MIDTERM

— Mon 3/18 2:00pm-3:15pm, Olsson Hall 011
— Based upon lecture materials and projects (lab 1)
— No laptops, cellphones or other electronic devices.
— You are allowed to bring one US letter or similar size double-sided note.

— Do not cheat
COVERAGE

— Cover topics before 2PC
  — 2PC is not included

— How to use today's slides:
  — Use it as a basis to develop your cheatsheet
  — Use it to self-test
INTRODUCTION TO CLOUD

— Difference between public and private cloud
— Why we need cloud computing
— What a single-site cloud (datacenter) consists of? distributed?
— Where does the idea of cloud computing come from?
INTRODUCTION TO CLOUD

— Difference between public and private cloud
— Why we need cloud computing
— What a single-site cloud (datacenter) consists of? distributed?
— Where does the idea of cloud computing come from?
  — Timesharing, Grids, Clusters, P2P...
— What are the trends? (hardware, users)
— Elaborate on four new features in today's cloud
— Can you tell difference between *AAS?
# INTRODUCTION TO CLOUD

<table>
<thead>
<tr>
<th>On-Premises</th>
<th>IaaS (Infrastructure as a Service)</th>
<th>PaaS (Platform as a Service)</th>
<th>SaaS (Software as a Service)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>Applications</td>
<td>Applications</td>
<td>Applications</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Runtime</td>
<td>Runtime</td>
<td>Runtime</td>
<td>Runtime</td>
</tr>
<tr>
<td>Middleware</td>
<td>Middleware</td>
<td>Middleware</td>
<td>Middleware</td>
</tr>
<tr>
<td>O/S</td>
<td>O/S</td>
<td>O/S</td>
<td>O/S</td>
</tr>
<tr>
<td>Virtualization</td>
<td>Virtualization</td>
<td>Virtualization</td>
<td>Virtualization</td>
</tr>
<tr>
<td>Servers</td>
<td>Servers</td>
<td>Servers</td>
<td>Servers</td>
</tr>
<tr>
<td>Storage</td>
<td>Storage</td>
<td>Storage</td>
<td>Storage</td>
</tr>
<tr>
<td>Networking</td>
<td>Networking</td>
<td>Networking</td>
<td>Networking</td>
</tr>
</tbody>
</table>

BMC
What are basic goals and challenges of DS?
— What are basic goals and challenges of DS?
  — Scalability, Fault-tolerance, Consistency (remember the Starbucks example?)
— What are "scale-out" and "scale-up"? How to understand perfect scalability?
— How we improve performance/fault-tolerance/scalability for Web Service Architecture?
— How system properties change after we add cache/replication/sharding?
MAPREDUCE

— Why we need MapReduce?
— How MapReduce workflow works? Can you use examples to show?
— How to write Map() and Reduce() functions for different workloads?
  — and when to use chaining?
MAPREDUCE

(Exercise) You are given a symmetric social network (like Facebook) where a is a friend of b implies that b is also a friend of a. The input is a dataset D (sharded) containing such pairs (a, b) – note that either a or b may be a lexicographically lower name. Pairs appear exactly once and are not repeated. Find the last names of those users who have at least 5 friends with "Mike" as their first name.
MAPREDUCE

— How MapReduce deal with different design challenges?
  — How to improve performance?
  — How to deal with failures?
  — How to deal with slow nodes?
MAPREDUCE

— How MapReduce deal with different design challenges?
  — How to improve performance? => scheduling policy for data locality
  — How to deal with failures? => heartbeats + replication
  — How to deal with slow nodes? => redundant execution
RPC

— What are the downsides of socket programming?
— What is the key idea behind this abstraction?
— What are the steps of a RPC workflow?
— Comparison between LPC and RPC
— What are different semantics of RPC?
  — show failure scenarios for each semantics
  — how to implement them?
TRANSACTION

- What system challenges do transactions solve?
- What semantics do transactions provide?
- Given an example of client codes, can you give an example of possible failure/concurrency problem?
- How does WAL work?
  - why different logging schemes don't work?
  - show how WAL helps to recover crashes or handle aborts
TRANSACTION (CONTD.)

— How to check serial equivalence?
— How does 2PL work?
  — can you show how 2PL ensures serial equivalence by blocking other transactions?
  — how to resolve deadlock?
— How does timestamp ordering work?
TRANSACTION (CONTD.)

Transaction T1
x = getSeats(ABC123);
// x = 10
if(x > 1)
    x = x - 1;
write(x, ABC123);
commit

Transaction T2
x = getSeats(ABC123);
if(x > 1) //x = 10
    x = x - 1;
write(x, ABC123);
commit
TRANSFER(src, dst, x)
00  txID = Begin() ← lock(src)
01  src_bal = Read(txID, src)
02  if (src_bal > x):
03    src_bal -= x
04  Write(txID, src_bal, src)
05  dst_bal = Read(txID, dst) ← lock(dst)
06  dst_bal += x
07  Write(txID, dst_bal, dst)
09  return Commit(txID) ← unlock(src,dst)
10  Abort(txID) ← unlock(src,dst)
11  return FALSE
TRANSACTION (CONTD.)

**SCHEDULE**

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN R(A)</td>
<td>BEGIN W(A) COMMIT</td>
</tr>
<tr>
<td>W(A) R(A) COMMIT</td>
<td></td>
</tr>
</tbody>
</table>

**DATABASE**

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>R-TS</th>
<th>W-TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Violation:

TS(T1) < W-TS(A)
TIME AND SYNCHRONIZATION

— Why Physical clocks are hard to synchronize?
— what protocols to use?
— How does Lamport Clock Protocol works?
AGREEMENT

— What is the difference between atomic commitment and consensus?
— Explain the insight of "Green cup, Red cup" game solution.
— Explain what challenge does two generals problem demonstrate.
— Explain CAP theorem.
TAKEAWAYS

– We provide an extra office hour this Friday: 3pm-4:30pm (3/15)

– Next class: midterm exam