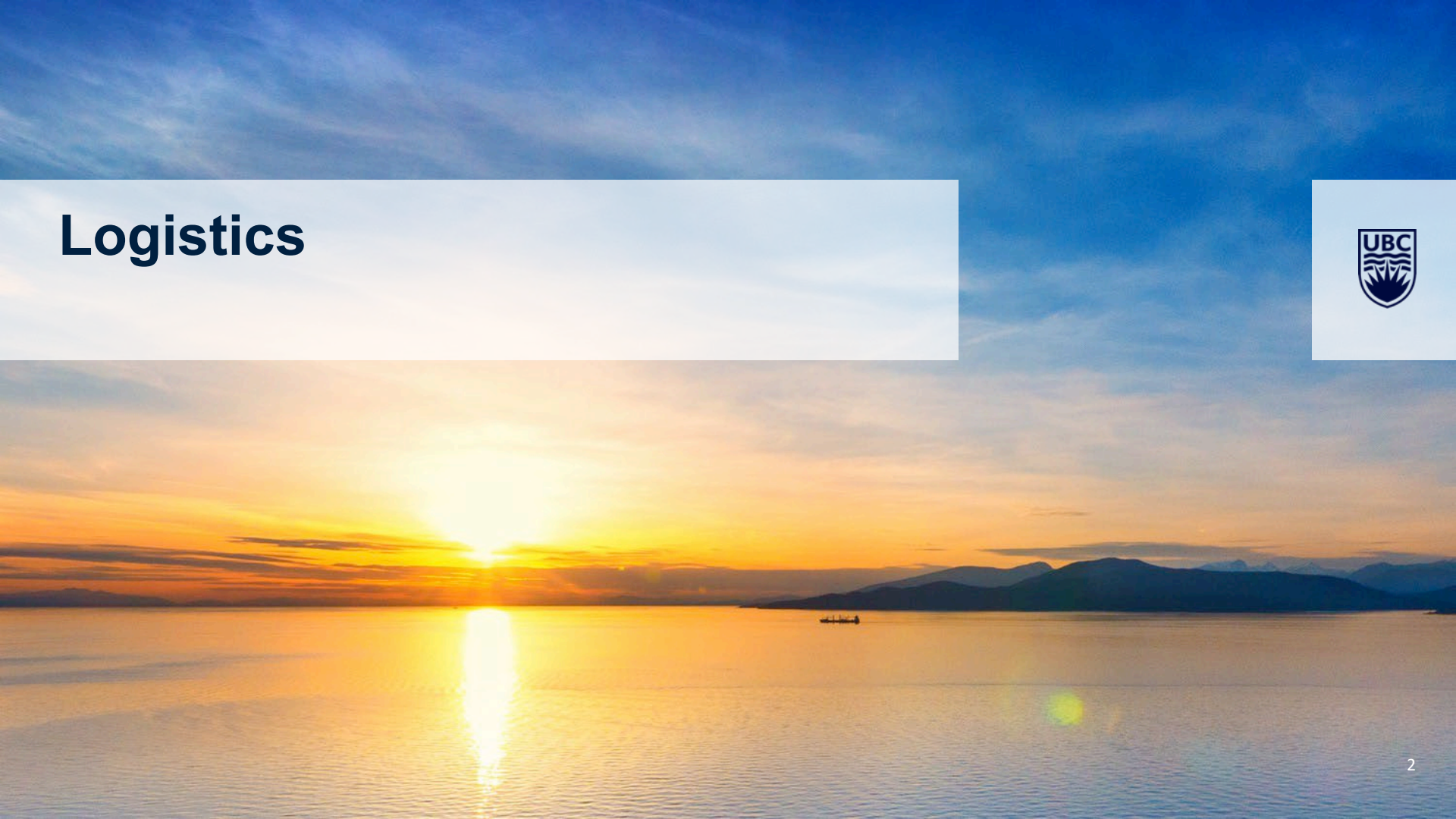
CPSC 416 Distributed Systems

Winter 2022 Term 2 (January 31, 2023)

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

Logistics



Deadlines

Project 1 Code: Grades posted to Gradescope

Project 1 Report: Grades posted to Gradescope

Project 2 Code: Pending (Due January 31, 2023)

Project 2 Report: Pending

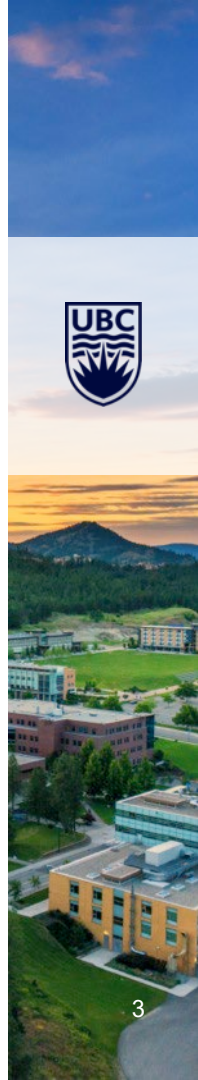
Project 3 Released. Initially Due: February 13, 2023.

- Note repo was updated Jan 31. (Minor updates)

Project 4 Released. Initially Due: March 13, 2023

Project 5 Released Due: April 13, 2023

All project work is due April 13, 2023. Late projects have a 75% score cap.



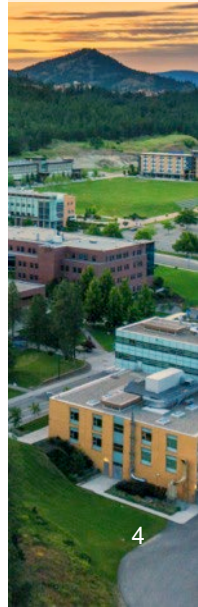
Deadlines

Alternate Path 1 & 2: Proposal was due January 30, 2023.

- Review in progress
- **Proceed according to your plan.**

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00



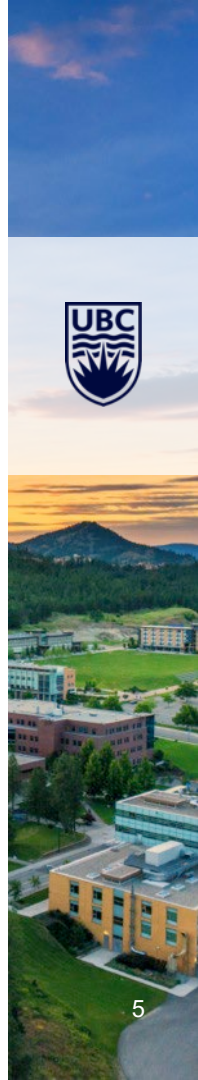
Readings

Required:

A Survey of Rollback-Recovery Protocols in Message-Passing Systems

Recommended:

- Fault Tolerant Techniques (Video)
- Microservices with Java (Blog Post)

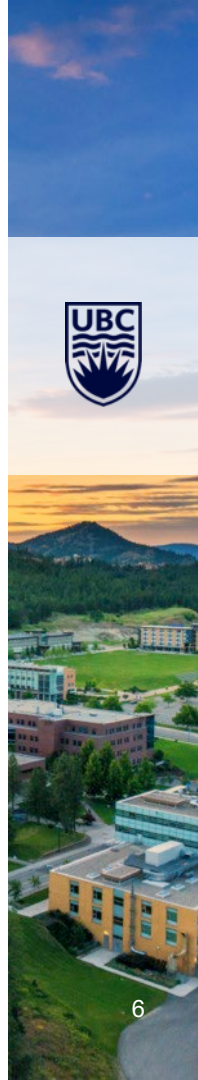


Questions?

Questions about the class?

Questions about the previous lecture?

Funny stories to share?



Today’s Failure

Microsoft Azure Outage

January 25, 2023 07:05 UTC

Between 07:05 UTC and 12:43 UTC on 25 January 2023, customers experienced issues with networking connectivity, manifesting as long network latency and/or timeouts when attempting to connect to resources hosted in Azure regions, as well as other Microsoft services including Microsoft 365 and Power Platform. While most regions and services had recovered by 09:00 UTC, intermittent packet loss issues were fully mitigated by 12:43 UTC. This incident also impacted Azure Government cloud services that were dependent on Azure public cloud.

What does this *mean*?

- Microsoft 365 = “Office applications, including E-mail, OneDrive Storage, SharePoint, etc”
- Microsoft Power Platform = “data analytics”
- Azure Government = “Governmental services”

Microsoft Azure Outage

Why did it happen?

- Wide Area Network (WAN) reconfiguration event
- Changed IP address on a WAN router
  - Triggered messages *to* other WAN routers
  - Routers reconstructed adjacency lists & forwarding tables
  - During reconstruction *packet forwarding stopped*
- Change command “had not been vetted”

We determined that a change made to the Microsoft Wide Area Network (WAN) impacted connectivity between clients on the internet to Azure, connectivity across regions, as well as cross-premises connectivity via ExpressRoute. As part of a planned change to update the IP address on a WAN router, a command given to the router caused it to send messages to all other routers in the WAN, which resulted in all of them recomputing their adjacency and forwarding tables. During this re-computation process, the routers were unable to correctly forward packets traversing them. The command that caused the issue has different behaviors on different network devices, and the command had not been vetted using our full qualification process on the router on which it was executed.

Microsoft Azure Outage

January 25, 2023 09:00 UTC

> “[N]early all network devices had recovered by 09:00 UTC… Final networking equipment recovered by 09:35 UTC.”

Outage over, right?

No… “Due to the WAN impact, our automated systems for maintaining the health of the WAN were paused… some paths in the network experienced increased packet loss from 09:35 UTC until those systems were **manually restarted**...” [emphasis added]

Microsoft Azure Outage

Corrective actions:

- Blocked highly impactful command from getting executed on the devices
- Require safe change guidelines for command execution

Translation: don’t allow people to do dangerous things.

Source: Azure status history | Microsoft Azure (entry for January 25, 2023)

Notice any patterns for failures?

Lesson Goals

Fault Tolerance

Techniques

- Fault tolerance
- Recovery

Failure Models

Basic Recovery Techniques

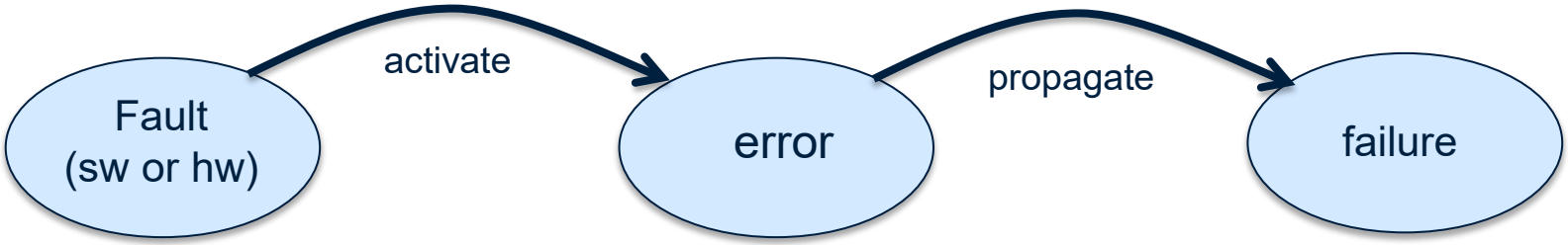
Terminology

Fault is some incorrect or unexpected behaviour

- Software
- Hardware

Error arises when a fault is *noticed* or *acted upon*

Failure occurs when an error impacts service



Failure types

Transient

Intermittent

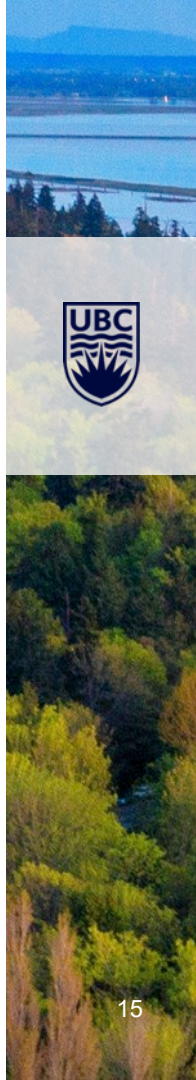
Permanent

Fail-stop
(fail-silent, crash)

Timing

Omission

Byzantine



Failure Management

Avoidance

- Consider all possible states and outcomes
- Predictive ability

Detection

- Heartbeat signal
- Error detection codes

Removal

- Rollback
- May not be possible

Recovery

- Despite occurrence of failures ensure correct execution
- Fault-tolerance

Rollback Recovery

Detect Failure?

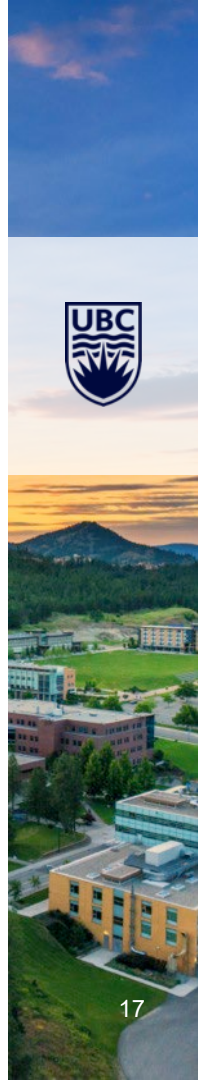
Rollback system state

- Prior to failure
- Consistent state

How?

- Undo recent work effect
- Reset process state

Try again (“re-execute”)



Rollback Recovery

Detect Failure?

Rollback system state

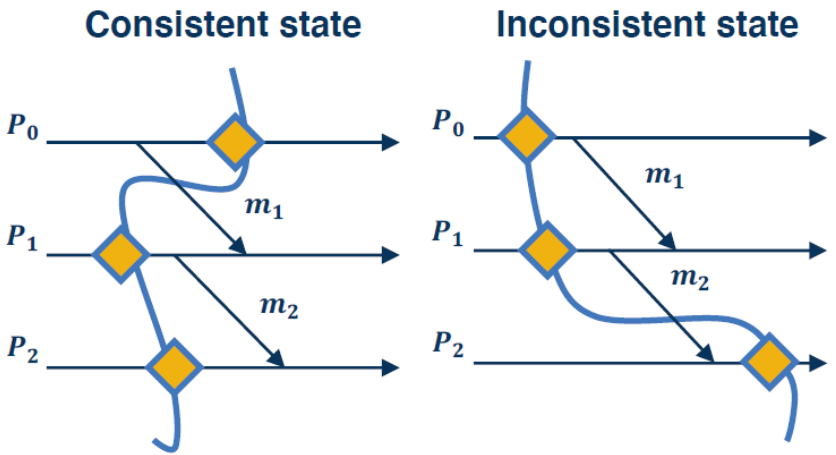
- Prior to failure
- Consistent state

How?

- Undo recent work effect
- Reset process state

Try again (“re-execute”)

1. Which previous state?



Rollback Recovery

Detect Failure?

Rollback system state

- Prior to failure
- Consistent state

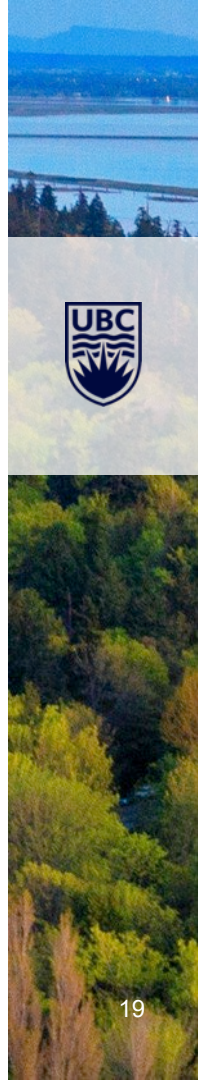
How?

- Undo recent work effect
- Reset process state

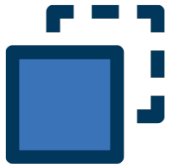
Try again (“re-execute”)

2. How to capture this state?

- Try to find by progressively rolling back to earlier consistent cuts
- Checkpointing
- Logging



Operation Granularity



Transparent (full-system)

- ⊕ no application modification
- ⊖ very high overheads

Transaction-based

- ⊕ relies on use of transactional API
- ⊖ overhead reduced to groups of related operations

```
tx_begin (inputs of all ops)
    operation1
    operation2
    operation2
    …
tx_end
```

Application-specific

- ⊕ applications know best what state is needed for recovery
- ⊖ limited applicability

Basic Rollback Mechanisms

Checkpointing

- Uncoordinated
- Coordinated
- Communications-induced

Logging

Which to use?

Checkpointing

Save {system,application} state

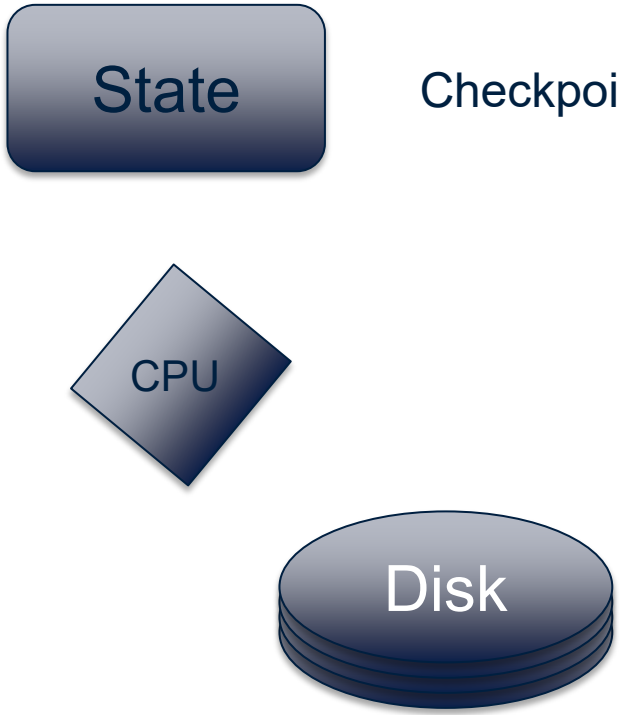
Flush checkpoint to *durable* storage

Benefits:

- Quick restart

Costs:

- I/O overhead for checkpoint
  - Can mitigate



Logging

Log information about operations

- Undo: record original value
- Redo: record new value
- Both (undo/redo)

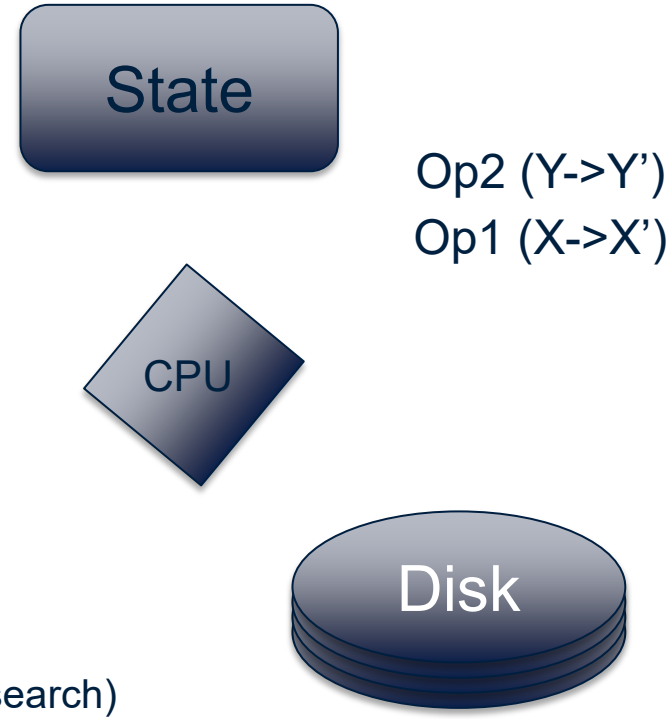
Write log to **persistent storage**

Benefits:

- Less I/O than checkpoint

Costs:

- Recovery takes time
- Operations may take longer (log search)



Checkpointing and Logging

Checkpoint

- Record consistent cut
- Use more recent

Logging

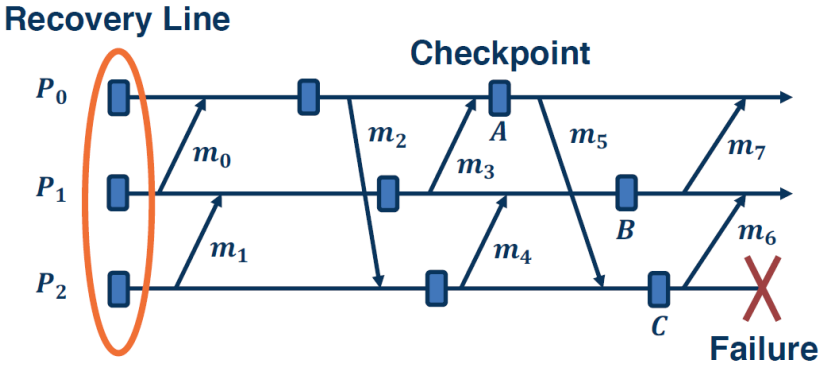
- Truncate log at checkpoint
- Record new operations

Benefits:

- Recover faster
- Minimize log overhead

Cost:

- Need stable, consistent cut



See: Required Reading
for original figure

Approaches for Checkpointing

Uncoordinated

Coordinated

Communications-triggered

Question: “*When* do we create a checkpoint?”

System Model

Fixed number of processes

Communications *only* via messages

Processes interact with external actors (“clients”)

No network partition

Variables:

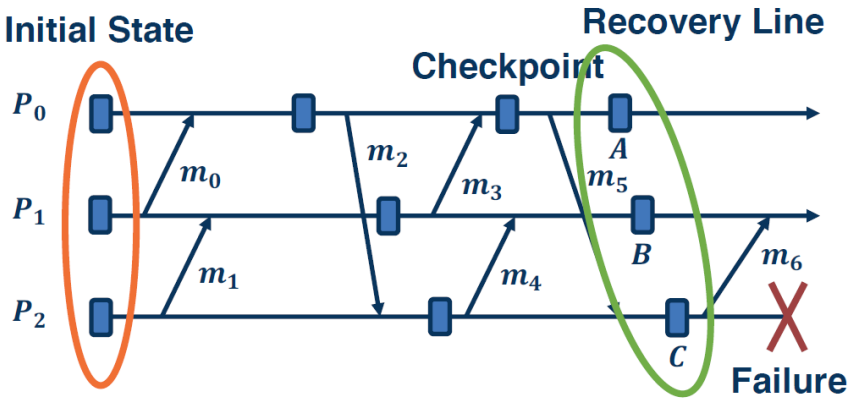
- FIFO protocol
- Reliable communications
- Number of failure tolerated

Uncoordinated Checkpointing

Processes checkpoint independently

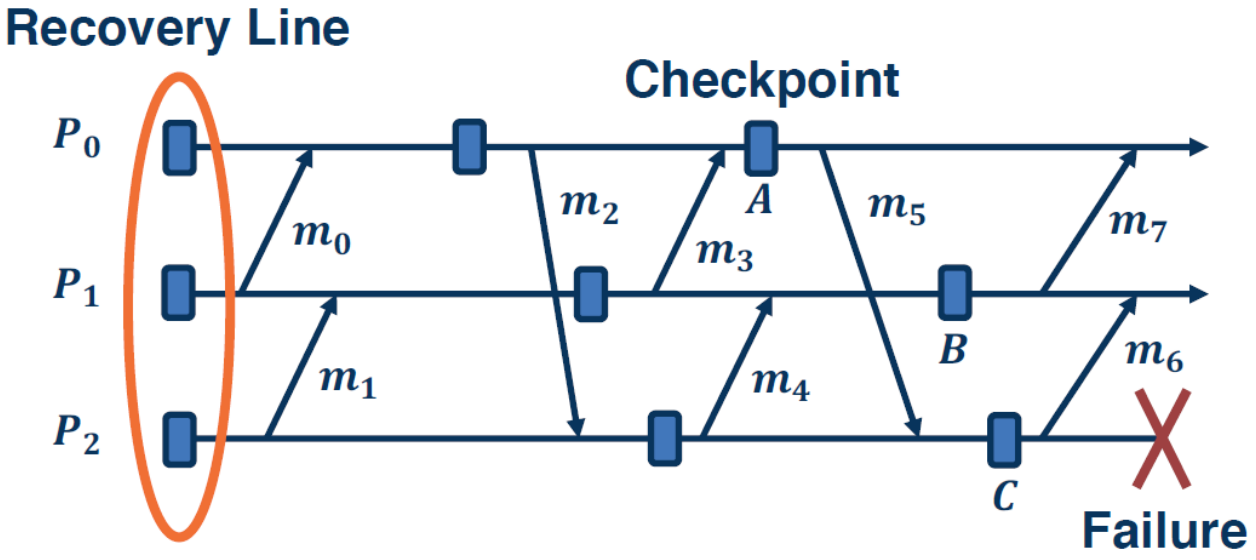
Must construct consistent state at failure point

- Compute recovery line
- Rollback
- Need *dependency* information

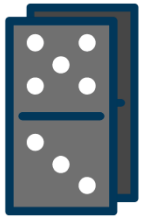


Approach Risk

How far do we roll back?



Uncoordinated Checkpoint Summary



Domino Effect

- could lose all your work

Useless Checkpoints

- checkpoints that can never form a globally consistent state may be taken

Multiple Checkpoints Per Process

- may need more than the most recent snapshots

Garbage Collection

- needed to identify obsolete checkpoints

Coordinated Checkpointing

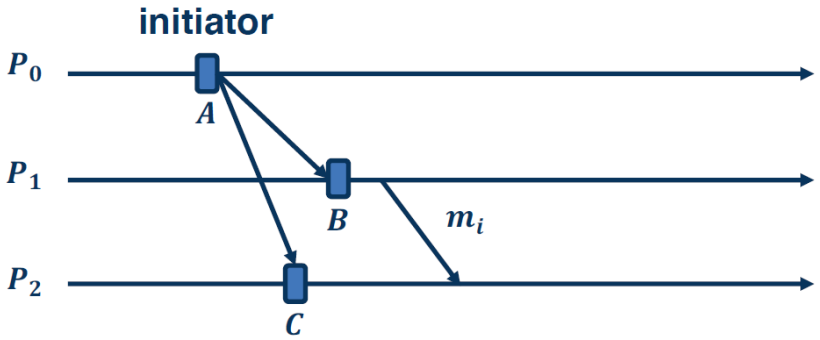
Coordinate *across* processes

Eliminates need for dependency graph

No domino effect

Single checkpoint *per process*

No garbage collection



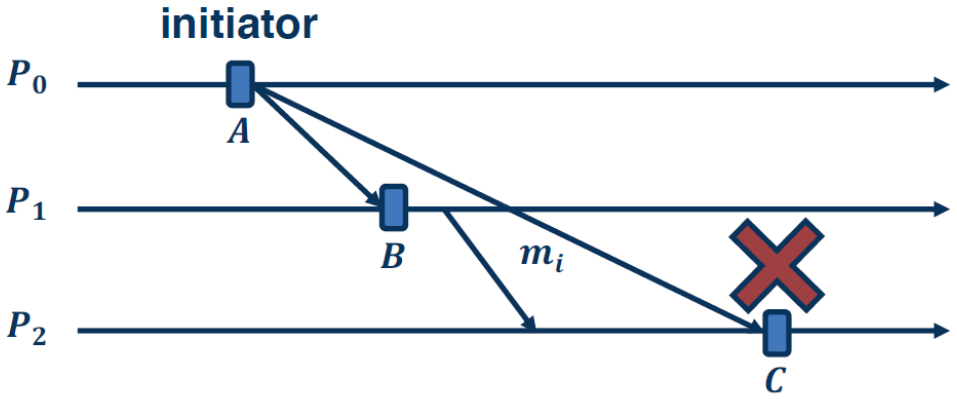
Coordinate Checkpoint Challenge

How to coordinate

No global clock

Message issues:

- Reliable
- Time bounded delivery



Can we eliminate any checkpoints?

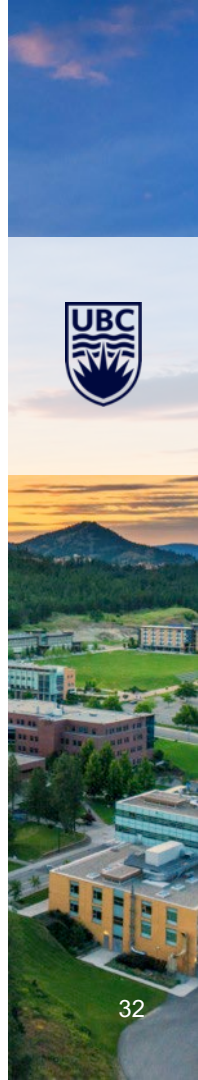
Communications Triggered Checkpoint

How to coordinate checkpoint?

Blocking

- Two-phase commit
  - Initiator: starts checkpoint
  - Non-initiator:
    - Blocks forward work
    - Waits for commit or abort
- Other consensus algorithms

Non-blocking: Global snapshot algorithm



Communications Triggered Checkpoint

Use piggyback information

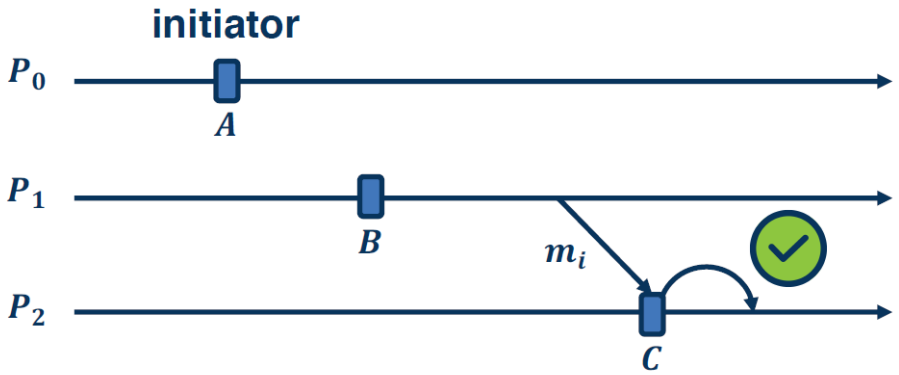
- FIFO not required

Nodes:

- Make independent decisions
- Works without communications

Message

- Can trigger snapshot
- Forces checkpoint decision

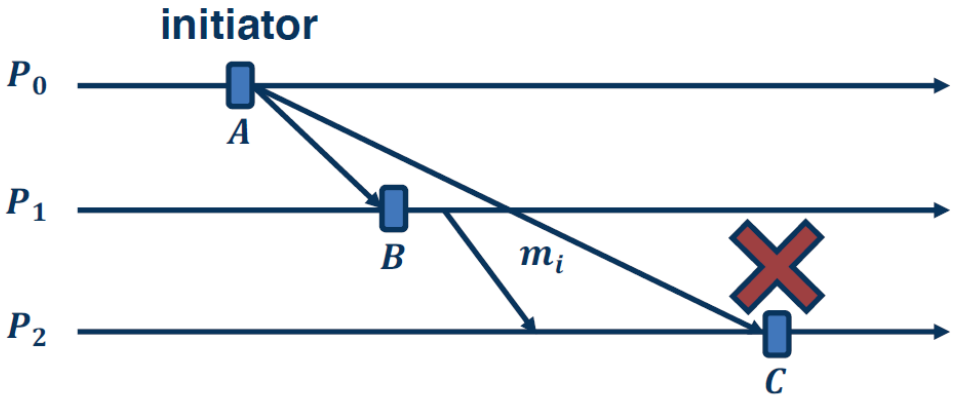


Logging Benefits

Computer versus I/O tradeoff

Requires:

- Node must allow rebuilding consistent state



Logging Approaches

Pessimistic:

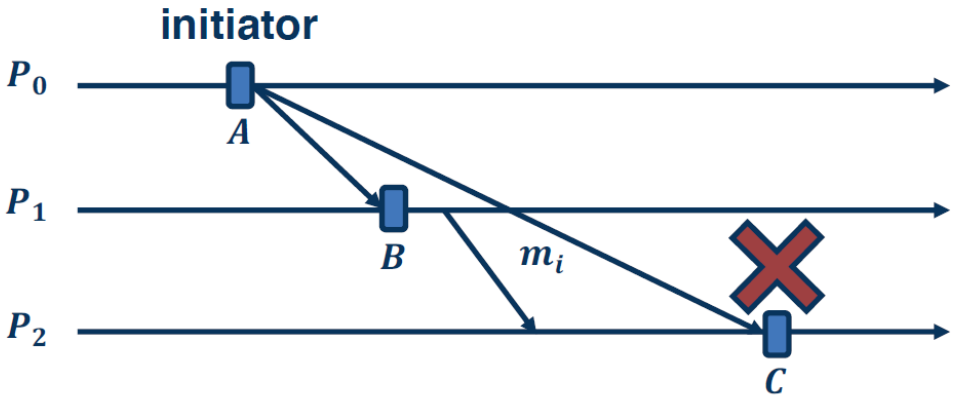
- Log everything
- Allow event to propagate

Optimistic:

- Log undo information
- Write log as necessary

Causality-tracking:

- Capture causality events
- Deterministic recording



Which Consistency Method Should You Use?

It depends:

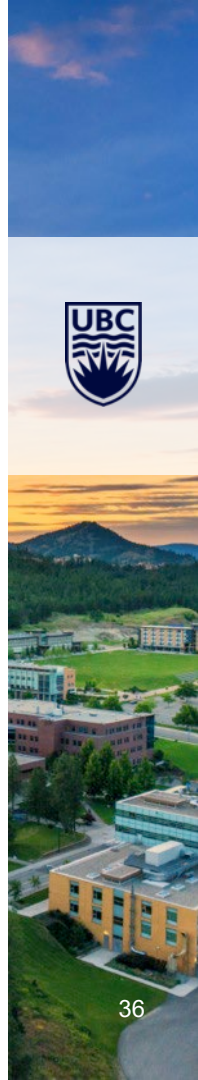
- Workload characteristics
- Failure characteristics
- System Characteristics
  - Communications cost
  - Storage cost
  - Scale

Anything changed since 2002?

- Networks: faster/more reliable
- Storage: faster/cheaper

	Uncoordinated Checkpointing	Coordinated Checkpointing	Communication Induced Checkpointing	Pessimistic Logging	Optimistic Logging	Causal Logging
PWD assumed?	No	No	No	Yes	Yes	Yes
Garbage collection	Complex	Simple	Complex	Simple	Complex	Complex
Checkpoints per process	Several	1	Several	1	Several	1
Domino effect?	Possible	No	No	No	No	No
Orphan processes?	Possible	No	Possible	No	Possible	No
Rollback extent	Unbounded	Last global checkpoint	Possibly several checkpoints	Last checkpoint	Possibly several checkpoints	Last checkpoint
Complex recovery?	Yes	No	Yes	No	Yes	Yes
Output commit	Not possible	Very slow	Very slow	Fastest	Slow	Fast

Comparison between different styles of rollback-recovery protocols



Lesson Summary

Fault Tolerance

We *can* deal with some failures

- Even in distributed systems

Costs associated with this

- Implementation: balance runtime versus recovery costs
- Increased compute, storage, network use
- Numerous trade-offs

Checkpointing and Logging: Key tools for enabling fault tolerance

Questions?


UBC
THE UNIVERSITY OF BRITISH COLUMBIA