Long Tail of Latency

Day 15
Agenda

• Why is Latency Important?

• Latency in Data Centers

• Reducing Latency through duplications
  – Duplicate Requests
  – Duplicate Storage
Cost of Additional Latency

- +50ms additional latency is okay
- +100ms or more leads to problems
  - Fewer clicks and follow through
  - Smaller revenue

- Important to keep latency low!!!
What Contributes To Latency?

The Internet!!!!!
What Contributes To Latency?

This Internet!

- Within an ISP: RCP, TeXCP
- Across ISP: Overlay Networks, BGP oscillations issues
# Ping Times to Some servers

<table>
<thead>
<tr>
<th>Service</th>
<th>Round Trip Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>10ms</td>
</tr>
<tr>
<td>Yahoo</td>
<td>37ms</td>
</tr>
<tr>
<td>Facebook</td>
<td>16ms</td>
</tr>
<tr>
<td>CNN</td>
<td>16ms</td>
</tr>
</tbody>
</table>

Much less than 50ms!!!! Why are we worried about latency?
What Contributes To Latency?
This Internet!

- Within an ISP: RCP, TeXCP
- Across ISP: Overlay Networks, BGP oscillations issues
What Contributes To Latency?
TCP and WebProtocol

- Network Multipath: MpTCP
- TCP Overheads: TFO
- Networks with losses: Reducing Web Latency
- Web protocols: SPDY
Server → Data Center

http://www.google.com/about/datacenters/gallery/#/all/14
Agenda

• Why is Latency Important?

• Latency in Data Centers

• Reducing Latency through duplications
  – Duplicate Requests
  – Duplicate Storage
What is a Data Center?

- **Servers**
  - Run multiple Applications
  - Run background jobs

- **Switches**
  - Connect servers together

Image courtesy of Mohammad Alizadeh
Source of Latency Within Data Center

**Server Issues**
- **Background jobs:**
  - E.g. back-up storage (daemon), clean up garbage, update software (maintenance)

- **Shared resources**
  - Imperfect sharing/scheduling

- **Bad Hardware:**
  - E.g. failing disk

- **Power Saving (energy management):**
  - Slow down CPU to save energy

**Network Issues (Global Resource)**
- **Not enough resources (e.g BW):**
  - Network devices are expensive
  - Day 16

- **Inefficient network protocol:**
  - Think more TCP overheads
  - Day 17

- **Physical server limits:**
  - Many-to-one problem: InCast
  - Day 17
How Do Request Get Processed in a Data Center
How Do Request Get Processed in a Data Center

Worker: mappers
Aggregation: reducers

find find find find
sort sort sort sort
merge

States A-M
States N-V
States W-Z
States A-M
States N-V
User-facing online services

**Application SLAs**
SLAs for components at each level of the hierarchy

**Cascading SLAs**

**Network SLAs**
Deadlines on communications between components

---

Other implications of Network Limits

Scalability

Scalability of Netflix-like recommendation system is bottlenecked by communication

Communication time increased faster than computation time decreased

Today's transport protocols: Deadline agnostic and strive for fairness

Application SLAs

Cascading SLAs

Network SLAs

Deadlines on communications between components
Importance of Tail Latency

• Tail Latency == 1 in X servers being slow

• Why is this bad?
Importance of Tail Latency

• Tail Latency == 1 in X servers being slow

• Why is this bad?
  – Each user request now needs several servers
  – Changes of experience tail is much higher

• If one in 100 servers has high latency (1% are bad)
  – If users needs 100 partitions then chances of latency is (63%): MUCH HIGHER!!!!
Respond with “Good Enough” Results

• Better to give the user less than perfect results rather than lose the user

• If a machine doesn’t respond before its deadline ignore it
Basic Latency Reduction Techniques

• Use priority queues
  – (Think HOV lanes)
  – User traffic Higher priority
  – Background traffic low priority

• Reduce head of line blocking
  – Break large requests into smaller ones

• Rate-limit background activity
• Stop low priority until high priority is done
# Source of Latency Within Data Center

## Server Issues
- **Background jobs:**
  - E.g. back-up storage (daemon), clean up garbage, update software (maintenance)
- **Shared resources**
  - Imperfect sharing/scheduling
- **Bad Hardware:**
  - E.g. failing disk
- **Power Saving (energy management):**
  - Slow down CPU to save energy

## Solutions
- Make them happen at the same time. Only affect jobs running at that time.
- Quarantine bad machines
- Minimize power savings
Agenda

• Why is Latency Important?

• Latency in Data Centers

• Reducing Latency through duplications
  – Duplicate Requests
  – Duplicate Storage
Dealing with Slow Processing With Replication

• Replicate Processing
  – If a request is slow: Start a new one!!
  – New request may run on a machine with no problems

• Why is this insufficient?
Dealing with Slow Processing With Replication

• All requests process data: e.g. queries about state tax processes US state data.
• Duplicating the request may not help if the new request uses the same data.
• **We need to perform data replication also.**
Agenda

• Why is Latency Important?

• Latency in Data Centers

• Reducing Latency through duplications
  – Duplicate Requests
  – Duplicate Storage
How to Replicate Processing?

• When to start replication?

• How many replicas to make?

• How to deal with replica results?

• Replicas waste resources: how to minimize waste?

Next set of slides are from Jeff Dean’s Achieving Rapid Response Times in Large Online Services
Backup Requests

Replica 1
- req 3
- req 6

Replica 2
- req 5

Replica 3
- req 8

Req 9

Monday, March 26, 2012
Backup Requests

Replica 1
- req 3
- req 6
- req 9

Replica 2
- req 5

Replica 3
- req 8

Client
Backup Requests

Replica 1

Replica 2

Replica 3

Client

req 3
req 6
req 9

req 5
req 9

req 8

Monday, March 26, 2012
Backup Requests

Replica 1
- req 3
- req 6
- req 9

Client

Replica 2
- reply

Replica 3
- req 8
Backup Requests

Replica 1

Replica 2

Replica 3

“Cancel req 9”

reply
Backup Requests Effects

• In-memory BigTable lookups
  – data replicated in two in-memory tables
  – issue requests for 1000 keys spread across 100 tablets
  – measure elapsed time until data for last key arrives

<table>
<thead>
<tr>
<th></th>
<th>Avg</th>
<th>Std Dev</th>
<th>95%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>No backups</td>
<td>33 ms</td>
<td>1524 ms</td>
<td>24 ms</td>
<td>52 ms</td>
<td>994 ms</td>
</tr>
<tr>
<td>Backup after 10 ms</td>
<td>14 ms</td>
<td>4 ms</td>
<td>20 ms</td>
<td>23 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>Backup after 50 ms</td>
<td>16 ms</td>
<td>12 ms</td>
<td>57 ms</td>
<td>63 ms</td>
<td>68 ms</td>
</tr>
</tbody>
</table>

• Modest increase in request load:
  – 10 ms delay: <5% extra requests; 50 ms delay: <1%
Can we reduce the back-up time even further? Maybe 0ms? How do we minimize overheads?

<table>
<thead>
<tr>
<th></th>
<th>Avg</th>
<th>Std Dev</th>
<th>95%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>No backups</td>
<td>33 ms</td>
<td>1524 ms</td>
<td>24 ms</td>
<td>52 ms</td>
<td>994 ms</td>
</tr>
<tr>
<td>Backup after 10 ms</td>
<td>14 ms</td>
<td>4 ms</td>
<td>20 ms</td>
<td>23 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>Backup after 50 ms</td>
<td>16 ms</td>
<td>12 ms</td>
<td>57 ms</td>
<td>63 ms</td>
<td>68 ms</td>
</tr>
</tbody>
</table>

- Modest increase in request load:
  - 10 ms delay: <5% extra requests; 50 ms delay: <1%
Backup Requests w/ Cross-Server Cancellation

Server 1

req 3

req 6

Server 2

req 5

req 9

Monday, March 26, 2012
Backup Requests w/ Cross-Server Cancellation

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Server 1
- req 3
- req 6
- req 9 also: server 2

Server 2
- req 9 also: server 1

Client

"Server 2: Starting req 9"

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Server 1

```
req 3
req 6
req 9
also: server 2
```

"Server 2: Starting req 9"

Server 2

```
req 9
also: server 1
```

Client

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Server 1
- req 3
- req 6

Server 2
- req 9
  - also: server 1

Client

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

- Read operations in distributed file system client
  - send request to first replica
  - wait 2 ms, and send to second replica
  - servers cancel request on other replica when starting read
- Time for bigtable monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td>Backup after 2 ms</td>
<td></td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
</tbody>
</table>

-43%
When Can this Go Wrong?
Agenda

• Why is Latency Important?

• Latency in Data Centers

• Reducing Latency through duplications
  – Duplicate Requests
  – Duplicate Storage
How to Replicate Storage?

• Which data to replicate?

• Where to place the replicated data?

• Replicas waste resources: how to minimize waste?

Next set of slides are from Jeff Dean’s Achieving Rapid Response Times in Large Online Services
Storage Issues

- What happens if all queries are for Wisconsin?
Popularity Skew

- According to Microsoft’s data
- Top 12% is 10x more popular than bottom third

Graph from Scarlett: Coping with Skewed Content Popularity in MapReduce Clusters
Solution: Make Copies of Popular Content

• If “W” is popular, I make copies of them:

Where to put this new chunk?

- States A-M
- States N-V
- States W-Z
- States A-M
- States N-V
Solution: Make Copies of Popular Content

• If “W” is popular, I make copies of them:
  – Avoiding putting both copies on the same server
  – Avoid putting the copy on a server with other popular content (Load Balancing)
Load balance chunk across servers

• Calculate predicted ‘load’: \textbf{Total Access} \times \textbf{Size}
  – Place on replica chunks on least ‘loaded’
When to Replicate Storage Chunks?

• Automated:
  – Monitor utilization of chunks
  – Replicate more utilized chunks

• Static:
  – Always replicate chunks of a particular type
Concluding Remarks

• Tail Latency is costly → Users will leave the system.

• Several approaches to improve tail latency leverage replication

• Replicate improves overheads, why are they acceptable?
  – Replication is also used to tackle failures:
  – These same copies can be used to tolerate variability
  – Times scales are very different:
    • Variability: requests with performance issues happen frequently: 1000s of disruptions/sec, scale of milliseconds
    • Faults: failure happen infrequently: 10s of failures per day, scale of tens of seconds
Reminder

• Project Proposal Due Tomorrow @ Noon!!!