Frequently asked questions from the previous class survey

- Synchronization vs Locks: Which one is better?
- Thread pools vs Creation of new threads
- Volatile
- Livelocks vs deadlocks: Which is worse? Debugging?
- Mutable state
- Time to create a thread vs Method invocations
- Is ConcurrentHashMap thread-safe?
- Is there a lighter weight concept than threads?
- Why no OS support for thread-safety inside a process?
- Thread-per-object is more thread-safe?
- Granularity of thread-synchronization
- Can thread priority be changed on the fly?
- How do you size thread pools when I/O is dominant?

Topics covered in this lecture

- Distributed Servers
- Performance
- Amdahl’s law
- Peer to Peer (P2P) Systems
  - Generations

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

- Logical switch
- Application/compute servers
- Distributed file/database

Client Requests

SLIDES CREATED BY: SHRIDEEP PALICKARA
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 server cluster is **transparent**
- Clients typically set up network connections over which requests are sent

The principle of TCP handoffs

- Logically a single TCP connection
- Client
- Switch (handed off)
- Server
- Response

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

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**

- **Client 1**
  - Believes server has address HA
  - Believes it is connected to X
  - Believes location of X is CA1

- **Client 2**
  - Believes server has address HA
  - Believes it is connected to X
  - Believes location of X is CA2

- **Distributed Server X**
  - Knows that Client 1 believes it is X
  - Access point with address CA1

- **Internet**
  - Access point with address CA2

**Performance**

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

**Performance and Scalability**
- Tuning for performance
  - Do same 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
Avoid premature optimizations

- First make it right, then fast
- Measure, don’t guess
- Quest for performance is one of the biggest source of bugs

Amdahl’s Law

September 10, 2015
CS555: Distributed Systems [Fall 2015]
Dept. Of Computer Science, Colorado State University
L6.28

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

Source: http://en.wikipedia.org/wiki/Amdahl's_law

September 10, 2015
CS555: Distributed Systems
Dept. Of Computer Science, Colorado State University

Know what to speed up

Two Independent parts
- Original process
- Make B 5x faster
- Make A 2x faster

Image from: http://en.wikipedia.org/wiki/Amdahl's_law

September 10, 2015
CS555: Distributed Systems
Dept. Of Computer Science, Colorado State University

PEER-TO-PEER (P2P) SYSTEMS

P2P systems
- Supports the construction of distributed systems
- Data and computational resources are contributed by many hosts
  - All participate in the provisioning of a uniform service

P2P systems
- Ability to share computing resources, storage, and data
  - Present in computers at the “edges of the internet”
- Have been used in several applications such as
  - File sharing, web caching, information distribution
  - 10s of thousands of machines harnessed by these applications

Goals
- Demand for Internet Services continues to grow
  - Scope for expanding popular services is limited when all hosts must be owned and managed by provider
- P2P systems aim to enable sharing of data and resources at a very large scale
  - They do so by eliminating requirements for separately managed servers and their associated infrastructure
P2P systems provide access to information resources

- Information located on computers throughout a network
- Algorithms for placement and retrieval of objects are a key aspect of system design

Traditional client-server systems

- Single computer or a cluster of tightly-coupled servers
- Simple decisions relating to the placement of resources
- Scale of service is limited by:
  - Server hardware capacity
  - Network connectivity

The delivered service must be

- Fully decentralized
- Self-organizing
- Dynamically balance storage and processing loads between all participating computers
  - Even as computers join and leave the service

Characteristics of P2P systems

- Each node contributes resources to the system
- Each node may differ in the quality of the resource that they contribute
  - But every node has the same functional capabilities and responsibilities
- Correct operation does not depend on the existence of any centrally administered systems
- Can be designed to provide a limited degree of anonymity to providers and users of resources

Key issue for the efficient operation of P2P systems

- Choice of algorithm for the placement of data across many hosts
- Subsequent access to the data in a manner that balances workload
  - Ensure availability without adding undue overheads
Coping with volatile resources in P2P systems

- Computers and network connections in P2P systems are owned by different entities
- A single node can become unavailable at any time
- P2P systems do not rely on guaranteed access to individual resources
- They are designed to make probability of failure to access a copy of a replicated object arbitrarily small
- Degree of resistance to tampering by malicious nodes

Realizing the potential of P2P systems

- Emerged when significant number of users had acquired always-on, broadband connections
- Made their desktops suitable for resource sharing
- TIMELINES
  - In the US, this occurred around 1999
  - By mid-2004, worldwide number of broadband connections exceeded 100 million

P2P Generations

- 1st Generation
  - Napster music exchange service
- 2nd Generation
  - Offered greater scalability, anonymity, and fault tolerance
    - Freenet, Gnutella, and BitTorrent

The 3rd Generation of P2P systems

- Emergence of middleware layers for application independent management of distributed resources
- Examples
  - Chord [Stoica et al. 2001]
  - Pastry [Rowstron and Druschel 2001]
  - Tapestry [Zhao et al. 2004]
  - Khademlia [Maymounkov and Mazières 2002]
Unlike 2nd generation systems, 3rd generation P2P systems:
- Provide guarantees of delivery for requests in a bounded number of network hops.
- Place replicas of resources on hosts in a structured manner taking account of their:
  - Volatile availability
  - Variable trustworthiness
  - Requirements for load balancing
  - Locality of information storage and use.

3rd Generation P2P systems: Resources are identified by globally unique identifiers (GUIDs):
- Derived as a secure hash from some or all of the resource's state.
- Make a resource self-certifying:
  - Clients receiving a resource can check the validity of the hash.
  - Protects it against tampering by untrusted nodes on which it might be stored.
  - Requires that states of the resource are immutable.
    - Change to the state will result in a different hash value.

Use of objects with changing values:
- Is much more challenging.
- Requires addition of trusted servers to manage sequence of versions.
  - Use this to identify the most current version.

Availability:
- Must avoid situations in which all replicas of an object are simultaneously unavailable.
- Use of randomly generated GUIDs assists by distributing object replicas:
  - To randomly located nodes.
  - If the underlying network spans multiple domains?
    - Risk of simultaneous unavailability is reduced significantly.

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