KVM / DR

TECHNICAL WHITE PAPER

# Near-Sync Replication & Changed-Block Tracking

How changed-block tracking turns continuous data protection into a practical, low-RPO reality for oVirt, RHV, and OLVM — and what to look for in a replication engine.

Published by **KVMDR** · kvmdr.ai · 2026

A vendor-neutral guide for infrastructure and platform teams

EXECUTIVE SUMMARY

# Your RPO is set by how you replicate.

The recovery point objective (RPO) of any environment is decided long before an incident — by how often, and how efficiently, data is copied to a recovery site. Periodic full or image-level copies impose an RPO measured in hours and a load that scales with total capacity, not with how much actually changed.

**Changed-block tracking (CBT) breaks that link.** By identifying exactly which blocks changed since the last cycle and shipping only those, replication cost scales with a workload's change rate rather than its size — making a recovery copy that trails production by seconds both practical and affordable.

This paper explains what CBT is, how it underpins near-sync replication on KVM, and the engine characteristics that separate a dependable, low-RPO design from a fragile one. It is written to be useful whatever tooling a team ultimately chooses.

## THE SHORTFALL OF PERIODIC COPIES

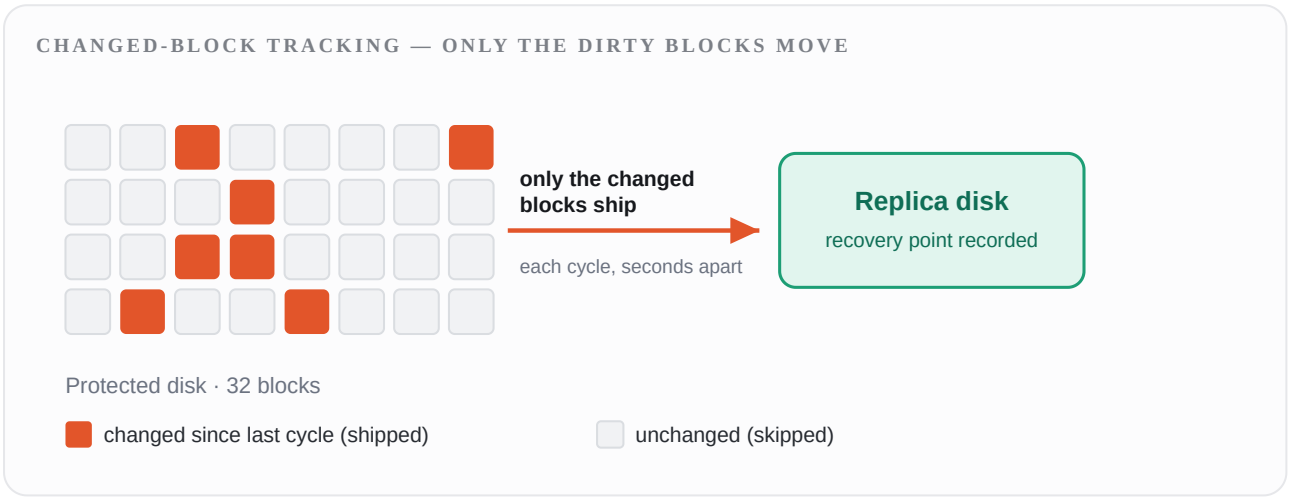
- **Cost scales with size, not change.** A full or image copy re-reads the entire disk every run, even when only a fraction of it changed.
- **RPO stuck in hours.** The interval between copies is, by definition, the window of data you stand to lose — and shortening it multiplies the cost above.
- **Production impact.** Large sequential reads compete with live workloads for storage and network I/O, which is why full copies get pushed into overnight windows.

The rest of this paper looks at how CBT removes those constraints, and what to demand of an engine that relies on it.

HOW IT WORKS

# What changed-block tracking actually is

At the storage layer, a virtual disk is a sequence of fixed-size blocks. Changed-block tracking keeps a **dirty bitmap** that marks which blocks were written since a reference point. On KVM this lives in the hypervisor's QEMU layer rather than inside the guest — so it is agentless and OS-agnostic, capturing every write at the block level.




Each cycle consults the bitmap, reads only the dirty regions, ships them to the recovery site, and resets the bitmap for the next interval. Nothing unchanged is ever re-read or re-sent, so replication cost tracks the workload's change rate rather than the size of the disk.

| Dimension | Periodic full / image copy | Changed-block delta |
|---|---|---|
| **Data moved per cycle** | The entire disk | Only blocks changed since the last cycle |
| **Cost driver** | Total capacity | Workload change rate |
| **Achievable RPO** | Hours | Seconds to minutes |
| **Guest impact** | In-guest agent or full read | Agentless; deltas only |

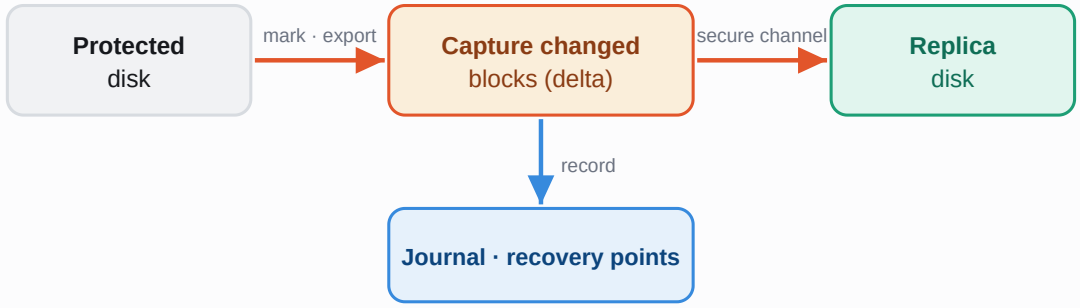
That single shift — from copying everything to tracking and shipping only what changed — is what makes a seconds-old recovery copy affordable on everyday hardware.

THE CYCLE

# From changed blocks to a near-sync replica

A near-sync replica is built by repeating the cycle on a short interval. Each pass marks the changed blocks, captures just those deltas, applies them to the replica over a secure channel, and records the result to a journal as a recovery point.

THE NEAR-SYNC CYCLE — MARK, CAPTURE, APPLY, RECORD



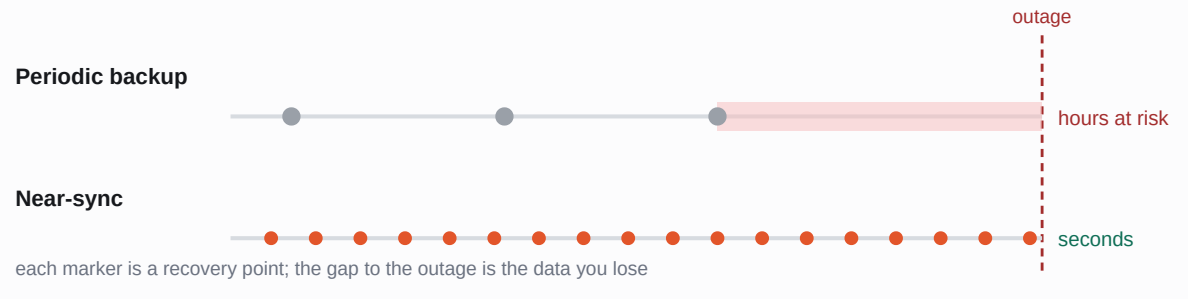

## CRASH-CONSISTENCY ACROSS DISKS

Multi-disk and multi-VM applications must be captured as a coherent set. Grouping related disks into a **consistency group** and marking their bitmaps in a single atomic operation makes every recovery point crash-consistent across the whole application — the same guarantee as cutting power to every disk at the same instant.

## TUNING THE RPO

The replication interval sets the floor for RPO. Adaptive designs tighten it when change rate is high and relax it when the workload is quiet, holding a target RPO without over-spending bandwidth. Because only deltas move, shortening the interval is inexpensive.

RECOVERY POINT — HOW FAR BEHIND PRODUCTION EACH APPROACH SITS



EVALUATION

# Characteristics of a dependable engine

Not all changed-block replication is equal. When evaluating an approach for oVirt, RHV, or OLVM, these properties separate a resilient low-RPO design from one that merely demonstrates well.

REPLICATION ENGINE CHECKLIST

### What a low-RPO design should provide

- ✓ Agentless capture using native QEMU / NBD facilities — no software inside every guest
- ✓ Atomic, multi-disk bitmap operations for crash-consistent consistency groups
- ✓ A durable journal and buffer so a cycle survives a control-plane or network interruption
- ✓ Immutable, tamper-proof recovery points with defined retention
- ✓ Encrypted transport of deltas between sites
- ✓ A configurable, adaptive replication interval to hold a target RPO
- ✓ Native support for oVirt, RHV, and OLVM on QEMU 5.2+

PLAN FOR CHANGE RATE, NOT CAPACITY

Size the recovery link to the workload’s sustained change rate, not its total capacity. A useful estimate: take peak daily change as a fraction of total data, divide by the replication window, and that gives the bandwidth a near-sync target requires. Quiet estates need surprisingly little; write-heavy databases need the most planning.

CONCLUSION

# Small deltas, continuous protection

Changed-block tracking is the mechanism that makes continuous data protection economical. By moving only what changed, it decouples replication cost from data size and brings a seconds-old recovery copy within reach for everyday KVM estates — not just the largest budgets.

Teams designing DR for oVirt, RHV, or OLVM should treat the replication engine as a first-order decision: how it tracks change, how atomically it captures multi-disk applications, and how durably it records each cycle determine the RPO they can actually promise.

Get those fundamentals right and the rest of the DR story — failover, testing, archive — has a solid base to stand on. Get them wrong and no amount of orchestration above will hide a recovery copy that is hours stale or quietly inconsistent.