Building a File System for Fun and Profit

Introduction

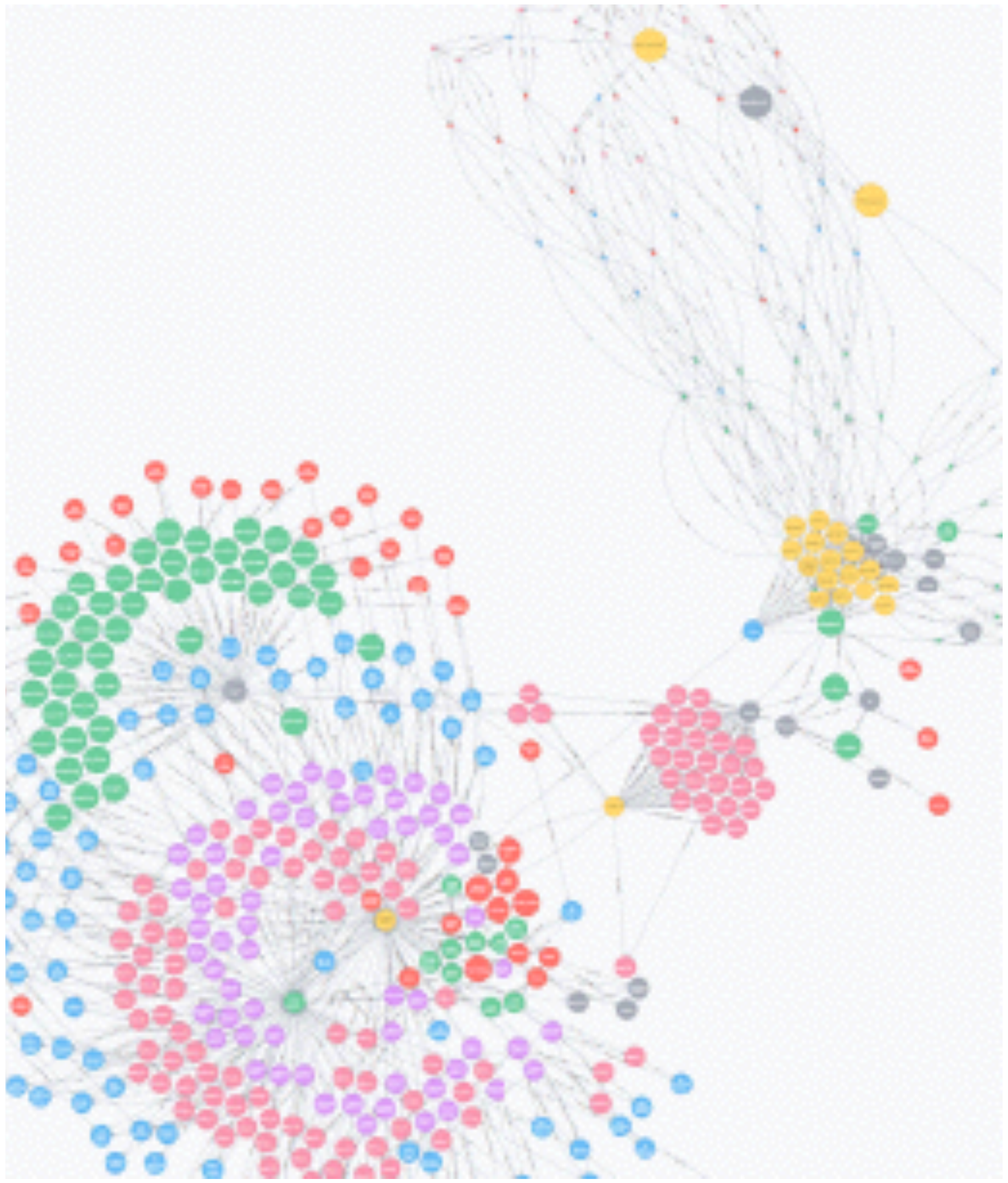
- About myself
- Neo4j in Brief
- What are File Systems
- Problems with Todays File Systems
- What do we Need from a File System
- DBFS Internals

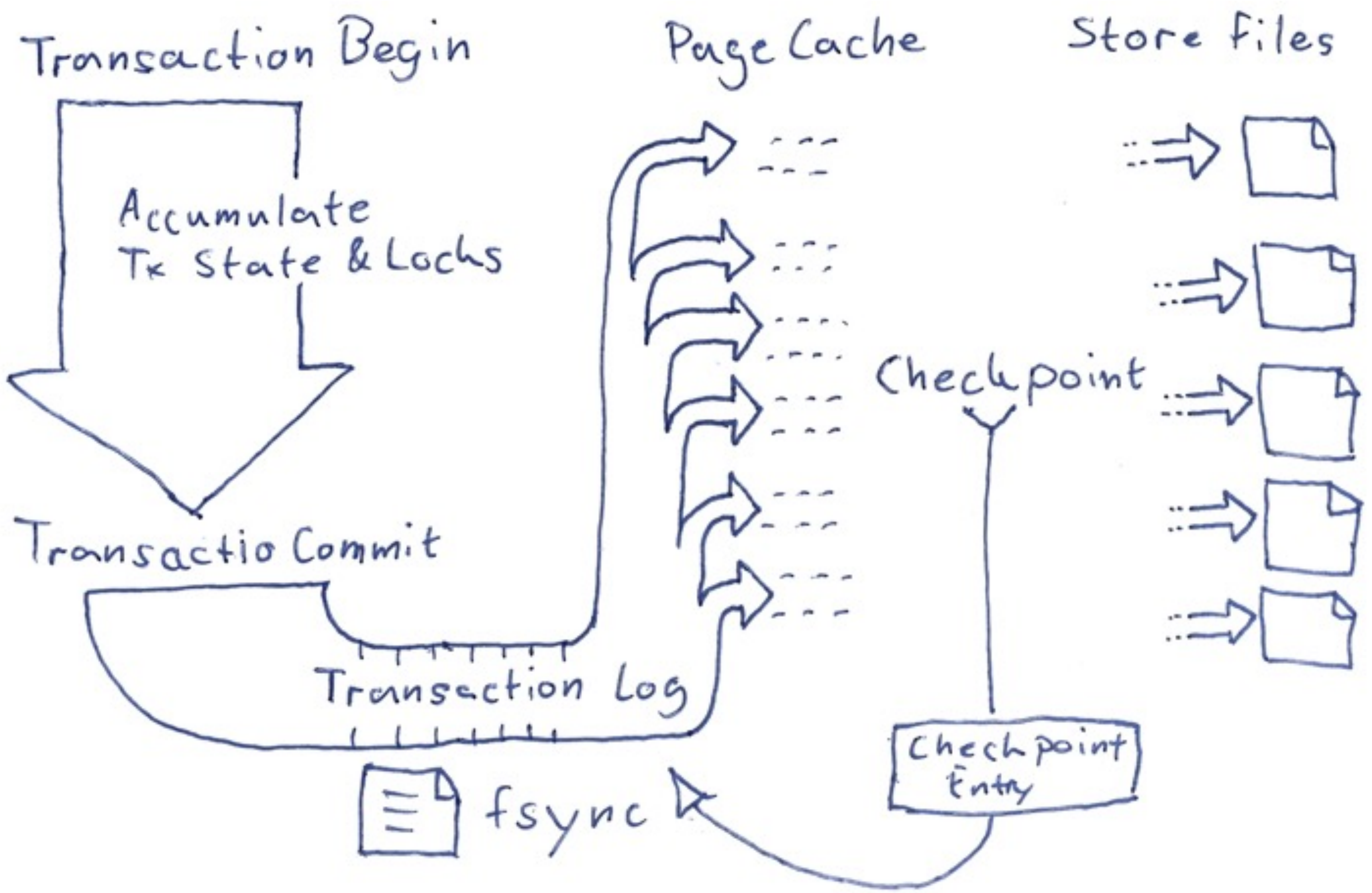
About Myself

- Chris Vest
- @chvest
- Writes database systems for fun and profit
- Specifically Neo4j – the worlds leading graph database

Neo4j in Brief

- ACID Transactional Database
- For Graphs; Vertices & Edges
- With Properties & Labels
- Cypher Query Language
- Single-image
- Replication for HA





Neo4j Internals

Basically a bog-standard database on the inside

File Systems

- Illusion of organised, durable, dynamically growing, and named byte arrays
- On top of giant array of blocks, each a fixed-size array of bytes

File Systems

- Directories; hierarchy
- Buffered IO
- Journaling
- Meta-data; permissions, modified time

File Systems

- Last access time
- Extended attributes
- Case preserving
- Sparse files
- Locks
- Snapshots
- Alternate data streams / Resource forks
- Hard/Soft Links
- Compression
- Encryption
- Various safety modes
- Direct IO
- Asynchronous IO
- Copy-On-Write
- Truncate
- Range sync
- Quotas

File System APIs

```
// write 3 bytes at offset 2
pwrite(fd, “bar”, 3, 2)
```

File System APIs

```
creat(/dir/log);
write(/dir/log, “2,3,foo”, 7);
pwrite(/dir/file, “bar”, 3, 2);
unlink(/dir/log);
```

👍 data=journal
❌ data=ordered
❌ data=writeback

File System APIs

```
creat(/dir/log);
write(/dir/log, “2,3,foo”, 7);
fsync(/dir/log);
pwrite(/dir/file, “bar”, 3, 2);
unlink(/dir/log);
```

👍 data=journal

👍 data=ordered

❌ data=writeback

File System APIs

```
creat(/dir/log);
write(/dir/log, “2,3,[chk],foo”, 7);
fsync(/dir/log);
pwrite(/dir/file, “bar”, 3, 2);
unlink(/dir/log);
```

👍 data=journal

👍 data=ordered

👍 data=writeback

File System APIs

```
creat(/dir/log);
write(/dir/log, “2,3,[chk],foo”, 7);
fsync(/dir/log);
fsync(/dir);
pwrite(/dir/file, “bar”, 3, 2);
fsync(/dir/file);
unlink(/dir/log);
fsync(/dir);
```

👍 almost all file systems

Let’s Take a Step Back

Back to a simpler time, with simpler file systems

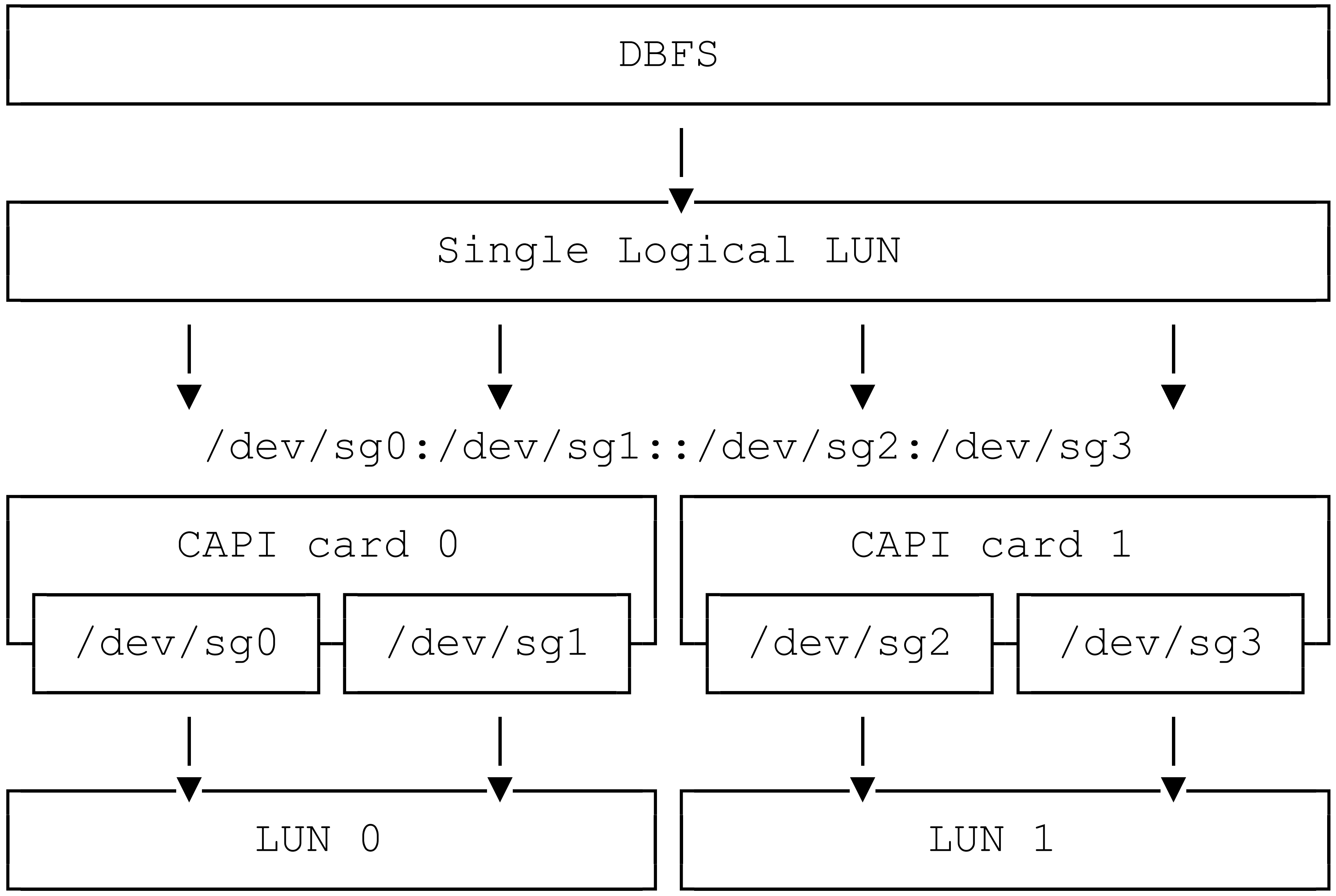
Let’s Take a Step Back

- Direct preadv, pwritev
- Create, truncate, delete
- No pre-cleaning allocation
- Rename, list
- Big files (16 TiB file size limit is not enough, ext4!)
- Small number of files (in the small tens)

DBFS

- DataBase File System
- (or Dumb Block File System)
- Exactly the functions we need
- Exabyte file sizes
- Transparent heterogeneous device striping
- Up to 500 files! 😄 And no directories 🤔

Device Striping



Device Striping

Device Striping

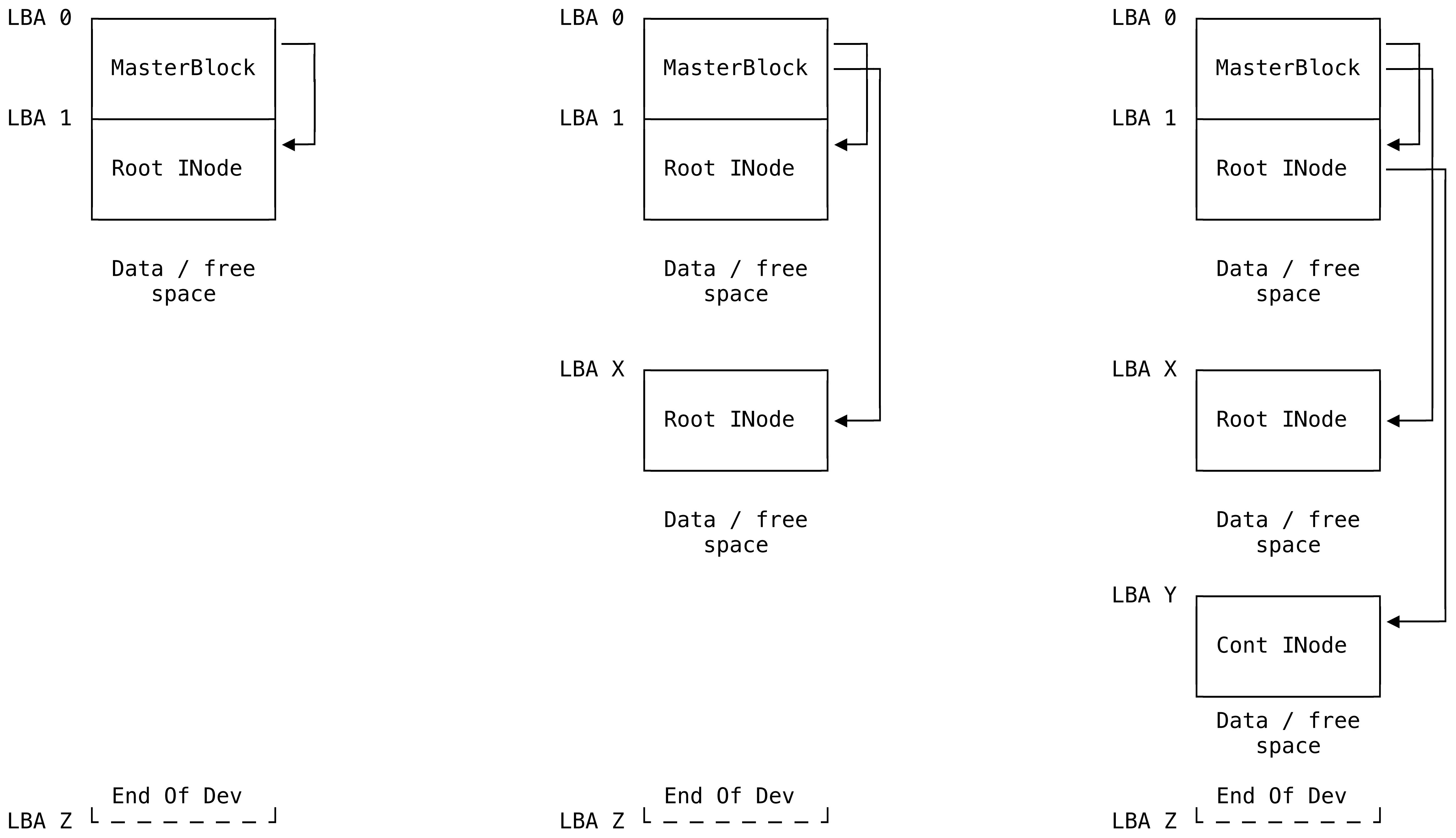
DBFS Internals

LBA #0: The MasterBlock

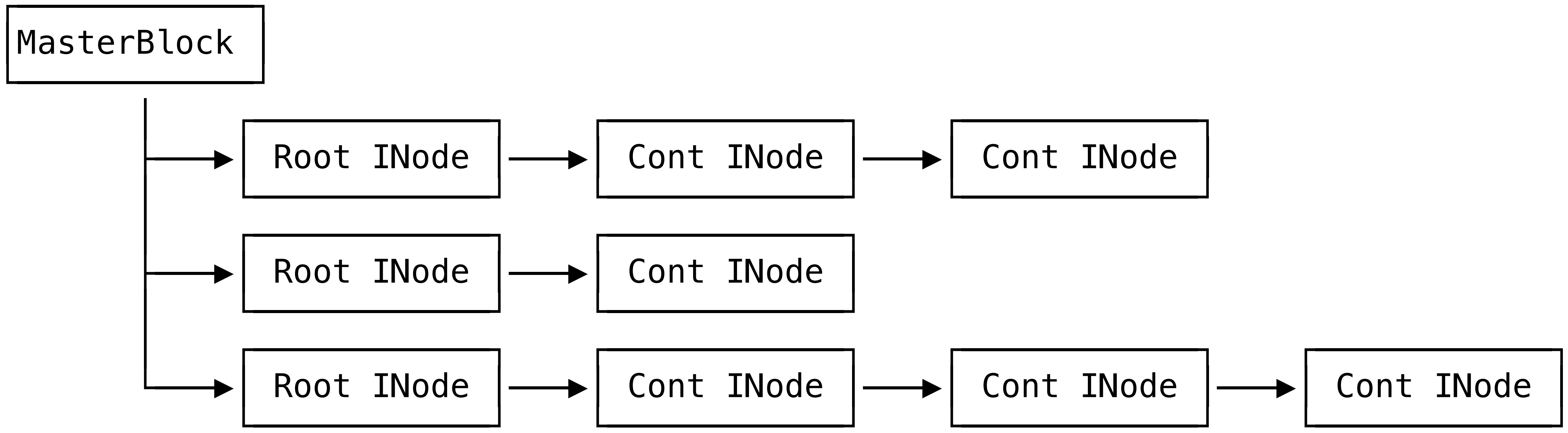
Root INode LBA: 8 bytes
Root INode LBA: 8 bytes
Root INode LBA: 8 bytes
...
0xDBF5_0042_7474_7306L + VERSION

```
Plus in-memory parts:
 * Root INode list
 * NavigableSet of all INodes
```

DBFS Internals



DBFS Internals



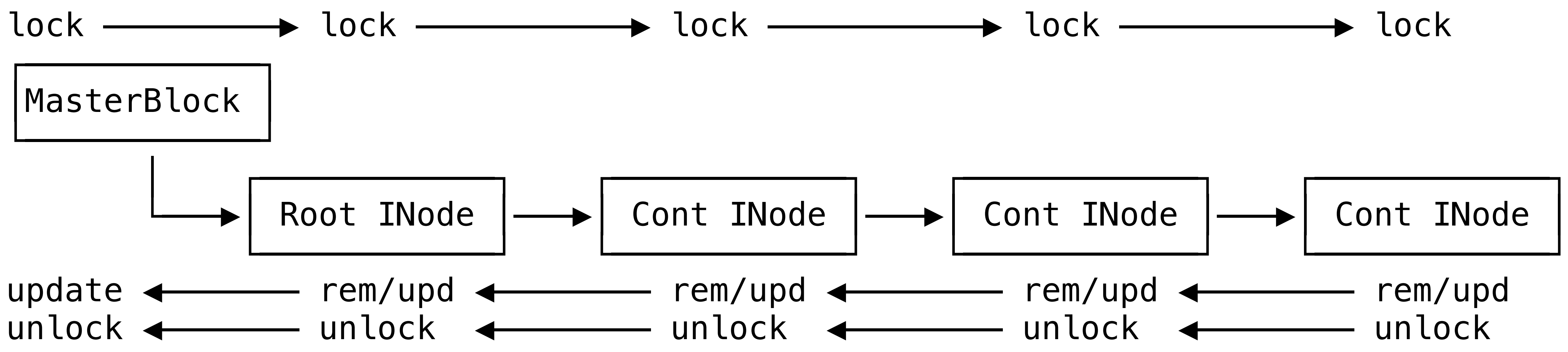
DBFS Internals

INode

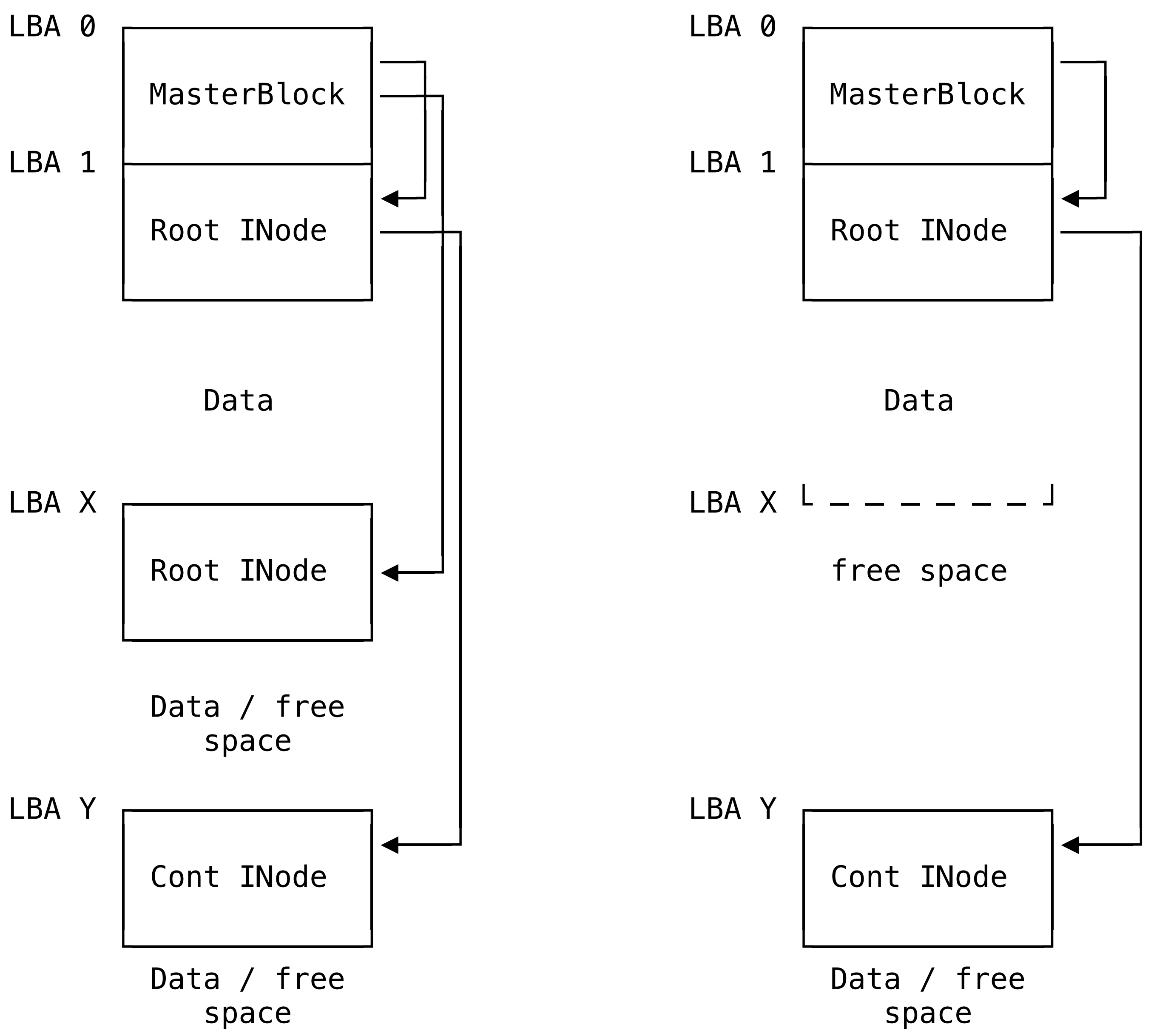
```
* size
* limit LBA
* continuation LBA
* continuation of LBA
* capacity

Optional NUL-terminated
filename for the rest of
the block
```

```
... data / free space ...
```

DBFS Internals



DBFS Internals



DBFS & Neo4j on CAPI Flash

- All reads & writes are direct & synchronous
- No journal; ordered meta-data updates
- Very low meta-data overhead
- High sequential access sympathy
- High random access sympathy
- Closed source add-on to Neo4j Enterprise Edition