CPSC 416 Distributed Systems

Winter 2023 Term 1 (October 19, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer

Logistics

Teaching Assistants

Andy Hsu (andy.hsu@alumni.ubc.ca)

Hamid Ramezanikebrya (hamid@ece.ubc.ca)

Jonas Tai (jonastai@student.ubc.ca)

Cathy Yang (kaiqiany@student.ubc.ca)

Office Hours

Remember: Use Piazza for **all** official course-related communications

- Not on Piazza? Not official.
- Canvas “comments/messages” **are not monitored**

Office Hours:

Who	When	Where
Tony	Monday 14:00-15:00 Wednesday 16:00-17:00	Discord
Andy	Thursday 19:00-20:30	Discord
Hamid	Friday 16:30-18:00	Kaiser 4075
Jonas	Thursday 13:00-14:00	X241
Cathy	Friday 09:00-10:30	X237

Due Dates

This week

- Design Project 3 Team Declaration Deadline (Today @ 23:59)

Next week

- Design Project 3 Due Tuesday (October 24 @ 17:00)
- Design Project 2 Code Due Thursday (October 26 @ 17:00)
- Design Project 2 Implementation Report Due Thursday (October 26 @ 23:59)

Note:

- You are strongly encouraged to collaborate with others on this
- You should use tools at your disposal to answer these questions
- **Do not forget to submit it.**

Today’s Failure

Microsoft Azure

Date: September 4, 2018

Time: 08:42 UTC

Source: [Azure status history | Microsoft Azure (archive.org)]

Lightning storm caused “sags and swells” to voltage feeds in Azure’s South Central US Region (Texas)

By design, this caused the air conditioning to shut down:

At 08:42 UTC, lightning caused a large swell which was immediately followed by a large sag in one of the region’s datacenters, which fell below the required voltage specification for the chiller plant. By design this sag triggered the chillers to power down and lock out, to protect this equipment.

Microsoft Azure (September 4, 2018)

Backup cooling was in *manual* mode:

In the impacted datacenter, the redundant cooling system had been switched to manual mode for invoking the failover of cooling. It was set to a manual failover following the installation of new equipment in the facility where all testing had not been completed.

The warnings were not noticed, bad things happened.
Onsite engineers in the impacted datacenter received multiple alarms so were investigating several situations reported in the facility. Cooling alerts which should have been manually actioned were not. Consequently, temperatures began to rise, eventually reaching temperatures that triggered infrastructure devices and servers to shutdown.

Microsoft Azure September 4, 2018

Things got bad faster than expected:

This shutdown mechanism is intended to protect infrastructure from overheating but, in this instance, temperatures increased so quickly in parts of the datacenter that a number of storage servers were damaged, as well as a small number of network devices.

Recovery is hard:

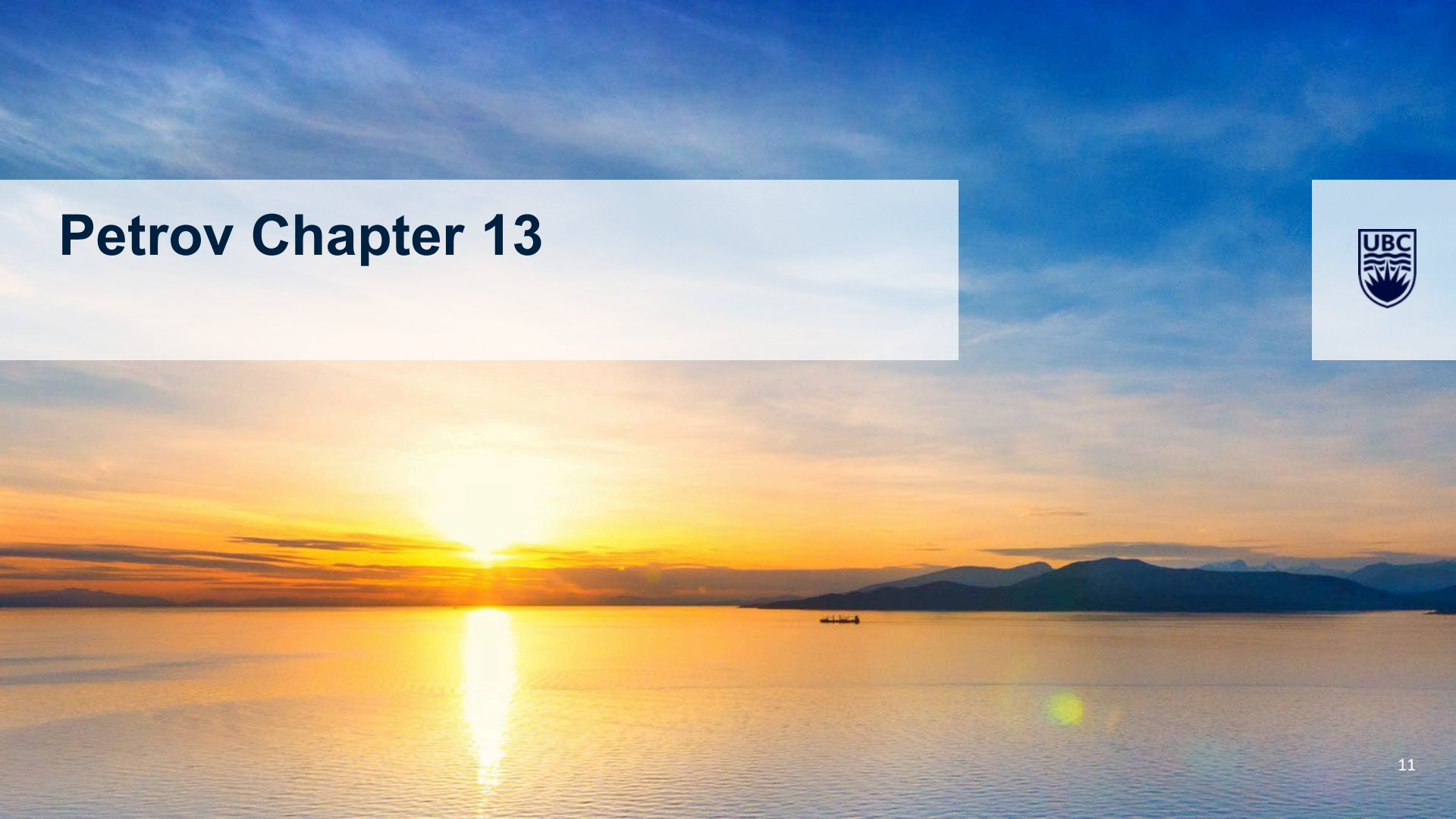
The decision was made to work towards recovery of data and not fail over to another datacenter, since a failover would have resulted in limited data loss due to the asynchronous nature of geo replication. Recovering the storage services involved replacing failed infrastructure components, migrating customer data from damaged servers to healthy servers, and validating the integrity of the recovered data. This process took time due to the number of servers that required manual intervention, and the need to work carefully to maintain customer data integrity above all else.

Microsoft Azure September 4, 2018

Takeaways:

- Unexpected things can happen at the worst of times
- Failures compound
- Recovery is slow and painful
- Core service failures lead to higher level service failures
  - VS Code/Visual Studio Marketplace
  - Azure Active Directory
  - Azure Service Manager
  - Etc.

Petrov Chapter 13



Learning Goals (Petrov Chapter 13)

Note: this is complementary with Kleppmann Chapter 7

Understand why we use transactions

Understand benefits and complexity of distributed transactions

Learn more about the techniques we use for distributed transactions

Atomicity

Atomic = “All or nothing” (literally “indivisible particle” in Latin)

Atomicity transitions from one “consistent state” to another “consistent state”

Observation:

- *Consistency* in systems means that the state of the system satisfies a set of invariants
  - Example: accounting records are required to “balance” across rows and columns
- Defining your *invariants* enables you to model and test
  - Formal modeling/state exploration
  - Assertions in your code

Two-Phase Commit (2PC)

Protocol Overview

Roles

- Coordinator
- Cohort

Phases:

- Initialization
- Prepare – sent to all cohorts
  - Cohort votes: yes or no
- Outcome
  - Unanimous yes
  - Non-unanimous yes (no, timeout)
- Record decision
- Send commit/abort message

Failures in 2PC

Cohort Failures

- Examples:
  - Network/node failure
  - Failure after voting
- Coordinator can handle
  - Wait for final response to decision
  - (or) implement recovery mechanism

Coordinator Failures

- Fail before prepare: no op
- Fail after prepare, before decision: cohorts wait
- Fail after decision, before acknowledgement: outcome uncertain
- Timeouts/recovery/backup coordinator

Three-Phase Commit (3PC)

Problem solved by 3PC:

- Coordinator (“pre-commit” phase)
- Allows “timeouts” and other recovery mechanisms

Protocol:

- Initialization
- CanCommit
- PreCommit (or Abort)
- DoCommit (or Abort)

Non-blocking, timeouts, allows coordinator replacement

Calvin and Distributed Transactions

Decouples transaction coordination from storage

Uses *deterministic locking* to avoid deadlocks

- Avoids cyclic lock acquisition

Uses *deterministic scheduling* to avoid 2PC usage

- Exploits known order of operations
- Eliminates transactions when possible

Transaction sequencer: define a global ordering of events (in this case “transactions.”)

Replication: primary/backup and multi-write.

Recovery/Fault tolerance: linear event schedule simplifies replication and recovery

Good performance – avoiding 2PC and making log replication with RSMs easy

Limitations: latency, assumes easy-to-sequence transactions (not true with contention)

Spanner and Distributed Transactions

Global clock (bounded)

- Globally consistent timestamps
- Known ϵ bound for time skew
- Linearizability across globally distributed data

Horizontal scalability (“sharding”)

Global transactions (ACID support)

Synchronous replication

SQL support

Integrated with GCP

Dynamic schema change support

TrueTime has limitations

Database Partitioning

Database constructed of multiple databases

- Horizontal partitioning: row level partitioning (“shard”)
- Vertical partitioning: column level partitioning

Sharding is quite common – KV stores shard quite well

Consistent Hashing – dynamic sizing of database partitioning to minimize data movement.

Benefits: Scalability, Performance, Manageability, Availability

Challenges: Data distribution, SQL join operation, locality management, rebalancing

Percolator and Distributed Transactions

Percolator: allows continuous updating of Google's search index

Transactions:

- Distributed transactions over rows/tables in Bigtable
- Uses locks, timestamps, and metadata operations
- Relies upon a centralized timestamp service – guarantees global ordering
- Supports 2PC
- Relies upon Bigtable for atomic operations (commit markers)

Observer Framework ("like snapshot isolation")

Advantages: incremental processing, scalability, strong consistency

Challenges: transactions aren't free (in fact they're complex)

Note: shows how transactions can be added to non-transactional system

Coordination Avoidance

Coordination: “agreeing to something” (consensus, locking)

Disadvantages:

- Relies on network communications (slow, dropped/misordered packets, etc.)
- Node failure

Define: “Invariant Confluence”

- States can be merged without violating system invariants *and* no coordination

Check Invariant Confluence

- State space checking: can merging a pair of operations be done to *any* database state without violating invariants.

Benefits: performance, lower latency, higher availability

Trade-offs: weaker consistency, defining invariants can be *hard*

Examples: NoSQL databases

Questions?

How to use this template

Please note: This template has a variety of slides for your use. To select what slide you would like, click on the drop down menu beside “new slide” button in the top left corner, and pick the corresponding slide. To insert text, simply double click on the text box and start typing. Please be aware that copying and pasting text may change how the font looks. It is better to type directly onto the slide. Also note that larger fonts (size 14+) work better for presentations than smaller sizes. This template uses the font Arial, as PowerPoint users will experience technical difficulties if using UBC’s official fonts. If desired, images can replaced by going into the “Master” view and applying your own image. Please ensure you have the rights to an image before using it.

The following slides are here for visual reference only. Please delete or edit as needed for your own presentation. If you have any questions about how to use this template, please contact UBC Communications and Marketing at comm.marketing@ubc.ca

Insert title here

Insert subtitle here

Name, position

Insert title here

Insert subtitle here

Name, position

Insert title here

Insert subtitle here

Name, position

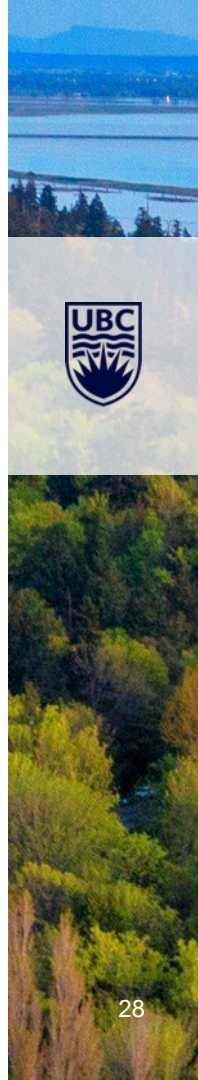
Insert title here

Insert subtitle here

Name, position

Page title

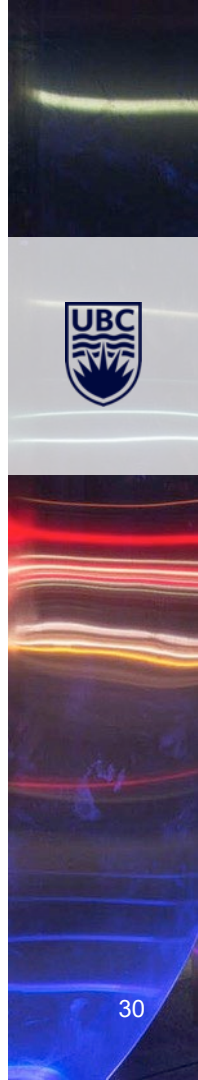
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list



Insert chapter title

Page title

- **Bullet point** list
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list



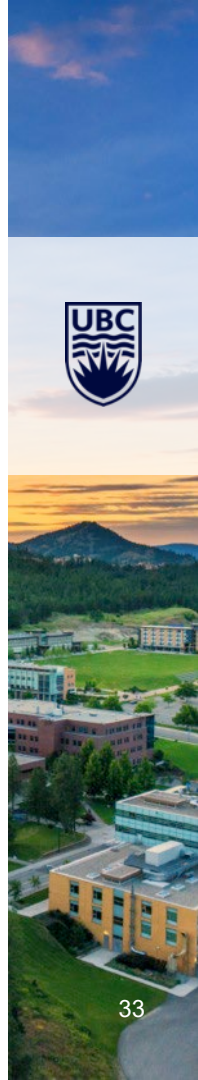
Insert chapter title

Page title

- Bullet point list
- Bullet point list
- Bullet point list
- Bullet point list

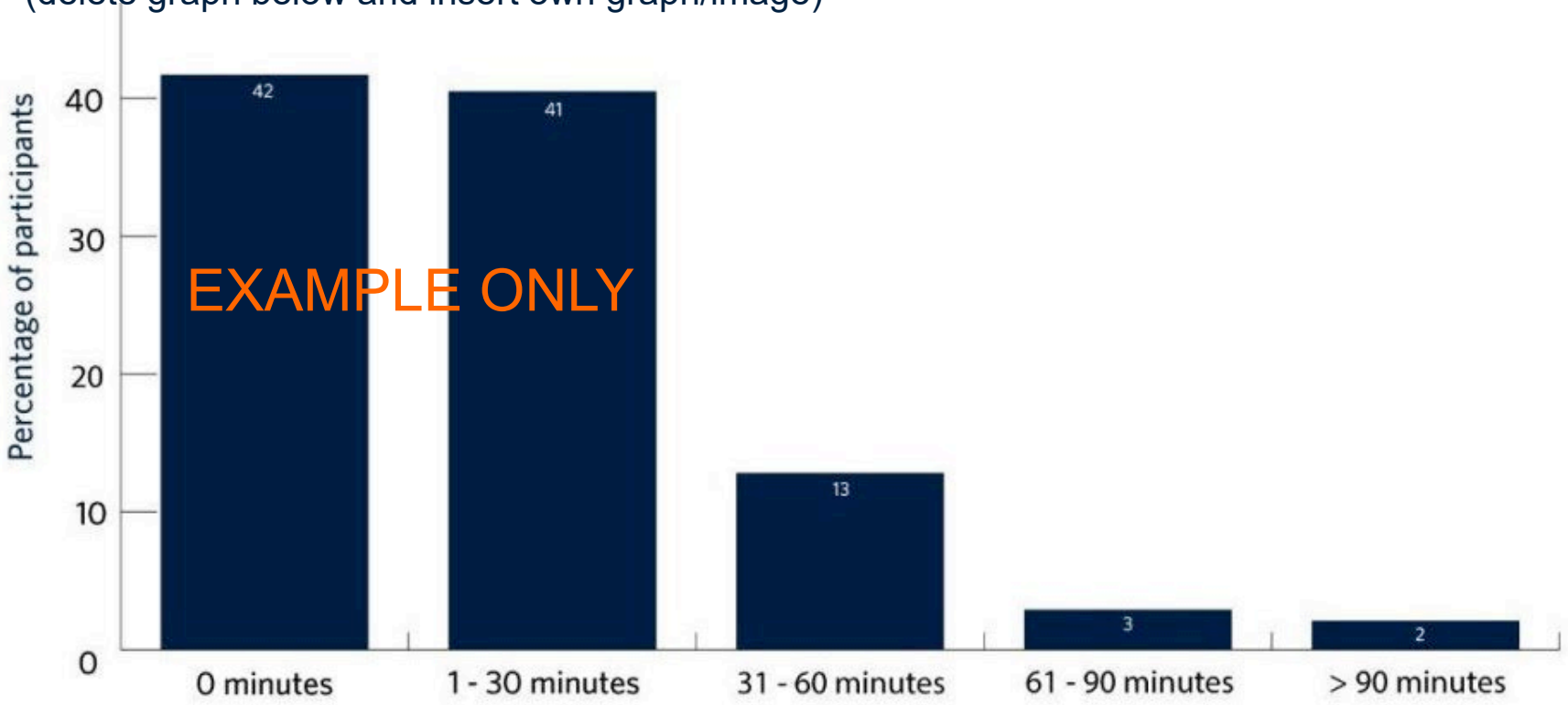
Page title

- **Bullet point** list
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list



Insert title

(delete graph below and insert own graph/image)




UBC
THE UNIVERSITY OF BRITISH COLUMBIA


THE UNIVERSITY OF BRITISH COLUMBIA
THE UNIVERSITY OF BRITISH