



Moving Apache® Ignite™ into Production: Best Practices for Native Persistence and Data Recovery

Ivan Rakov
June 19, 2019



Agenda



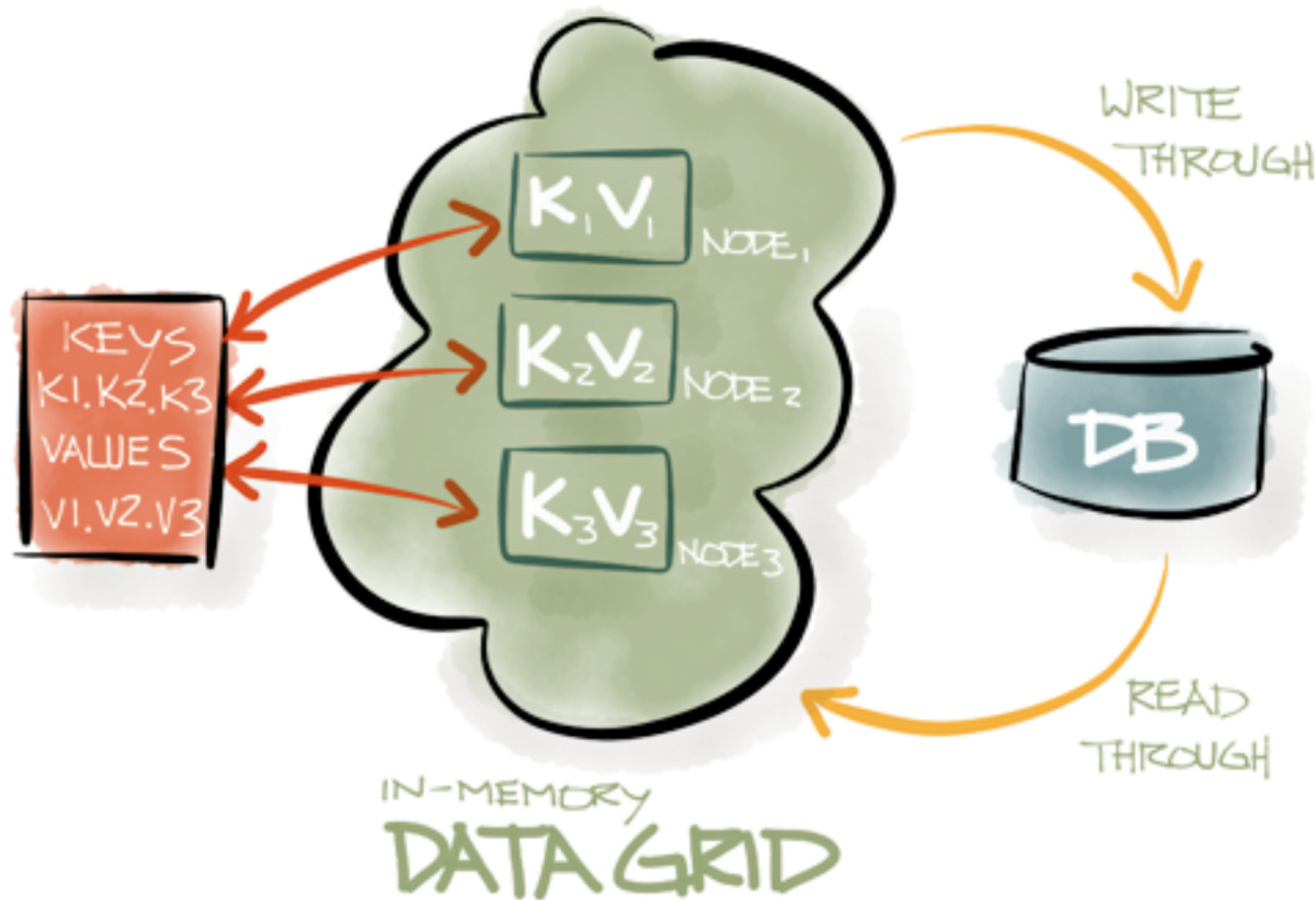
- Features that in-memory data grids lack
- Apache Ignite way: durability through page memory architecture
- Durability: use cases and solutions
 - Storage management use cases
 - Data backups use cases
- Durability: performance tricks

Agenda



- Features that in-memory data grids lack
- Apache Ignite way: durability through page memory architecture
- Durability: use cases and solutions
 - Storage management use cases
 - Data backups use cases
- Durability: performance tricks

In-memory Data Grid



In-memory Data Grid



Good, but

In-memory Data Grid



Do you need to access all your data at in-memory speed?

In-memory Data Grid



Good, but

- Storing all data in RAM is expensive
RAM ~8\$ per GB, SSD ~0.2\$ per GB

In-memory Data Grid



Sooner or later, cluster will require maintenance

In-memory Data Grid



Good, but

- Storing all data in RAM is expensive
RAM ~8\$ per GB, SSD ~0.2\$ per GB
- Cluster maintenance is complicated
Grid restart requires data reloading

In-memory Data Grid



Anything that can go wrong will go wrong

In-memory Data Grid



Good, but

- Storing all data in RAM is expensive
RAM ~8\$ per GB, SSD ~0.2\$ per GB
- Cluster maintenance is complicated
Grid restart requires data reloading
- Disaster protection
Data backups would be handy

Agenda



- Features that in-memory data grids lack
- Apache Ignite way: durability through page memory architecture
- Durability: use cases and solutions
 - Storage management use cases
 - Data backups use cases
- Durability: performance tricks

How to gain in-memory speed and durability?

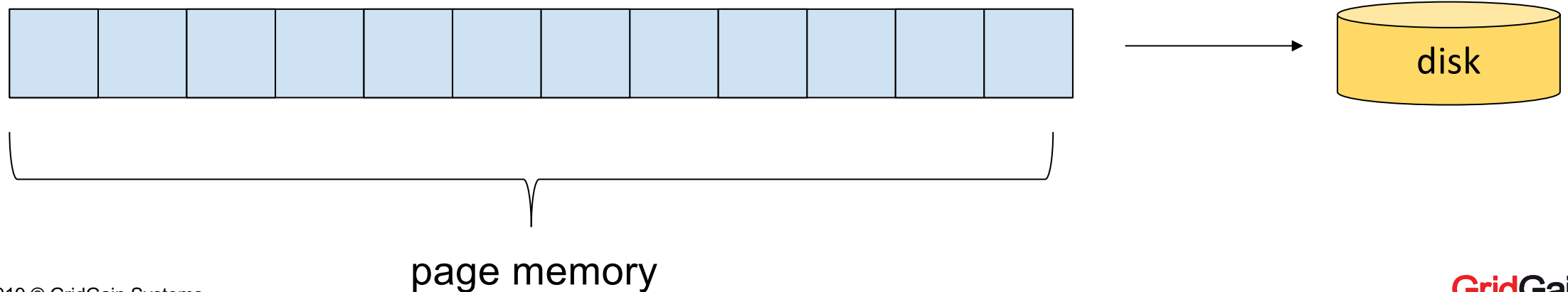


- Apache Ignite: transparent page memory architecture

Transparent Page Memory Architecture



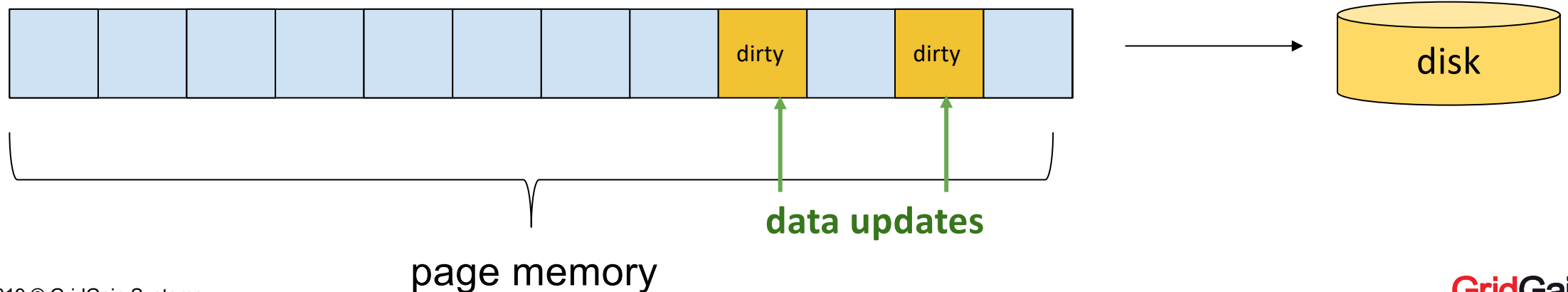
- Pages are always on disk, optionally in RAM



Transparent Page Memory Architecture



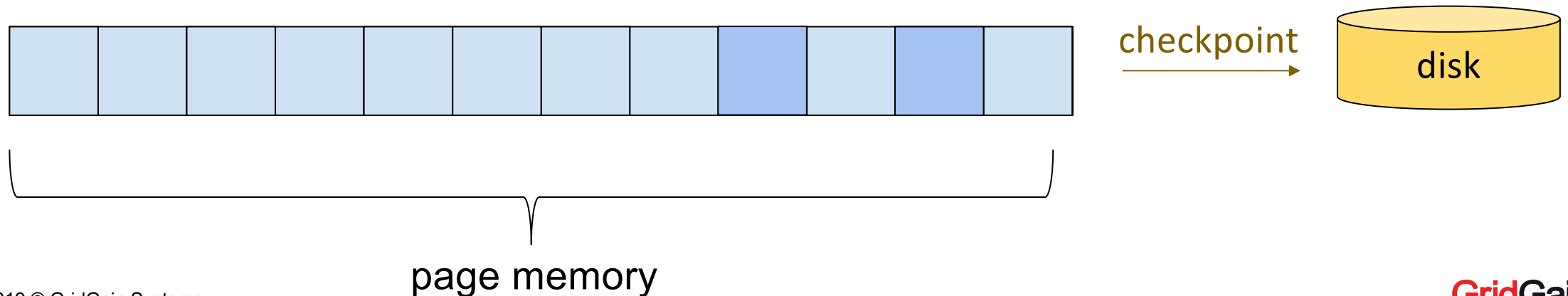
- Pages are always on disk, optionally in RAM
- Dirty pages are accumulated in RAM



Transparent Page Memory Architecture



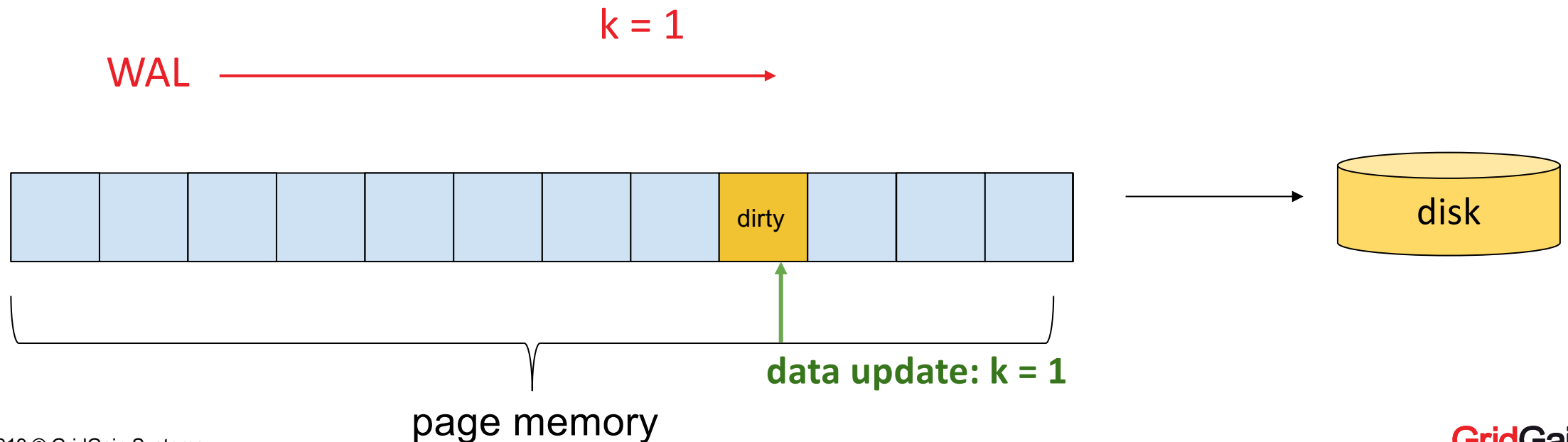
- Pages are always on disk, optionally in RAM
- Dirty pages are accumulated in RAM
- Checkpoint: batch of dirty pages is written to disk



Transparent Page Memory Architecture



- Pages are always on disk, optionally in RAM
- Dirty pages are accumulated in RAM
- Checkpoint: batch of dirty pages is written to disk
- WAL: updates between checkpoints are logged



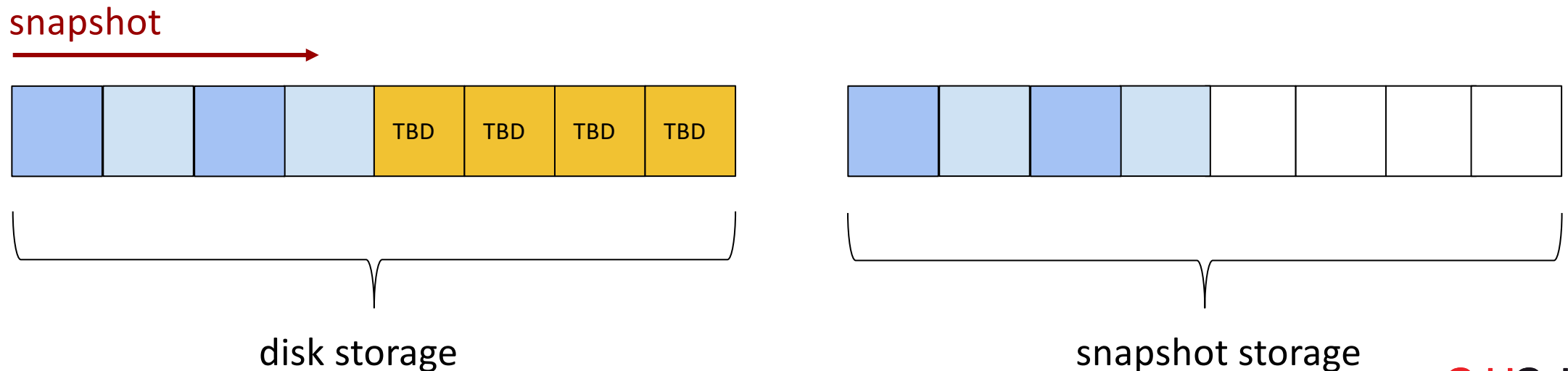
Snapshot under load: copy-on-write



Snapshot under load: copy-on-write



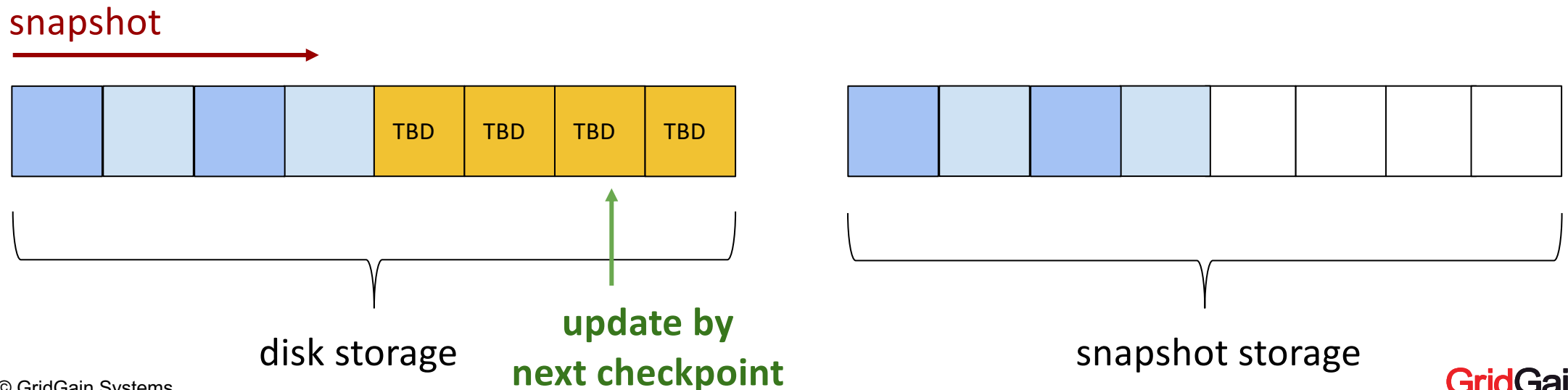
- Scan disk storage, copy pages to snapshot



Snapshot under load: copy-on-write



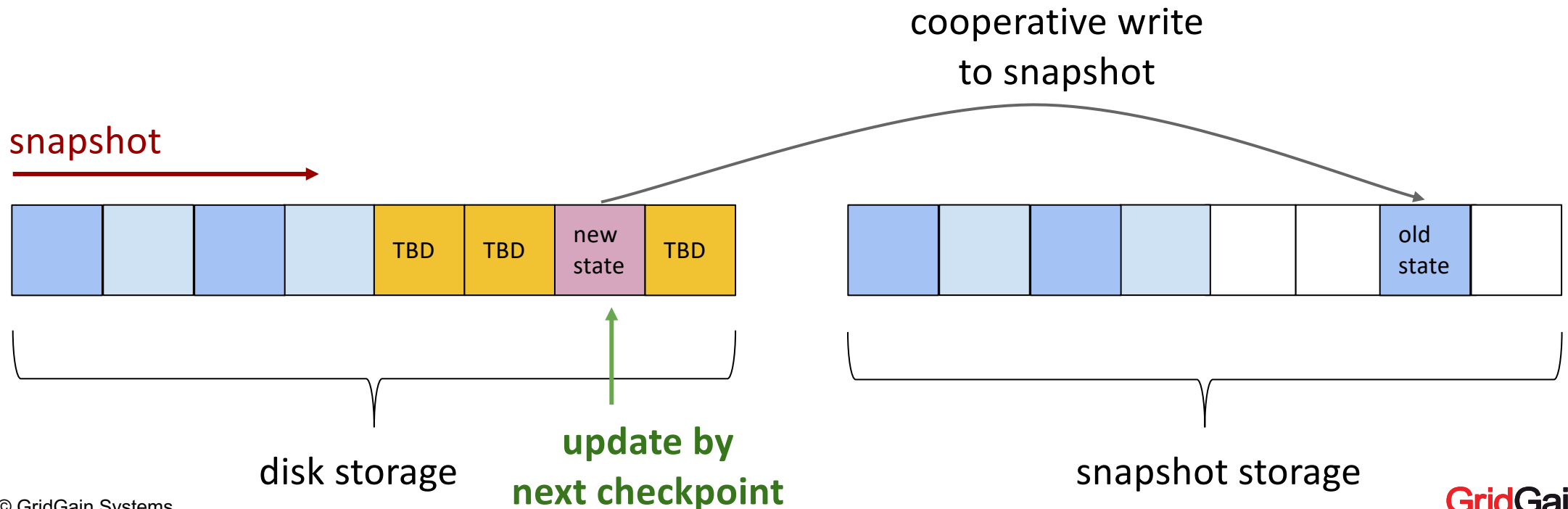
- Scan disk storage, copy pages to snapshot
- Next checkpoint is going to update yet not written page?



Snapshot under load: copy-on-write



- Scan disk storage, copy pages to snapshot
- Next checkpoint is going to update yet not written page?
- Let it write page to snapshot first!



Agenda



- Features that in-memory data grids lack
- Apache Ignite way: durability through page memory architecture
- **Durability: use cases and solutions**
 - Storage management use cases
 - Data backups use cases
- Durability: performance tricks

From theory to practice: Data Storage Configuration

- Use cases:
 - Limit RAM usage

From theory to practice: Data Storage Configuration



- Use cases:
 - Limit RAM usage
 - Different RAM limitations for different caches

From theory to practice: Data Storage Configuration



- Use cases:
 - Limit RAM usage
 - Different RAM limitations for different caches
 - Fast cluster restart and cheaper data storing

From theory to practice: Data Storage Configuration



- Use cases:
 - Limit RAM usage
 - Different RAM limitations for different caches
 - Fast cluster restart and cheaper data storing
 - Hot and cold data

Use case: limit node RAM consumption



- Default: in-memory mode

Use case: limit node RAM consumption



- Default: in-memory mode
- Overall RAM usage limit is configurable

Use case: limit node RAM consumption



- Default: in-memory mode
- Overall RAM usage limit is configurable
- Available RAM allocated by caches on demand

Use case: limit node RAM consumption



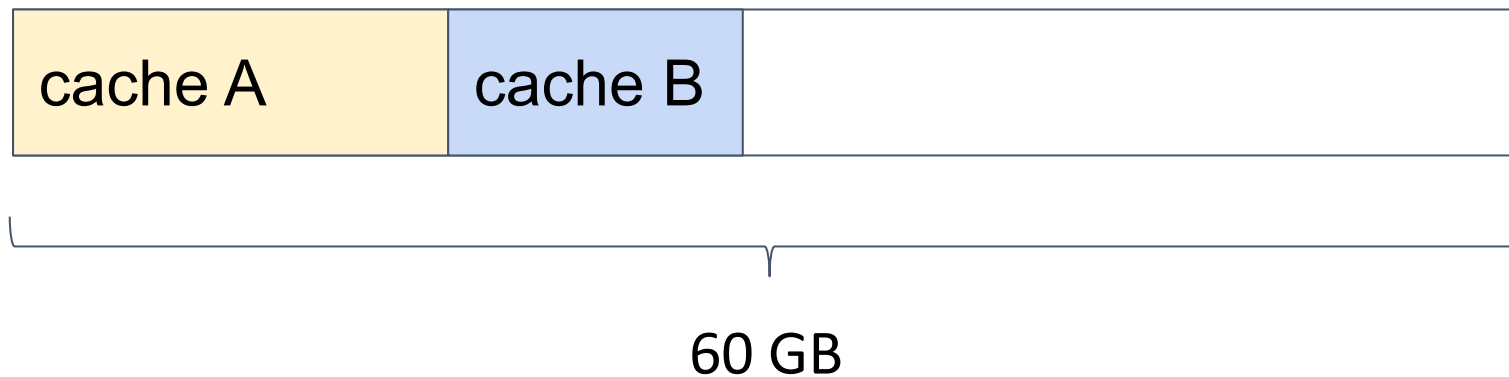
- Default: in-memory mode
- Overall RAM usage limit is configurable
- Available RAM allocated by caches on demand

```
new DataStorageConfiguration()  
    .setDefaultDataRegionConfiguration(  
        new DataRegionConfiguration().setMaxSize(60L * 1024 * 1024 * 1024));
```


Use case: limit node RAM consumption



- Default: in-memory mode
- Overall RAM usage limit is configurable
- Available RAM allocated by caches on demand



Use case: limit RAM consumption for specific cache

- Several “data regions”

Use case: limit RAM consumption for specific cache

- Several “data regions”
- Each region has its own limit

Use case: limit RAM consumption for specific cache

- Several “data regions”
- Each region has its own limit
- Optional eviction mode: old data above the limit is removed

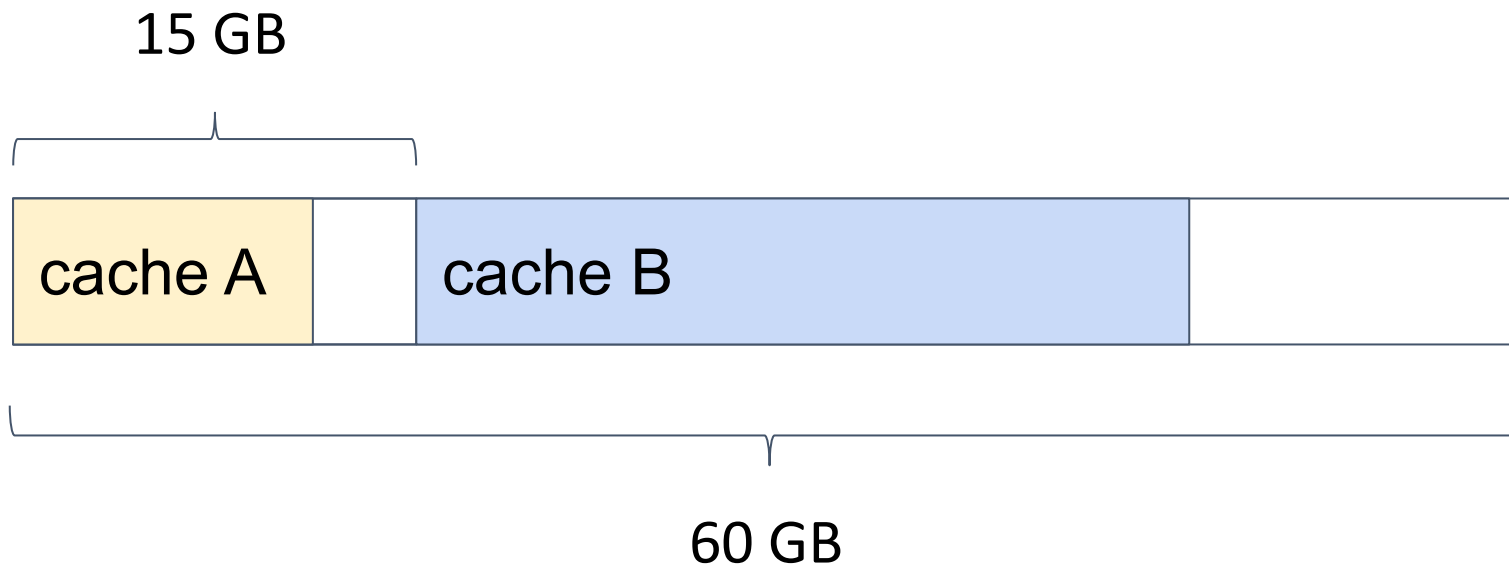
Use case: limit RAM consumption for specific cache

- Several “data regions”
- Each region has its own limit
- Optional eviction mode: old data above the limit is removed

```
new DataStorageConfiguration()  
    .setDefaultDataRegionConfiguration(  
        new DataRegionConfiguration().setMaxSize(45L * 1024 * 1024 * 1024))  
    .setDataRegionConfigurations(  
        new DataRegionConfiguration().setName("region-with-eviction")  
            .setMaxSize(15L * 1024 * 1024 * 1024)  
            .setPageEvictionMode(DataPageEvictionMode.RANDOM_LRU));
```

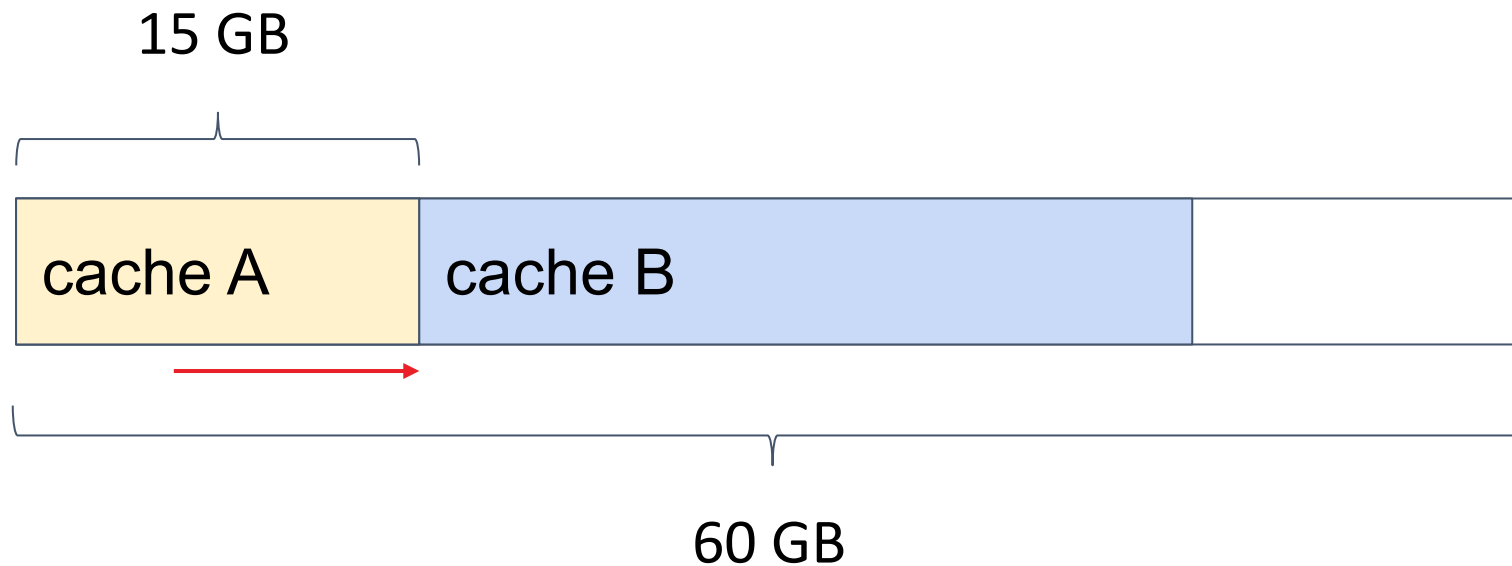
Use case: limit RAM consumption for specific cache

- Several “data regions”
- Each region has its own limit
- Optional eviction mode: old data above the limit is removed



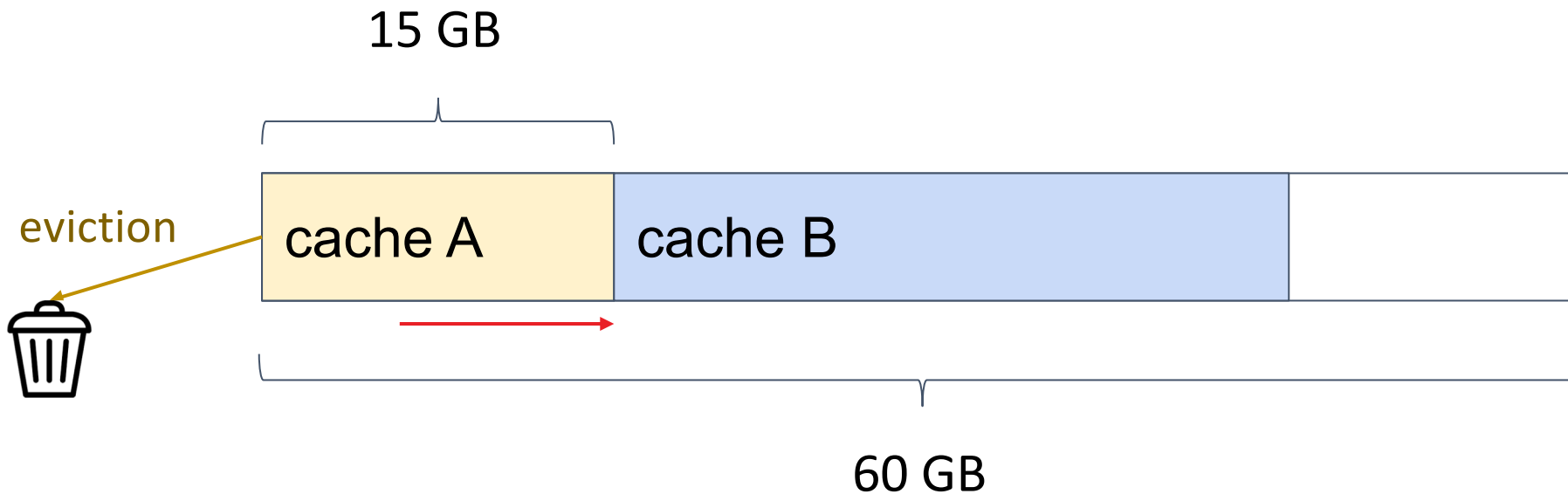
Use case: limit RAM consumption for specific cache

- Several “data regions”
- Each region has its own limit
- Optional eviction mode: old data above the limit is removed



Use case: limit RAM consumption for specific cache

- Several “data regions”
- Each region has its own limit
- Optional eviction mode: old data above the limit is removed



Use case: fast restart and cheaper storing



Use case: fast restart and cheaper storing



- Persistent mode: all pages on disk, subset of pages in RAM

Use case: fast restart and cheaper storing



- Persistent mode: all pages on disk, subset of pages in RAM
- Cold pages replaced to disk on demand

Use case: fast restart and cheaper storing



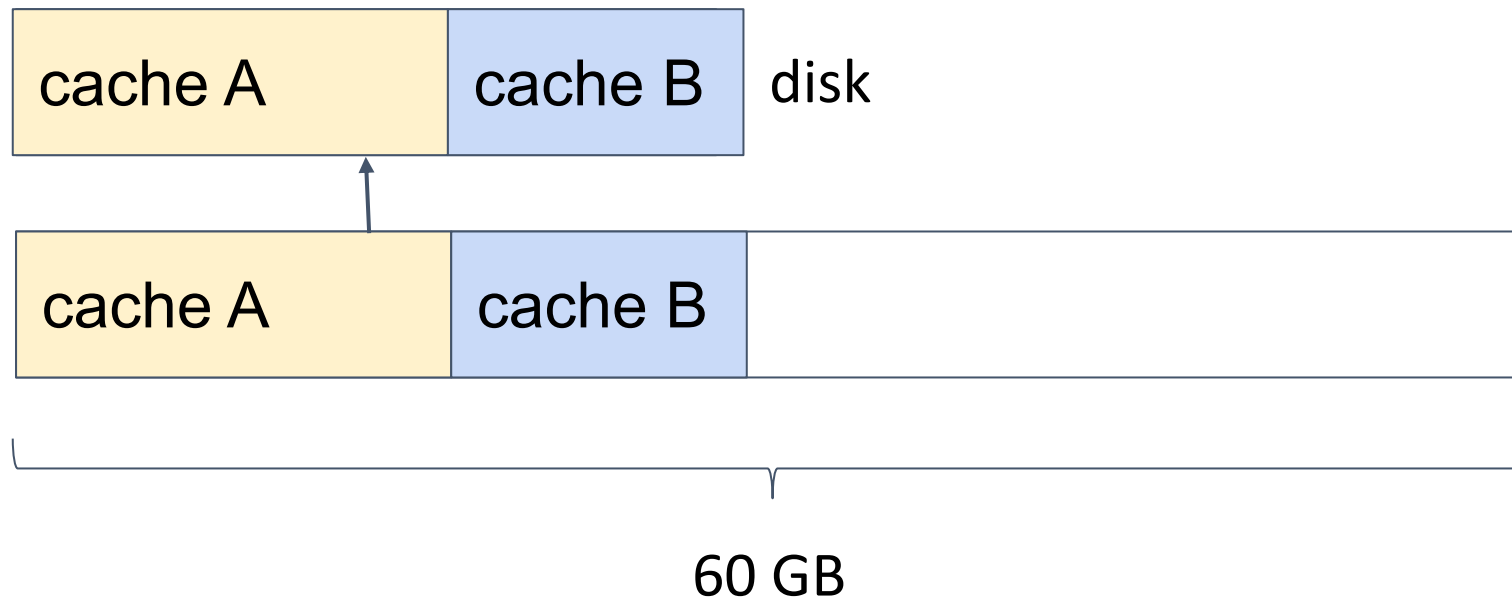
- Persistent mode: all pages on disk, subset of pages in RAM
- Cold pages replaced to disk on demand

```
new DataStorageConfiguration()  
    .setDefaultDataRegionConfiguration(  
        new DataRegionConfiguration().setMaxSize(60L * 1024 * 1024 * 1024)  
        .setPersistenceEnabled(true));
```

Use case: fast restart and cheaper storing



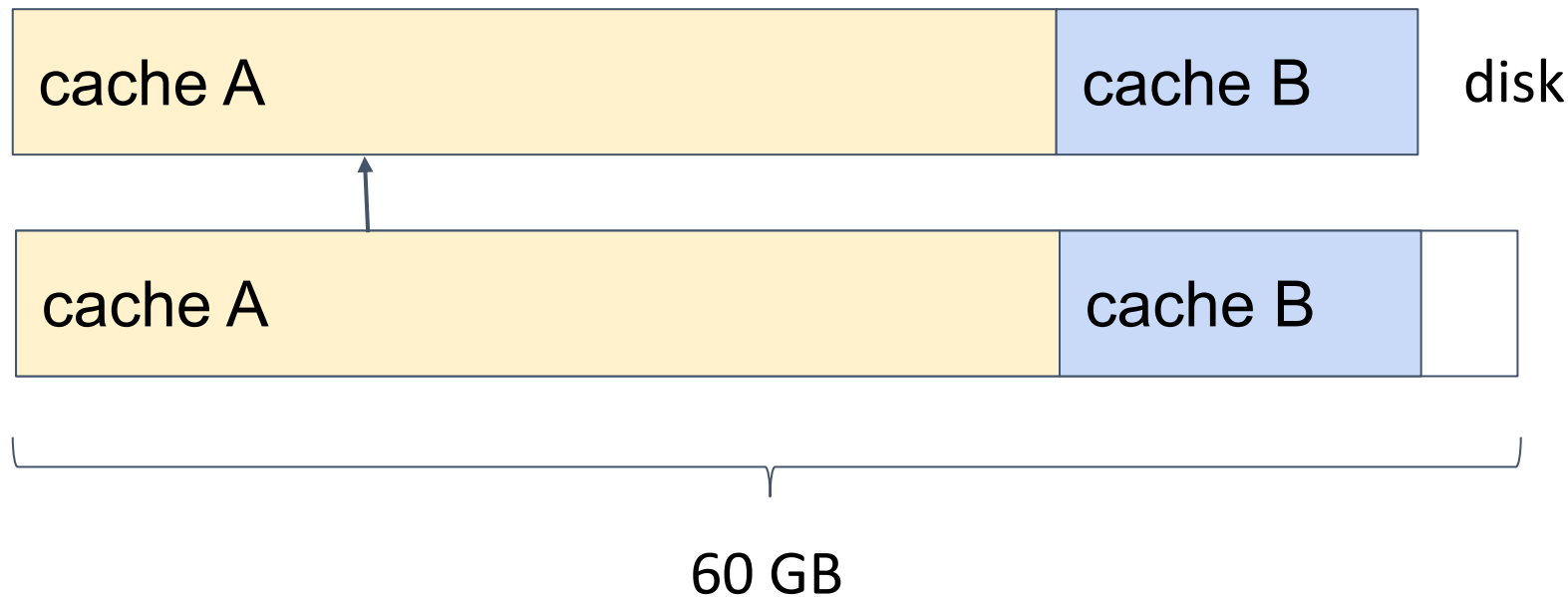
- Persistent mode: all pages on disk, subset of pages in RAM
- Cold pages replaced to disk on demand



Use case: fast restart and cheaper storing



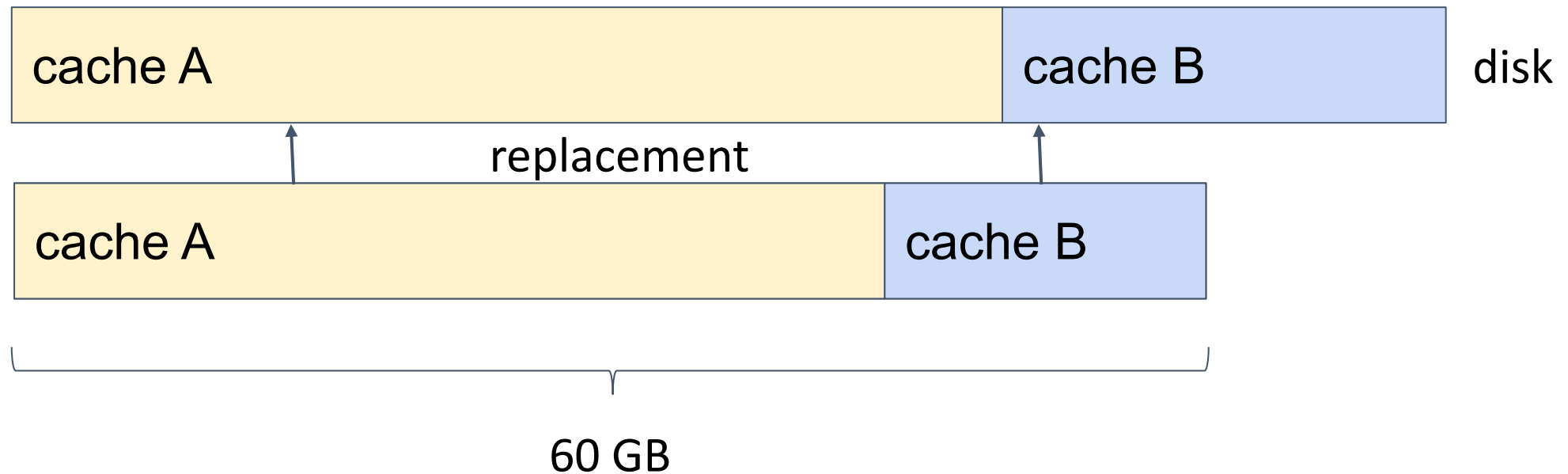
- Persistent mode: all pages on disk, subset of pages in RAM
- Cold pages replaced to disk on demand



Use case: fast restart and cheaper storing



- Persistent mode: all pages on disk, subset of pages in RAM
- Cold pages replaced to disk on demand



Use case: hot and cold data



Use case: hot and cold data



- Small memory region for big cold dataset

Use case: hot and cold data



- Small memory region for big cold dataset
- Large memory region for small hot dataset

Use case: hot and cold data



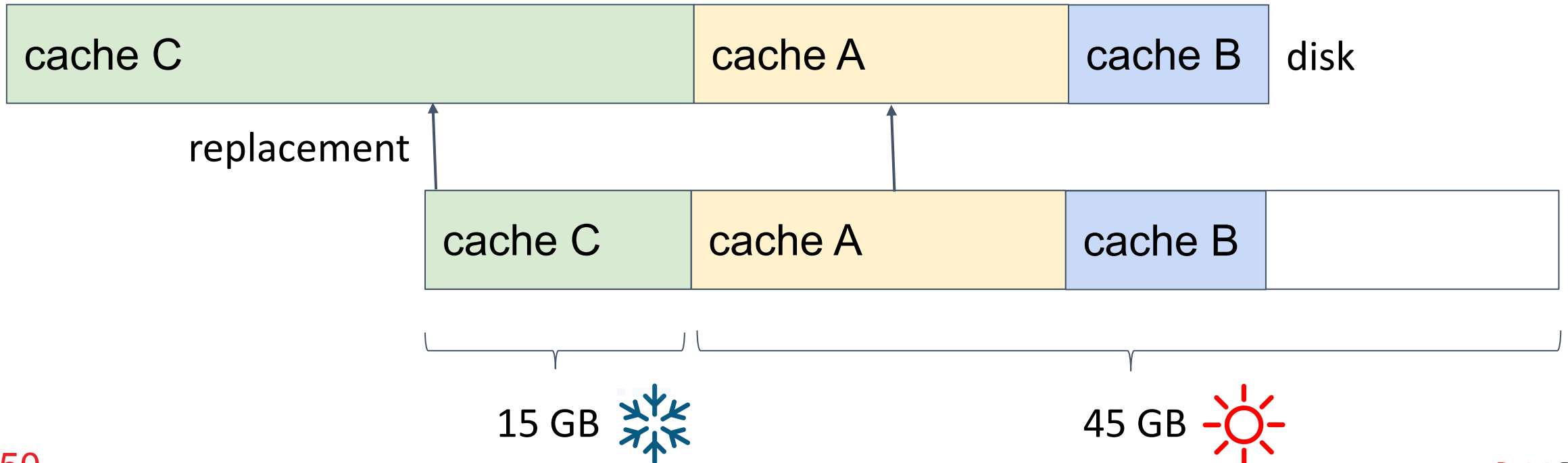
- Small memory region for big cold dataset
- Large memory region for small hot dataset

```
new DataStorageConfiguration()  
    .setDefaultDataRegionConfiguration(  
        new DataRegionConfiguration().setMaxSize(45L * 1024 * 1024 * 1024)  
        .setPersistenceEnabled(true))  
    .setDataRegionConfigurations(new DataRegionConfiguration().setName("cold")  
        .setMaxSize(15L * 1024 * 1024 * 1024)  
        .setPersistenceEnabled(true)));
```

Use case: hot and cold data



- Small memory region for big cold dataset
- Large memory region for small hot dataset



Agenda



- Features that in-memory data grids lack
- Apache Ignite way: durability through page memory architecture
- **Durability: use cases and solutions**
 - Storage management use cases
 - Data backups use cases
- Durability: performance tricks

From theory to practice: Data Snapshots



- Use cases:
 - Disaster protection

From theory to practice: Data Snapshots



- Use cases:
 - Disaster protection
 - Optimization: snapshots of non-volatile data

From theory to practice: Data Snapshots



- Use cases:
 - Disaster protection
 - Optimization: snapshots of non-volatile data
 - When local snapshot is not enough: remote snapshot catalog

Use case: snapshot for disaster protection



Use case: snapshot for disaster protection



- Snapshot create
 - Background process
 - Current state of disk store copied to snapshot directory



Use case: snapshot for disaster protection



- Snapshot create
 - Background process
 - Current state of disk store copied to snapshot directory
- Snapshot restore
 - Disk storage is replaced by previously saved state



Use case: regular snapshots of non-volatile data



Use case: regular snapshots of non-volatile data



- Incremental snapshot create
 - Only changed pages are written

Use case: regular snapshots of non-volatile data



- Incremental snapshot create
 - Only changed pages are written
- Special page type to track changes

idx=0	Meta page
idx=1	Tracking page 0101010100 1 110001001
idx=2	Regular page
idx=3	Regular page
idx=4	Regular page

← data update

Use case: local snapshot is not enough



Use case: local snapshot is not enough



- Complete disaster (local snapshots are lost as well)

Use case: local snapshot is not enough



- Complete disaster (local snapshots are lost as well)
- Daily snapshot catalog

Use case: local snapshot is not enough



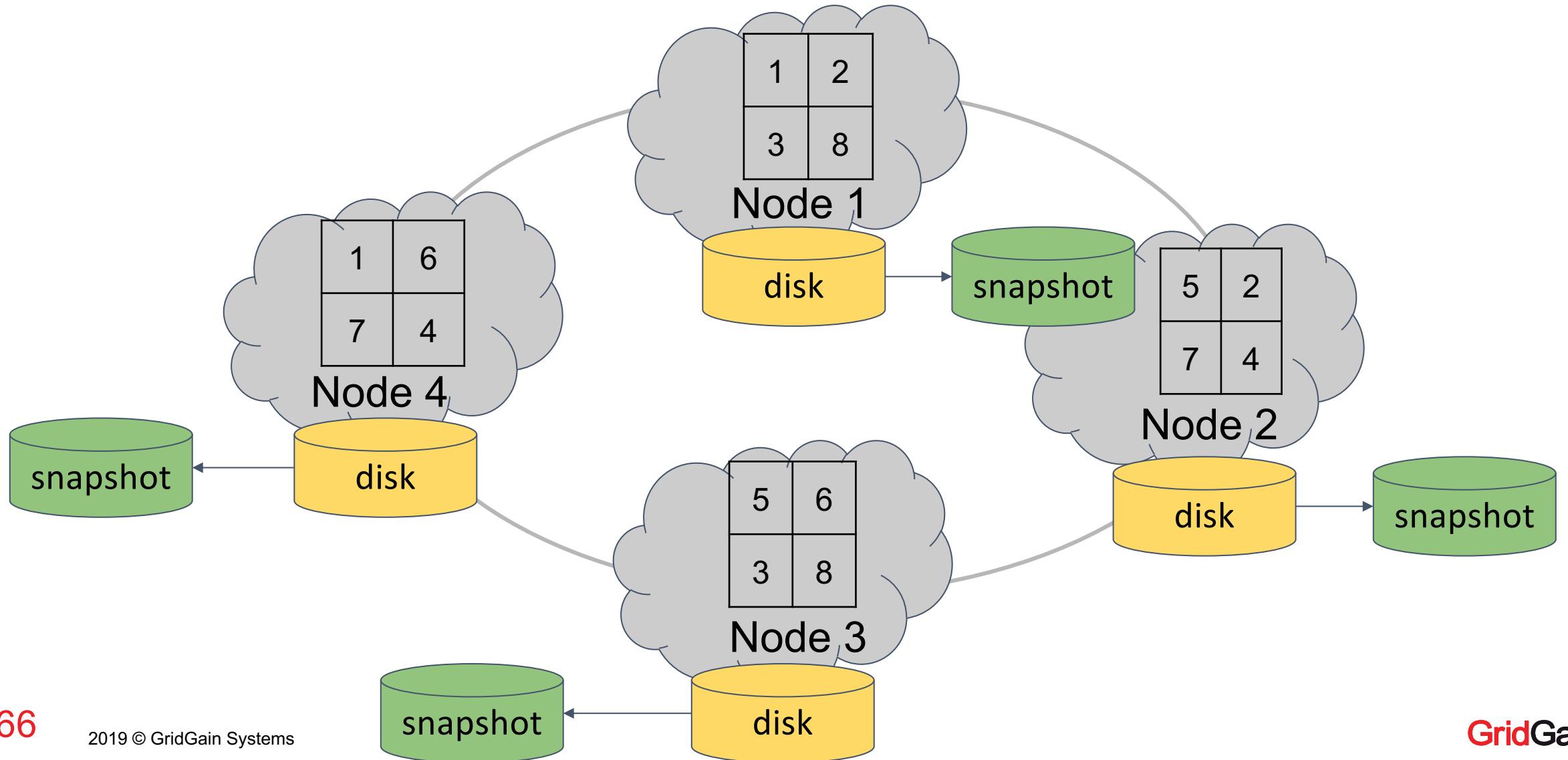
- Complete disaster (local snapshots are lost as well)
- Daily snapshot catalog
- Restore after topology change

Use case: local snapshot is not enough

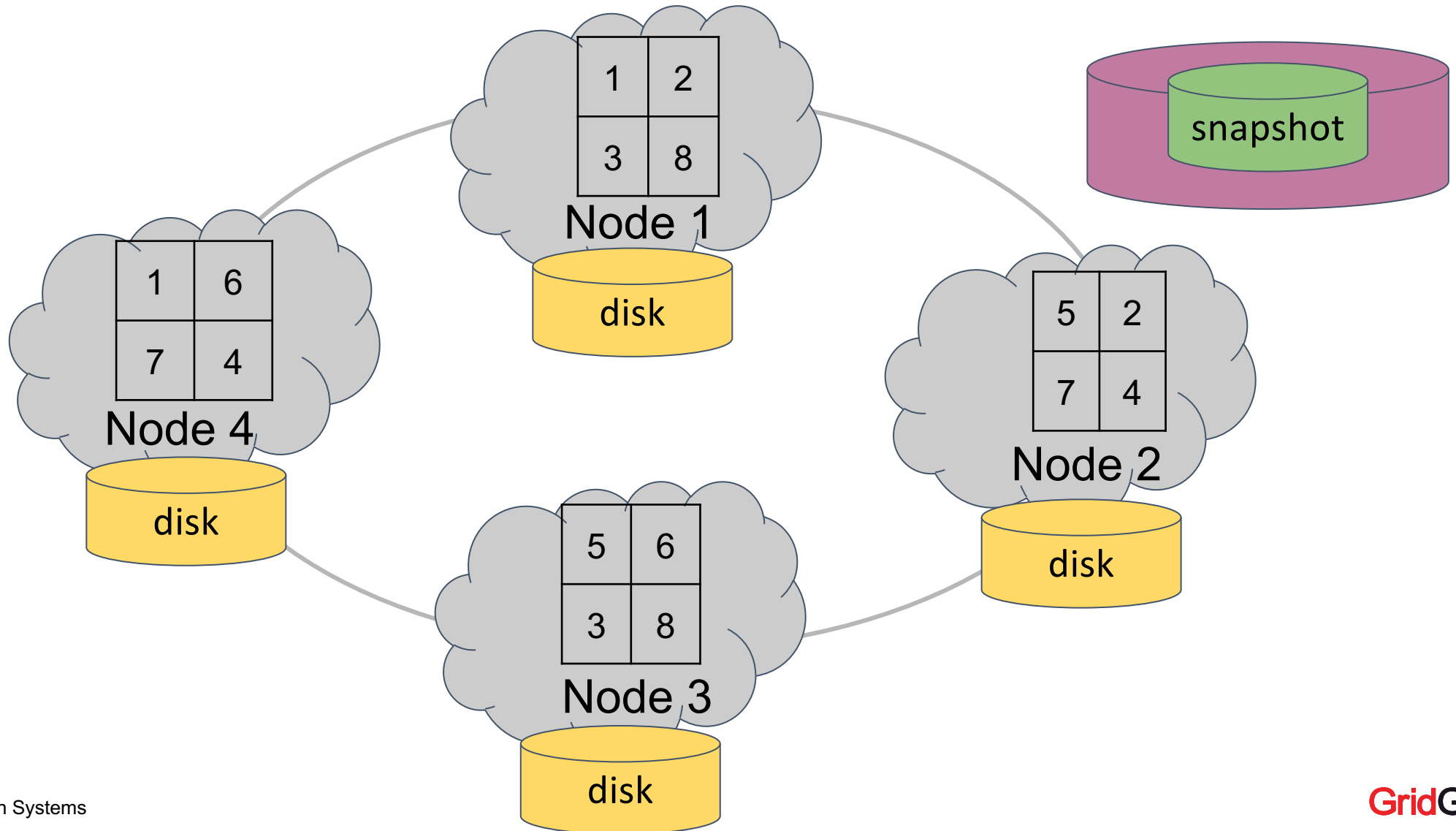


- Snapshot move to shared folder
 - Data from the whole cluster is moved to reliable network storage

Snapshot move to shared folder



Snapshot move to shared folder



Use case: local snapshot is not enough



- Snapshot move to shared folder
 - Data from the whole cluster is moved to reliable network storage
- Snapshot restore from shared folder
 - Even if topology was changed, all data partitions will be found

Agenda



- Features that in-memory data grids lack
- Apache Ignite way: durability through page memory architecture
- Durability: use cases and solutions
 - Storage management use cases
 - Data backups use cases
- Durability: performance tricks

Memory / disk ratio affects performance directly

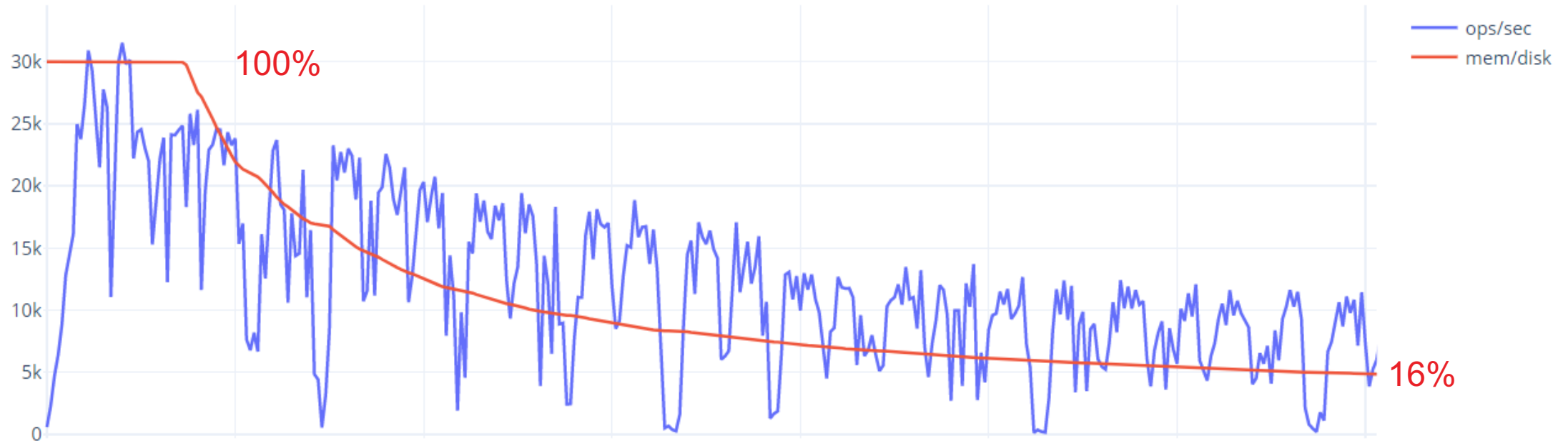


- Every page absent in RAM will require synchronous read

Memory / disk ratio affects performance directly



- Every page absent in RAM will require synchronous read
- Latency grows along with share of “disk only” pages



Use Throttling when disk is slower than load

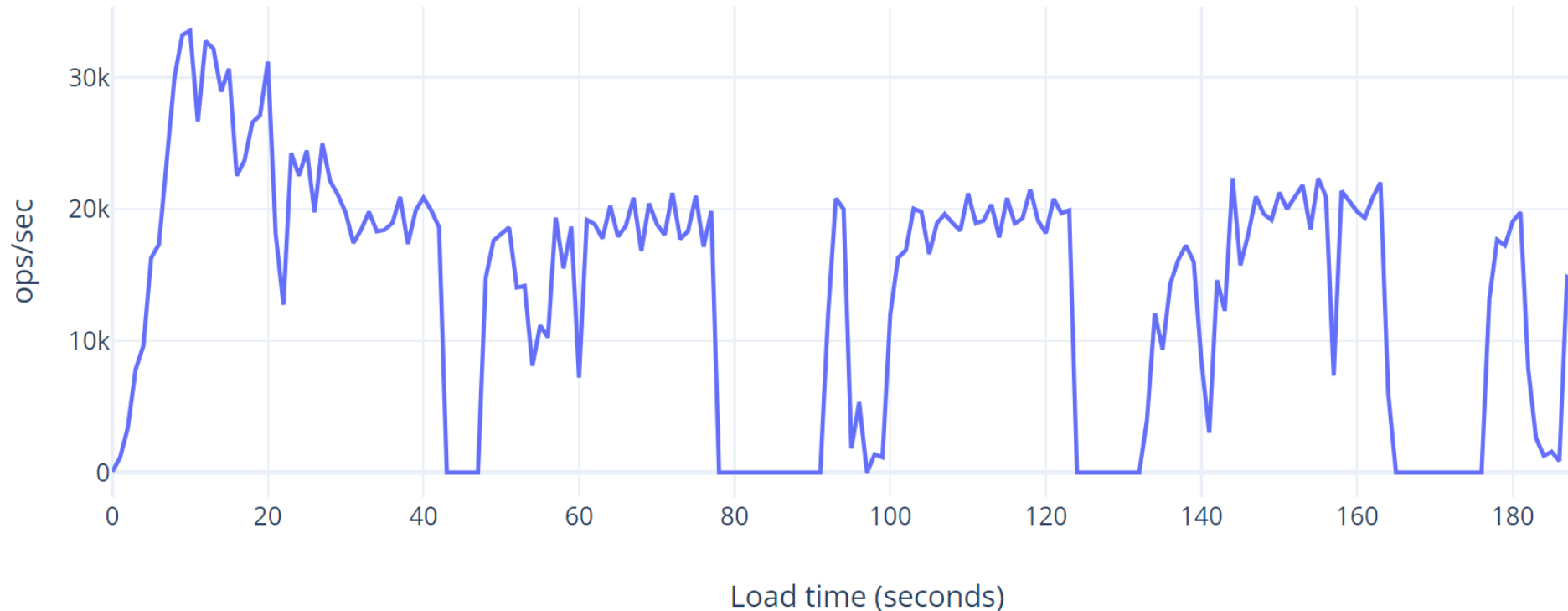


- Peak load throughput can be higher than disk throughput

Use Throttling when disk is slower than load



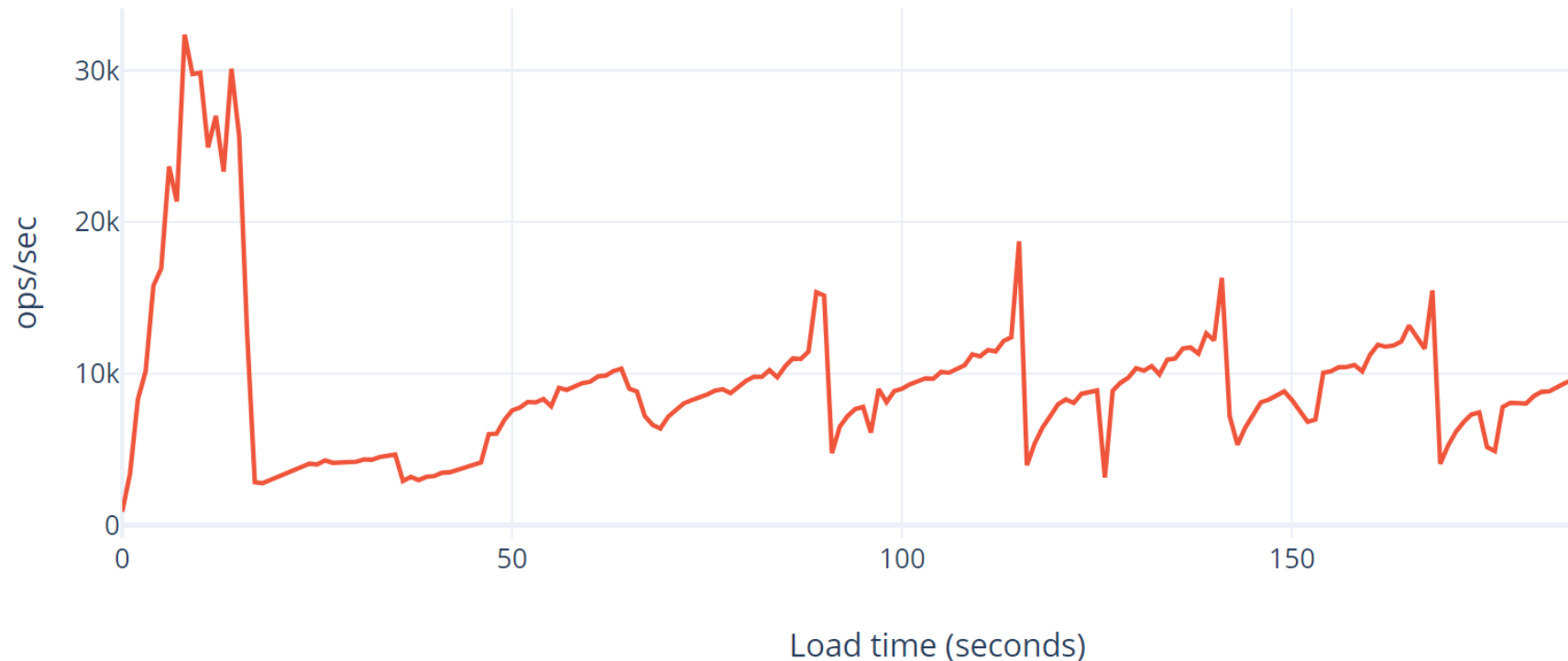
- Peak load throughput can be higher than disk throughput



Use Throttling when disk is slower than load



- Peak load throughput can be higher than disk throughput
`dataStorageCfg.setWriteThrottlingEnabled(true);`



Overprovision your SSD

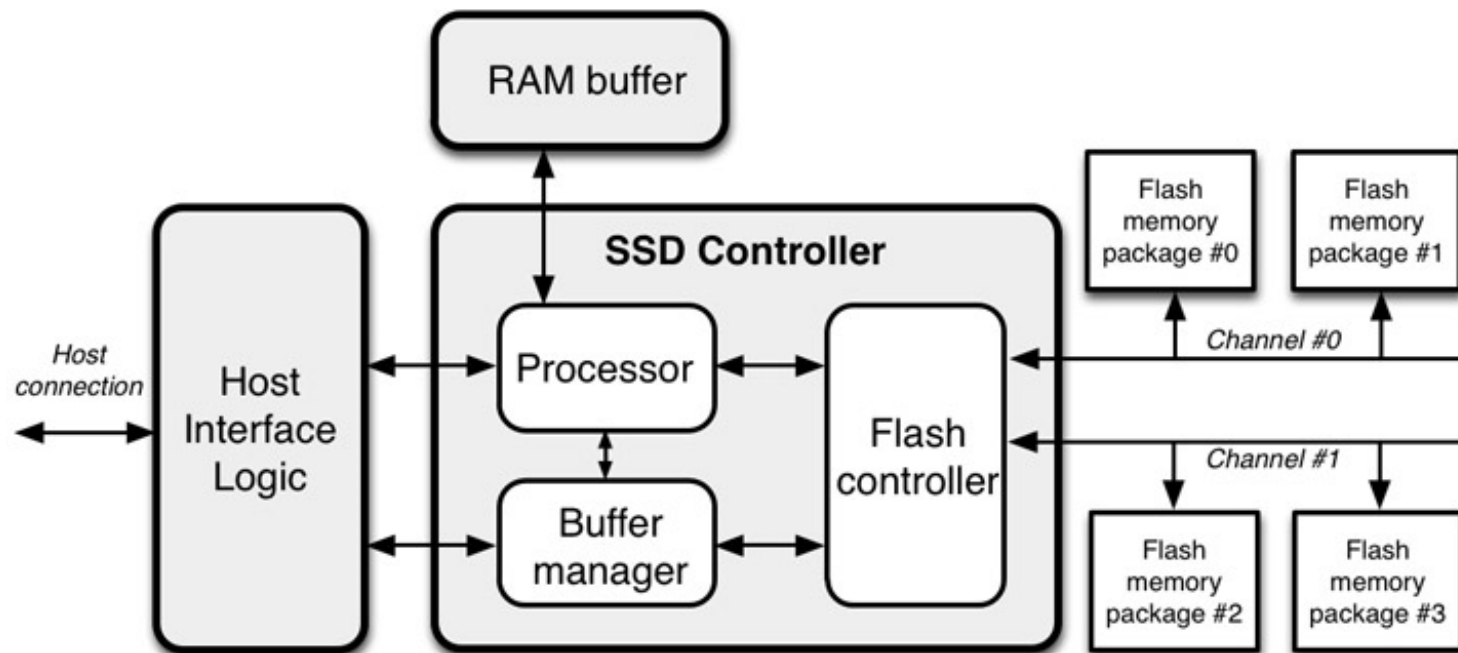


- SSDs are designed to be easily adopted by OS
 - “write K bytes to 0xFF...” interface like HDD

Overprovision your SSD



- SSDs are designed to be easily adopted by OS
 - “write K bytes to 0xFF...” interface like HDD
- But actually SSD is a complex computer itself



Overprovision your SSD



- SSD has pages and blocks (64/128/256 pages)

Overprovision your SSD



- SSD has pages and blocks (64/128/256 pages)
- Data is written in page granularity, erased in block granularity

Overprovision your SSD



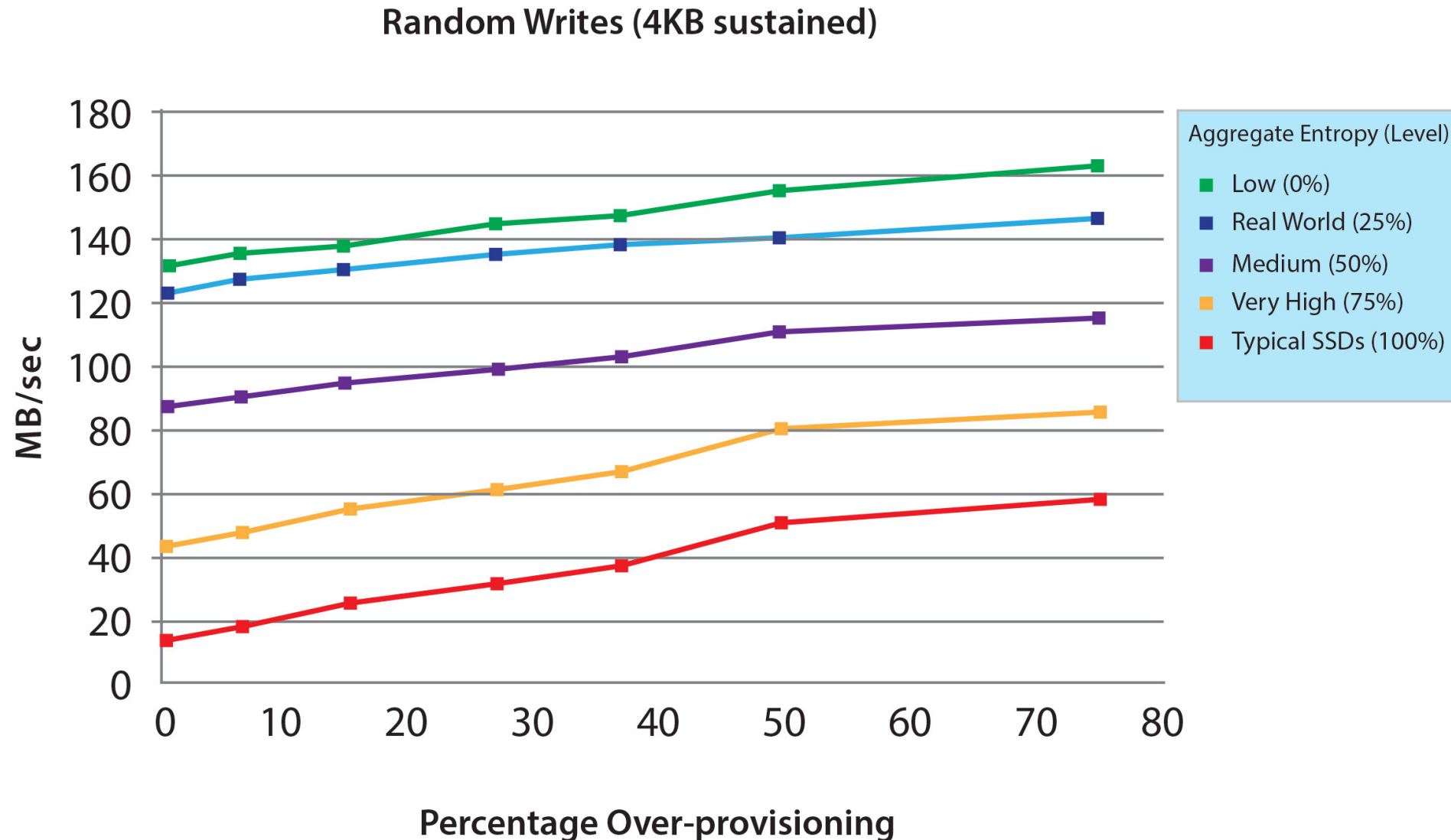
- SSD has pages and blocks (64/128/256 pages)
- Data is written in page granularity, erased in block granularity
- Block erase requires shifting useful data to another block

Overprovision your SSD



- SSD has pages and blocks (64/128/256 pages)
- Data is written in page granularity, erased in block granularity
- Block erase requires shifting useful data to another block
- Shifting is easier when more free blocks are available

Overprovision your SSD



Overprovision your SSD

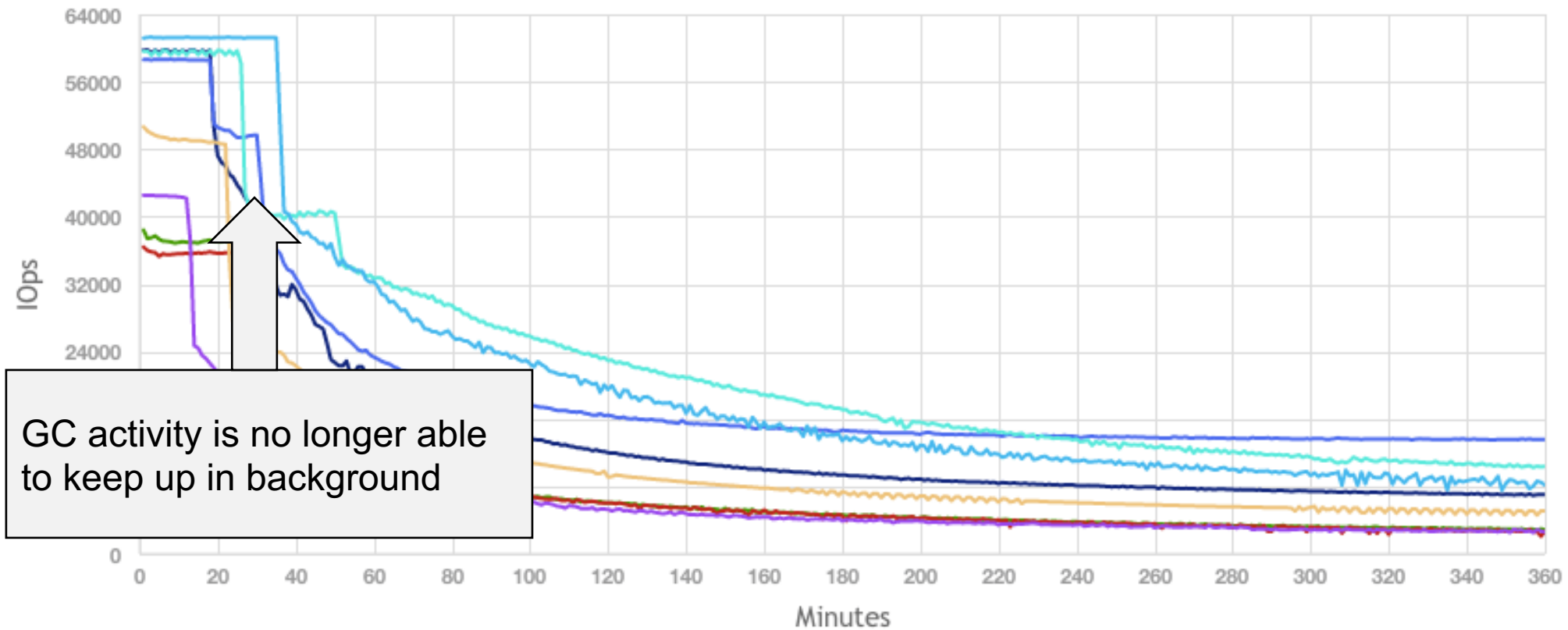


- Beware: SSD performance issues do not appear immediately

Overprovision your SSD



- Beware: SSD performance issues do not appear immediately



Disable WAL on initial data load



- Every data update is written twice in persistent mode

Disable WAL on initial data load



- Every data update is written twice in persistent mode
 - To journal (write-ahead log), synchronously

Disable WAL on initial data load



- Every data update is written twice in persistent mode
 - To journal (write-ahead log), synchronously
 - To disk storage (on checkpoint), asynchronously

Disable WAL on initial data load



- Every data update is written twice in persistent mode
 - To journal (write-ahead log), synchronously
 - To disk storage (on checkpoint), asynchronously
- WAL can be disabled on purpose

Disable WAL on initial data load



- Every data update is written twice in persistent mode
 - To journal (write-ahead log), synchronously
 - To disk storage (on checkpoint), asynchronously
- WAL can be disabled on purpose
 - Crash recovery is not guaranteed

Disable WAL on initial data load



- Every data update is written twice in persistent mode
 - To journal (write-ahead log), synchronously
 - To disk storage (on checkpoint), asynchronously
- WAL can be disabled on purpose
 - Crash recovery is not guaranteed
 - At least 2x load throughput boost

Disable WAL on initial data load



- Every data update is written twice in persistent mode
 - To journal (write-ahead log), synchronously
 - To disk storage (on checkpoint), asynchronously
- WAL can be disabled on purpose
 - Crash recovery is not guaranteed
 - At least 2x load throughput boost
- `igniteCluster.disableWal(cacheToLoad);`

Consider using separate devices



- Persistent Ignite node has four disk write activities

Consider using separate devices



- Persistent Ignite node has four disk write activities
 - Checkpointing

Consider using separate devices



- Persistent Ignite node has four disk write activities
 - Checkpointing
 - Writing WAL

Consider using separate devices



- Persistent Ignite node has four disk write activities
 - Checkpointing
 - Writing WAL
 - Transferring old WAL segments to WAL archive dir

Consider using separate devices



- Persistent Ignite node has four disk write activities
 - Checkpointing
 - Writing WAL
 - Transferring old WAL segments to WAL archive dir
 - Data snapshotting

Consider using separate devices



- Persistent Ignite node has four disk write activities
 - Checkpointing
 - Writing WAL
 - Transferring old WAL segments to WAL archive dir
 - Data snapshotting
- Separate path can be configured for each
 - `dataStorageCfg.setStoragePath(...);`
 - `dataStorageCfg.setWalPath(...);`
 - `dataStorageCfg.setWalArchivePath(...);`
 - `snapshotCfg.setSnapshotsPath(...);`

Performance tips: summary



- Plan memory / disk ratio for your performance requirements
- Use throttling for smooth throughput
- Overprovision your SSD
- Disable WAL on initial data load
- Split disk activities on separate storage devices



Thanks for your attention!
Questions?