LARGE SCALE COMPUTING SYSTEMS OVERVIEW
Outline

- Course Introduction
- Overview and Examples of Large Scale Systems
- Problems and Solutions
- Architecture Patterns
Course Introduction
What is this course all about?

- Focus on the architecture of large systems
- Answer the following questions
  - How can we provide service for millions of users?
  - What are the issues we should consider?
  - How can we expand our systems to support more users?
  - What are the current trends in this area?
- No textbook, we are talking about the state of the arts of many issues
3 Pillars of this Course

Large-Scale Computing Systems

<table>
<thead>
<tr>
<th>Large-Scale Architecture</th>
<th>High-Performance Cluster</th>
<th>Scalable Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Large-Scale Internet Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Cloud Computing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Scalable Data Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Content Caching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Cluster Architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Cluster Usage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Cluster Administration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Cluster Programming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Distributing Algorithms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Map Reduce Framework</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Volunteer Computing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Case Studies</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scores and Assignments

- 3 exams (30 each) = 90%
- Assignments = 10%
What is large-scale computing?

- A computing system that can
  - support large amount of workloads
    - user requests / submitted tasks / service requests
  - provide reliable services
    - guarantee SLA (Service Level Agreement)
  - manage large amount of data
    - data for search engine / data mining / business intelligence
  - Involve in large sets of (distributed) resources
    - 10,000+ computing resources
  - Some or all of the above
Why do we need large scale system?

- Online services are mandatory
  - Online Registration system
  - Online Ticketing system
- Internet allows business to reach more customers at all time
  - Internet banking
  - eGovernment
Why do we need large scale system?

- The booming of social networks and online services
  - Facebook serves 570 billion page views per month
  - YouTube reaches 1 billion viewers per day (>10,000 views per second)
  - Amazon has more than 55 million active customer accounts
  - Playfish has more than 10 million players a day
Real World Example: Twitter

- SMS of the Internet
- Sending short 140-character message to followers
- Start in 2006
- From 120,000 tweets/month (in 2007) to 1,500,000,000 tweets/day (in 2010) - 750 tweets/second
- 300,000 new subscribers a day
- Just reach 20,000,000,000 tweets in July 31st, 2010 (a Japanese graphic designer)
Twitter vs. World Cup 2010

2110414 - Large Scale Computing Systems
FAIL WHALE

Twitter: Failure is an option. At least once a day, or whenever you need it.
Real World Example: Fastest Computer in the World

- Tianhe-1A (China)
  - 2.5 petaFlops (sustained)
  - CPUs
    - 14,336 Xeon X5670
    - 7,168 Nvidia Tesla
    - 2,048 NUDT FT1000
  - Memory: 262TB
Problems and Solutions
Requirements of Large Scale Computing Systems

- Focus on throughput
- Need high-availability
- Must be scalable
- Simple to manage
Throughput Oriented

- When you have many users, throughput is as important as service time
  - Upgrading systems allows system to serve requests faster (shorter service time + more throughput)
  - Adding more resources allows system to serve more requests (same service time + more throughput)
  - Similar to pipeline technique
Example: customers using ATM service

Suppose there are 2 workloads
- light = every 5 mins, 1 custs arrive
- heavy = every 5 mins, 2 custs arrive

2 server types
- slow = each customer spends 2 mins
- fast = each customer spends 1 min

Consider 3 cases
- Case 1: single slow ATM
- Case 2: single fast ATM
- Case 3: two slow ATMs
## Delay vs. Throughput

<table>
<thead>
<tr>
<th>Case</th>
<th>Light Workload</th>
<th></th>
<th>Heavy Workload</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wait</td>
<td>ATM</td>
<td>Total</td>
<td>Wait</td>
</tr>
<tr>
<td>1. One slow ATM</td>
<td>1m 20s</td>
<td>2m</td>
<td>3m 20s</td>
<td>8m</td>
</tr>
<tr>
<td>2. One fast ATM</td>
<td>20s</td>
<td>1m</td>
<td>1m 20s</td>
<td>40s</td>
</tr>
<tr>
<td>3. Two slow ATMs</td>
<td>5s</td>
<td>2m</td>
<td>2m 5s</td>
<td>23s</td>
</tr>
</tbody>
</table>

Large Scale Computing Systems

2110414 - Large Scale Computing Systems
Availability

- System must be able to provide services for certain period of time
- There are 2 possible status: uptime and downtime
- Downtime includes any time that user cannot use or access the system
  - Any failures
  - Maintenance period
- Availability = uptime / (uptime + downtime)
## Availability

<table>
<thead>
<tr>
<th>Availability %</th>
<th>Downtime per year</th>
<th>Downtime per month*</th>
<th>Downtime per week</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>36.5 days</td>
<td>72 hours</td>
<td>16.8 hours</td>
</tr>
<tr>
<td>95%</td>
<td>18.25 days</td>
<td>36 hours</td>
<td>8.4 hours</td>
</tr>
<tr>
<td>98%</td>
<td>7.30 days</td>
<td>14.4 hours</td>
<td>3.36 hours</td>
</tr>
<tr>
<td>99%</td>
<td>3.65 days</td>
<td>7.20 hours</td>
<td>1.68 hours</td>
</tr>
<tr>
<td>99.5%</td>
<td>1.83 days</td>
<td>3.60 hours</td>
<td>50.4 min</td>
</tr>
<tr>
<td>99.8%</td>
<td>17.52 hours</td>
<td>86.23 min</td>
<td>20.16 min</td>
</tr>
<tr>
<td>99.9% (&quot;three nines&quot;)</td>
<td>8.76 hours</td>
<td>43.2 min</td>
<td>10.1 min</td>
</tr>
<tr>
<td>99.95%</td>
<td>4.38 hours</td>
<td>21.56 min</td>
<td>5.04 min</td>
</tr>
<tr>
<td>99.99% (&quot;four nines&quot;)</td>
<td>52.6 min</td>
<td>4.32 min</td>
<td>1.01 min</td>
</tr>
<tr>
<td>99.999% (&quot;five nines&quot;)</td>
<td>5.26 min</td>
<td>25.9 s</td>
<td>6.05 s</td>
</tr>
<tr>
<td>99.9999% (&quot;six nines&quot;)</td>
<td>31.5 s</td>
<td>2.59 s</td>
<td>0.605 s</td>
</tr>
</tbody>
</table>
Scalability

- The ability to either handle growing amounts of work in a graceful manner or to be enlarged
- If we have more users, can we add some resources to the system to support the user’s growth?
Scalability Approaches

- **Scale-Up (Vertical scale)**
  - Adding more resources without changing number of servers
  - Simple, but can be expensive (sometimes)

- **Scale-Out (Horizontal scale)**
  - Increase number of servers
  - All servers are usually identical
  - More cost effective, but require the right design + hardware support
Manageability

- **Operational**
  - Monitoring
  - Cooling issues

- **Cost**
  - Electricity

- **Maintainability**
  - How difficult is it for the admin to deploy, maintain, and upgrade the system?
Design Principles

- Workload Partitioning
- Relaxed Data Consistency
- Effective Resource Management
- Memory/Storage Hierarchy
Workload Partitioning

- We can improve performance by distributing workloads to many servers

- Two possible approaches
  - Vertical partitioning
  - Horizontal partitioning
Vertical vs. Horizontal Partitioning

**Vertical Partitioning**
- Client
  - Web Browser
  - Web Server
  - AppServer
  - Database

Simple, but limited

**Horizontal Partitioning**
- Client
  - Web Browser
  - Web Server
  - AppServer
  - Database

Improve availability, may require special hardware
Systems with multiple data sources usually require strong consistency

- All servers must have the same data (user can send request to any server and expect the same result)
- Consistency algorithm slows down the entire system

What if we have 10,000 servers?
Relaxed Data Consistency

- Some services can rely on relaxed data consistency
  - Data on all servers are not required to be exactly the same at all time
  - Still perform data consistency algorithm, but at less frequent periods
  - Google search engine (400,000 servers)

- Some services utilize data partitioning (data shards)
  - Data are distributed across multiple servers
  - Each data resides on only one server
  - Require intelligence hardware to distribute requests correctly
  - Used by many high-end relational databases
Effective Resource Management

- Traditional transaction systems are synchronous and stateful operations
  - Synchronous operation holds server’s resources until the operation is complete
  - Stateful operation occupies server’s resources until user logouts
- Modern architectures rely on
  - Asynchronous operations
  - Stateless operation
Memory/Storage Hierarchy

- Large scale systems require data transfer across the networks
- Network latency is non-uniform by nature
  - Client usually has to connect to servers via WAN
  - LAN has shorter latency and more bandwidth than WAN
- Utilize caching system for latency reduction and bandwidth saving
  - Web Caching
  - Memcached (to be explained in other session)
Web Caching Structure

[Diagram showing web caching structure with labels:
1. Web browsers
2. Proxy caches
3. Caching load balancer
4. Origin servers

Client-side caching
Server-side caching

Internet

Connection to Internet or intranet web servers]

2110414 - Large Scale Computing Systems
Architecture Patterns and Case Study
Popular Architecture Patterns

- Web Based Architecture
- Cluster Computing
- Peer-to-Peer Architecture
- Service Oriented Architecture
- Cloud Computing
Web Based Architecture

- Based on multi-tier architecture
  - Presentation
  - Business
  - Data Access
- Very popular and standard
  - A lots of frameworks
Cluster Computing

- Group of computers working together via network as a single computer
- Very popular and cost effective
  - Utilize COTS
  - Dominate top 500
Peer-to-Peer Architecture

- All participants become both clients and servers
- Popular among data sharing
  - Napster
  - Bittorrent
  - P2P IPTV
- Require special P2P algorithms and structures
Service Oriented Architecture

- Very vague but popular architecture
- Everything is “service”
- Promise a lot of great things
  - Flexible
  - Extensible
- Not really deliver
  - Too complex
Cloud Computing

- Current trends
- Provide all infrastructures, resources and services via the Internet technology
  (mostly) use web browser as a front-end
- Utilize resource sharing based on virtualization concepts
- Allow cheap “on-demand” resources and dynamic scalability
Sample Cloud Services
Sample Cloud Services

2110414 - Large Scale Computing Systems
References