Speculative Recovery: Cheap, Highly Available Fault Tolerance with Disaggregated Storage

Nanqin Qin Li, Anja Kalaba, Michael J. Freedman, Wyatt Lloyd, and Amit Levy
Princeton University
Disk “disaggregated:” highly durable and available

Storage pool

Compute instance

MySQL

network-attached

Data is replicated

Highly-available

Highly-durable
Highly available
Highly durable

Storage pool

Data is replicated

MySQL

Compute instance

High disk availability = High application availability?

Highly-durable
A nascent fault-tolerance technique

1. The primary failed
2. Spin up a new backup
3. Detach the disk
4. Attach the disk to the backup
5. Restart the application

REcovery from Disaggregated Storage (REDS)

Fail, then recover
REDS has low cost, and low availability as well

- Low cost
- Generally applicable to crash-consistent applications
- Low availability

- Must persist writes before externalization
- Can recover from crashed state
Why low availability #1: failover must be sequential

(I). Failure detection on the primary

Timeout unresponsiveness

Step (I) first, then step (II)

(II). Recovery on the backup

Move over the disk and restart the application

How long should step one (timeout) be?
Why low availability #2: the future is unknown

When to start the recovery?
- The primary truly failed?
- Can it self-heal quickly?

How long does the recovery take?
- How much “crash” to fix?
Why low availability #2: the future is unknown

- Short timeout
  - Long recovery when failure is only transient
  - Always an opportunity cost!

- Long timeout
  - Slow reaction to true failures
Suppose an Oracle knew the future
Suppose an Oracle knew the future

Oracle
Okay, I pick you

Please wait, BRB in 30 seconds!

But I can finish recovery in 20 seconds!
Speculative Recovery: similarly optimal decision!

The primary stopped responding

Clone
Speculative Recovery: similarly optimal decision!

Hang tight, be right back!

Two racing paths: Whoever finishes first wins!

Recovery completes in 10...9...8...

Clone
Outline

- Introduction
- Speculative recovery and disk superposition
- Two new primitives super and collapse
- Evaluation
Speculative Recovery creates a “superposition”

Hang tight, be right back!

Two independent versions of the application?

Recovery completes in 10…9…8…

Clone
Speculative Recovery creates a “superposition”

Hang tight, be right back!

Superposition: two versions, but only one observed

Recovery completes in 10...9...8...

Clone
Speculative Recovery creates a “superposition”

(Schroedinger's Box)

(Photo credit: https://www.sciencefocus.com/science/what-is-schrodinger-cats/)
Speculative Recovery creates a “superposition”

Hey, I've been here the whole time!
Speculative Recovery creates a “superposition”
Speculative Recovery creates a “superposition”

- **Creating** a superposition by creating a disk clone
  - The disk clone must be fast-to-create and performant
    - Copy-on-write has bad I/O performance after creation, especially under highly parallel write workload

- **Collapsing** a superposition when one disk version is observed
  - If the primary’s disk is updated, then the primary has been observed
    - Deallocating the backup
  - Otherwise if no write until the recovery completes
    - Promoting the backup and deallocating the primary
Outline

- Introduction
- Speculative recovery and disk superposition
- Two new primitives super and collapse
- Evaluation
A set of primitives: super and collapse

- **super**: creating a superposition by creating a disk clone
- **collapse**: collapsing the superposition by tracking writes to the primary’s disk
super uses copy-on-write

Copy-on-write

Fast to create

Bad I/O performance

Serialized concurrent COW operations

Data copying over the network

The average write latency could be 30x higher!!!
super uses copy-on-write

Copy-on-write

Fast to create

Bad I/O performance

Collocated-clone

I/O performance close to a regular disk!

The average write latency could be 30x higher!!!

More details in the paper!
collapse uses a dirty bit to monitor writes

- A dirty bit for the primary’s disk: **any write sets it dirty**

More details in the paper!
Three components of a speculative recovery system

The instance pool
- primary
- backup

The storage cluster

The failure monitor

I/O requests

Tracking primary writes

Creating a superposition!
Three components of a speculative recovery system

- The instance pool
  - primary
  - backup

- The storage cluster

- The failure monitor
  - spawn()
  - heat eat()
  - collapse()

Tracking primary writes

Collapsing a superposition: which version to keep?
Outline

- Introduction
- Speculative recovery and disk superposition
- Two new primitives super and collapse
- Evaluation
The experiment setup

- We implemented a prototype speculative recovery system: **SpecREDS**
  - Based on Ceph's block device interface **rbd**.
- The instance pool: **docker containers**
- The storage cluster: high-end **NVMe SSD** drives
The experiment setup

- We compare three disk types
  - \texttt{rbd} (a regular disk)
  - \texttt{rbd-clone} (with Ceph’s existing clone implementation)
  - \texttt{super} (with collocated-clone)

- We compare three systems
  - \texttt{REDS} (using rbd)
  - \texttt{SpecREDS} (using super)
  - \texttt{Oracle} (using rbd)

- Three database applications running \textit{TPC-C} workload with \texttt{oltpbench}:
  - \texttt{MySQL} with InnoDB
  - \texttt{PostgreSQL}
  - \texttt{MariaDB} with RocksDB
Application recovery latency from various disk states

Time it takes to complete recovery

Failure type:
- S - docker stop
- P - kernel panic

Size of the write-ahead-log in GB

A regular disk
collocated-clone
Ceph’s existing clone

Recovery latency (s)

Postgres

S/.4G  S/2G  P/5G
Application recovery latency from various disk states

Failure type:
S - docker stop
P - kernel panic

Size of the write-ahead-log in GB
Application recovery latency from various disk states

- Failure type: S - docker stop
- Failure type: P - kernel panic
- Size of the write-ahead-log in GB

Diagram showing recovery latency for Postgres with different failure types and disk sizes.
Application recovery latency from various disk states

![Chart showing recovery latency for different disk states and failure types.]

- **Failure type:**
  - S - docker stop
  - P - kernel panic

- **Size of the write-ahead-log in GB**

**Close-to-normal** recovery latency
End-to-end failover latency with varying timeout/recovery

Simulation: adds latencies together

Long recovery - around 1 minute
Short recovery - around 5 seconds
End-to-end failover latency with varying timeout/recovery

No waiting for a timeout!
End-to-end failover latency with varying timeout/recovery

Similar latency when timeout is short
End-to-end failover latency with varying timeout/recovery

Simulated false positive:
The primary self-heals after 10 seconds
End-to-end failover latency with varying timeout/recovery

Avoids unnecessary failovers
End-to-end failover latency

SpecREDS avoids long timeout and unnecessary failovers
Speculation once again improves performance!

Speculative recovery safely and efficiently parallelizes self-heal on the primary and recovery on the backup to push failover latency to the lower bound of REDS

Thank you!