Google Megastore & Google Spanner
The Problem

I have a great app idea, but I don’t want to learn much about databases. Can’t it just be easy and scalable and reliable?
Because that’s impossible!

Consistency

Availability

Scalability

RDBMS

NoSQL
Google Megastore

- Highly Scalable
- Rapid Development
- Low Latency
- Consistent
- Highly Available
BigTable, Google’s distributed KV store

```
BEGIN BigTable

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t3,</td>
<td>t4,</td>
<td>t5,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;&lt;html&gt;...&quot;</td>
<td>&quot;CNN&quot;</td>
<td>&quot;CNN.com&quot;</td>
</tr>
<tr>
<td></td>
<td>t9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t8</td>
</tr>
</tbody>
</table>

END BigTable
```
Paxos Basics - Read and Write
Read, Phase 1 - Reading replica polls other replicas
Read, Phase 2 - Available replicas respond with their values
Write, Prepare - Writing replica asks other replicas to conduct vote
Write, Promise -- Available replicas promise to ignore lower proposals
Write, Accept - Writing replica proposes its value
Write, Commit - Available replicas accept the value
Paxos Basics -- Write Conflicts
Prepare - 2 writing replicas want to make proposals
Promise
More prepares - Lower numbered proposals get rejected / replaced
Accept - Only 1 replica gets the majority of promises required
How could Paxos be made more efficient?
Local Reads via a “coordinator”
Local replica up-to-date
Local replica out-of-date
Local replica out-of-date - normal Paxos read
Local replica out-of-date - update replica and coordinator
Faster Writes: Accept & Prepare in 1 communication
Writing replica requests to be proposal 0
Immediate Accept & Prepare for next round
Megastore Performance Analysis
Our experimental setup is that we’ve used Megastore for 100+ applications for several years and it works!
Figure 9: Distribution of Availability
Figure 10: Distribution of Average Latencies
What are some drawbacks to this solution?
Google Megastore

- Highly Scalable
- Rapid Development
- Low Latency
- Consistent
- Highly Available
Google Spanner

- Highly Scalable
- Rapid Development
- Low Latency
- Consistent
- Highly Available
What time is it?

TT.now() → TTinterval:

Lower bound  Absolute time  Upper bound

2016-02-10, 4:35:30  2016-02-10, 4:35:31  2016-02-10, 4:35:32  2016-02-10, 4:35:33
Improved Paxos: Serialization with Locking
Improved Paxos: Long Leader Leases
Spanner Performance Analysis
Setup
<table>
<thead>
<tr>
<th>replicas</th>
<th>latency (ms)</th>
<th>throughput (Kops/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>write</td>
<td>read-only transaction</td>
</tr>
<tr>
<td>1D</td>
<td>9.4±.6</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>14.4±1.0</td>
<td>1.4±.1</td>
</tr>
<tr>
<td>3</td>
<td>13.9±.6</td>
<td>1.3±.1</td>
</tr>
<tr>
<td>5</td>
<td>14.4±.4</td>
<td>1.4±.05</td>
</tr>
</tbody>
</table>

Table 3: Operation microbenchmarks. Mean and standard deviation over 10 runs. 1D means one replica with commit wait disabled.
Figure 5: Effect of killing servers on throughput.
Next steps?
Supporting more complex SQL queries with an underlying key-value structure
The Future

SQL

NoSQL

NewSQL