motr

nikita.danilov@seagate.com

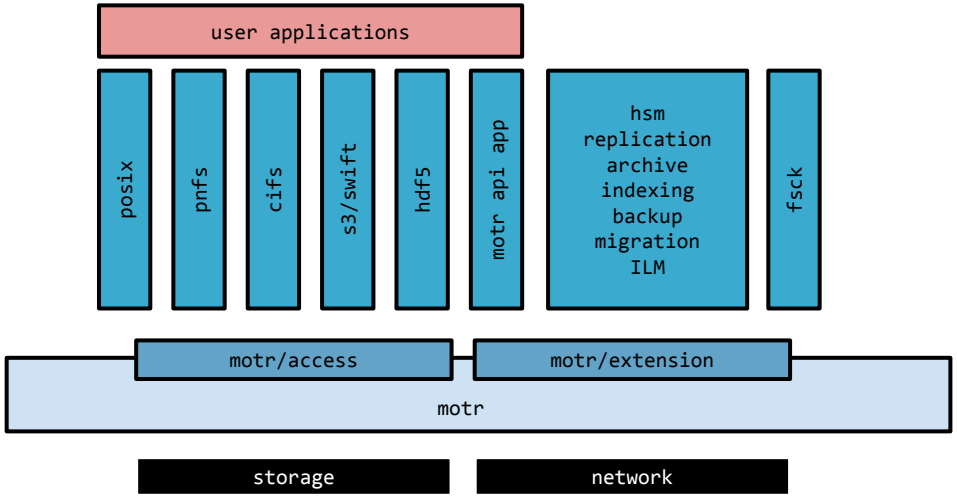
a scalable storage platform

Contents

- What is motr?
- Architecture overview
- Use

What is motr

- Storage platform
  - library
  - interface (C, Go, Python)
  - processes
- Typically used through font-ends



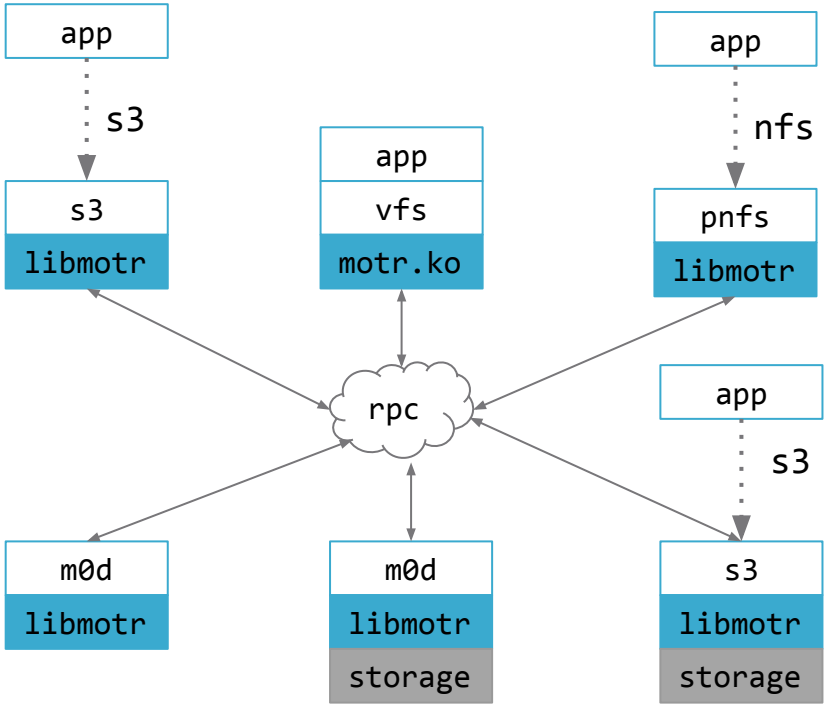
History

- Lustre: parallel file system
- “Let’s try to design for exascale!”
  - Lustre
  - pNFS
  - databases
- Startup (Clusterstor) -> acquisition -> … -> Seagate
- Can the same be applied to cloud?

Interesting features

- Scalability
  - Horizontal: add more nodes to the cluster
    - shared-nothing IO path, no global meta-data
    - scalable tunable fault-tolerance
    - 0-copy (RDMA + direct-IO + checksum-parity offloading)
  - Vertical: use more powerful nodes
    - small number of threads, non-blocking state machines
    - processor affinity, NUMA awareness
- Observability
  - Fine-grained cross-referenced telemetry
  - Analysis tools
- Extensibility
  - Extension interface for *plugins*
- Portability (user space, linux, macos)

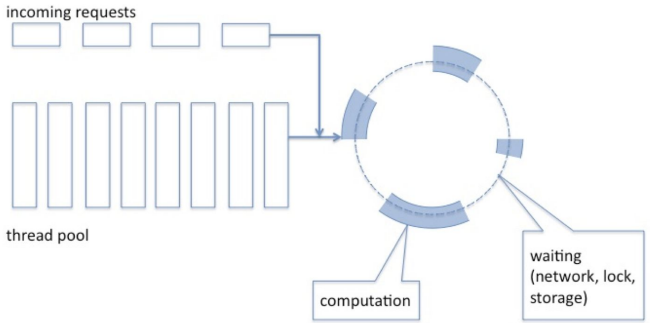
Generic deployment

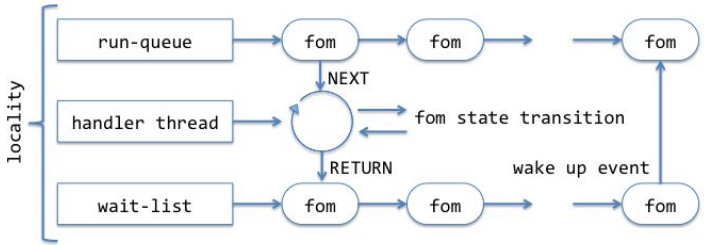


- `libmotr`, `motr.ko` — motr instances
- an instance can be attached to some local storage
- HA sub-system (hare) monitors liveness
- m0d wraps libmotr in a standalone process
- rpc goes over network transport: libfabric or LNet or sockets
- instances provide services (locally and over network):
  - object io (ios)
  - key-value indices (cas)
  - resource management (rm)
  - transactions (dtm), *etc*.
- front-ends interoperate

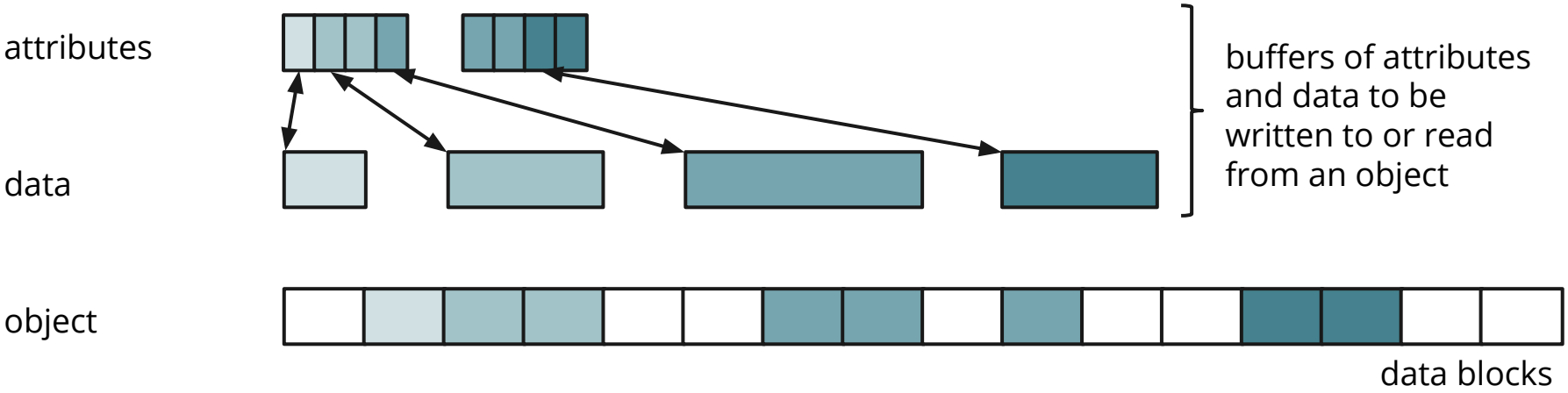
Anatomy of an instance

- thread-per-request:
  - multiple cores, NUMA,
  - locking,
  - cache ping-pong,
  - c10K, many threads
- non-blocking state machines:
  - thread per core
  - non-blocking scheduler
  - locality of reference
  - load balancing
  - long-term scheduling



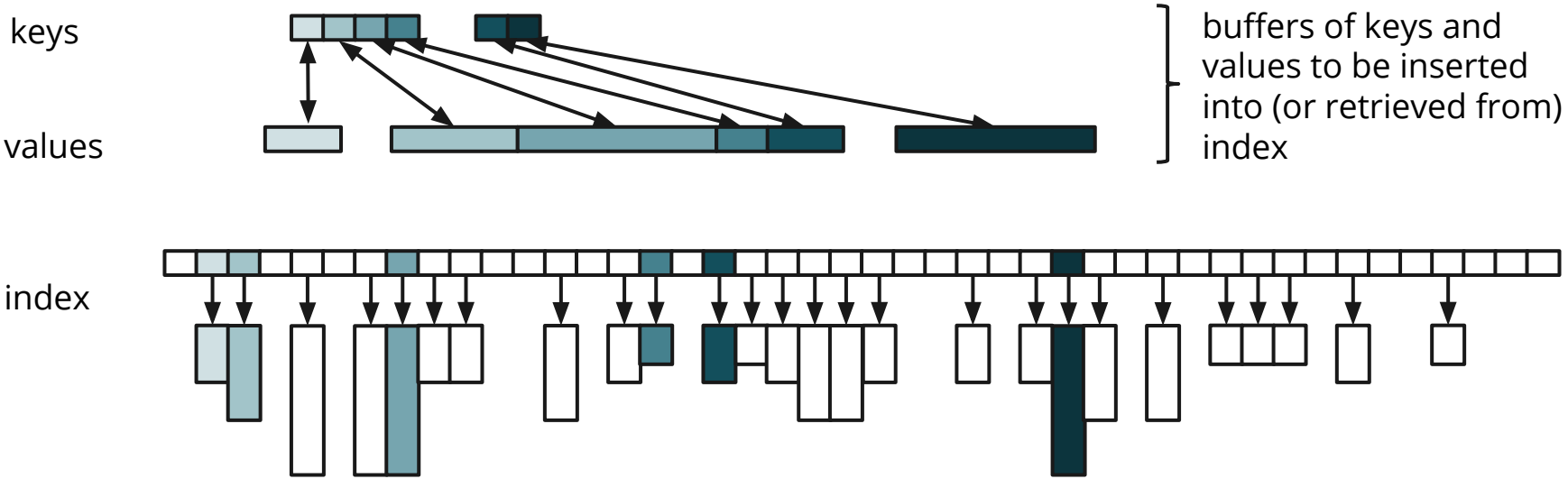


Object api



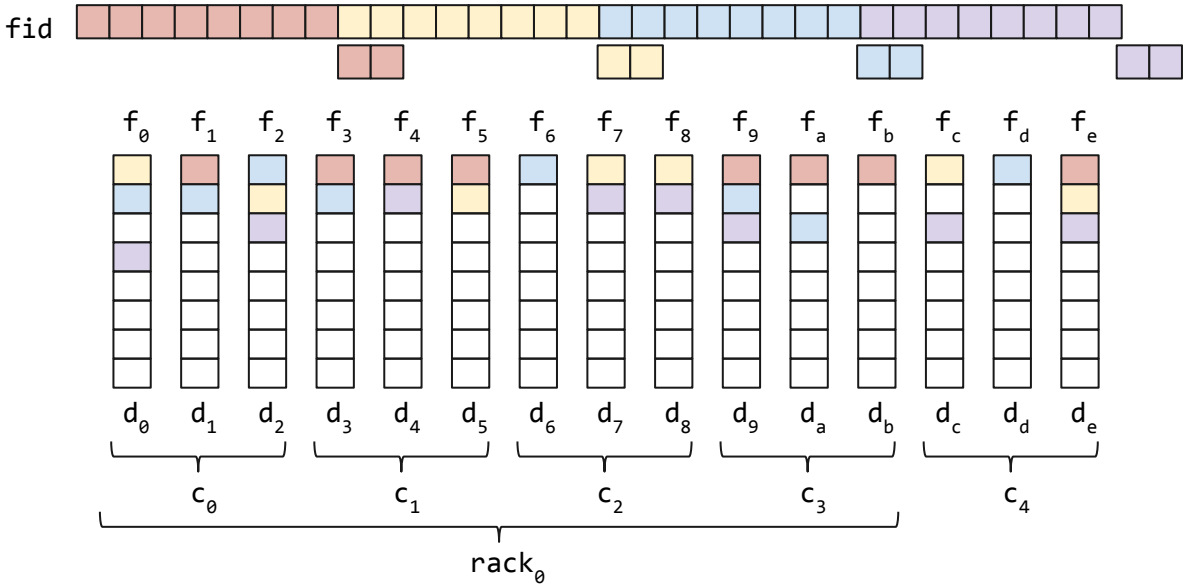

```
obj_read(op, obj, data_buf_vec, attr_buf_vec, extent_vec)
obj_write(op, obj, tx, data_buf_vec, attr_buf_vec, extent_vec)
obj_alloc(op, obj, tx, extent_vec)
obj_free(op, obj, tx, extent_vec)
```

Index api




```
idx_lookup(op, index, key_buf_vec, val_buf_vec)
idx_insert(op, index, tx, key_buf_vec, val_buf_vec)
idx_next(op, index, key_buf_vec, val_buf_vec)
```

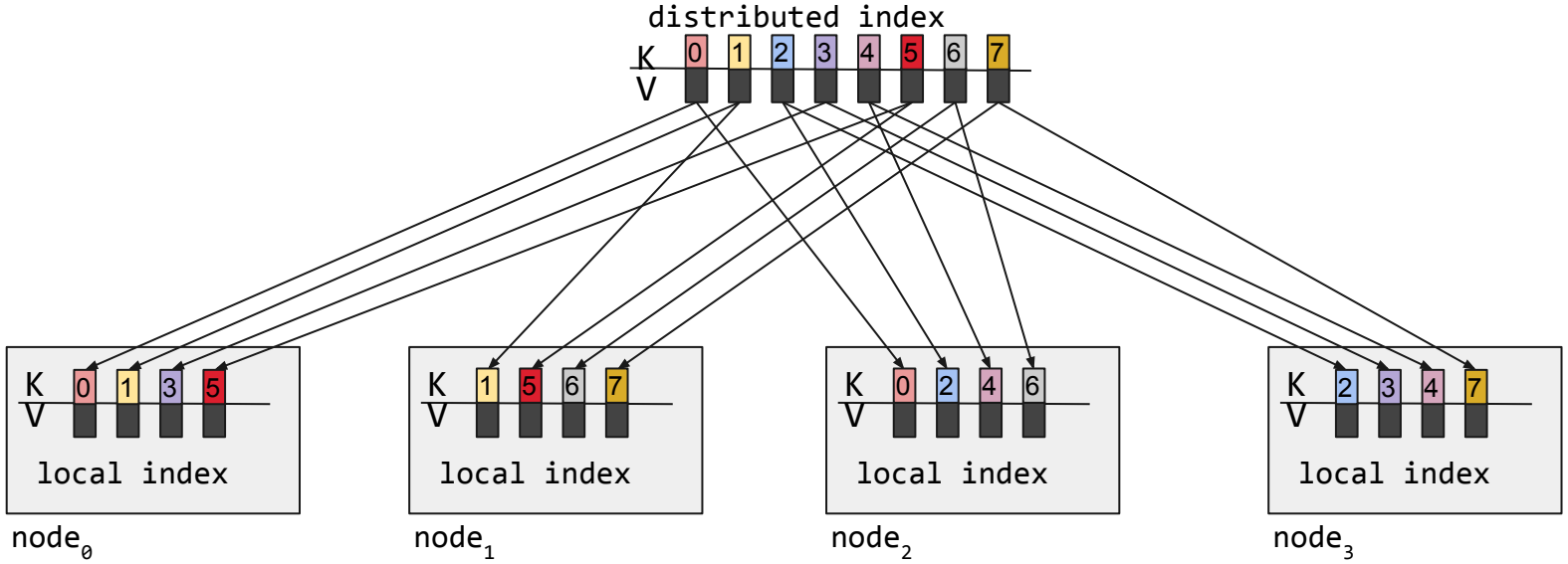
Object layout



- N+K+S striping
  - N - data units
  - K - parity units
  - S - spare units
  - 8+2+0
- Distributes units of each group over large number of devices
- Takes hardware topology into account (drives, expanders, nodes, racks, sites, *etc*.)
- Parity de-clustering: uniform distribution

- Distribution is deterministic—no global meta-data
- Repair to distributed spare space—scales with system size
  - Each drive in the system reads and writes
- *Elastic spare*: repair into free space

Index layout



- Distributed indices: store all *external* meta-data: S3 buckets, POSIX directories, attributes, *etc*.
- Use parity de-clustered layout as objects, but N+K+S with N = 1 (K+1-way replication), fast repair
- Local indices are stored in local transactional back-end
  - Based on direct-mapped memory
  - Suitable for low latency storage like NVMe

Observability: addb

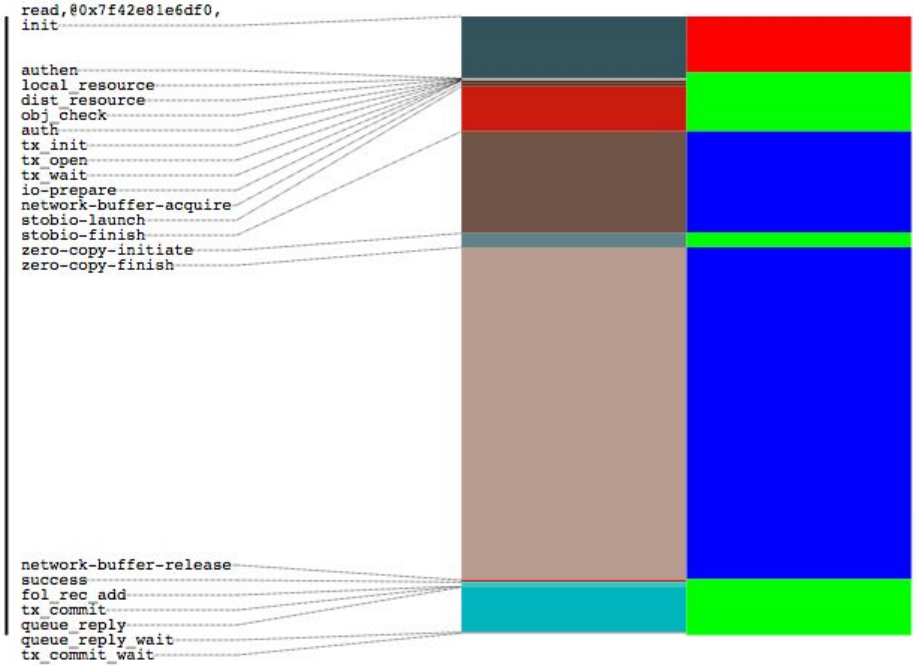
- Systems grow larger and more complex
- How well the system is utilised?
- Is it failure or expected behaviour?
- Is it system or application behaviour?
- Sources of data:
  - system logs
  - operating system
  - application traces
- Very large amount of collected data
- ... or insufficiently detailed, or both
- Difficult to analyse and correlate

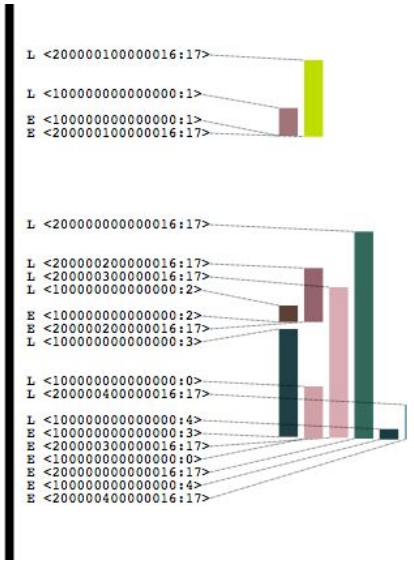
addb: analytics and diagnostics database

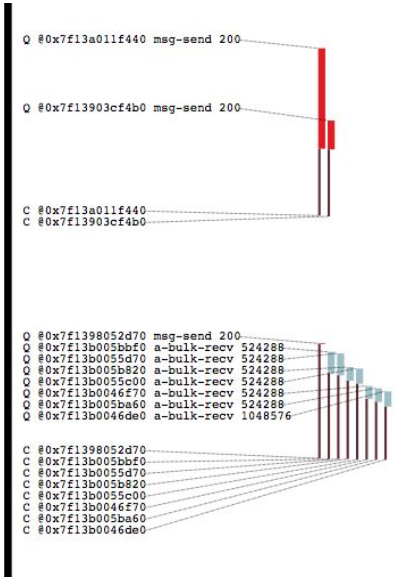
- instrumentation on client and server
- data about operation execution and system state
- passed through network
- cross-referenced
- always on (post-mortem analysis, first incident fix)
- simulation (change configuration, larger system, load mix)

```
* 2021-04-20-14:36:13.687531192 alloc size: 40, addr: @0x7fd27c53eb20
|  node             <f3b62b87d9e642b2:96a4e0520cc5477b>
|  locality         1
|  thread           7fd28f5fe700
|  fom              @0x7fd1f804f710, 'IO fom' transitions: 13 phase: Zero-copy finish
|  stob-io-launch   2021-04-20-14:36:13.629431319, <200000000000003:10000>, count: 8, bvec-nr: 8, ivec-nr: 1, offset: 0
|  stob-io-launch   2021-04-20-14:36:13.666152841, <100000000adf11e:3>, count: 8, bvec-nr: 8, ivec-nr: 8, offset: 65536
```

addb: analysis tools



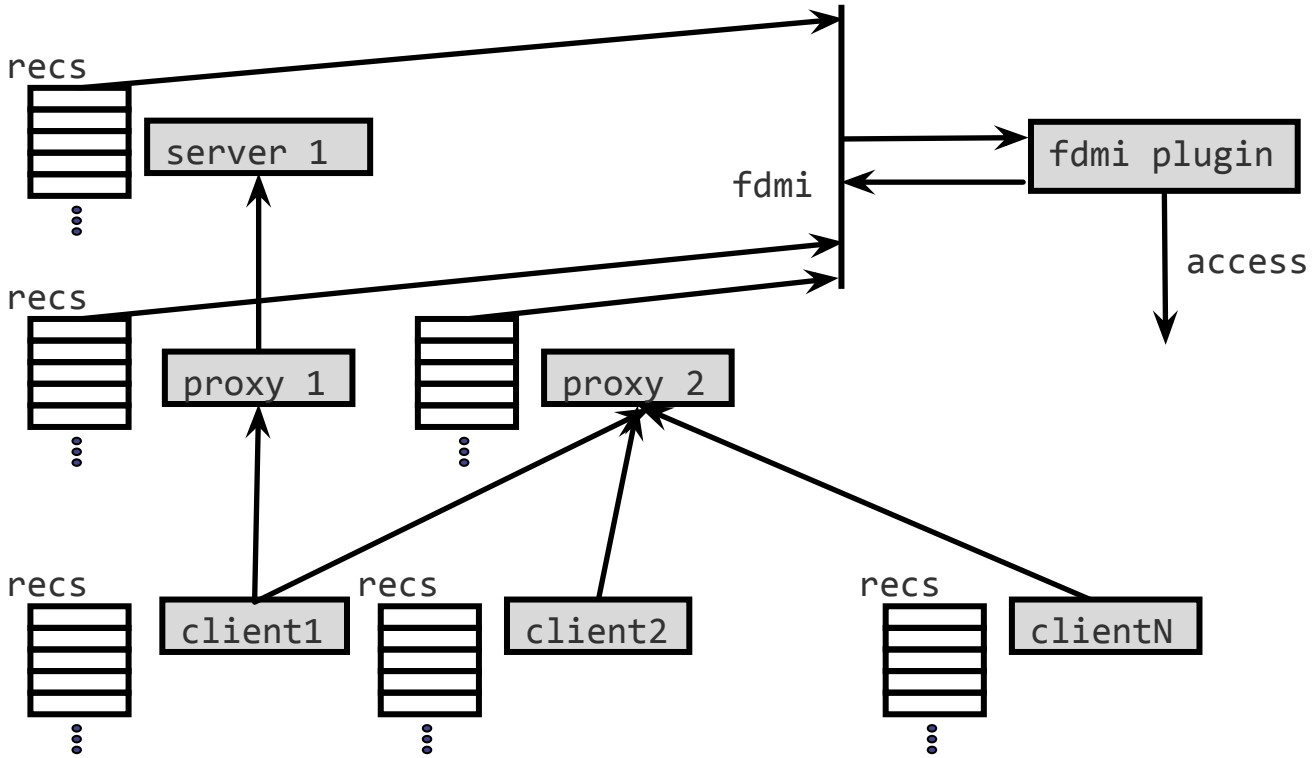




Extensibility: fdmi

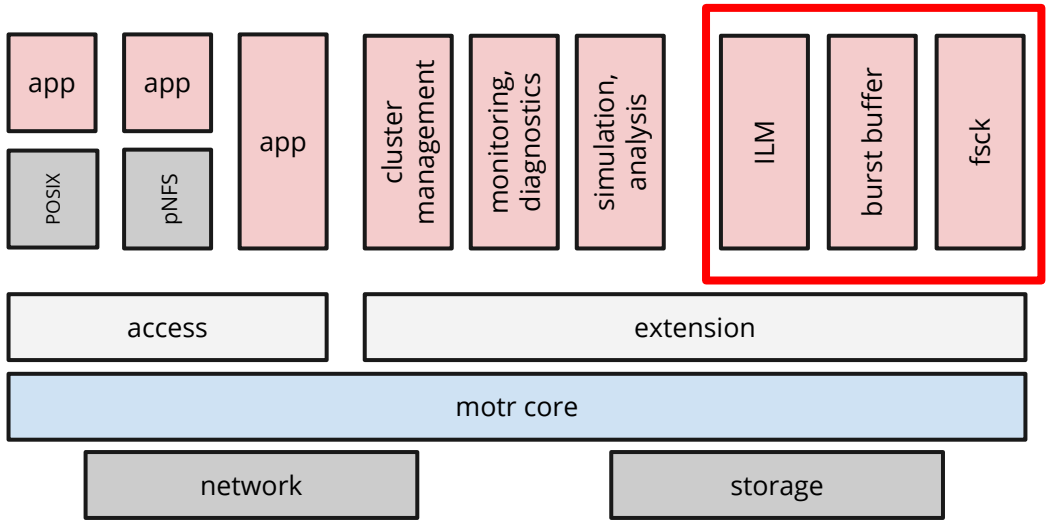
- operation log (fol):
  - record each operation executed by the instance
  - log consists of records, is maintained by each node
- file-system definition and manipulation interface (fdmi)
  - scalable publish-subscribe interface
  - subscribe to records matching certain filter
  - map-reduce-style mechanism to deliver matching records to the subscribers
  - transactional delivery (EOS)
  - delivers: fol records, addb records, HA events, others
- horizontal scalability
  - offload plugin processing and data-structures
  - asynchronous processing, batching

fdmi


recs
server 1
fdmi
fdmi plugin
access
recs
proxy 1
proxy 2
recs
client1
recs
client2
recs
clientN

fdmi

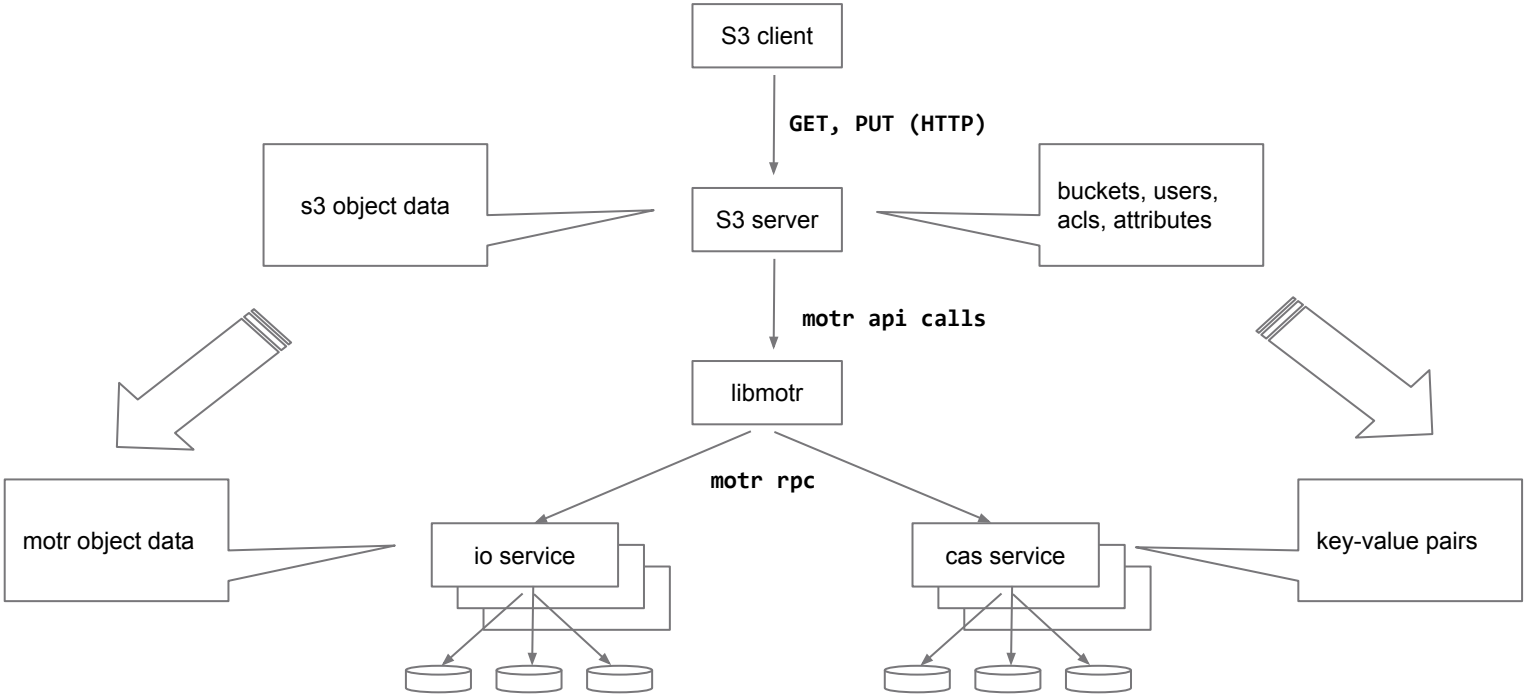
- ILM
  - replication
  - migration
  - backup, archival,
  - hsm
- indexing
- fsck
- data re-structuring
  - proxy de-staging
  - RAID re-striping
- guided interfaces
  - profiling
  - prefetching, destaging



Other services

- DTM: distributed transaction manager
  - groups operations on objects and indices into atomic transactions
  - used by front-ends to implement complex meta-data (*e.g.*, POSIX directories)
- RM: resource manager
  - distributed coherent caching
  - distributed locking, concurrency control
- Other layouts
  - composite layout: glue objects from subobjects (hsm, snapshots)
  - de-duplication, compression, encryption
- containers, function shipping

CORTX



- Motr-based Seagate product
- S3-based object store (CORTX)

Questions?