sendrecv</td>
<td>Send message and wait for reply</td>
</tr>
<tr>
<td>MPI_isend</td>
<td>Pass reference to outgoing message, and continue</td>
</tr>
<tr>
<td>MPI_isendrecv</td>
<td>Pass reference to outgoing message. Wait until receipt starts</td>
</tr>
<tr>
<td>MPI_recv</td>
<td>Receive a message; block if there is none</td>
</tr>
<tr>
<td>MPI_irecv</td>
<td>Check if there is an incoming message; but do not block</td>
</tr>
</tbody>
</table>

MPI offers a very large number of communication primitives

- **Official reason**
  - More possibilities to improve performance
  - Pick and choose most suitable one
- **Cynical view**
  - Committee could not make up its collective mind
  - Threw in everything!

MULTICAST COMMUNICATIONS

- Uses datagram packets
  - Sender sends packet to multicast address
    - 224.0.0.0 to 239.255.255.255 Class D
- IPv6
  - Multicast addresses have the prefix ff00::/8

<table>
<thead>
<tr>
<th>Bits</th>
<th>8</th>
<th>4</th>
<th>4</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Prefix</td>
<td>Flags</td>
<td>Scope</td>
<td>Group ID</td>
</tr>
</tbody>
</table>

Multicast communications: Routing data

- Routers make sure packet delivered to all hosts in multicast group
- Choose points where streams are duplicated
- Pay attention to TTL for the datagrams
  - Maximum number of routers a datagram is allowed to cross

Multicast issues

- **Packet size** restrictions in Multicast
- Practical considerations
  - Turned off at several institutions to curb free-riding
Multicast is not particularly suitable in some situations

- Consumption patterns change dynamically
  - Groups cannot be pre-allocated

- Representing consumption profiles as groups
  - Enormous number of groups – potentially $2^N$ for $N$ consumers
  - Eliminating impossible groups would still require millions of groups

**APPLICATION LEVEL MULTICASTING**

Application-level multicasting

- Sending data to multiple receivers
  - Multicasting
  - Belonged in the domain of network protocols
    - Set up communication paths
  - No convergence on protocols
    - ISPs are reluctant to support this

Construction of the overlay network

- Nodes organize into a tree
  - Unique path between every pair of nodes

- Organize nodes into a mesh network
  - Every node has multiple neighbors
  - Multiple paths exist between pairs of nodes
  - More robust to failures

Use of rendezvous nodes

- New node contacts rendezvous node to get a partial list of nodes
- New node uses this list to determine which nodes to connect to
  - Metrics used in decision making
    - Load, delays, bandwidths, etc

**INDIRECT COMMUNICATIONS**
Indirect communications

- Communications between entities in a distributed system through an intermediary
- Senders and receivers have no direct coupling

Direct coupling

- E.g. TCP communications between 2 endpoints
- Leads to a rigidity in terms of dealing with change
- Example
  - A client-server system
  - Difficult to replace a server with another one that offers equivalent functionality

Key properties of using an intermediary for communications

- Space uncoupling
  - Sender does not know (or need to know) the identity of receivers
  - And vice versa
- Time uncoupling
  - Developer has a degree of freedom in dealing with change
  - Participants (senders or receivers) can be replaced, updated, replicated or migrated

Space uncoupling

- Sender does not know (or need to know) the identity of receivers
- And vice versa
- Developer has a degree of freedom in dealing with change
- Participants (senders or receivers) can be replaced, updated, replicated or migrated

Time uncoupling

- Sender and receiver(s) can have independent lifetimes
- Do not need to exist at the same time to communicate
- Useful in volatile environments
  - Where senders and receivers join and leave constantly

Indirect communications: Primary uses

- In distributed systems where change is anticipated
  - E.g. mobile environments
- Event disseminations
  - Situations where receivers are unknown and may be liable to change
Indirect communications: Disadvantages

- **Performance overhead** due to the added level of indirection
- Systems developed using this concept are more **difficult to manage**
  - Precisely because of the lack of coupling

Indirection may imply decoupling in space and time .... but not always the case (1/2)

- **Space-uncoupled, time-coupled**
  - **IP multicast**
    - Space uncoupled
      - Messages directed to a group, not a specific receiver
    - Time coupled
      - All receivers must exist at the time of the message-send to

Indirection may imply decoupling in space and time .... but not always the case (2/2)

- **Space-coupled but time-uncoupled**
  - Sender knows the identity of specific receiver or a set of receivers
  - Receiver(s) may not exist at the time of sending

  - **Example:**
    - E-mail

Space and time coupling in distributed systems

<table>
<thead>
<tr>
<th>Time-coupled</th>
<th>Time-uncoupled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space coupling</strong></td>
<td>Communication directed towards a given receiver(s)</td>
</tr>
<tr>
<td>Receiver(s) must exist at that moment in time</td>
<td></td>
</tr>
<tr>
<td><strong>E.g.: Remote invocation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Space uncoupling</strong></td>
<td>Communication directed towards a given receiver(s)</td>
</tr>
<tr>
<td>Sender and Receiver(s) can have independent lifetimes</td>
<td></td>
</tr>
<tr>
<td><strong>E.g.: E-mail</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**
- **Remote invocation**
- **E-mail**

**Example:**
- **IP Multicast**
- **Publish/subscribe**

Time-coupling and asynchronous communications

- **Asynchronous communications**
  - Sender sends message and continues without blocking

- **Time uncoupling**
  - Extra dimension where sender and receiver can have independent existence

GROUP COMMUNICATIONS
Group communications

- Service where message is sent to a group
- Message is then delivered to all members of the group
- Sender is not aware of the identities of the receivers

Group communications

- May be implemented over
  - IP Multicast
  - Overlay network
- Adds extra value in terms of
  - Managing group membership, detecting failures and providing reliability, and ordering guarantees
- Group communication is to IP Multicast
  - What TCP is to point-to-point service in IP

The programming model

- Central concept is a group
  - This is associated with group membership
  - Processes may join or leave the group
- Process sends message to this group
  - Propagated to all members with guarantees relating to reliability and ordering
- Communications:
  - All members in a group: broadcast
  - Communication with a single member: unicast

Key feature of group communications

- Process issues only one operation to send the message to the group
  - E.g. aGroup.send(aMessage)
  - It does not issue multiple send operations to individual processes
- This has much more than convenience implications for the programmer

Advantage of a single group-send over multiple individual-send operations

- Enables efficient utilization of bandwidth
  - Take steps to send the message no more than once over a communication link
    - For e.g., send it over a distribution tree
    - Use network hardware support for multicast when available
- Minimize total time taken to deliver message to all destinations
  - As compared to transmitting separately and serially

This is also important in terms of delivery guarantees

- If a process performs multiple, independent send operations?
  - No way to provide guarantees that affect the group as a whole
  - If sender fails halfway through sending?
    - Some members receive while others don’t
  - Relative ordering of two messages delivered to any two group members is also undefined
As opposed to multiple, independent individual messages ...

- Group communications can provide a range of guarantees for:
  - Reliability
  - Ordering

Process groups

- Closed
  - If only members of the group may multicast to it
- Open
  - If processes outside the group may send messages to it
- Overlapping
  - Entities within a group may be members of multiple groups
- Non-overlapping groups: memberships do not overlap

Reliability in one-to-one communications?

- Integrity
  - Message received is the same as the one that was sent
- Validity
  - Any outgoing message is eventually delivered
- Duplicate elimination
  - Exactly one copy of the message is delivered

Group communications

- Extends semantics to cover delivery to multiple receivers
- Adds an additional property to the reliable delivery requirements
- Agreement
  - If message is delivered to one process, it is delivered to all processes in the group

Relative ordering of messages delivered to multiple destinations

- FIFO ordering
  - Preserve order from the perspective of a sending process
- Causal ordering
  - Based on the happens before relationship
- Total ordering
  - Same order will be preserved at all processes

Process Group Implementation Issues

Group communications

- Extends semantics to cover delivery to multiple receivers
- Adds an additional property to the reliable delivery requirements
- Agreement
  - If message is delivered to one process, it is delivered to all processes in the group
Group membership management [1/4]
- Provide interface for group membership changes
- Interfaces to:
  - Create and destroy process groups
  - Add or withdraw process from a group

Group membership management [2/4]
- Failure detection
  - Monitor group members
    - Not just for crash failures at the nodes
    - But also for unreachability due to communication failures
    - Relies on failures detectors to reach decisions about group membership
      - Detector marks processes as Suspected or Unsuspected

Group membership management [3/4]
- Notify members about membership changes
- When a process
  - Is added
  - Excluded
    - Due to failure detection
    - Deliberate withdrawal

Group membership management [4/4]
- Perform group address expansion
- When process multicasts, it supplies group identifier rather than a list of processes
- Membership service expands identifier into the current membership for delivery

WHY IP MULTICAST IS A WEAK CASE OF GROUP MEMBERSHIP SERVICE
- Allows processes to join or leave groups dynamically
- Performs address expansion
  - Senders only specify the IP multicast address as the destination of the message
  - But it does not:
    - Provide members with information about membership
    - Delivery is not coordinated with membership changes
    - Achieving these properties is referred to as view-synchronous group communication
The contents of this slide set are based on the following references