Frequently asked questions from the previous class survey

- How do new memory/hardware technologies impact threading?
- Use urgent data to cancel requests?
- Is message passing IPC? Can we implement using sockets?
- How is shared memory IPC done?

Topics covered in this lecture

- Distributed Servers
- Performance
- Messaging Systems

Mean time for failures and the premise for distributed servers

- Group several machines together
- Don’t rely on the availability of any single machine
- Together achieve better stability than each component individually
  - The sum is greater than the parts

Server Clusters

![Server Clusters Diagram]
Server Clusters

- Switch is also responsible for load balancing requests
- simplest way to do this is using round-robin
- If there are different services offered within the cluster?
- Switch needs to dispatch requests appropriately

But what about transparency?

- An important consideration is that the server cluster is transparent
- Clients typically set up network connections over which requests are sent

But TCP expects an answer from the switch not some arbitrary node

- When server responds to client
  - Inserts switch’s IP address in source field of the IP packet
  - Requires OS-level modifications
  - Also used in content-aware request distribution

The principle of TCP handoffs

When a cluster offers a single point ...

- When there is a failure at that access point?
  - The entire cluster becomes unavailable
- Several access points are typically provided
  - DNS can return several addresses all mapped to the same host name
  - Client makes several attempts if there are failures
  - Still requires static access points

Pulls and trade-offs

- Stability
  - Long lived access point
- Flexibility
  - Ability to configure a server cluster including the switch
What would be really nice
- Distributed server with a dynamically changing set of machines
- And also varying access points

Mobility support in IP version 6 (MIPv6)
- A mobile node has a home-network
- This node has a home-address
- The node has a home-agent
- Takes care of traffic to the mobile node while it is away

Mobility support in IP version 6 (MIPv6)
- When a mobile node attaches to a foreign network
  - Gets a temporary care-of address
- Care-of address reported to the home-agent
  - Forward all traffic to the mobile node

Apps communicating with mobile node only see the home address and not the care-of-address
- Offers a stable address for a distributed server
  - A single, unique contact address is initially assigned
- Contact address is server's lifetime address

Any node can act operate as the access point
- Record own address as the care-of address
- All traffic will be directed to the access point
- If there's a failure at the access point?
  - Another node takes over
- Potential bottlenecks?
  - Home agent and access point
  - All traffic must flow through them

The route optimization feature in MIPv6
- When a mobile node reports its care-of address (CA) to the home-agent (HA)
  - The HA reports the CA to a client
- Client keeps (HA, CA)
- Communications will be with the CA
  - Applications can still use the HA
  - MIPv6 protocol stack will translate HA to CA
Depicting Route Optimizations

Measures of performance
- Service time
- Latency
- Throughput
- Capacity
- Efficiency
- Scalability

Performance and Scalability
- Tuning for performance
  - Do some work with less effort
  - Caching, choice of algorithms $O(n^2)$ to $O(n \log n)$
- Scalability
  - Find ways to parallelize problem
  - Do more work with more resources

The quest for performance
- What do you mean by faster?
- Under what conditions?
  - Small or large datasets
  - Perform measurements to substantiate arguments
- How often do these conditions arise?
- What are the hidden costs?
  - Development/maintenance risks
  - Tradeoffs
  - Ripple effects of decision

How fast and how much
- Separate and can (at times) be at odds with each other
- To scale or for better hardware utilization
  - We often end up increasing the amount of work for each task
  - Divide tasks into multiple pipelined tasks
  - Orchestration overhead
Avoid premature optimizations
- First make it right, then fast
- Measure, don’t guess
- Quest for performance is one of the biggest sources of bugs

Amdahl’s Law

How much can we speed things up
- Harvesting crops
  - The more the number of workers
  - The faster the crop can be harvested
- But some things are fundamentally serial
  - Adding additional workers does not make the crop grow faster

Everything is not a nail
- Make sure that problem is amenable to parallel decomposition
- Most programs have a mix of parallelizable and serial portions

Amdahl’s law describes how much a program can be theoretically sped up
- \( F \): Fraction of components that must be executed serially
- \( N \): Number of available processors

\[
\text{Speedup} = \frac{1}{F + \frac{(1-F)}{N}}
\]

\[
\text{Utilization} = \frac{\text{Speedup}}{N}
\]

As \( N \) approaches infinity; maximum speedup converges to \( 1/F \)
- With 50% serial code
  - Maximum speedup is 2
- With 10% serial code
  - Maximum speedup is 10
    - With \( N = 10 \)
      - Speedup = 5.3 at 53% utilization
    - With \( N = 100 \)
      - Speedup = 9.2 at 9% utilization
Speedups for different parallelization portions

Know what to speed up

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

Interfaces to the transport layer

Socket primitives for TCP/IP
The communication pattern using sockets

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

Client
socket \\ connect \\ write \\ read \\ close

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

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

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

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

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
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 for reply</td>
</tr>
<tr>
<td>MPI_isend</td>
<td>Pass reference to outgoing message, and continue</td>
</tr>
<tr>
<td>MPI_isend</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 1100/8

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

- Pocket 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

October 12, 2017

The contents of this slide-set are based on the following references