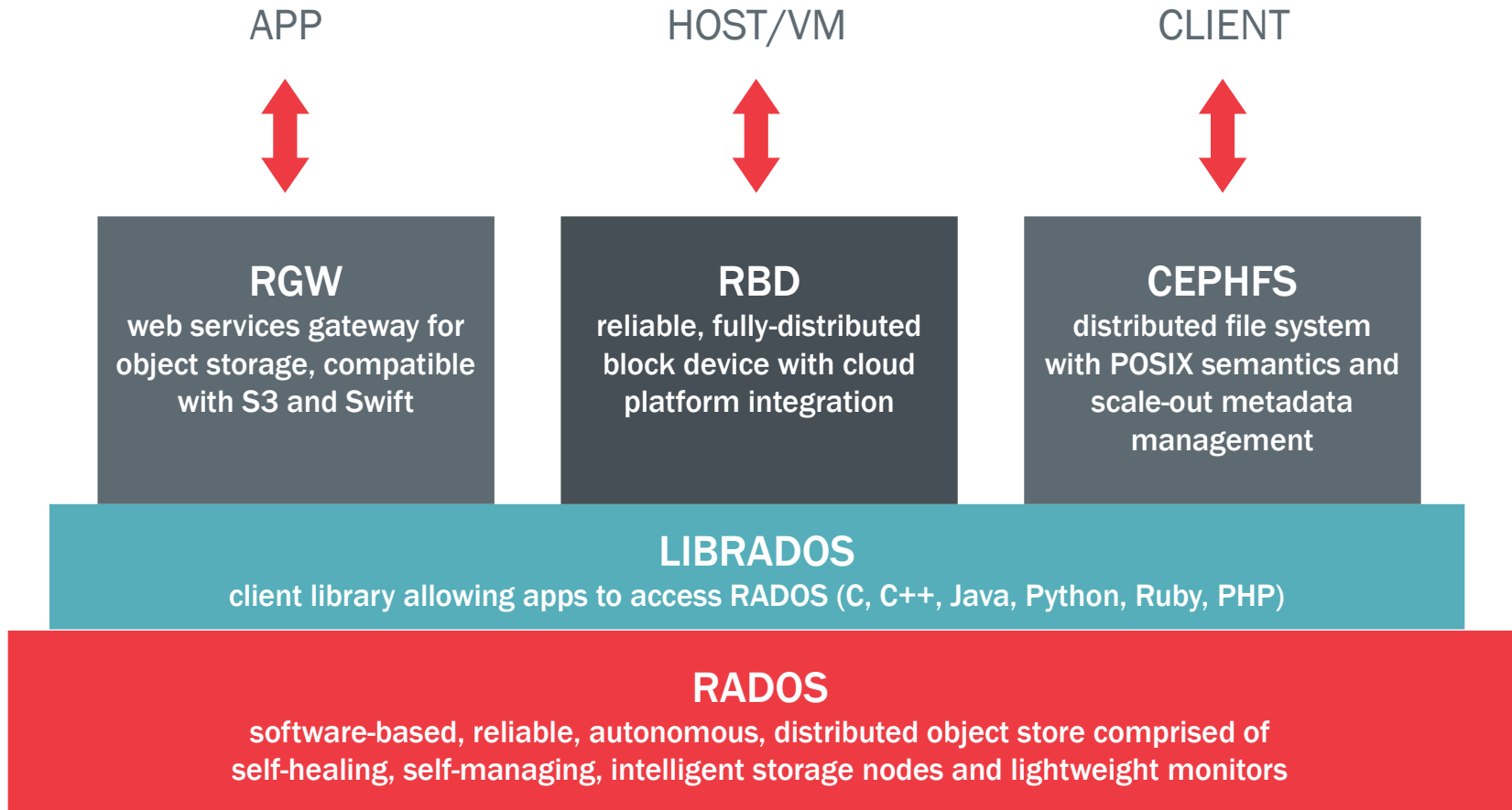


# Ceph Internals & Data Processing Capabilities

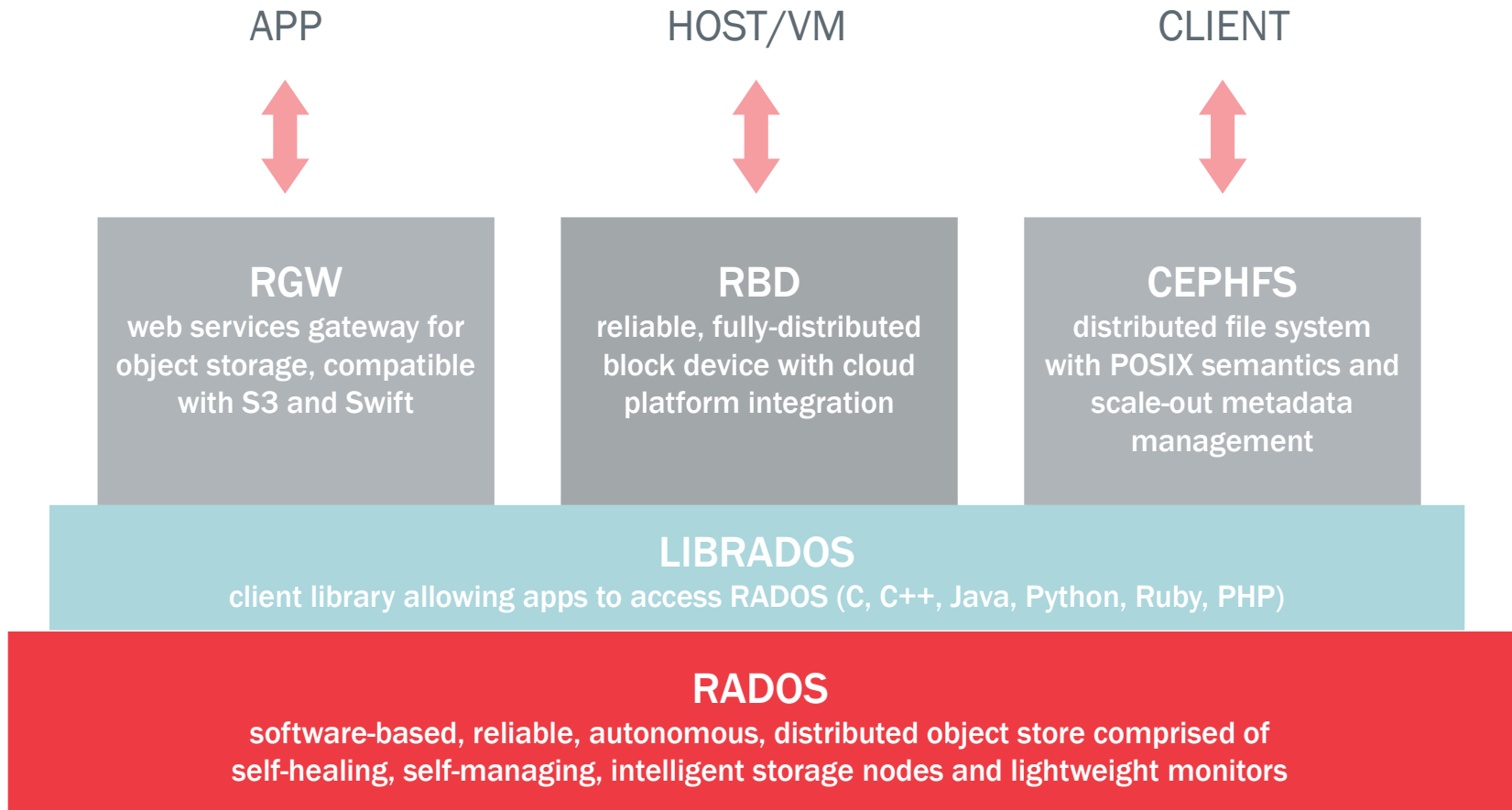
**Joao Eduardo Luis**  
**Senior Software Engineer**  
[jluis@suse.com](mailto:jluis@suse.com) / [joao@suse.de](mailto:joao@suse.de)



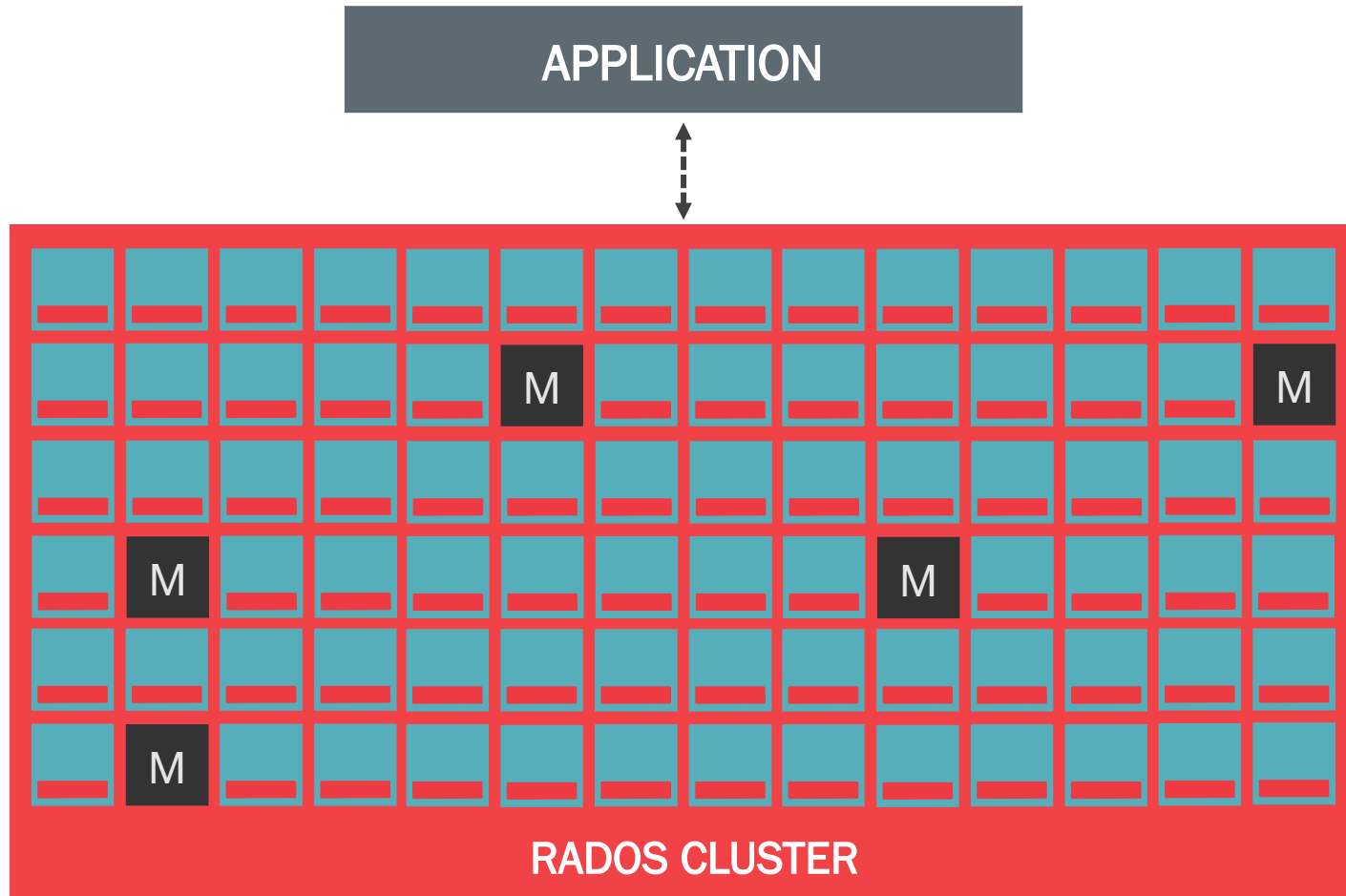
# OVERALL ARCHITECTURE



# OVERALL ARCHITECTURE



# RADOS CLUSTER



# RADOS COMPONENTS



## OSDs

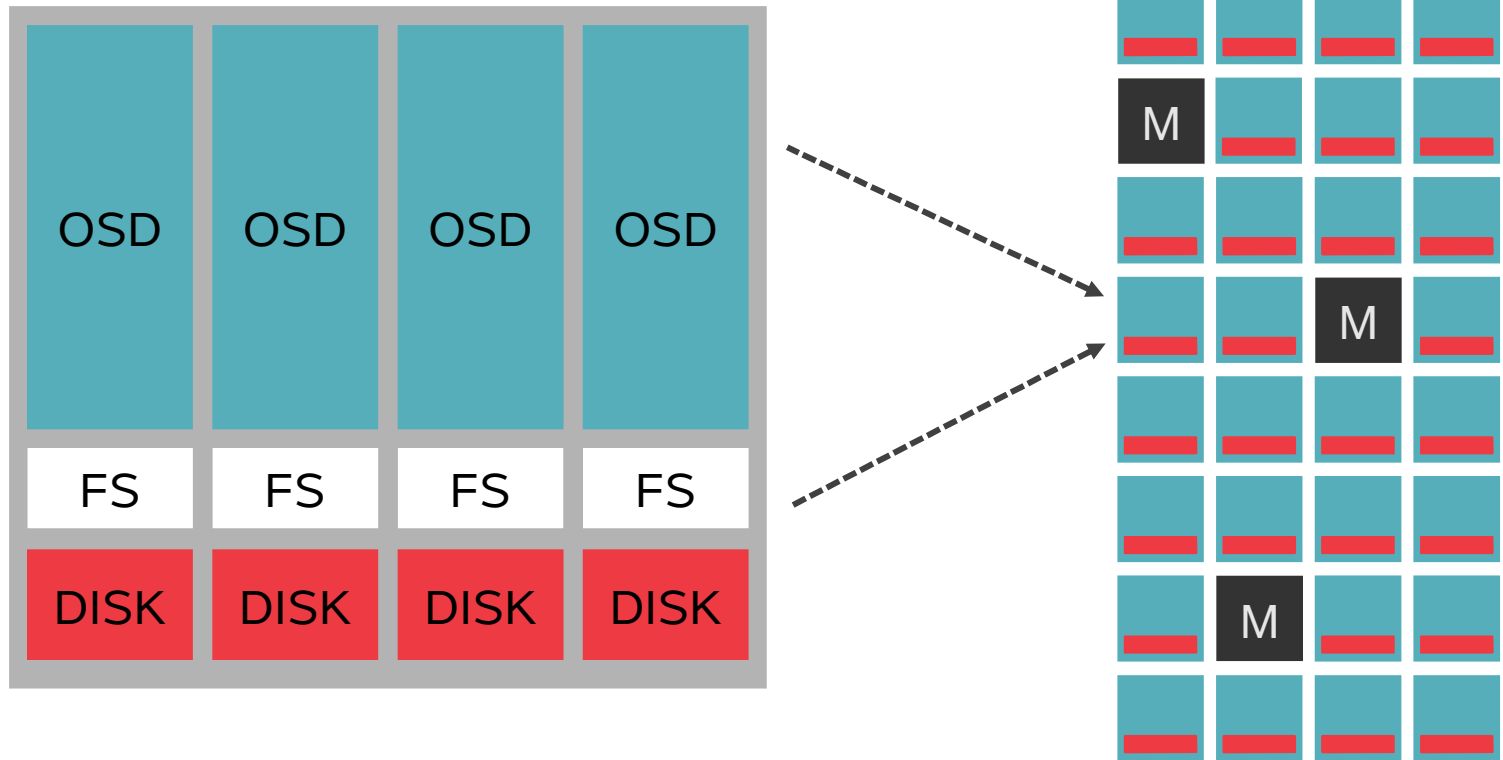
- Smart storage
- Resilient, Distributed, Self-healing, etc
- 100's to thousands



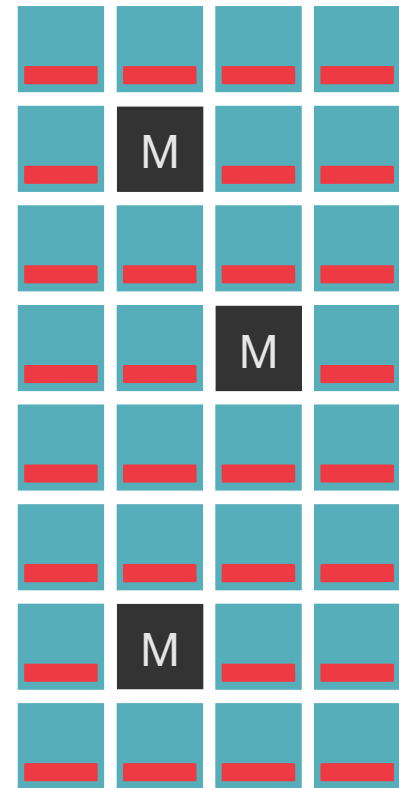
## Monitors

- Keep track of cluster state
- Always consistent, or otherwise...
- 3, 5, 7, ...

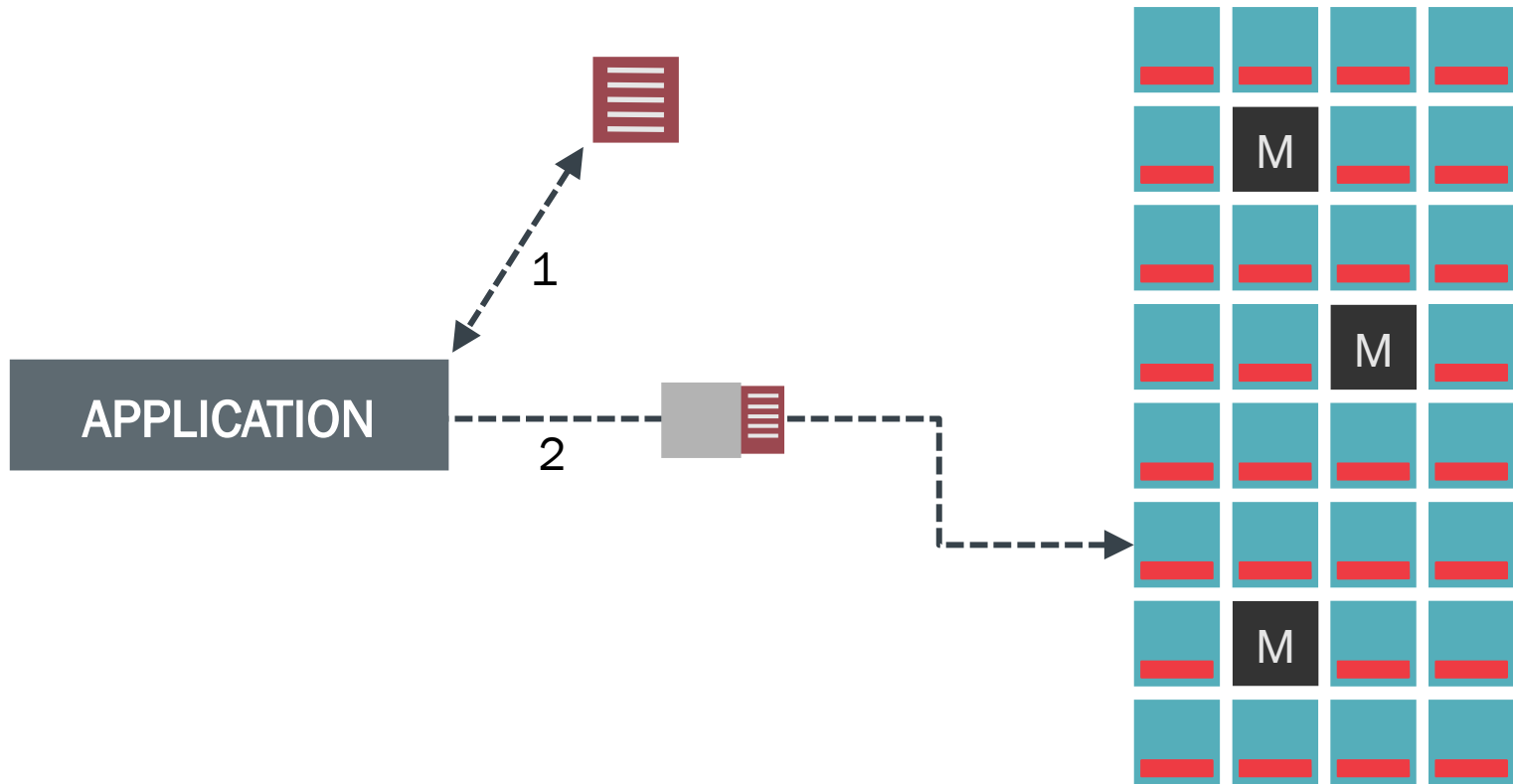
# OBJECT STORAGE DAEMONS



# WHERE IS MY OBJECT?

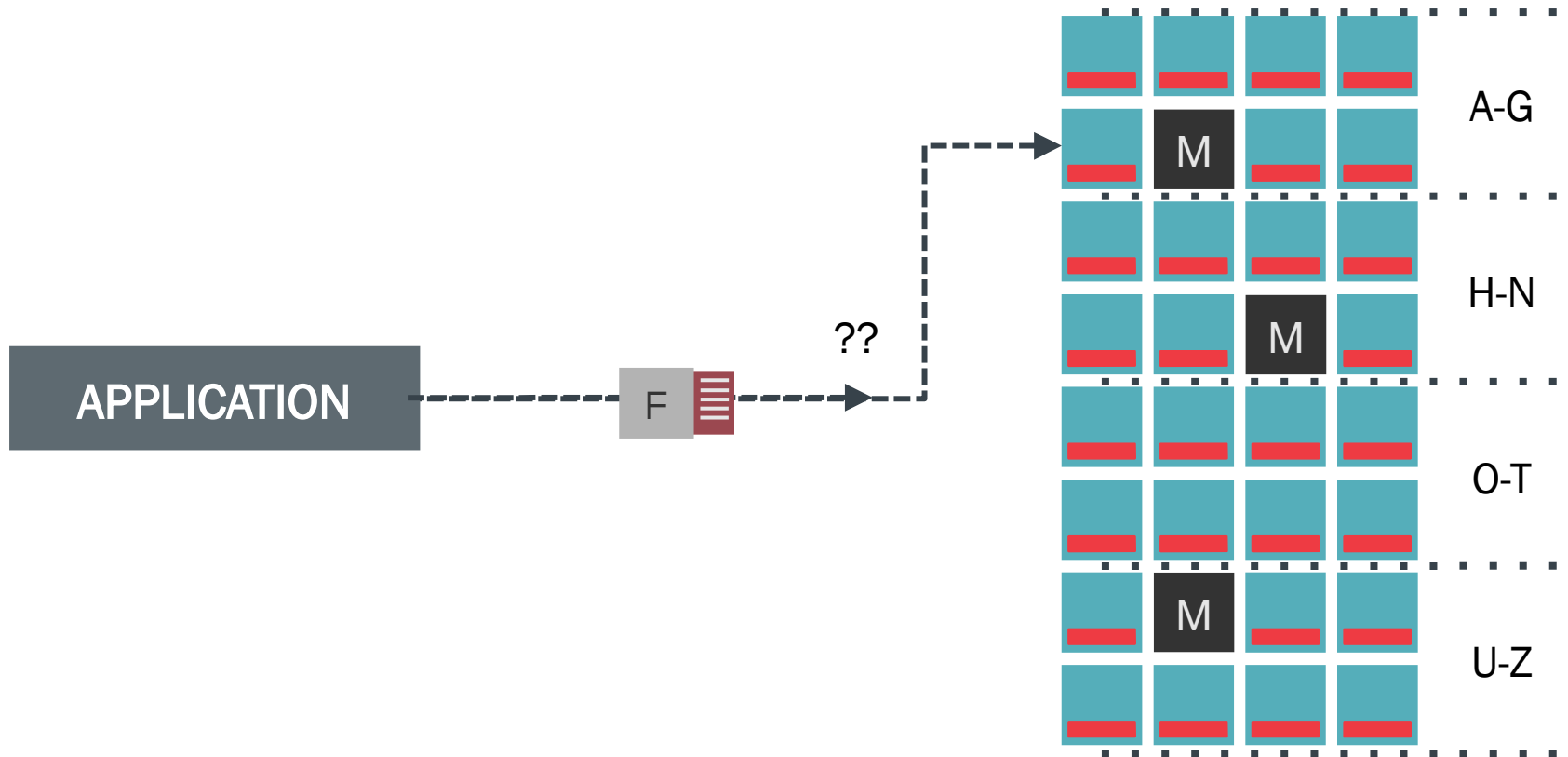


# METADATA SERVER?

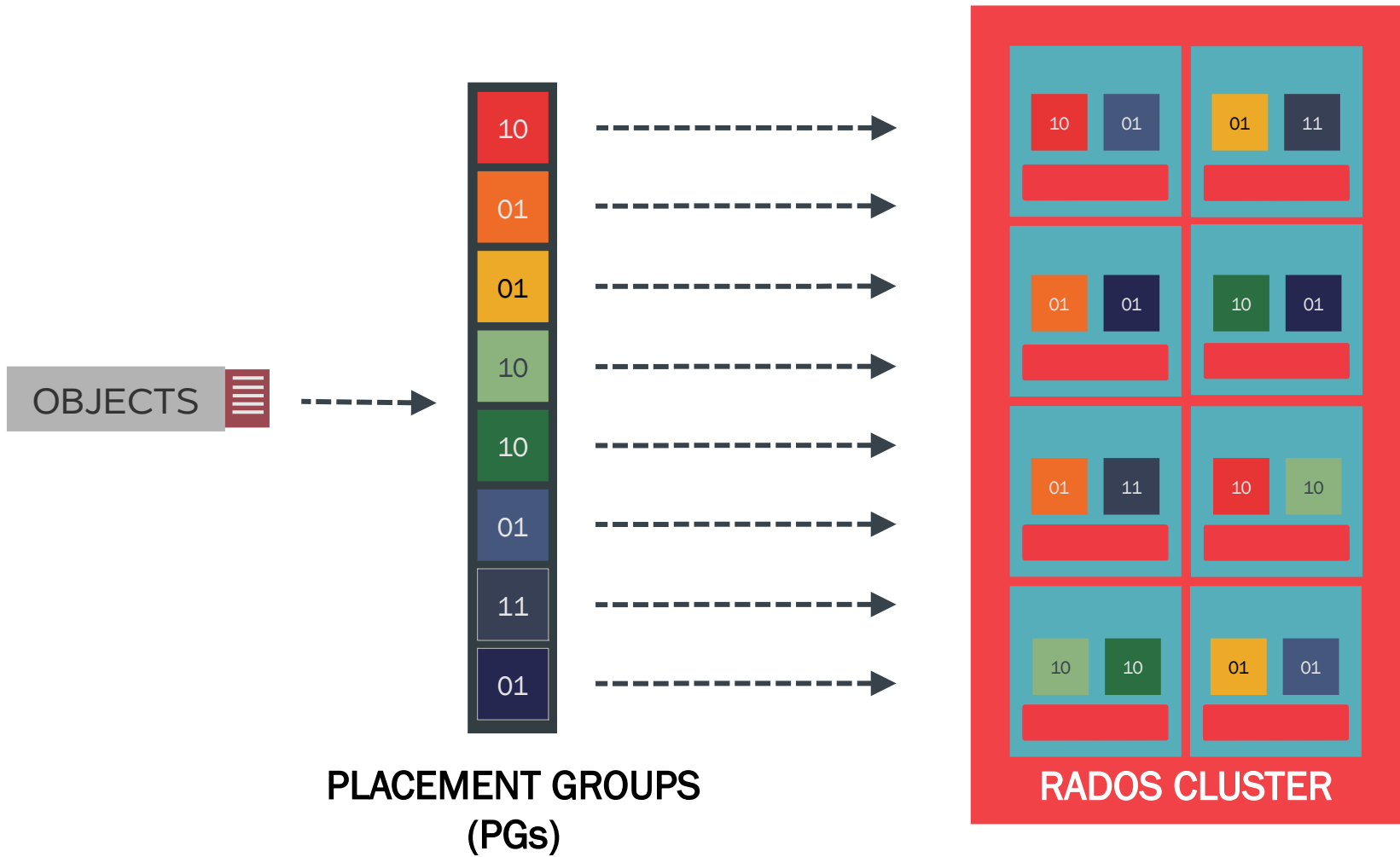




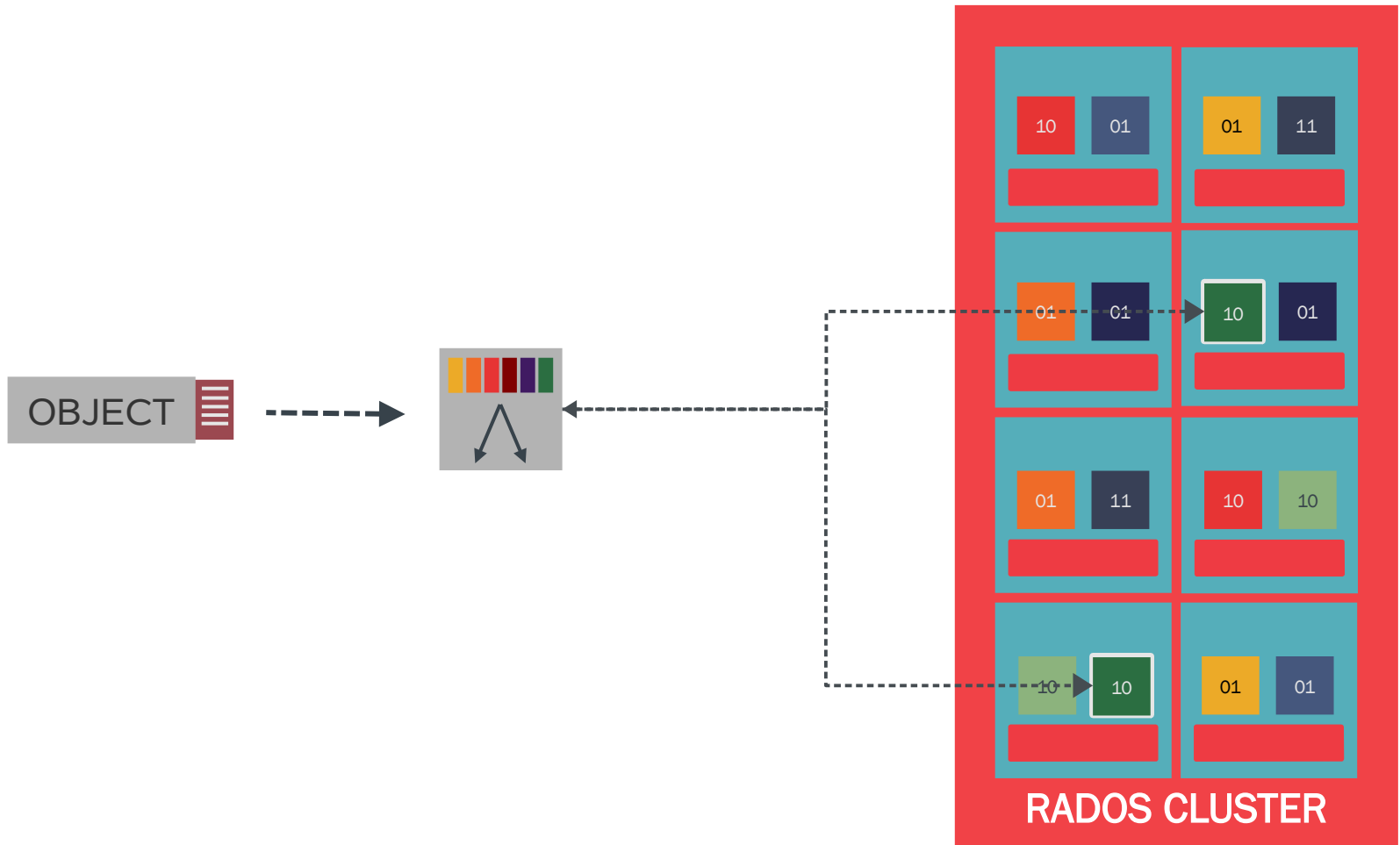
# CALCULATED PLACEMENT?



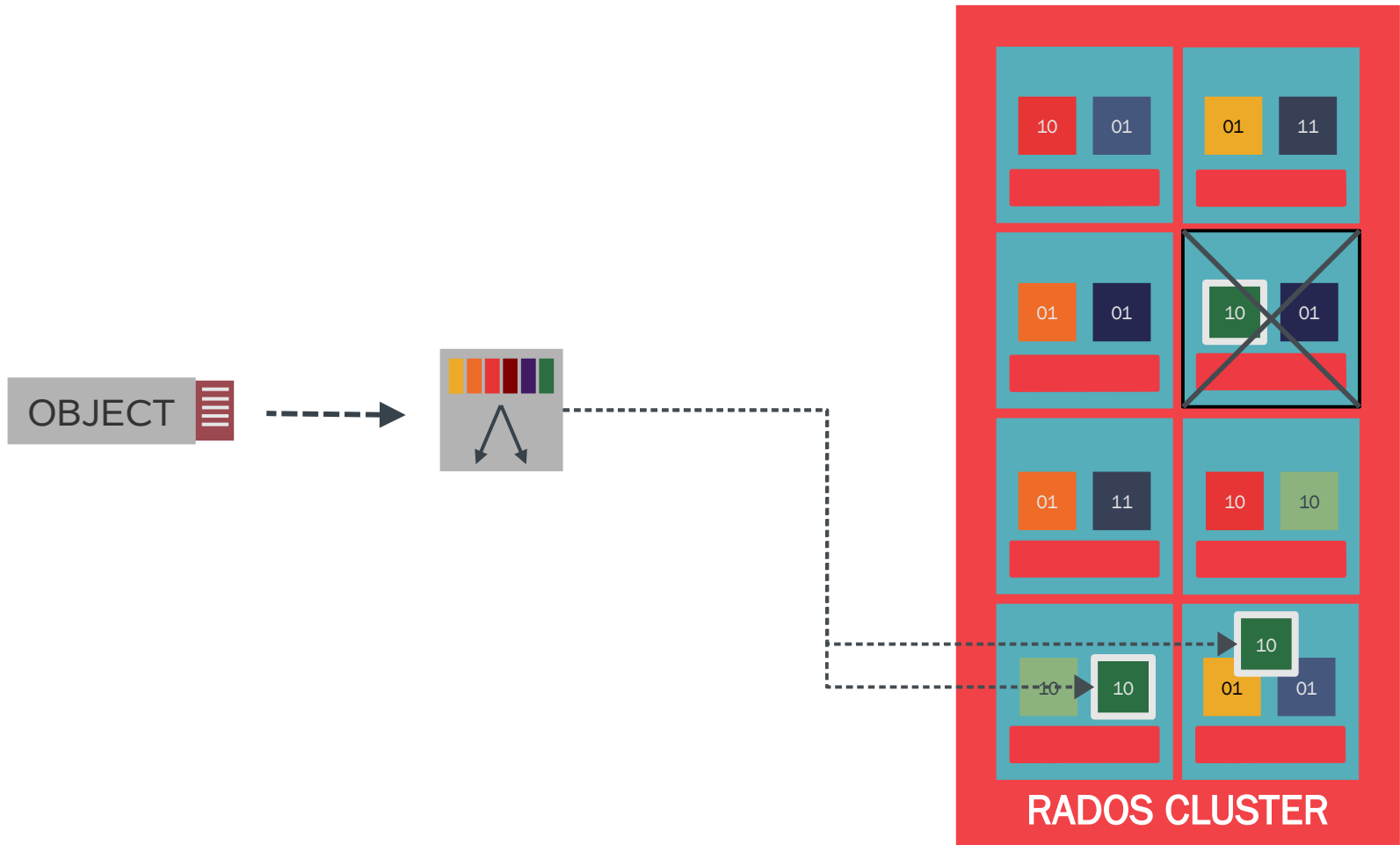
# CRUSH



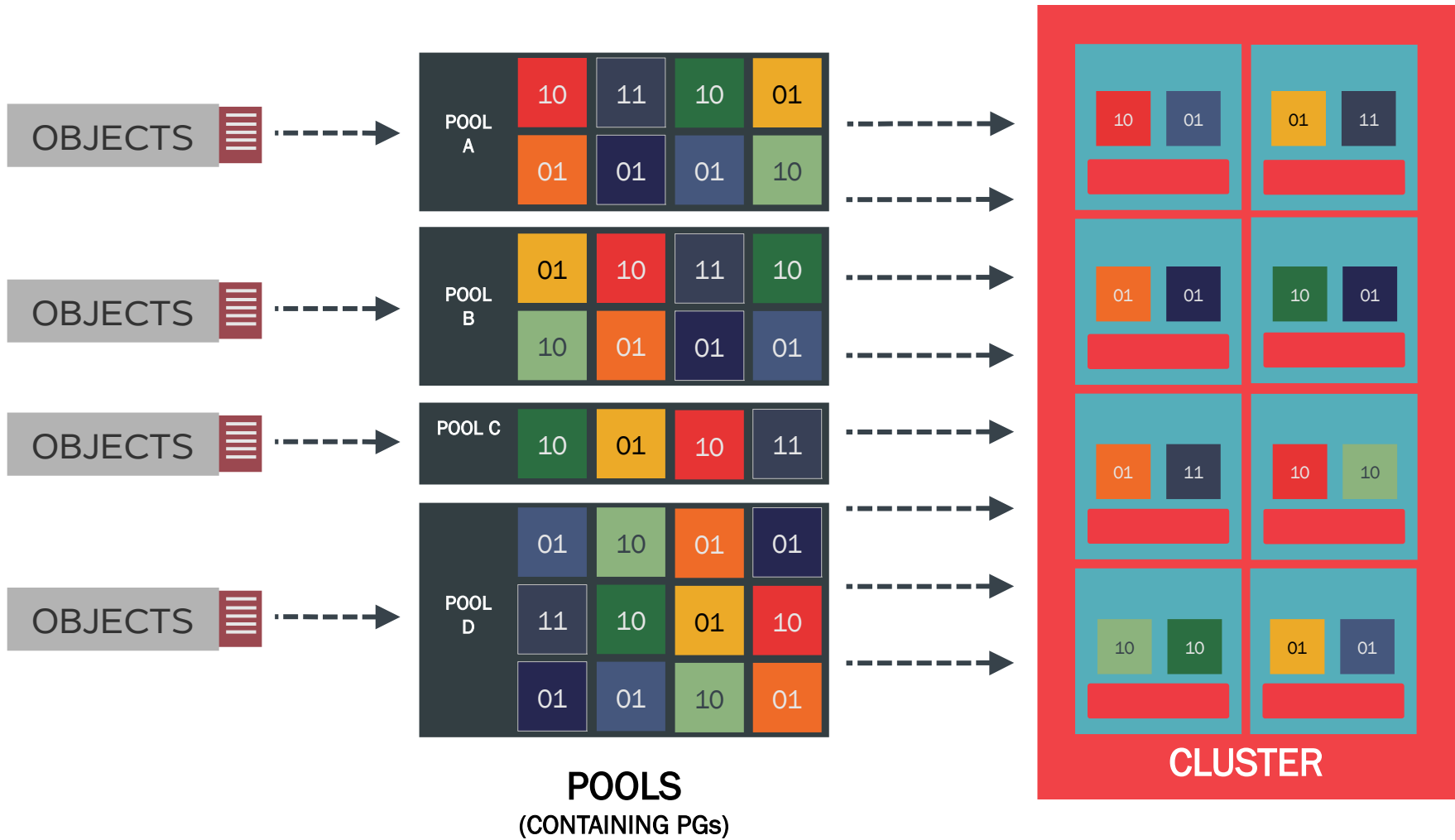
# CRUSH



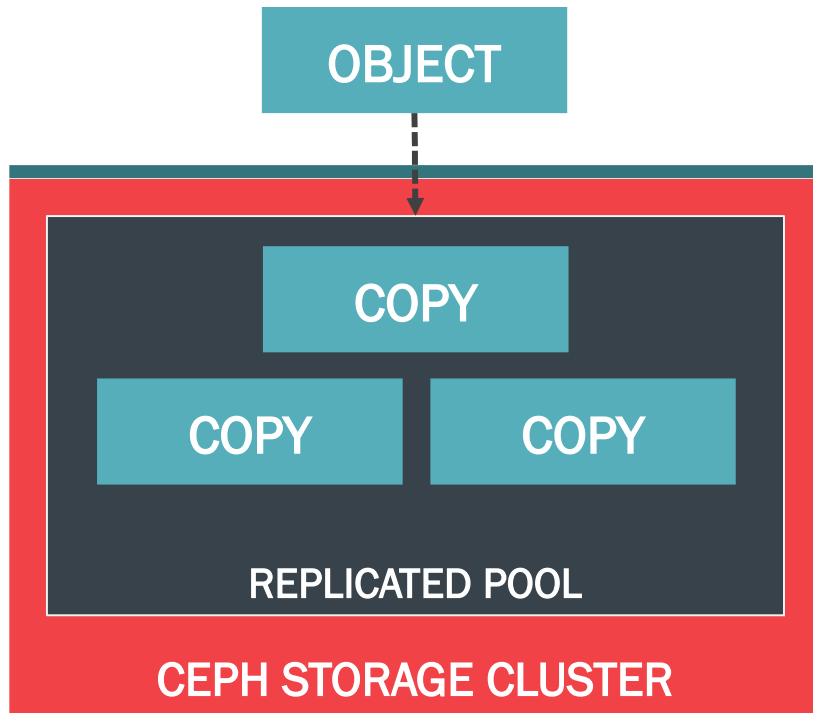
# CRUSH – Failure?



# DATA ORGANIZED INTO POOLS

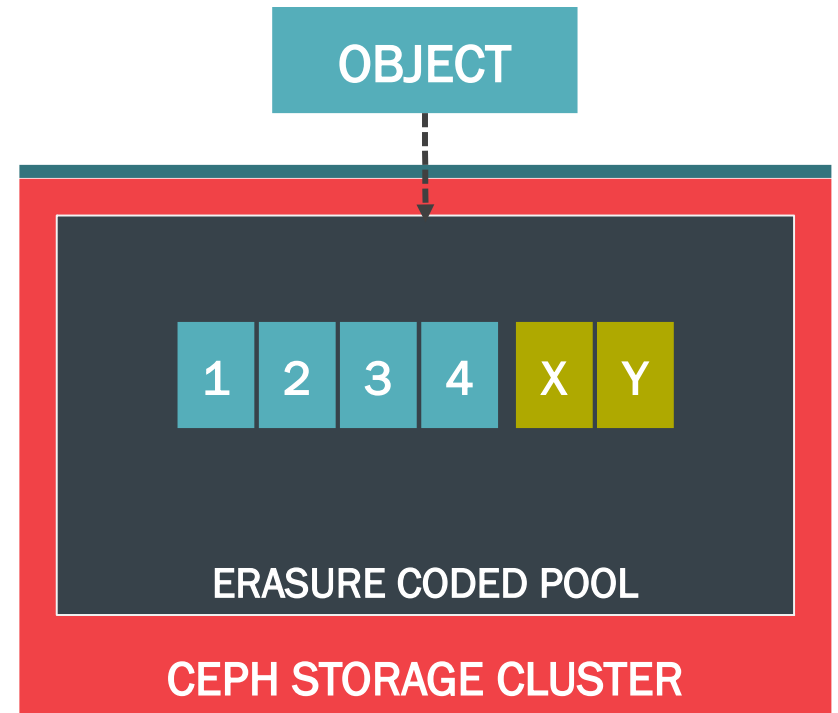


# POOLS



Full copies of stored objects

- Very high durability
- 3x (200% overhead)
- Quicker recovery

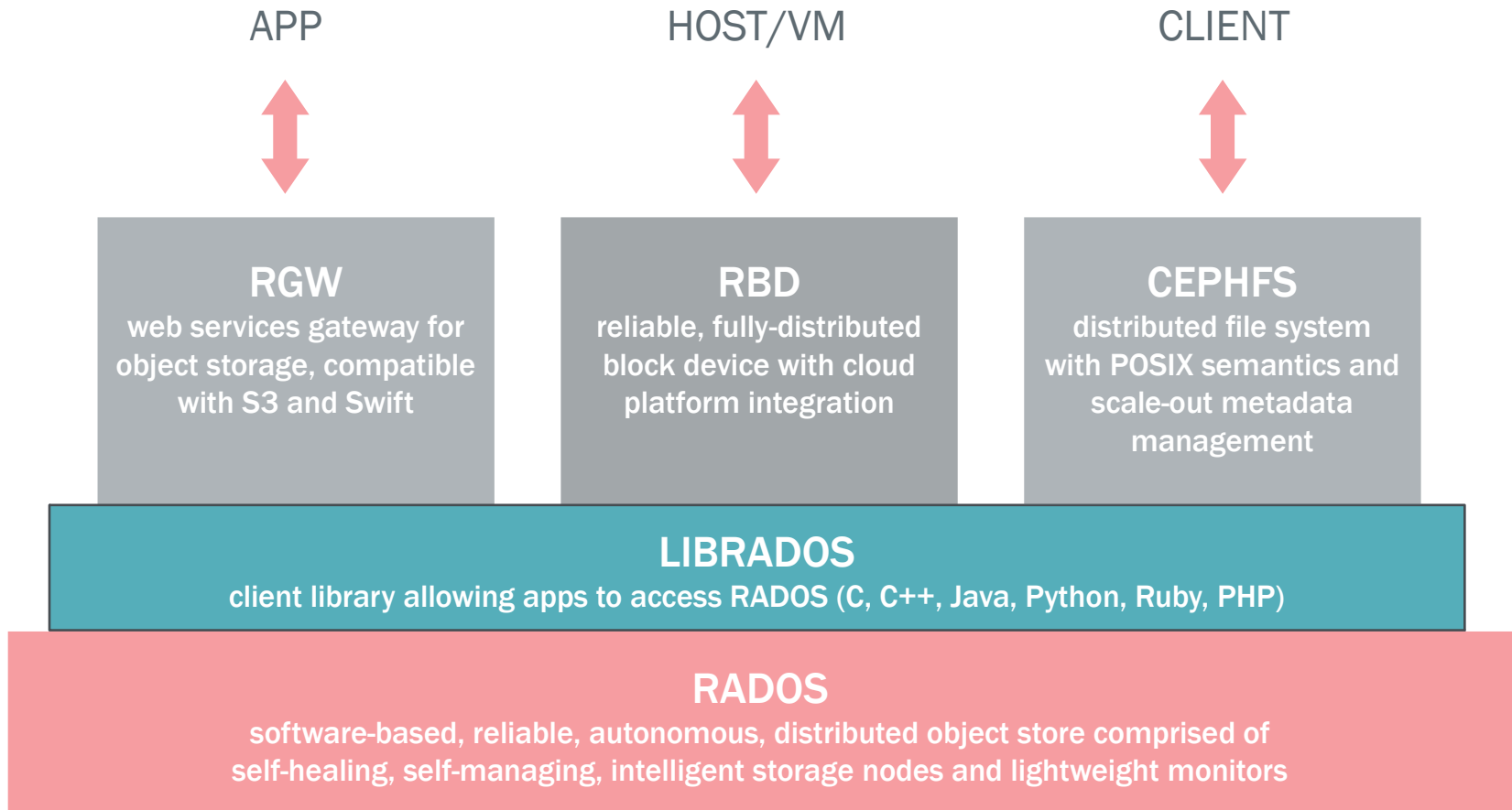


One copy plus parity

- Cost-effective durability
- 1.5x (50% overhead)
- Expensive recovery

LIBRADOS

# OVERALL ARCHITECTURE





# EXAMPLE `HELLO WORLD!`

```
#include <rados/librados.hpp>
```

```
librados::Rados rados;  
rados.init("admin");  
rados.connect();
```

Connect to cluster

```
rados.pool_create("hello_pool");
```

Create pool

```
librados::IoCtx ctx;  
rados.ioctx_create("hello_pool", ctx);
```

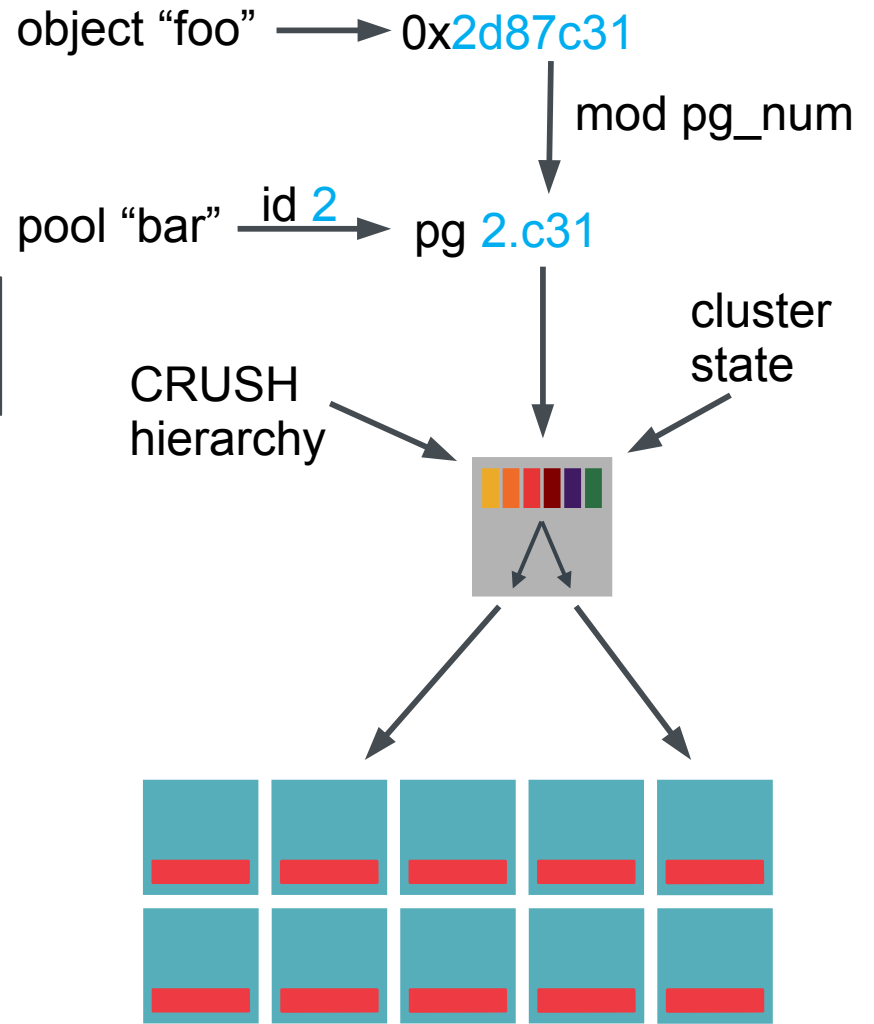
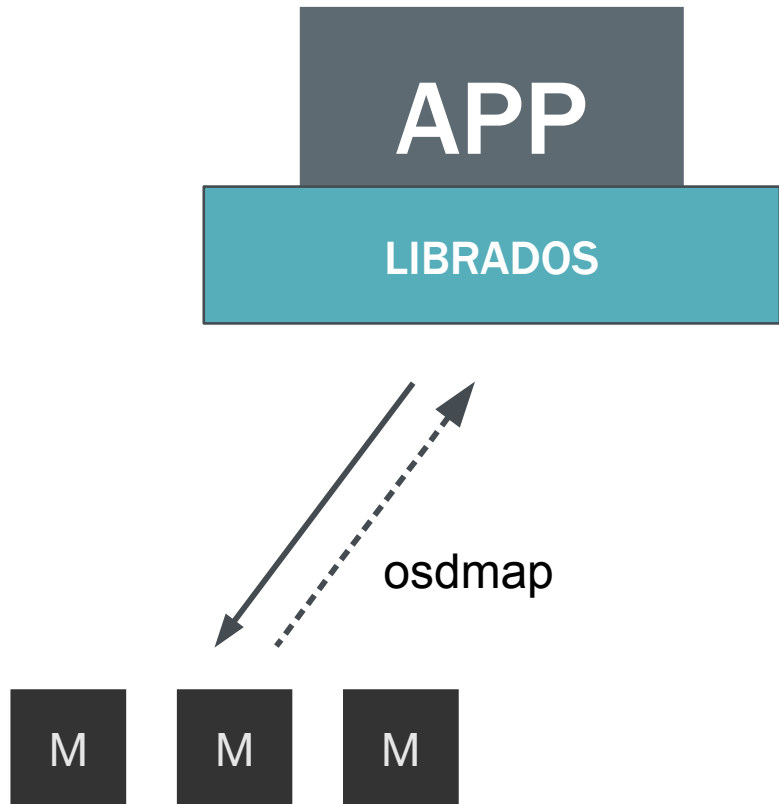
```
bufferlist data;  
data.append("hello world!");  
ctx.write_full("hello_object", data);
```

Atomic (re)write

```
bufferlist attr;  
attr.append("1");  
ctx.setxattr("hello_object", "version", attr);
```

```
rados.shutdown();
```

# LIBRADOS



# COMPOUND OBJECT OPERATIONS

```
#include <rados/librados.hpp>
```

```
librados::Rados rados;  
rados.init("admin");  
rados.connect();
```

Connect to cluster

```
rados.pool_create("hello_pool");
```

Create pool

```
librados::IoCtx ctx;  
rados.ioctx_create("hello_pool", ctx);
```

```
ObjectWriteOperation op;
```

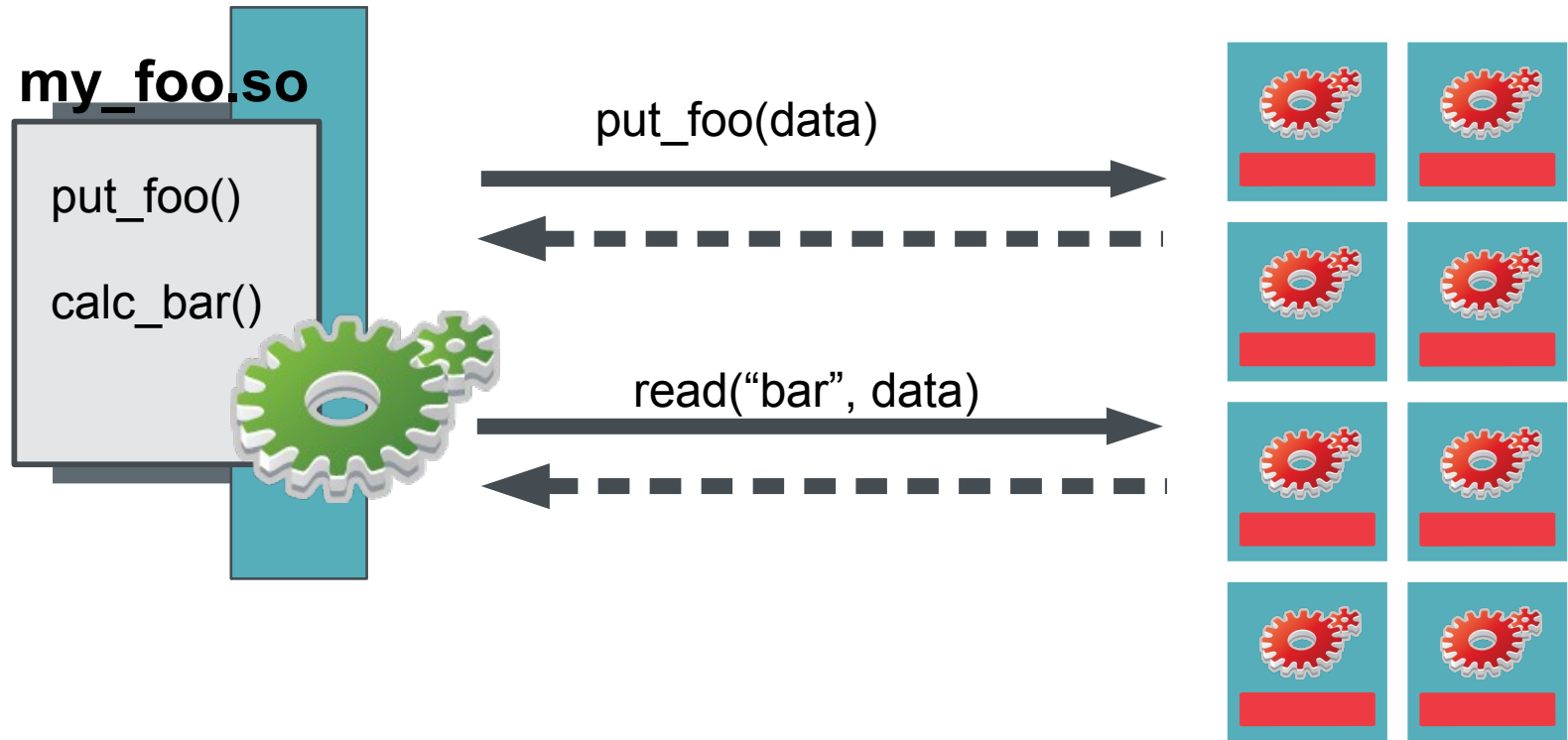
```
bufferlist data;  
data.append("hello world!");  
op.write_full(data);
```

```
bufferlist attr;  
attr.append("1");  
op.setxattr("version", attr);
```

```
ctx.operate("hello_object", &op);
```

```
rados.shutdown();
```

# RADOS OBJECT CLASSES



# EXAMPLE

## Server-side class (hello\_hash\_class)

```
int compute_md5(cls_method_context_t hctx,
                bufferlist *in, bufferlist *out)
{
    size_t size;
    int ret = cls_cxx_stat(hctx, &size, NULL);
    if (ret < 0)
        return ret;

    bufferlist data;
    ret = cls_cxx_read(hctx, 0, size, data);
    if (ret < 0)
        return ret;

    byte digest[AES::BLOCKSIZE];
    MD5().CalculateDigest(digest,
                          (byte*)data.c_str(),
                          data.length());

    out->append(digest, sizeof(digest));

    return 0;
}
```

## librados client

```
bufferlist input, output;
```

```
ioctx.exec("hello_object", "hello_hash_class",
           "compute_md5", input, output);
```

## Compound operations

```
ObjectReadOperation op;
```

```
uint64_t size;
time_t m_time;
op.stat(&size, &m_time, NULL);
```

```
bufferlist in, out;
op.exec("hello_hash_class", "compute_md5",
        in, &out);
```

```
int r = op.operate("hello_object", &op);
```

# REAL APPLICATIONS

- Cooperative Locking
- Simple Object Reference Counting
- Image manipulation
- RADOS Block Device (RBD) & Gateway (RGW)

sources in `src/cls/*`

# DYNAMIC OBJECT CLASSES IN LUA

- Noah Watkins (UCSC / Red Hat)
  - <http://ceph.com/rados/dynamic-object-interfaces-with-lua/>

```
local script = [[
function say_hello(input, output)
  output:append("Hello, ")
  if #input == 0 then
    output:append("world")
  else
    output:append(input:str())
  end
  output:append("!")
end
cls.register(say_hello)
]]
```

```
local ret, outdata = clslua.exec(ioctx,
                                "oid", script,
                                "say_hello", "")
print(outdata)
```

```
local ret, outdata = clslua.exec(ioctx,
                                "oid", script,
                                "say_hello", "John")
print(outdata)
```

ceph-devel@vger.kernel.org  
ceph-users@lists.ceph.com  
#ceph / #ceph-devel @ OFTC  
[www.ceph.com](http://www.ceph.com)

Thank you.







## **Unpublished Work of SUSE LLC. All Rights Reserved.**

This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

## **General Disclaimer**

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

