Using Lua in the Ceph distributed storage system

Noah Watkins
noahwatkins@gmail.com
@noahdesu

Michael Sevilla
mikesevilla3@gmail.com

Lua Workshop, October 2017, San Francisco

yet another Lua embedding

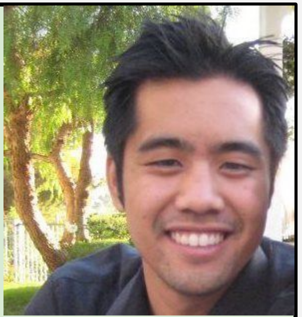
- Ceph is massively **distributed**, highly dynamic **storage system**
- Lua is embedded throughout Ceph
    - Defining policies
    - Domain-specific I/O interfaces
    - Remote data processing

Who we are

Noah Watkins

- PhD candidate
- UC Santa Cruz
- Experience
  - LANL, HGST
  - IBM, Red Hat
- Contact
  - noahwatkins@gmail.com
  - @noahdesu
- Project Lead
  - Distributed shared-log on SDS
    - https://github.com/noahdesu/zlog
  - **Lua integration in Ceph object store**

Michael Sevilla



- PhD candidate
- UC Santa Cruz
- Experience
  - HP Enterprise
  - LANL
- Contact
  - mikesevilla3@gmail.com
- Project Lead
  - Mantle: Programmable load balancer in Ceph
  - **Lua integration in Ceph file system**

Where are we from

UC SANTA CRUZ

SRL UCSC Systems Research Lab

- Funding from: CROSS | CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

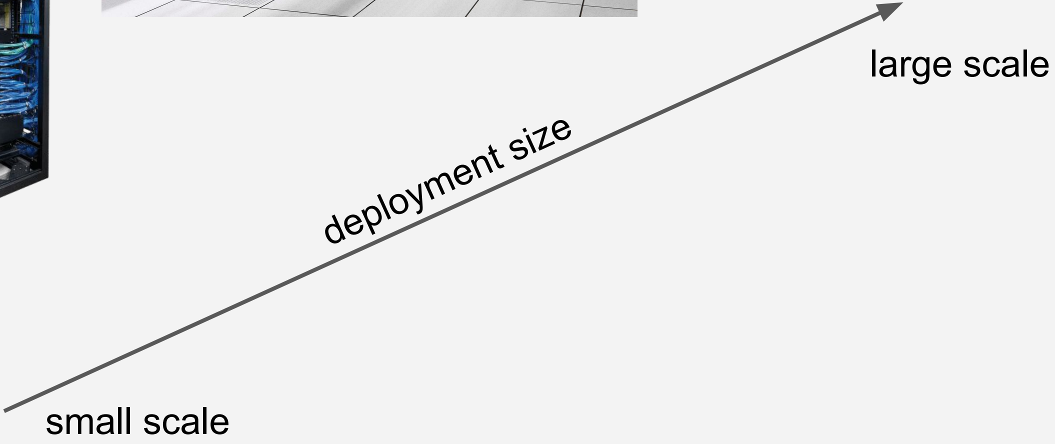
Advisory Board

Doug Cutting	Karen Sandler	Sage Weil	Charlie McDowell	Carlos Maltzahn
Lucene, Nutch, Avro, Hadoop	GNOME Foundation, SW Freedom Law	WebRing, Dreamhost, Inktank (Ceph)	Associate Dean (Undergrad) @ UCSC	Director, Adjunct Professor

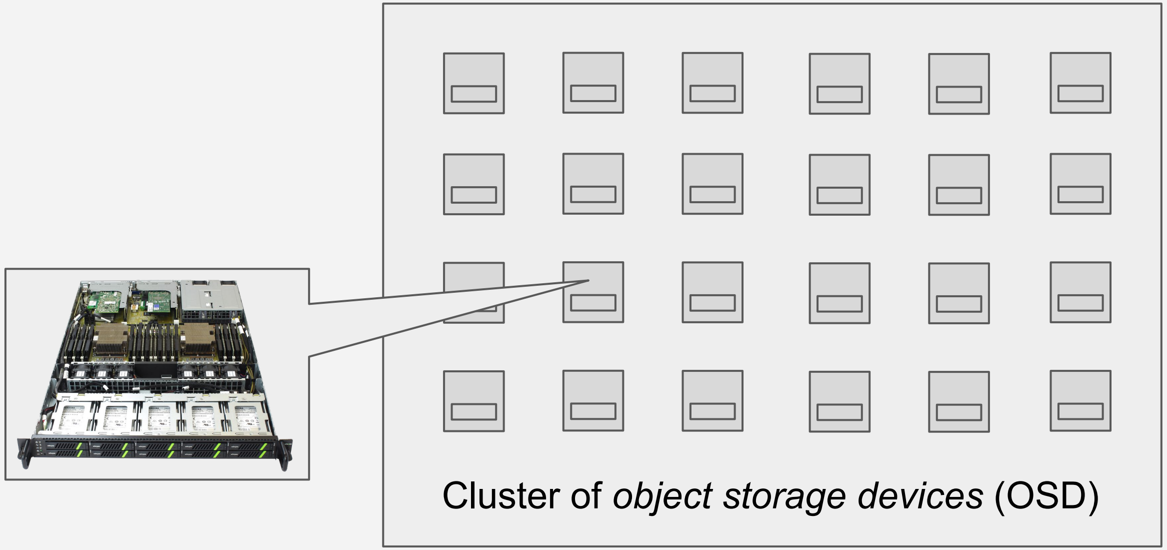
- Bridges gap between research & open source
  - Incubator projects, Research projects, College courses
- Proposals considered every 6 months

Ceph is a distributed storage system

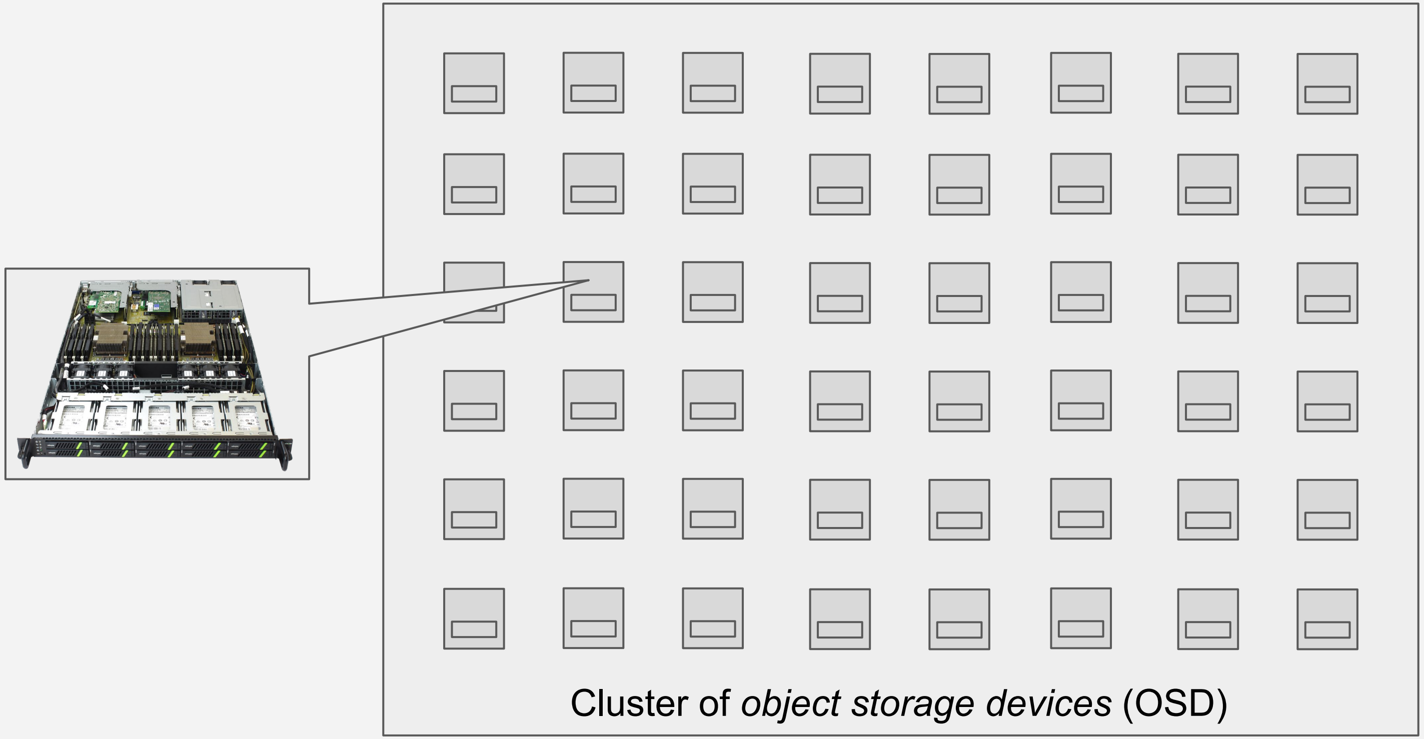
- Horizontal scalability
- No single point of failure
- Software controlled
- Hardware agnostic
  - COTS



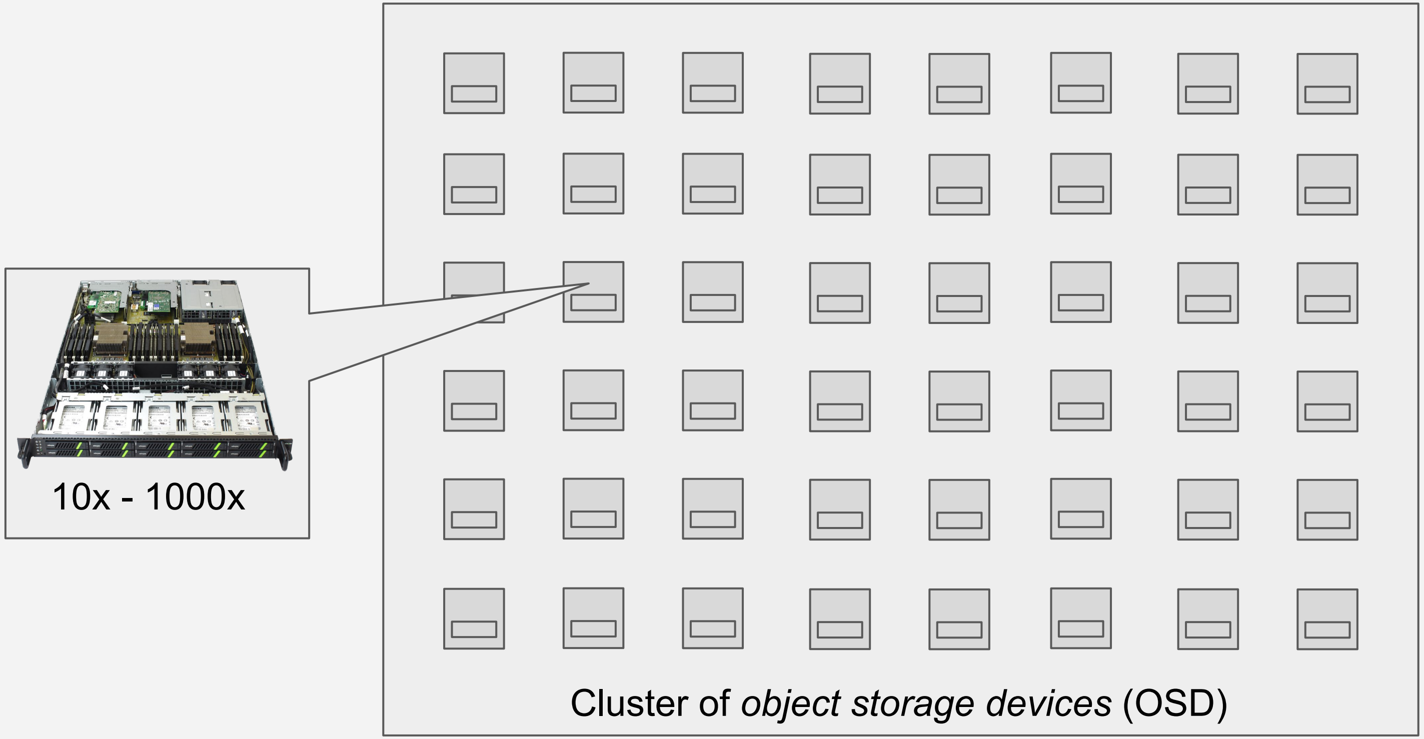
Ceph is a distributed storage system



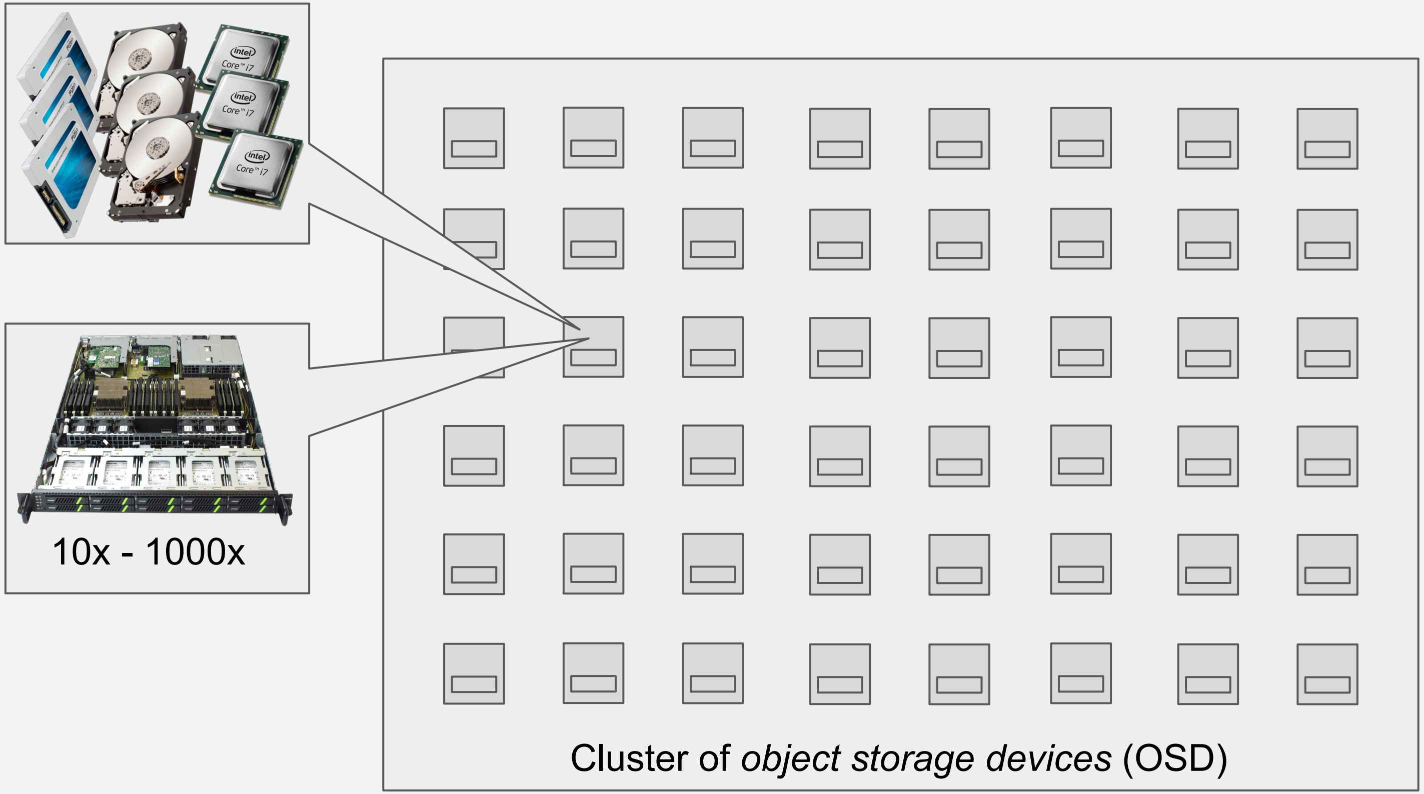
Ceph scales horizontally. Just add more nodes!



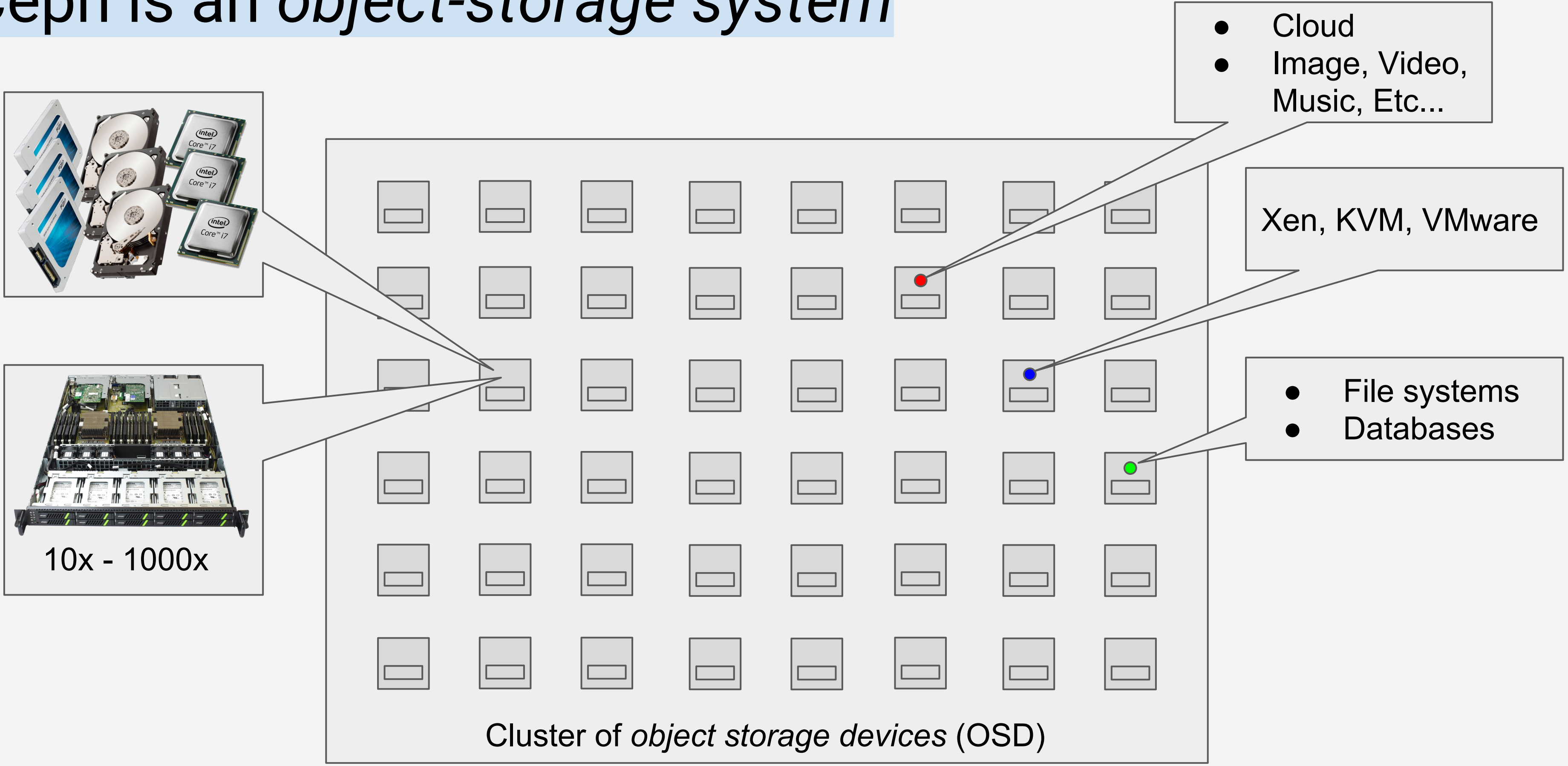
Ceph scales horizontally. Just add more nodes!



Ceph works with all sorts of hardware configurations



Ceph is an *object-storage system*



- Cloud
- Image, Video, Music, Etc...

Xen, KVM, VMware

- File systems
- Databases

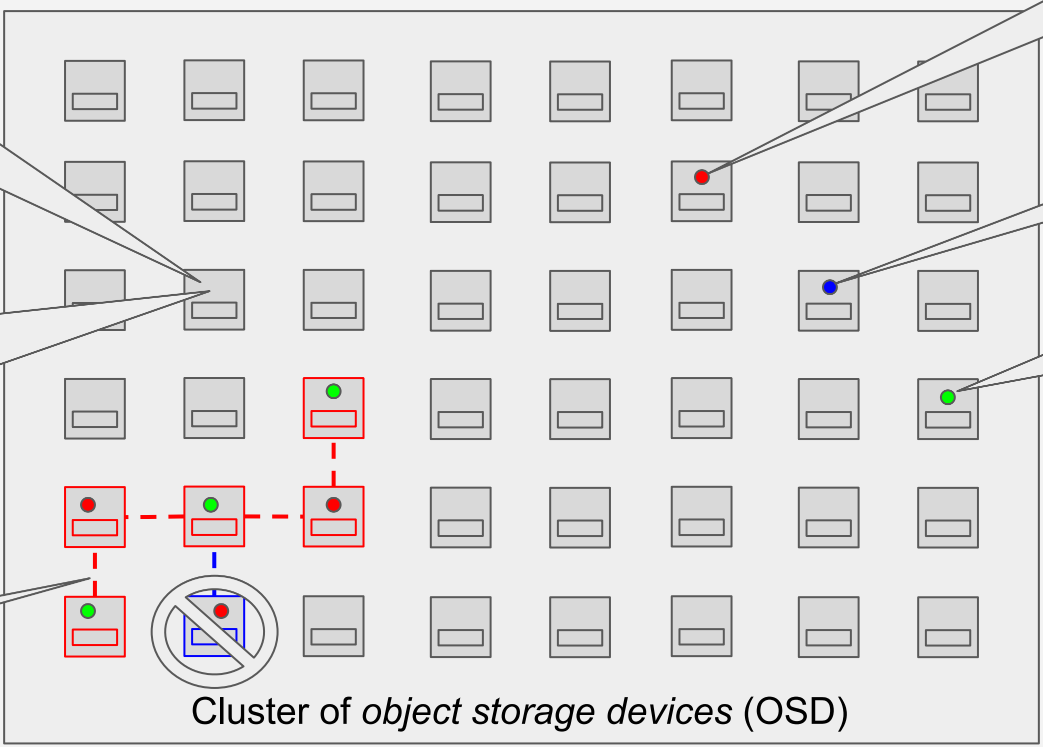
10x - 1000x

Cluster of *object storage devices* (OSD)

Ceph is an *object-storage system*

10x - 1000x

- Peer-to-peer
- Self-healing



Cluster of *object storage devices* (OSD)

- Cloud
- Image, Video, Music, Etc...

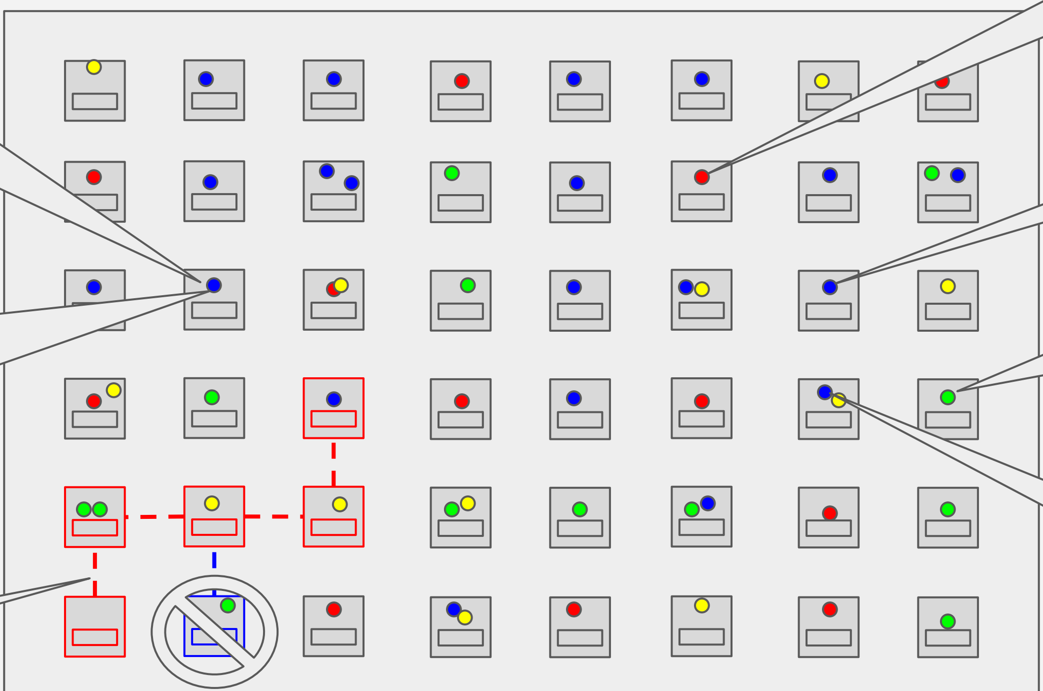
Xen, KVM, VMware

- File systems
- Databases

Ceph is an *object-storage system*

10x - 1000x

- Peer-to-peer
- Self-healing



Cluster of *object storage devices* (OSDs)

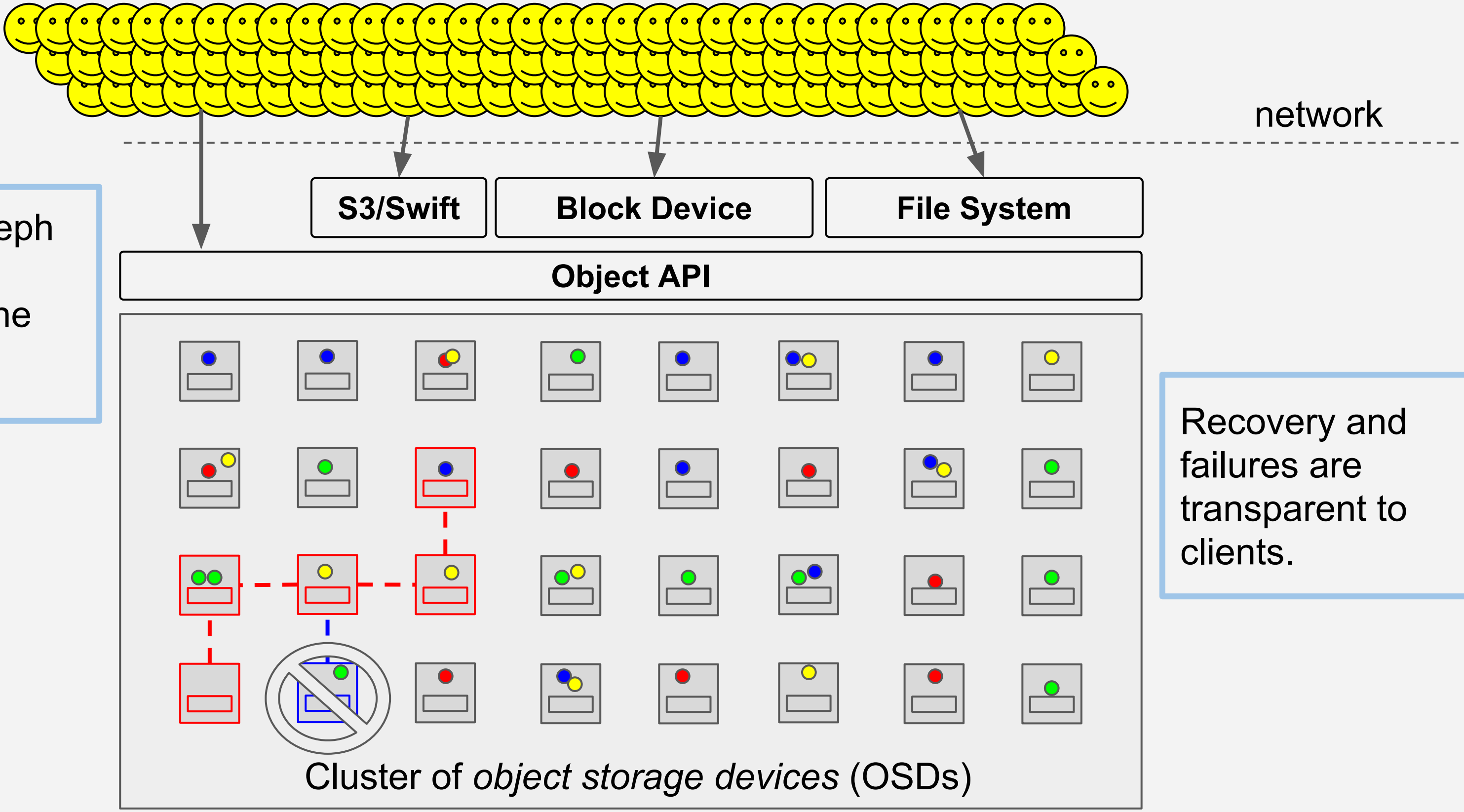
- Cloud
- Image, Video, Music, Etc...

Xen, KVM, VMware

- File systems
- Databases

- Billions of objects
- Object API

Ceph is an *object-storage system*

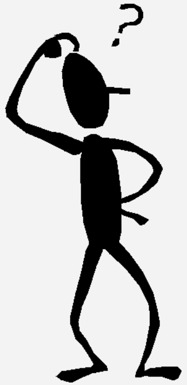


Clients access Ceph via **services**, or directly through the **object API**.

Recovery and failures are transparent to clients.

Ceph is widely deployed and developed

- 100% open-source
  - Hundreds of contributors
  - Industry, academics, government
- https://github.com/ceph/ceph
- Recently tested at CERN
  - 10,800 OSDs
  - 65 petabytes
- Embedded devices
  - 504 WDLabs converged Ethernet drives (4 PB)
  - ARM CPU, memory, network, storage
  - OSD and disk plug directly into network



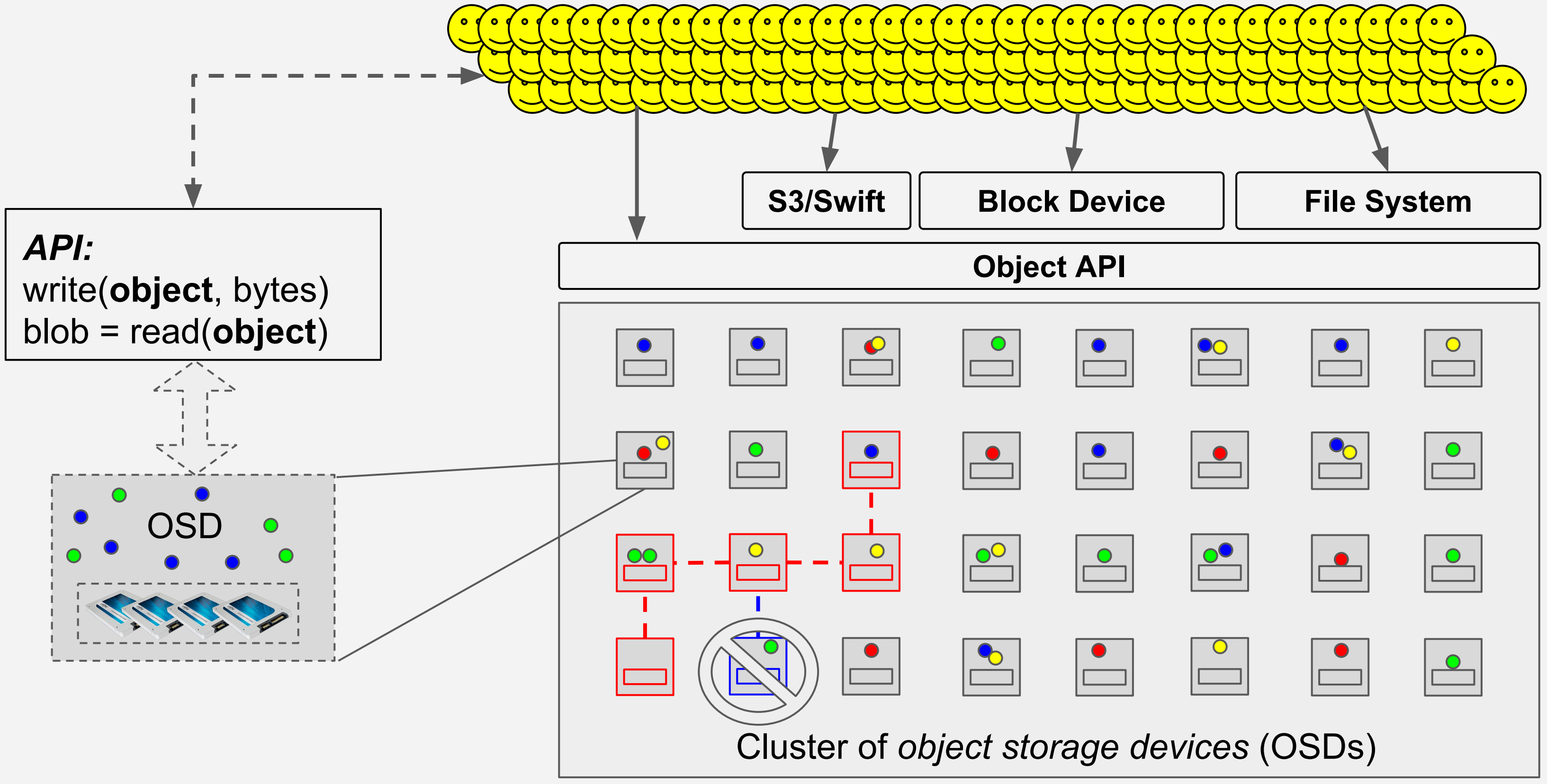
questions?

Recap: Ceph is a distributed storage system, that:

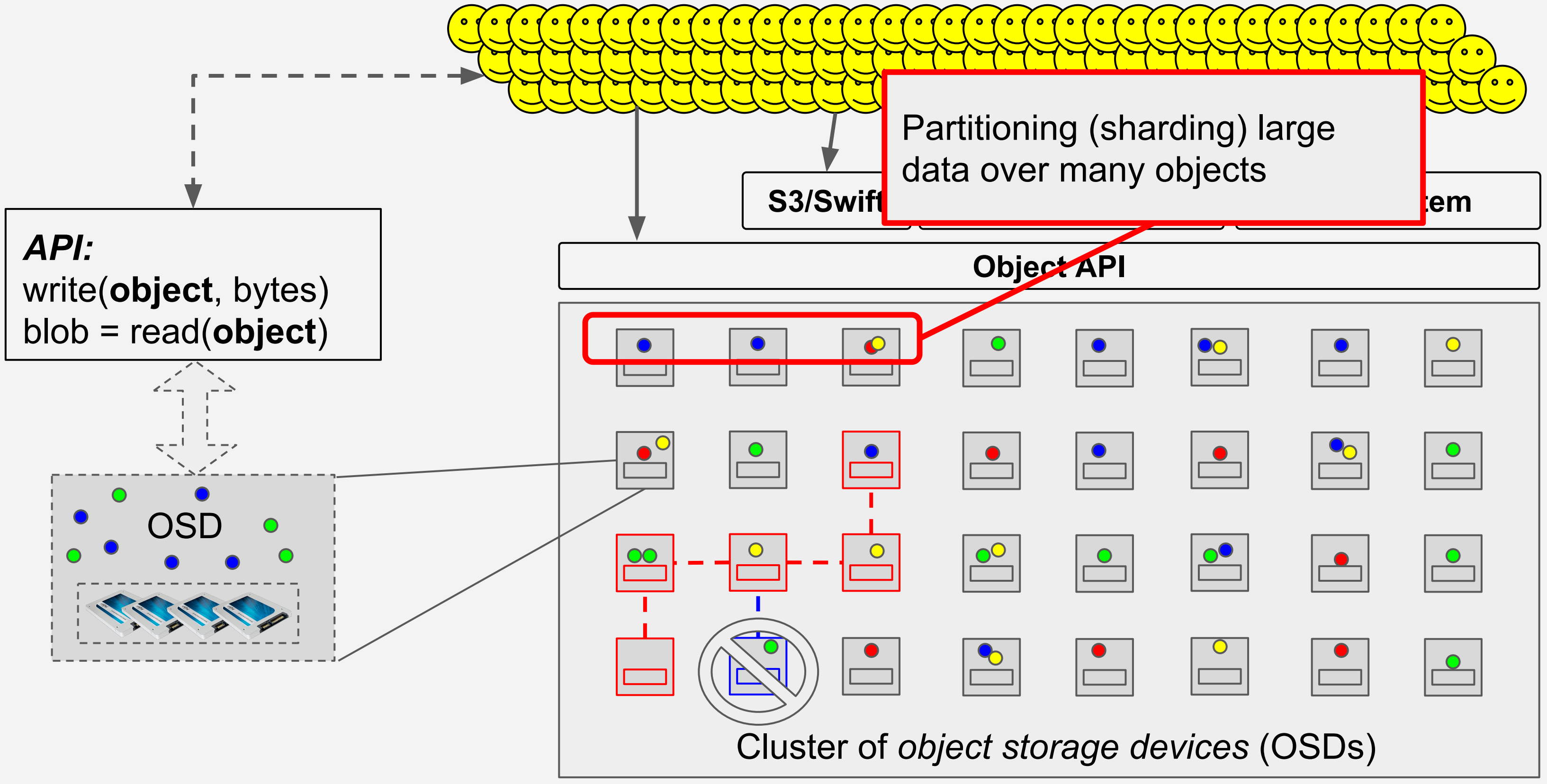
- scales horizontally. Just add more nodes!
- works with all sorts of hardware configurations
- is an object-storage system
- is widely deployed and developed

Lua in your objects

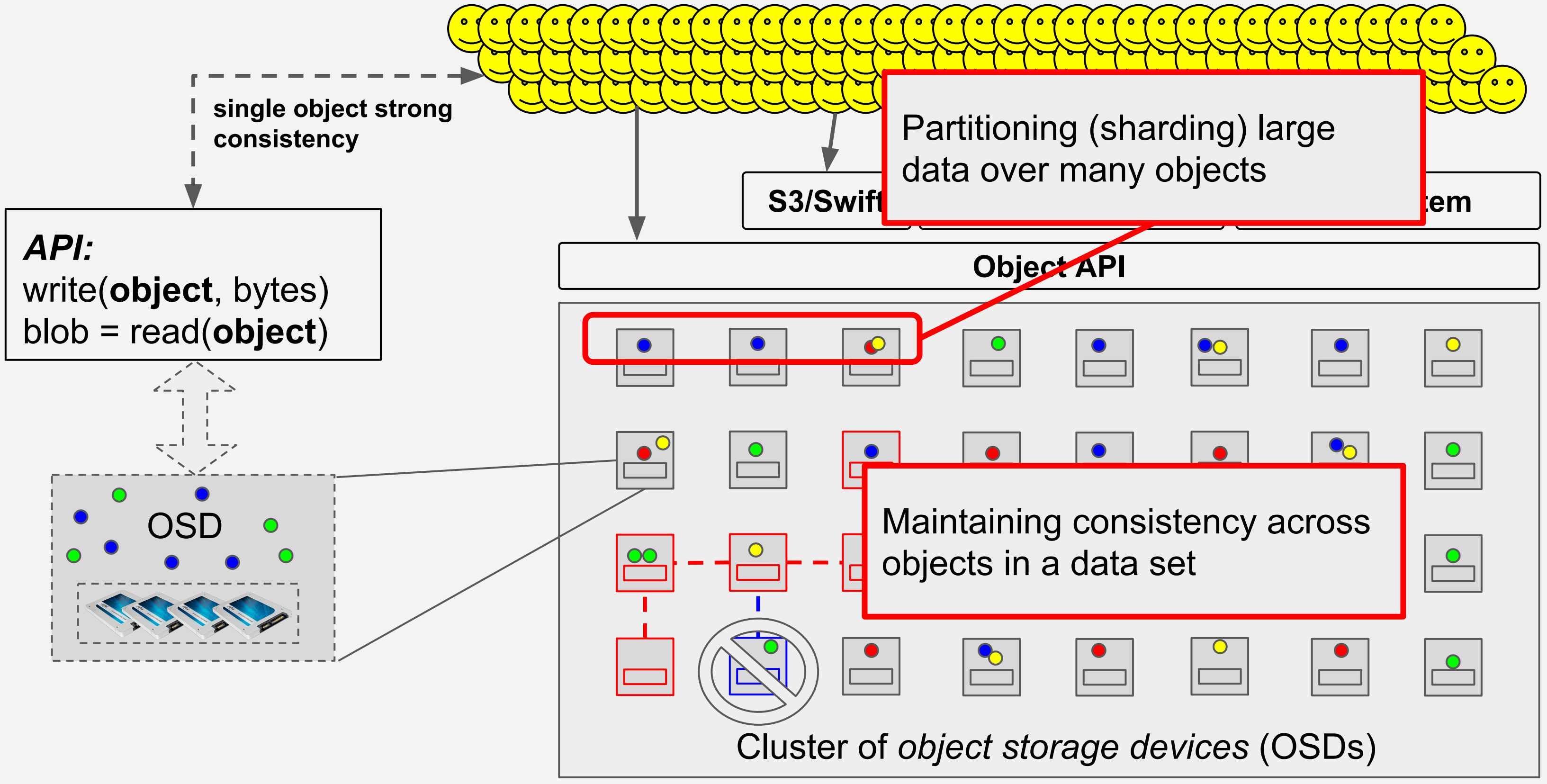
Targeting object APIs can be challenging



Targeting object APIs can be challenging


Partitioning (sharding) large data over many objects
S3/Swift
tem
Object API
API:
write(object, bytes)
blob = read(object)
OSD
Cluster of object storage devices (OSDs)

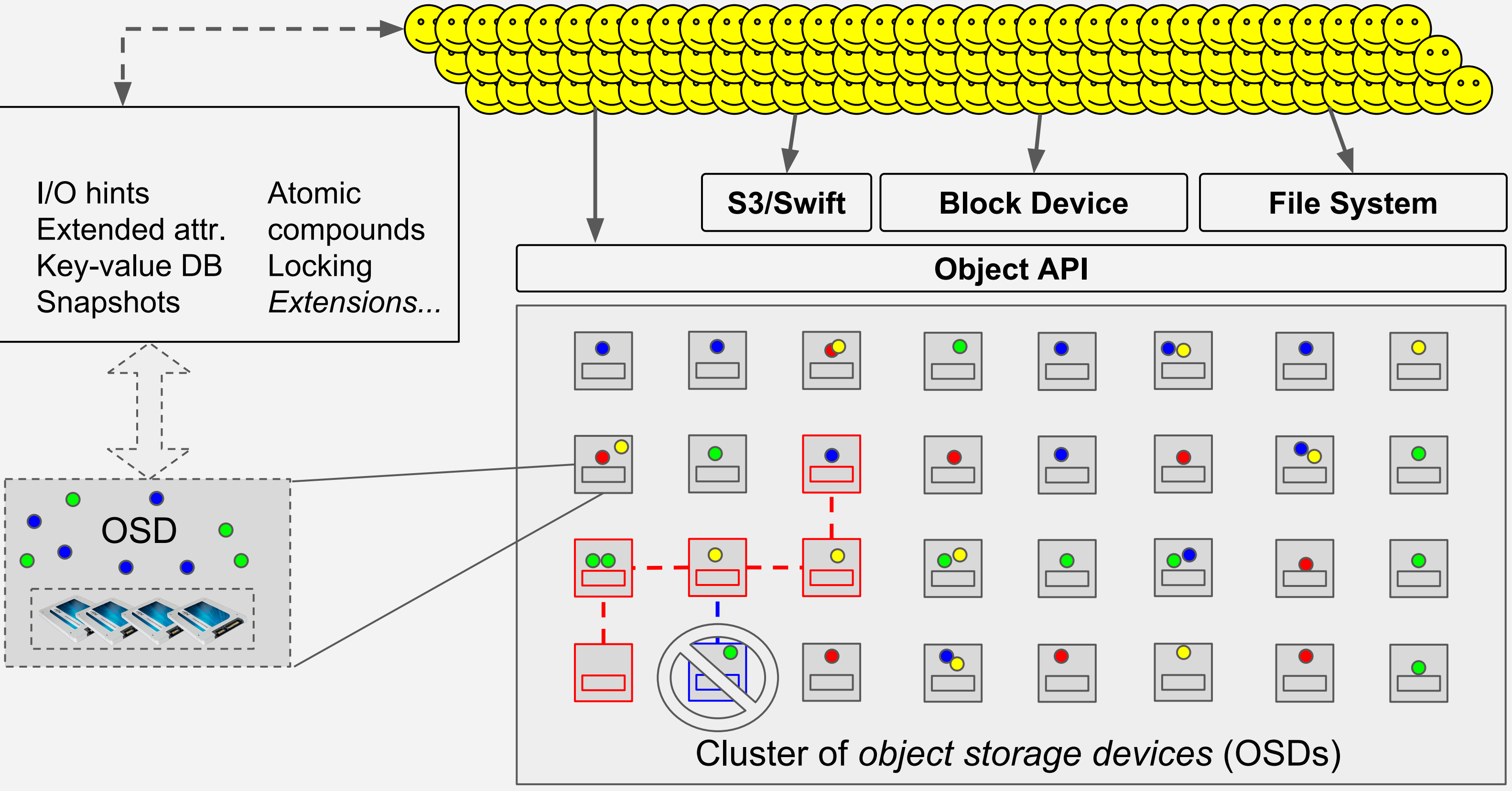
Targeting object APIs can be challenging


single object strong consistency
API:
write(**object**, bytes)
blob = read(**object**)
OSD
S3/Swift
tem
Object API
Partitioning (sharding) large data over many objects
Maintaining consistency across objects in a data set
Cluster of *object storage devices* (OSDs)

A rich object API makes things better

API:

Enumeration	I/O hints	Atomic compounds
Watch/Notify	Extended attr.	Locking
Sync, Async	Key-value DB	*Extensions...*
Caching	Snapshots	



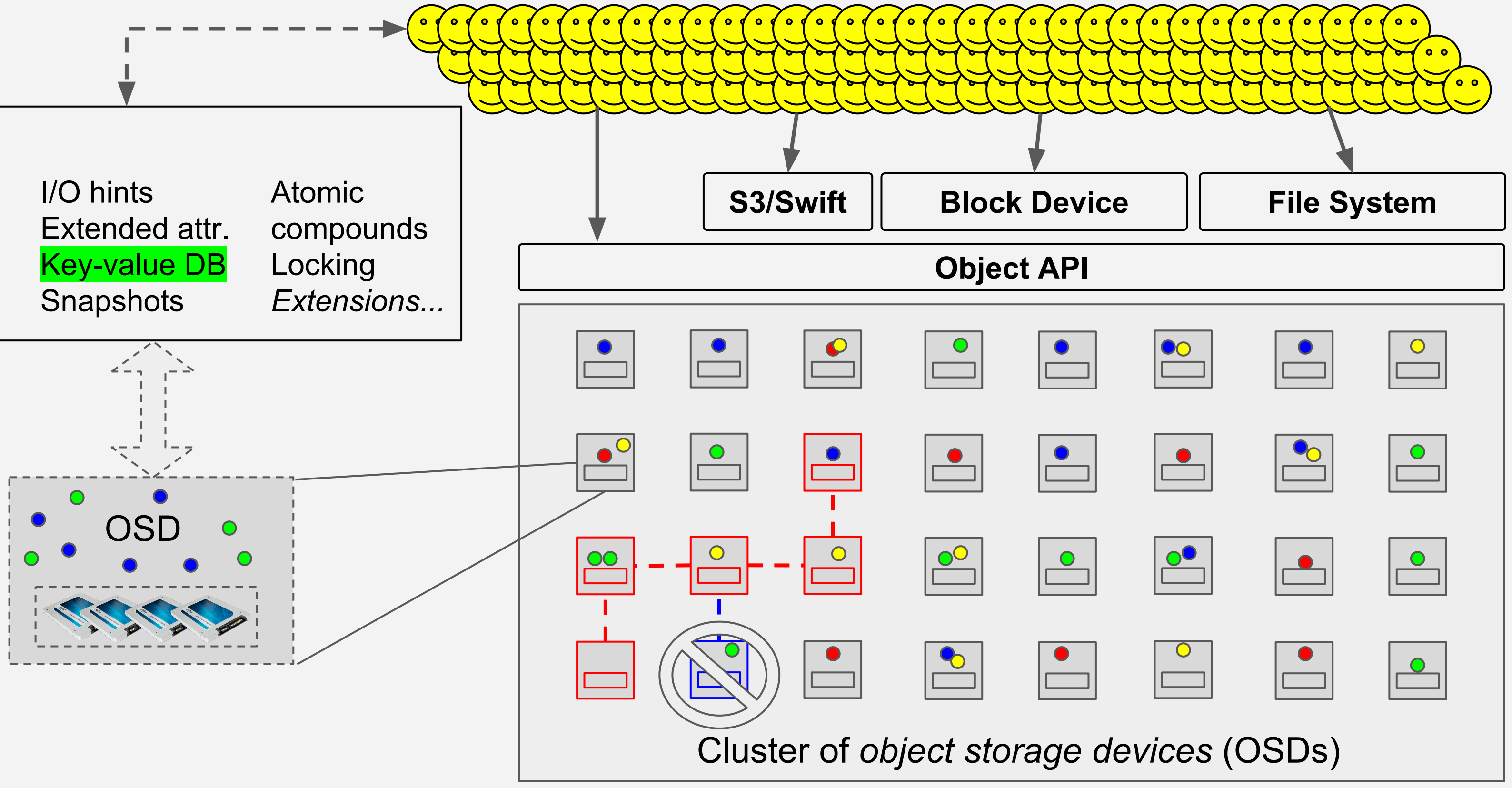
A rich object API makes things better

API:

Enumeration
Watch/Notify
Sync, Async
Caching

I/O hints
Extended attr.
Key-value DB
Snapshots

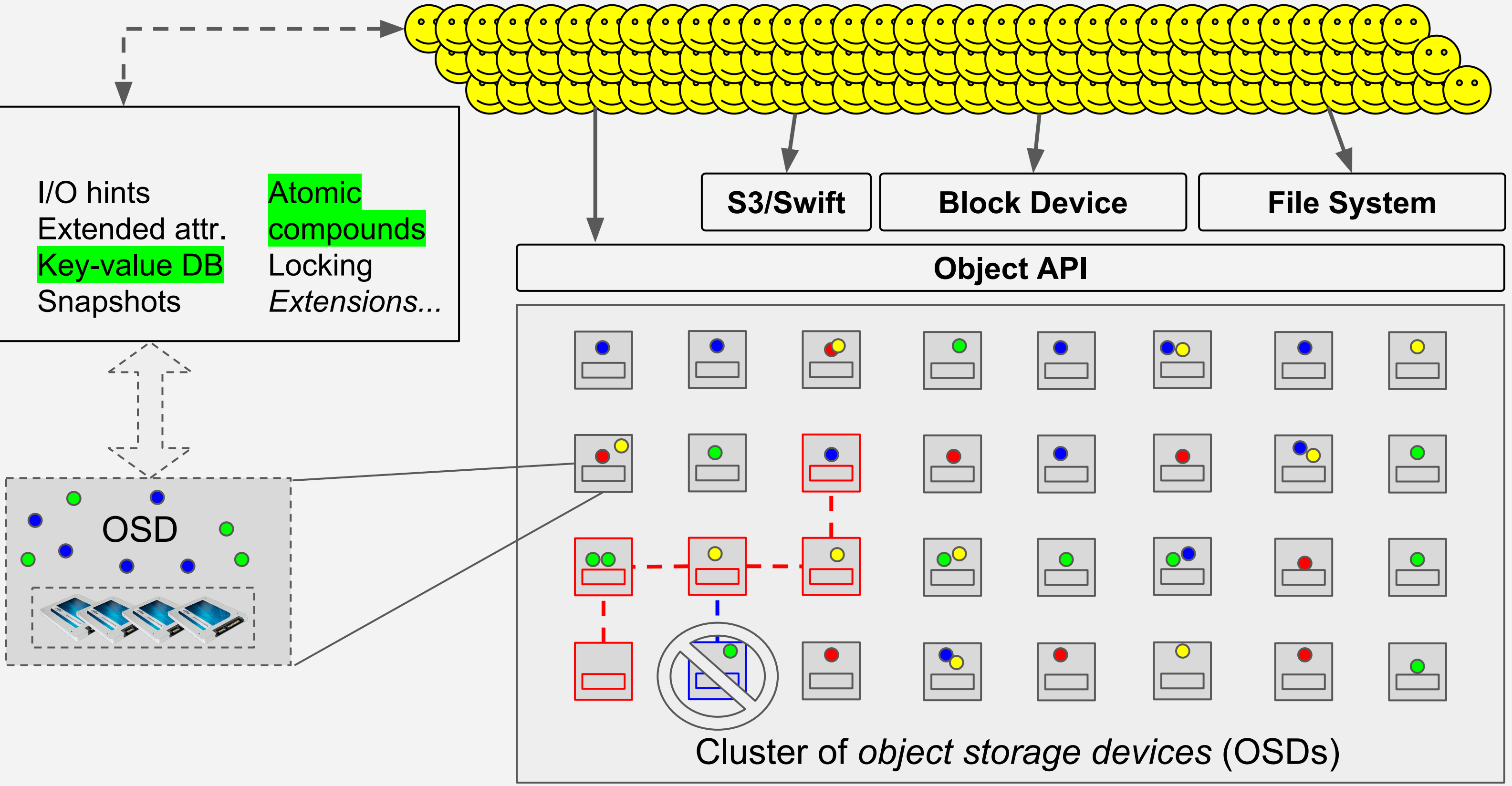
Atomic
compounds
Locking
Extensions...



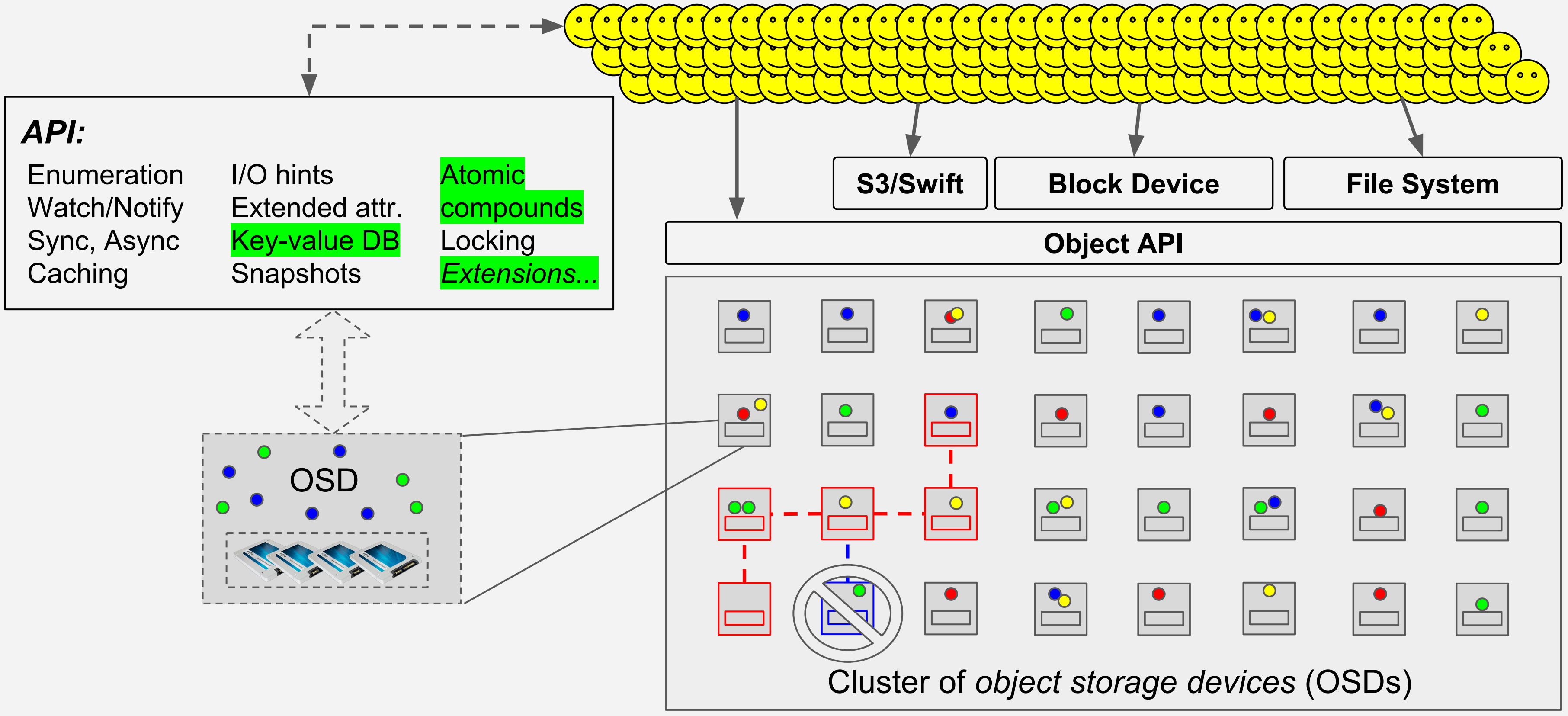
A rich object API makes things better

API:

Enumeration	I/O hints	Atomic compounds
Watch/Notify	Extended attr.	Locking
Sync, Async	Key-value DB	*Extensions...*
Caching	Snapshots	



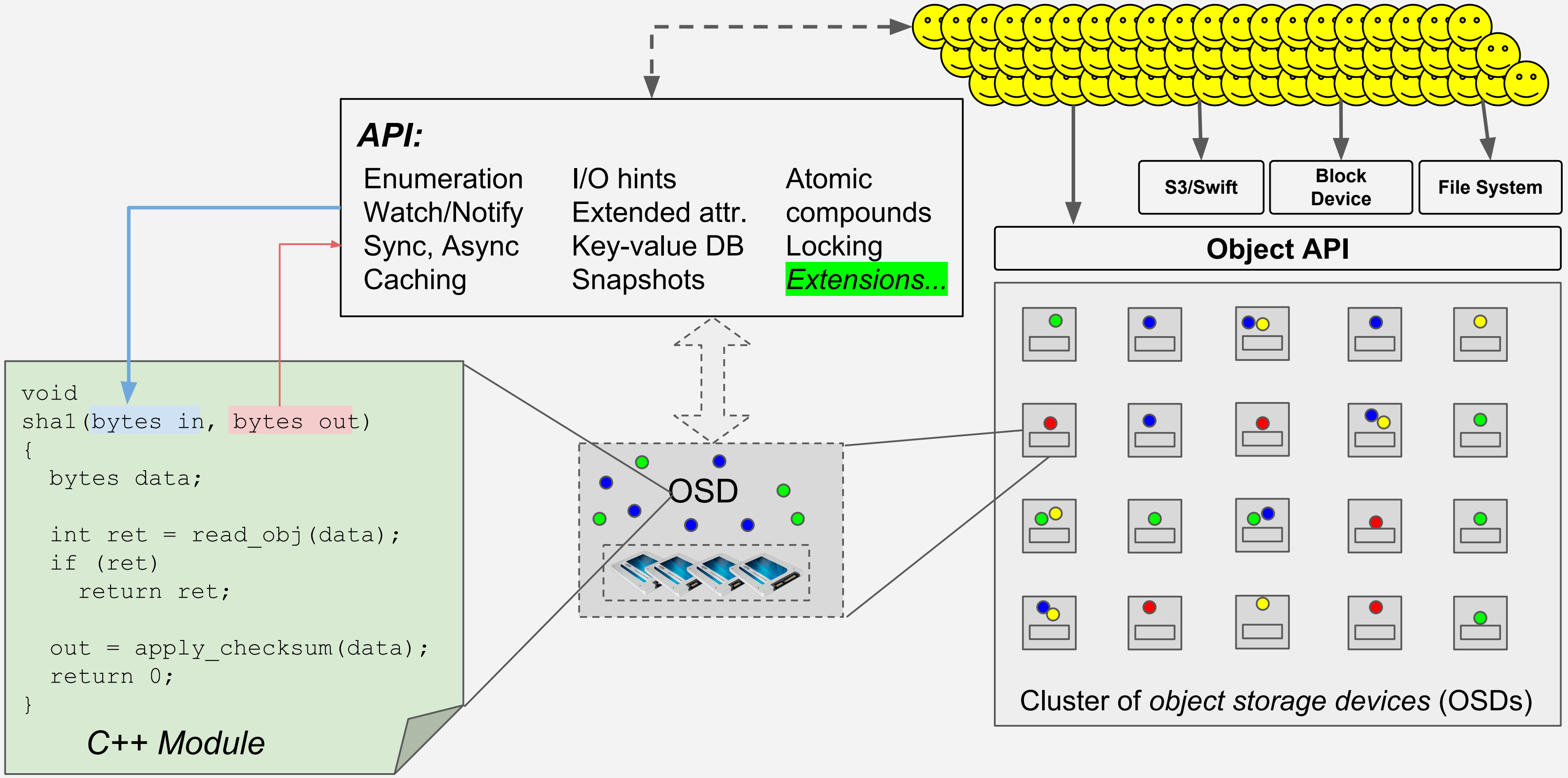
A rich object API makes things better



API:

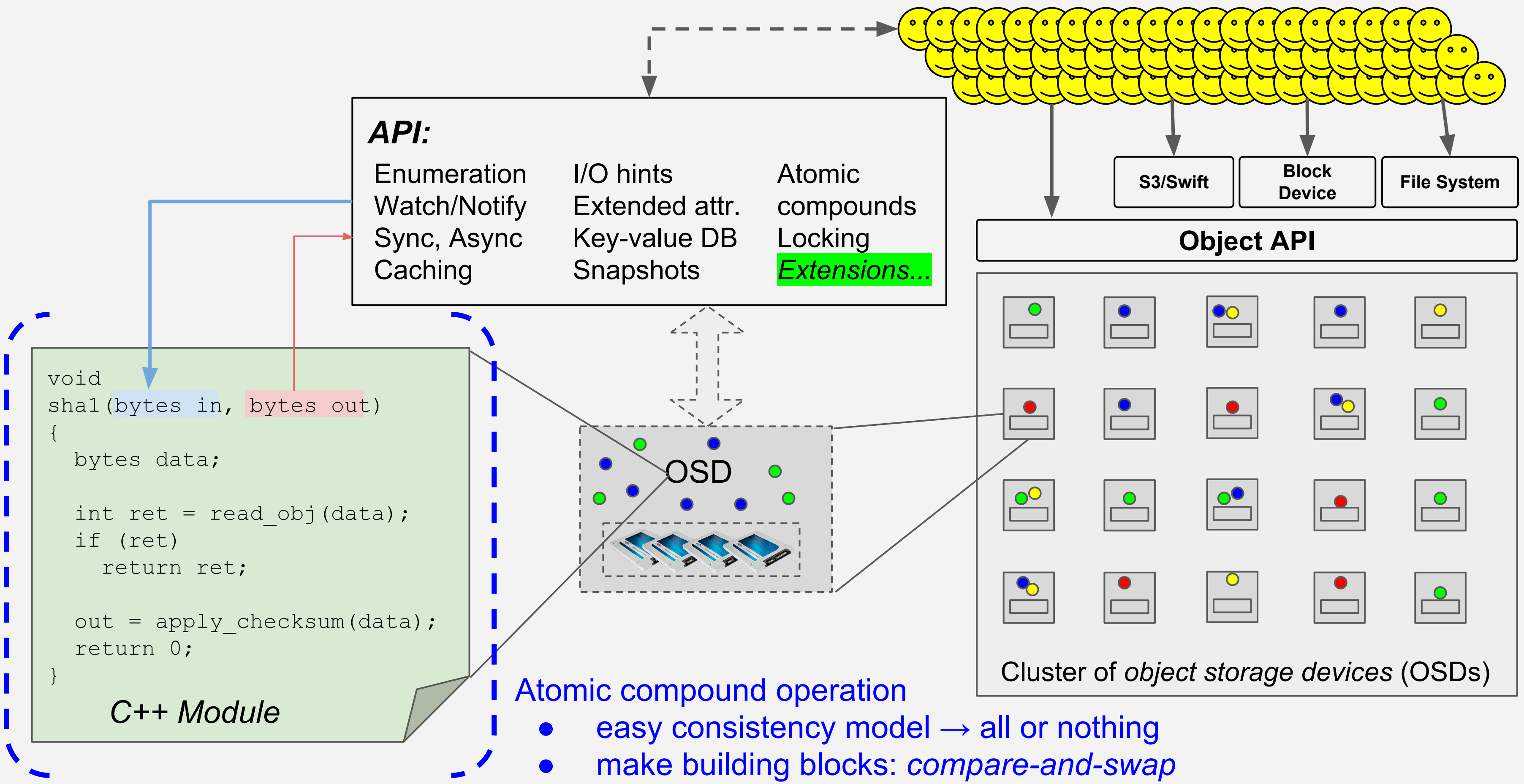
Enumeration	I/O hints	Atomic compounds
Watch/Notify	Extended attr.	Locking
Sync, Async	Key-value DB	*Extensions...*
Caching	Snapshots	

Object classes *extend* the object API



```
void
sha1(bytes in, bytes out)
{
  bytes data;

  int ret = read_obj(data);
  if (ret)
    return ret;

  out = apply_checksum(data);
  return 0;
}
```

Object classes *extend* the object API

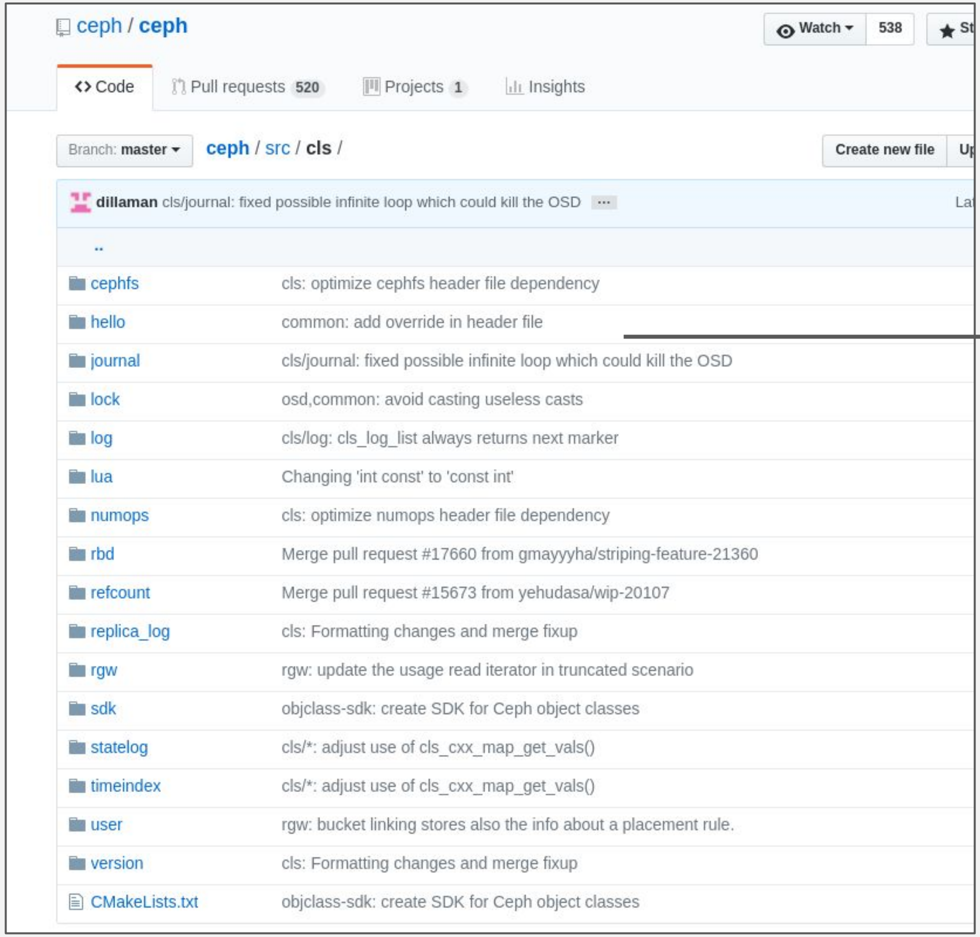


```
void
sha1(bytes in, bytes out)
{
  bytes data;

  int ret = read_obj(data);
  if (ret)
    return ret;

  out = apply_checksum(data);
  return 0;
}
```

Atomic compound operation

- easy consistency model → all or nothing
- make building blocks: *compare-and-swap*

Applications want to build object classes



ceph / ceph

Watch 538

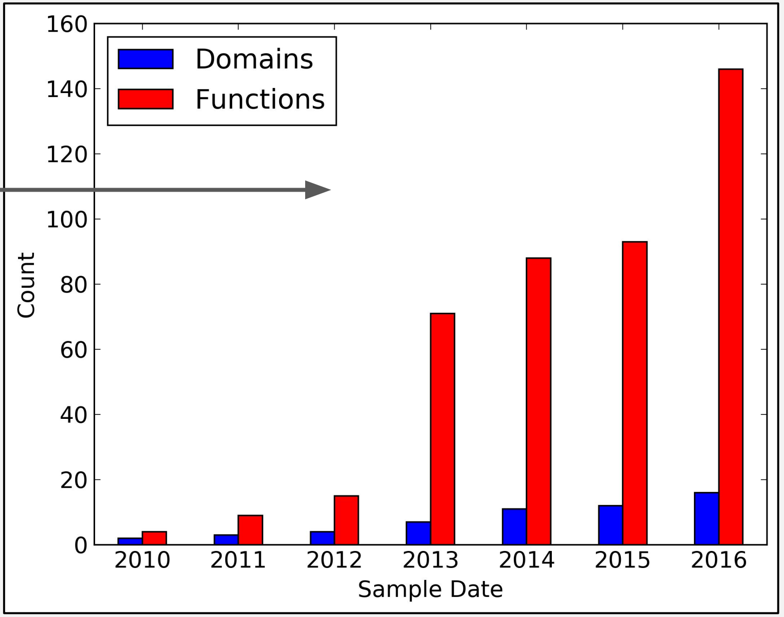
Code | Pull requests 520 | Projects 1 | Insights

Branch: master ceph / src / cls /

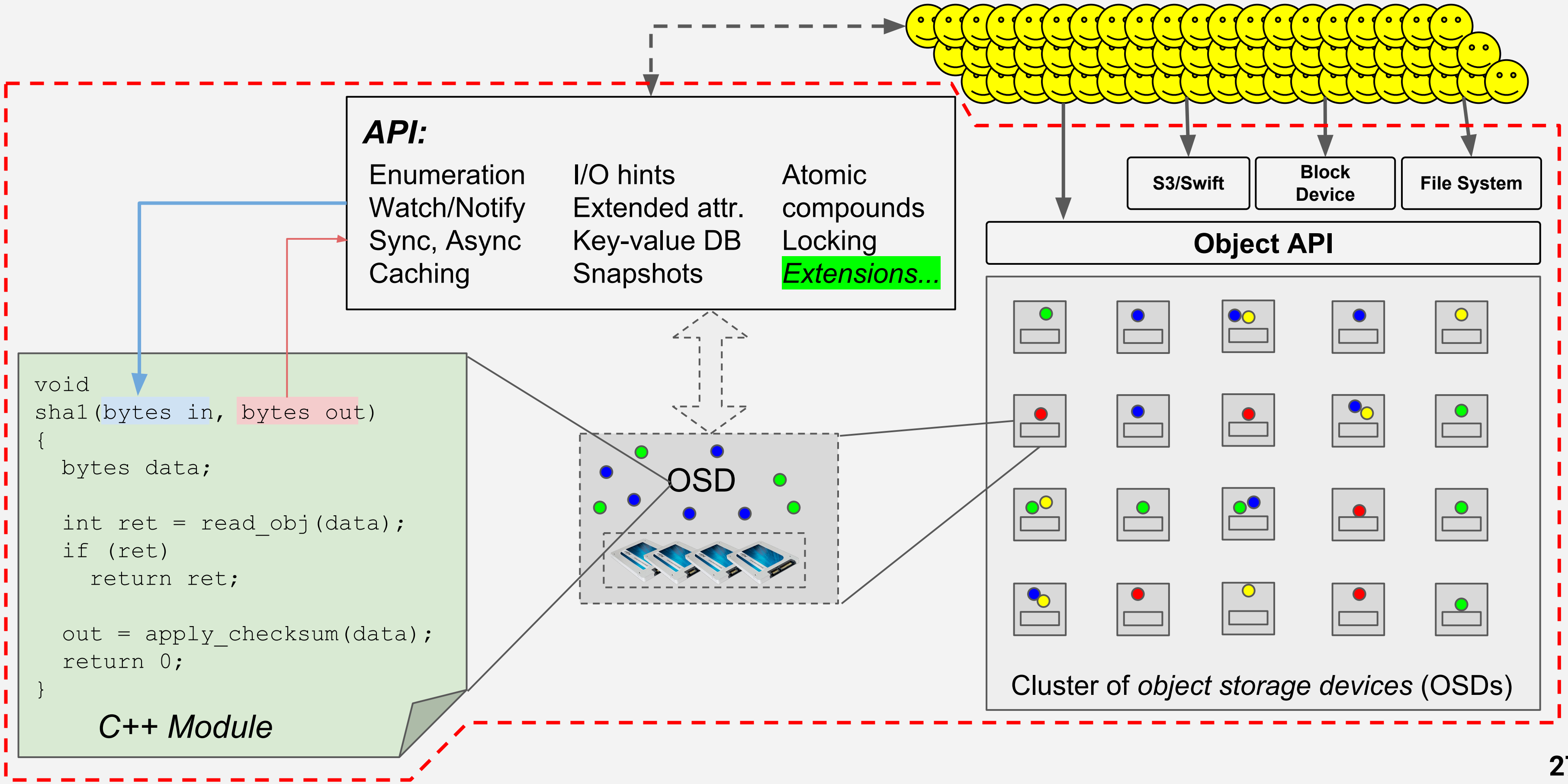
Create new file

dillaman cls/journal: fixed possible infinite loop which could kill the OSD

..	
cephfs	cls: optimize cephfs header file dependency
hello	common: add override in header file
journal	cls/journal: fixed possible infinite loop which could kill the OSD
lock	osd,common: avoid casting useless casts
log	cls/log: cls_log_list always returns next marker
lua	Changing 'int const' to 'const int'
numops	cls: optimize numops header file dependency
rbd	Merge pull request #17660 from gmayyyha/striping-feature-21360
refcount	Merge pull request #15673 from yehudasa/wip-20107
replica_log	cls: Formatting changes and merge fixup
rgw	rgw: update the usage read iterator in truncated scenario
sdk	objclass-sdk: create SDK for Ceph object classes
statelog	cls/*: adjust use of cls_cxx_map_get_vals()
timeindex	cls/*: adjust use of cls_cxx_map_get_vals()
user	rgw: bucket linking stores also the info about a placement rule.
version	cls: Formatting changes and merge fixup
CMakeLists.txt	objclass-sdk: create SDK for Ceph object classes



Object classes exist in a walled garden



API:

Enumeration
Watch/Notify
Sync, Async
Caching

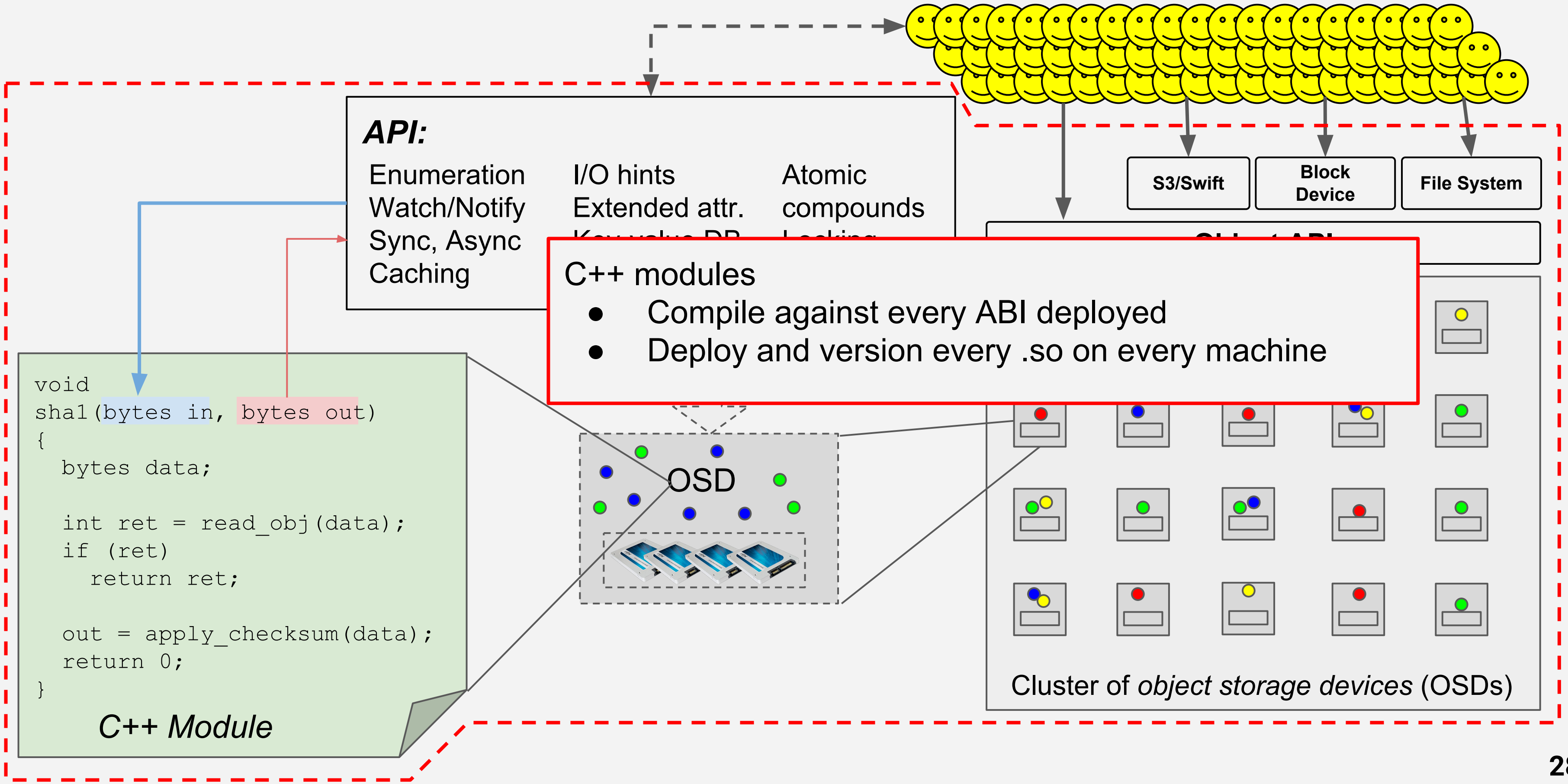
I/O hints
Extended attr.
Key-value DB
Snapshots

Atomic
compounds
Locking
Extensions...

```
void
sha1(bytes in, bytes out)
{
  bytes data;

  int ret = read_obj(data);
  if (ret)
    return ret;

  out = apply_checksum(data);
  return 0;
}
```

C++ Module

Object classes exist in a walled garden

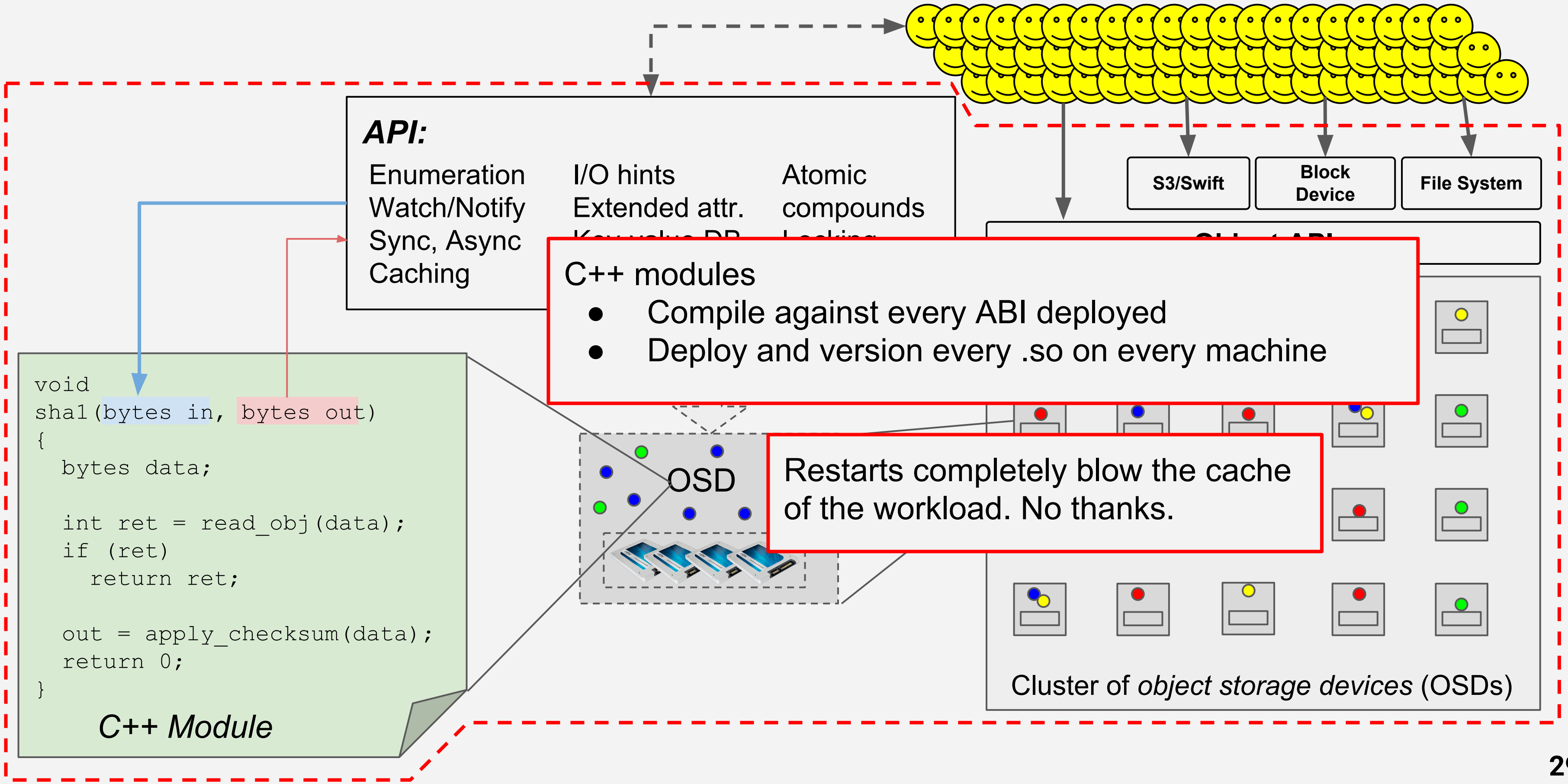


```
void
sha1(bytes in, bytes out)
{
  bytes data;

  int ret = read_obj(data);
  if (ret)
    return ret;

  out = apply_checksum(data);
  return 0;
}
```

C++ modules

- Compile against every ABI deployed
- Deploy and version every .so on every machine

Object classes exist in a walled garden



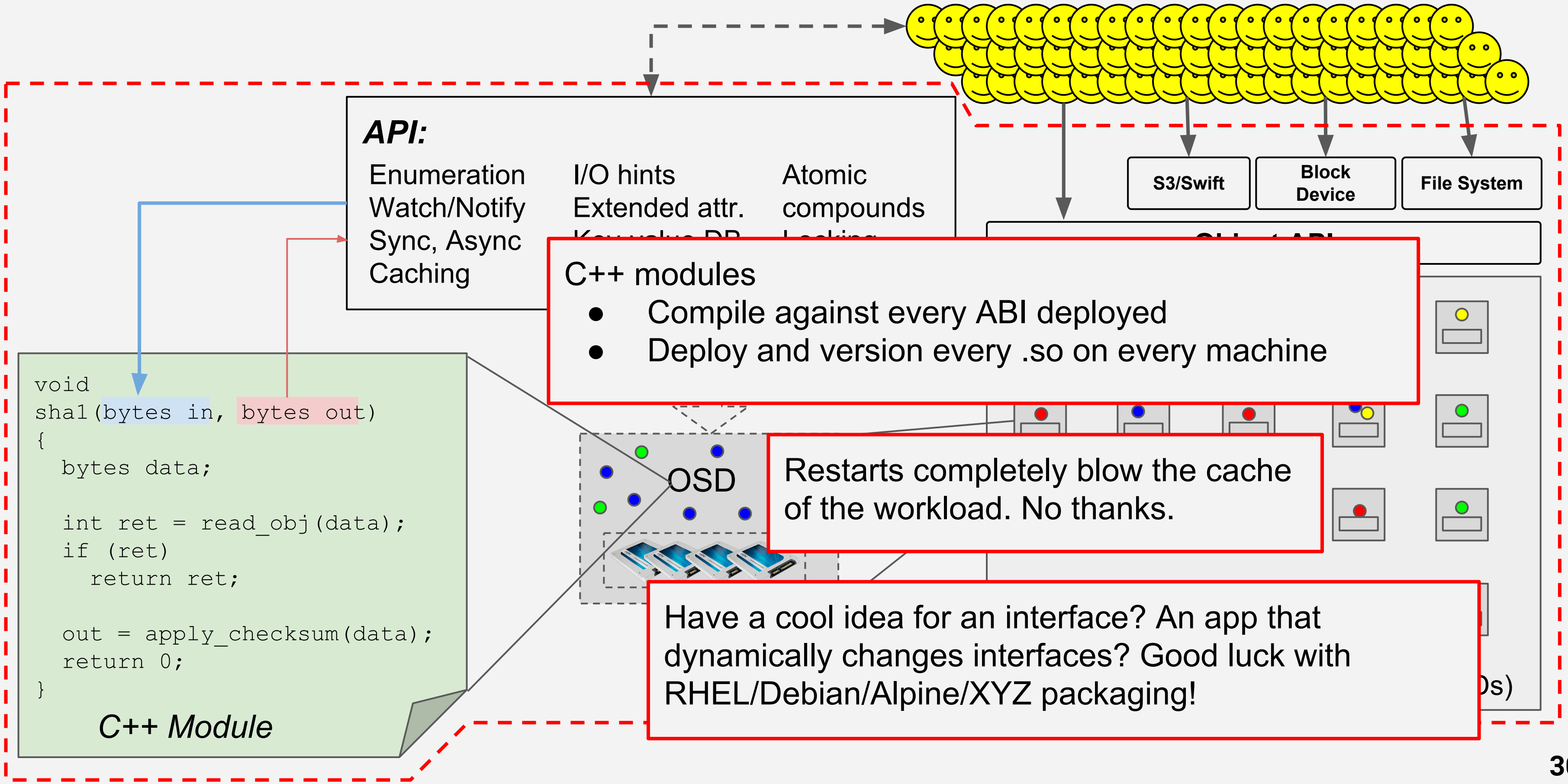
```
void
sha1(bytes in, bytes out)
{
  bytes data;

  int ret = read_obj(data);
  if (ret)
    return ret;

  out = apply_checksum(data);
  return 0;
}
```

C++ modules

- Compile against every ABI deployed
- Deploy and version every .so on every machine

Restarts completely blow the cache of the workload. No thanks.

Object classes exist in a walled garden



```
void
sha1(bytes in, bytes out)
{
  bytes data;

  int ret = read_obj(data);
  if (ret)
    return ret;

  out = apply_checksum(data);
  return 0;
}
```

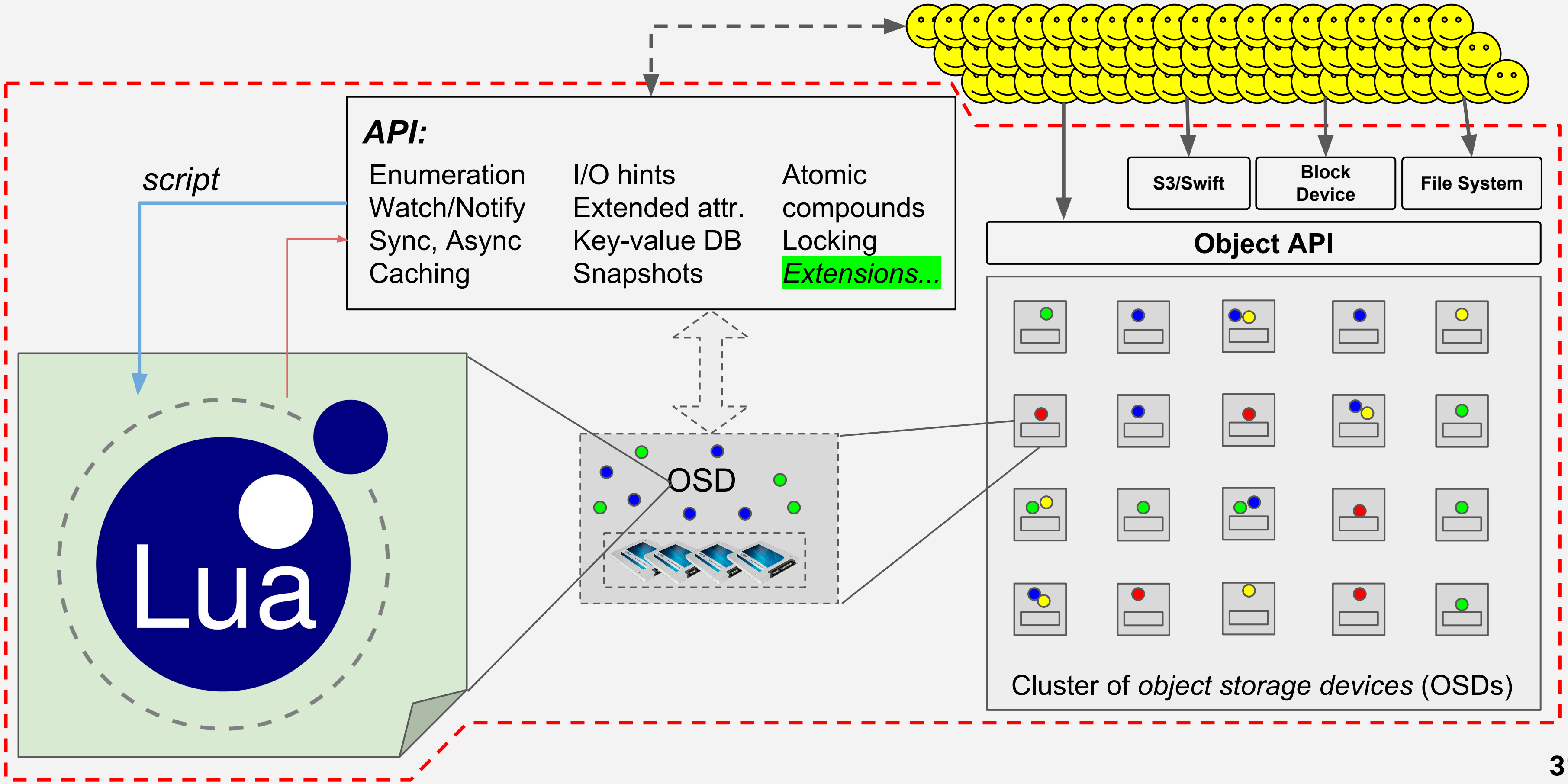
C++ modules

- Compile against every ABI deployed
- Deploy and version every .so on every machine

Restarts completely blow the cache of the workload. No thanks.

Have a cool idea for an interface? An app that dynamically changes interfaces? Good luck with RHEL/Debian/Alpine/XYZ packaging!

Lua object class: tunnel requests through LuaVMs



Example: remotely compute md5 hash

C++ API

```
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;
}
```

Read object

Compute result

Equivalent Lua

```
???
```

Example: remotely compute md5 hash

C++ API

```
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;
}
```

Read object

Compute result

Equivalent Lua

```
local md5 = require 'md5’

function compute_md5(input, output)
  local data = objclass.read()
  output = md5.sumhexa(data)
end
```

Error handling is easy and robust

C++ API

```
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;
}
```

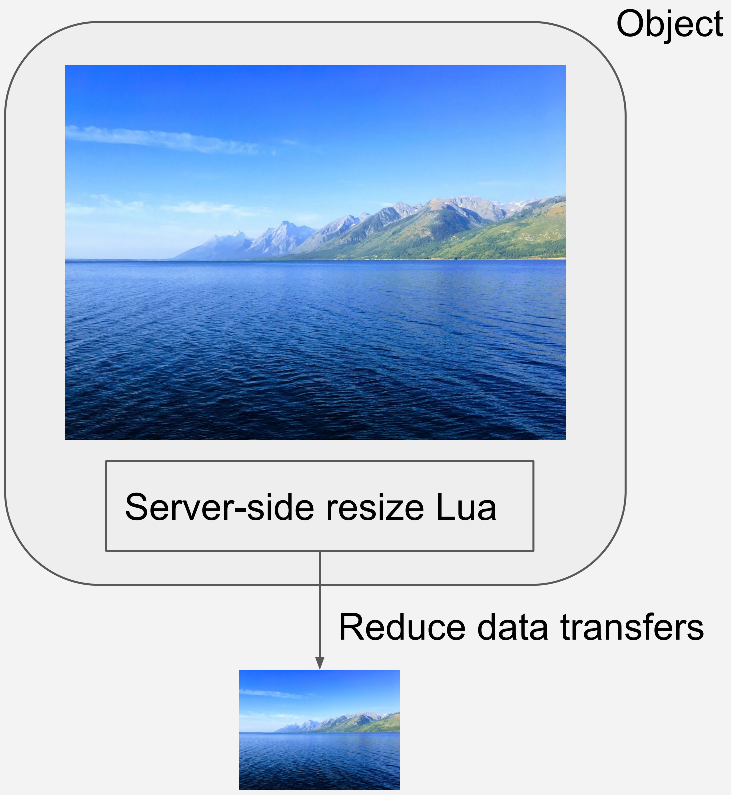
Equivalent Lua

```
local md5 = require 'md5’

function compute_md5(input, output)
  local data = objclass.read()
  output = md5.sumhexa(data)
end
```

- Low-level errors are usually not caught
- Common error handling patterns
  - transparent in Lua integration
- Sometimes you want the error
  - wrap in *ok, ret, args = pcall(...)*

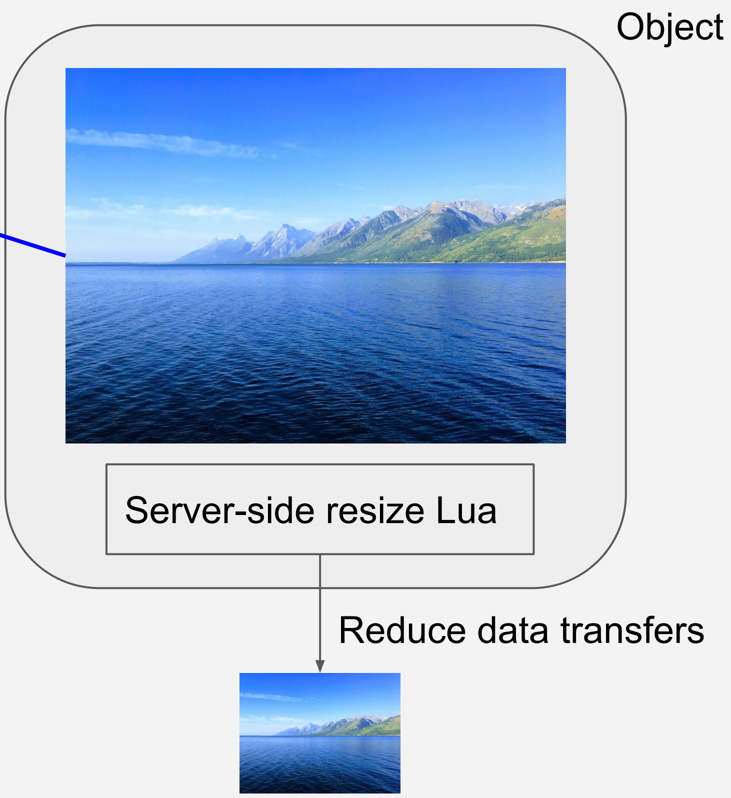
One more example: image thumbnail generation

```
local magick = require "magick"

function thumb(input, output)
  local blob = objclass.read()
  local img = assert(
    magick.load_image_from_blob(blob:str()))

  local spec_string = input:str()
  img = magick.thumb(img, spec_string)

  output:append(img)
end

objclass.register(thumb)
```



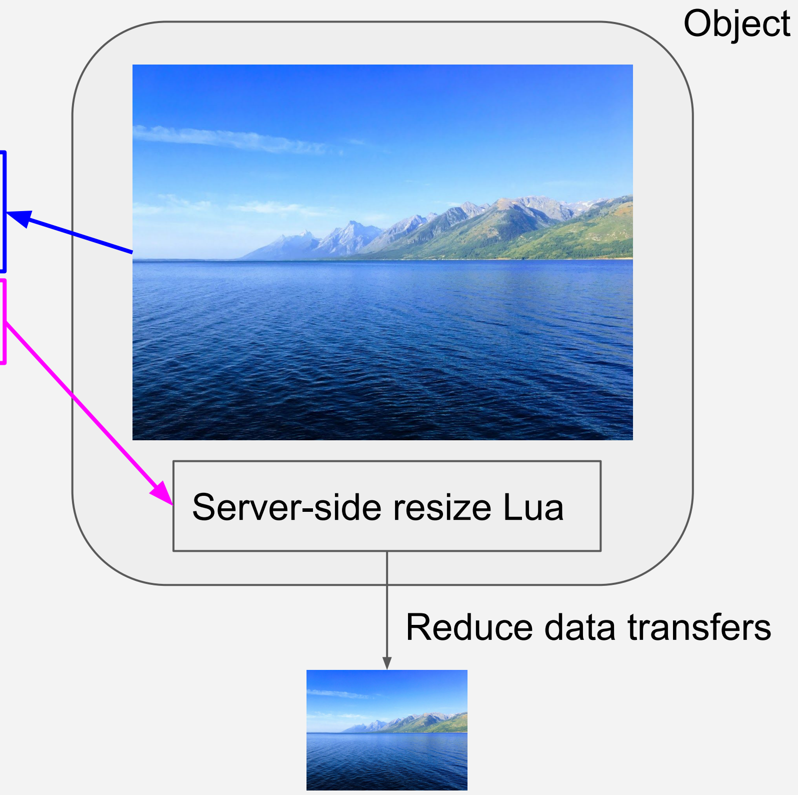
One more example: image thumbnail generation

```
local magick = require "magick"

function thumb(input, output)
  local blob = objclass.read()
  local img = assert(
    magick.load_image_from_blob(blob:str()))

  local spec_string = input:str()
  img = magick.thumb(img, spec_string)

  output:append(img)
end

objclass.register(thumb)
```



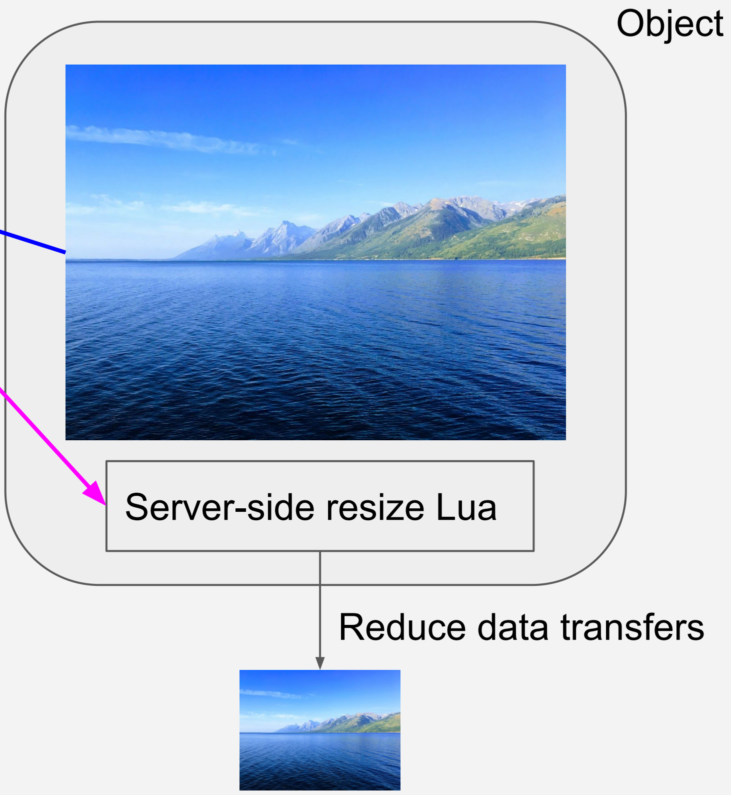
One more example: image thumbnail generation

```
local magick = require "magick"

function thumb(input, output)
  local blob = objclass.read()
  local img = assert(
    magick.load_image_from_blob(blob:str()))

  local spec_string = input:str()
  img = magick.thumb(img, spec_string)

  output:append(img)
end

objclass.register(thumb)
```



One more example: image thumbnail generation

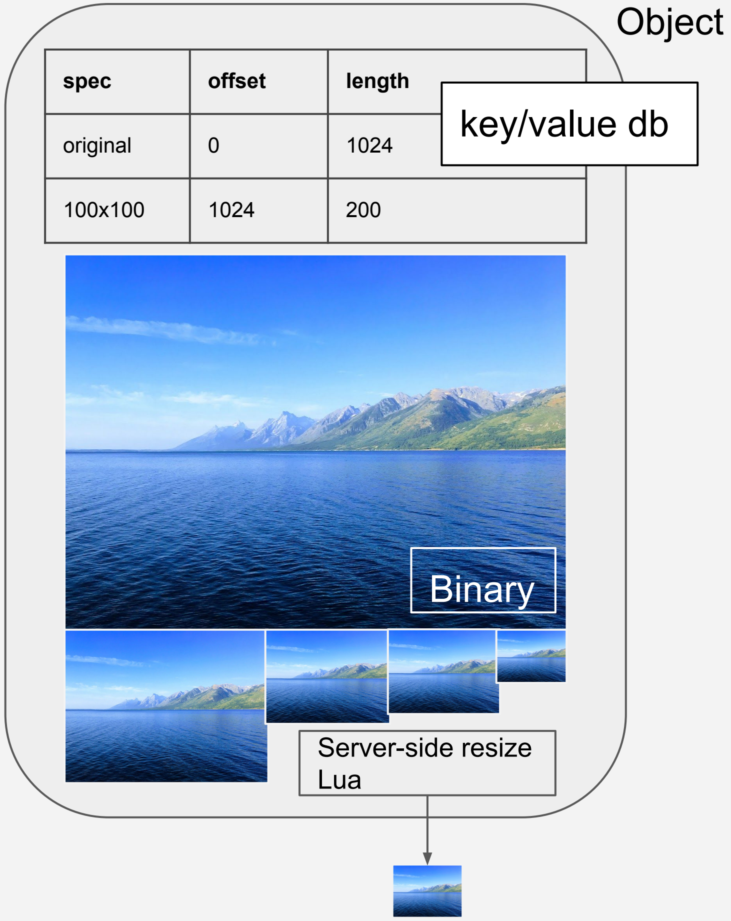
```lua
local magick = require "magick"

function thumb(input, output)
  local blob = objclass.read()
  local img = assert(
    magick.load_image_from_blob(blob:str()))

  local spec_string = input:str()
  img = magick.thumb(img, spec_string)

  output:append(img)
end

objclass.register(thumb)
```



Avoid *recomputation* by caching thumbnails!

Example: image thumbnail generation *with caching*

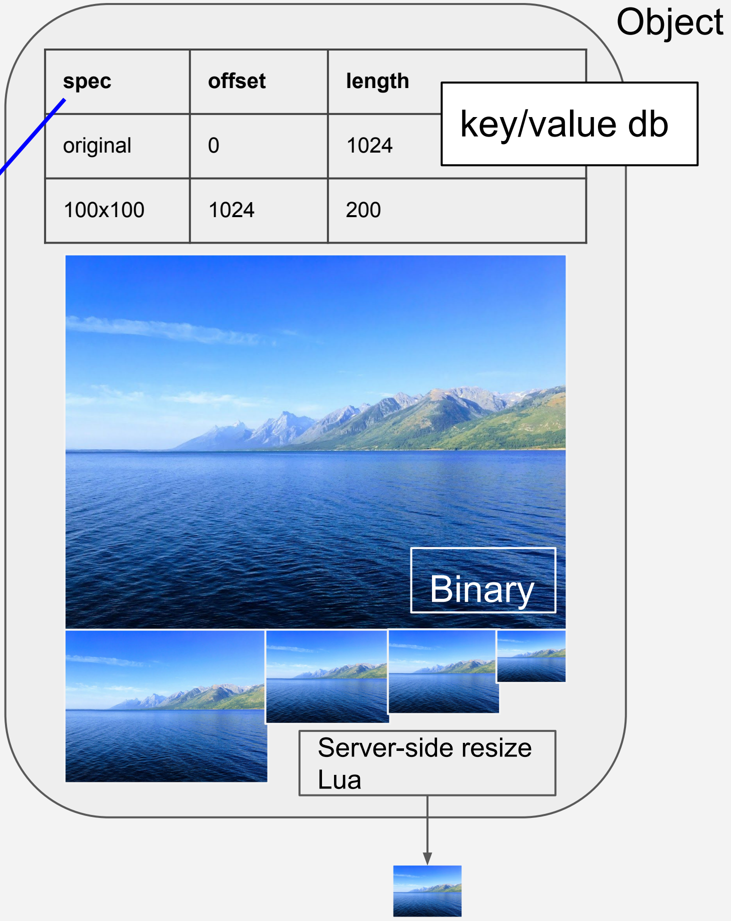
```
local magick = require "magick"

function thumb(input, output)
  local build_thumb = false
  local spec = input:str()
  ok, ret, loc = pcall(objclass.get_map_val, spec)
  if ret == -objclass.ENOENT then
    loc = objclass.get_map_val(“original”)
    build_thumb = true
  end

  local size, off = string.match(loc:str(), “(%d+)@(%d+)”)
  local blob = objclass.read(off, size)
  if not build_thumb then
    output:append(blob)
  else
    img = magick.load_image_from_blob(blob:str()).
    img = magick.thumb(img, spec)

    local obj_size, mtime = objclass.stat()
    loc = #img .. “@” .. obj_size
    objclass.write(off, #img, img)
    objclass.map_set_val(spec, loc)
    output:append(img)
  end
end
objclass.register(thumb)
```



spec	offset	length
original	0	1024
100x100	1024	200

Example: image thumbnail generation *with caching*

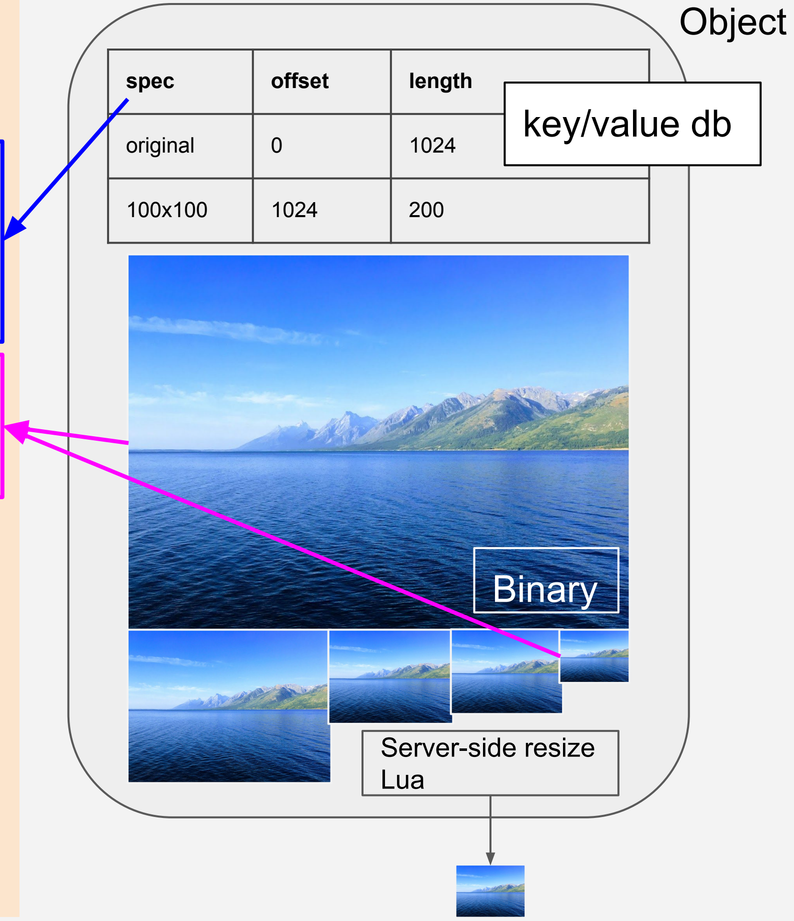
```
local magick = require "magick"

function thumb(input, output)
  local build_thumb = false
  local spec = input:str()
  ok, ret, loc = pcall(objclass.get_map_val, spec)
  if ret == -objclass.ENOENT then
    loc = objclass.get_map_val(“original”)
    build_thumb = true
  end

  local size, off = string.match(loc:str(), “(%d+)@(%d+)”)
  local blob = objclass.read(off, size)
  if not build_thumb then
    output:append(blob)
  else
    img = magick.load_image_from_blob(blob:str()).
    img = magick.thumb(img, spec)

    local obj_size, mtime = objclass.stat()
    loc = #img .. “@” .. obj_size
    objclass.write(off, #img, img)
    objclass.map_set_val(spec, loc)
    output:append(img)
  end
end
objclass.register(thumb)
```



spec	offset	length
original	0	1024
100x100	1024	200

Example: image thumbnail generation *with caching*

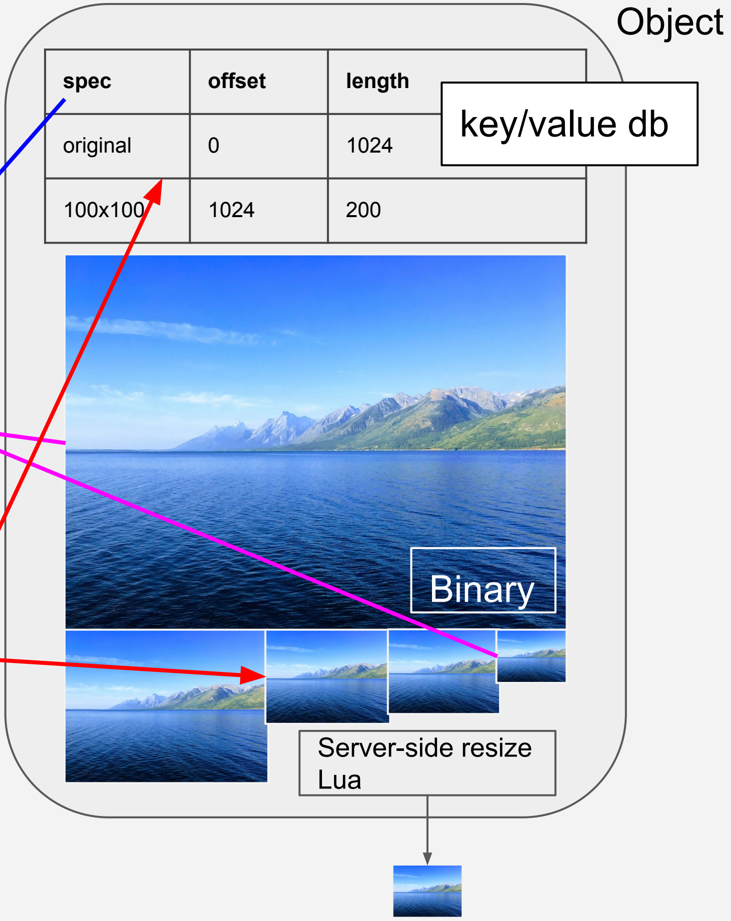
```
local magick = require "magick"

function thumb(input, output)
  local build_thumb = false
  local spec = input:str()
  ok, ret, loc = pcall(objclass.get_map_val, spec)
  if ret == -objclass.ENOENT then
    loc = objclass.get_map_val(“original”)
    build_thumb = true
  end

  local size, off = string.match(loc:str(), “(%d+)@(%d+)”)
  local blob = objclass.read(off, size)
  if not build_thumb then
    output:append(blob)
  else
    img = magick.load_image_from_blob(blob:str()).
    img = magick.thumb(img, spec)

    local obj_size, mtime = objclass.stat()
    loc = #img .. “@” .. obj_size
    objclass.write(off, #img, img)
    objclass.map_set_val(spec, loc)
    output:append(img)
  end
end
objclass.register(thumb)
```



spec	offset	length
original	0	1024
100x100	1024	200

Example: image thumbnail generation *with caching*

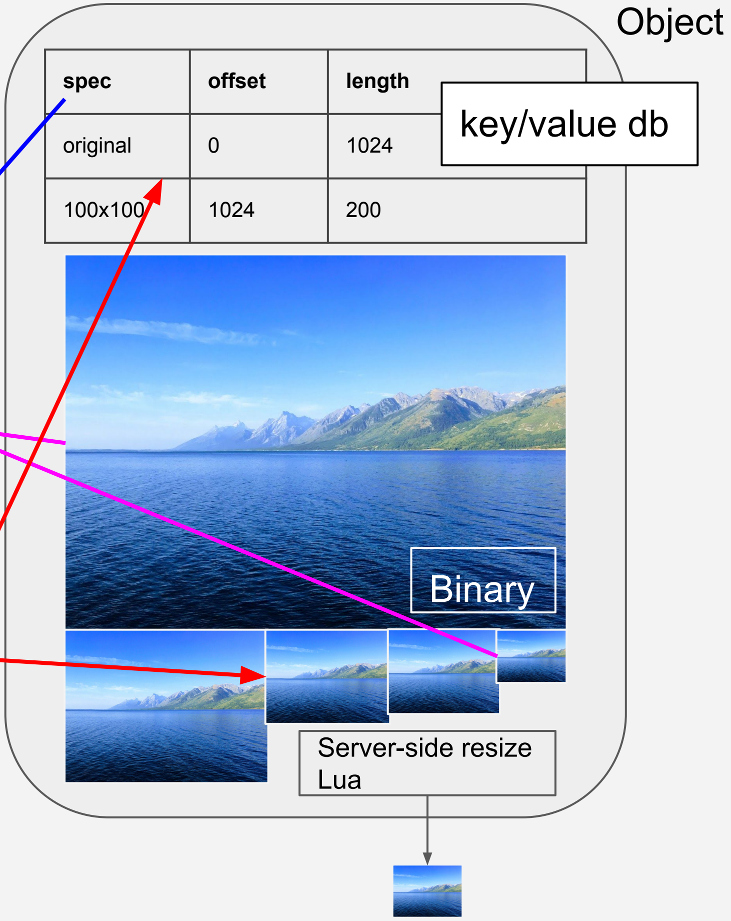
```
local magick = require "magick"

function thumb(input, output)
  local build_thumb = false
  local spec = input:str()
  ok, ret, loc = pcall(objclass.get_map_val, spec)
  if ret == -objclass.ENOENT then
    loc = objclass.get_map_val(“original”)
    build_thumb = true
  end

  local size, off = string.match(loc:str(), “(%d+)@(%d+)”)
  local blob = objclass.read(off, size)
  if not build_thumb then
    output:append(blob)
  else
    img = magick.load_image_from_blob(blob:str()).
    img = magick.thumb(img, spec)

    local obj_size, mtime = objclass.stat()
    loc = #img .. “@” .. obj_size
    objclass.write(off, #img, img)
    objclass.map_set_val(spec, loc)
    output:append(img)
  end
end
objclass.register(thumb)
```



spec	offset	length
original	0	1024
100x100	1024	200

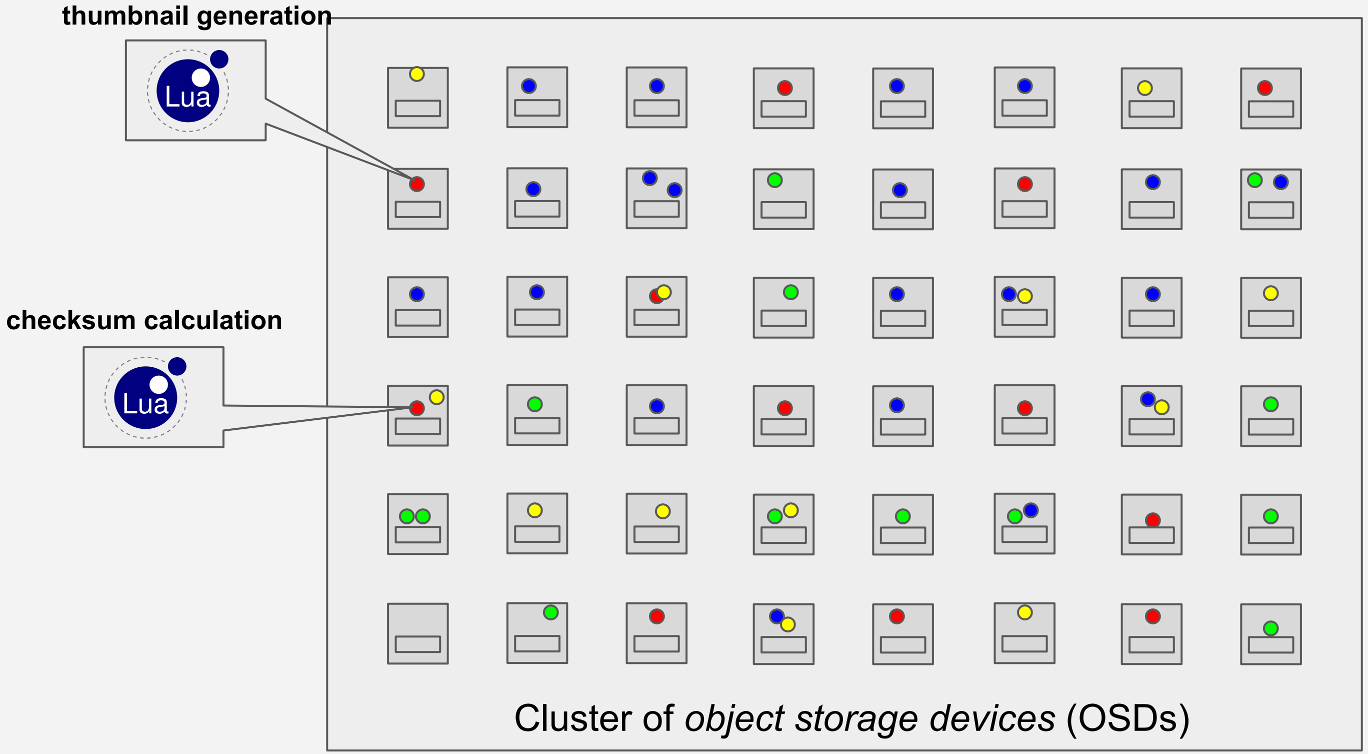
Example: image thumbnail generation *with caching*

```
local magick = require "magick"

function thumb(input, output)
  local build_thumb = false
  local spec = input:str()
  ok, ret, loc = pcall(objclass.get_map_val, spec)
  if ret == -objclass.ENOENT then
    loc = objclass.get_map_val(“original”)
    build_thumb = true
  end

  local size, off = string.match(loc:str(), “(%d+)@(%d+)”)
  local blob = objclass.read(off, size)
  if not build_thumb then
    output:append(blob)
  else
    img = magick.load_image_from_blob(blob:str()).
    img = magick.thumb(img, spec)

    local obj_size, mtime = objclass.stat()
    loc = #img .. “@” .. obj_size
    objclass.write(off, #img, img)
    objclass.map_set_val(spec, loc)
    output:append(img)
  end
end
objclass.register(thumb)
```

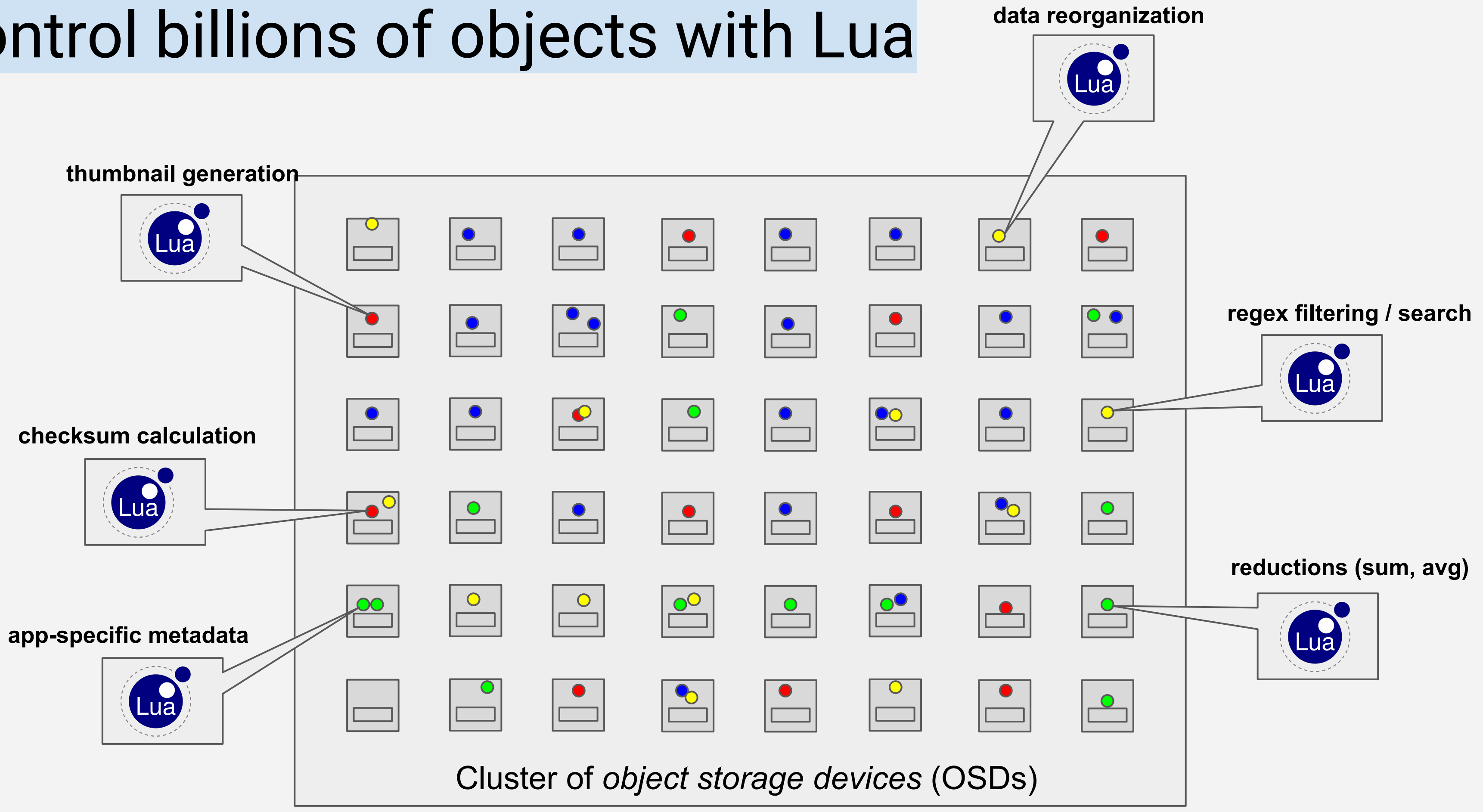


spec	offset	length
original	0	1024
100x100	1024	200

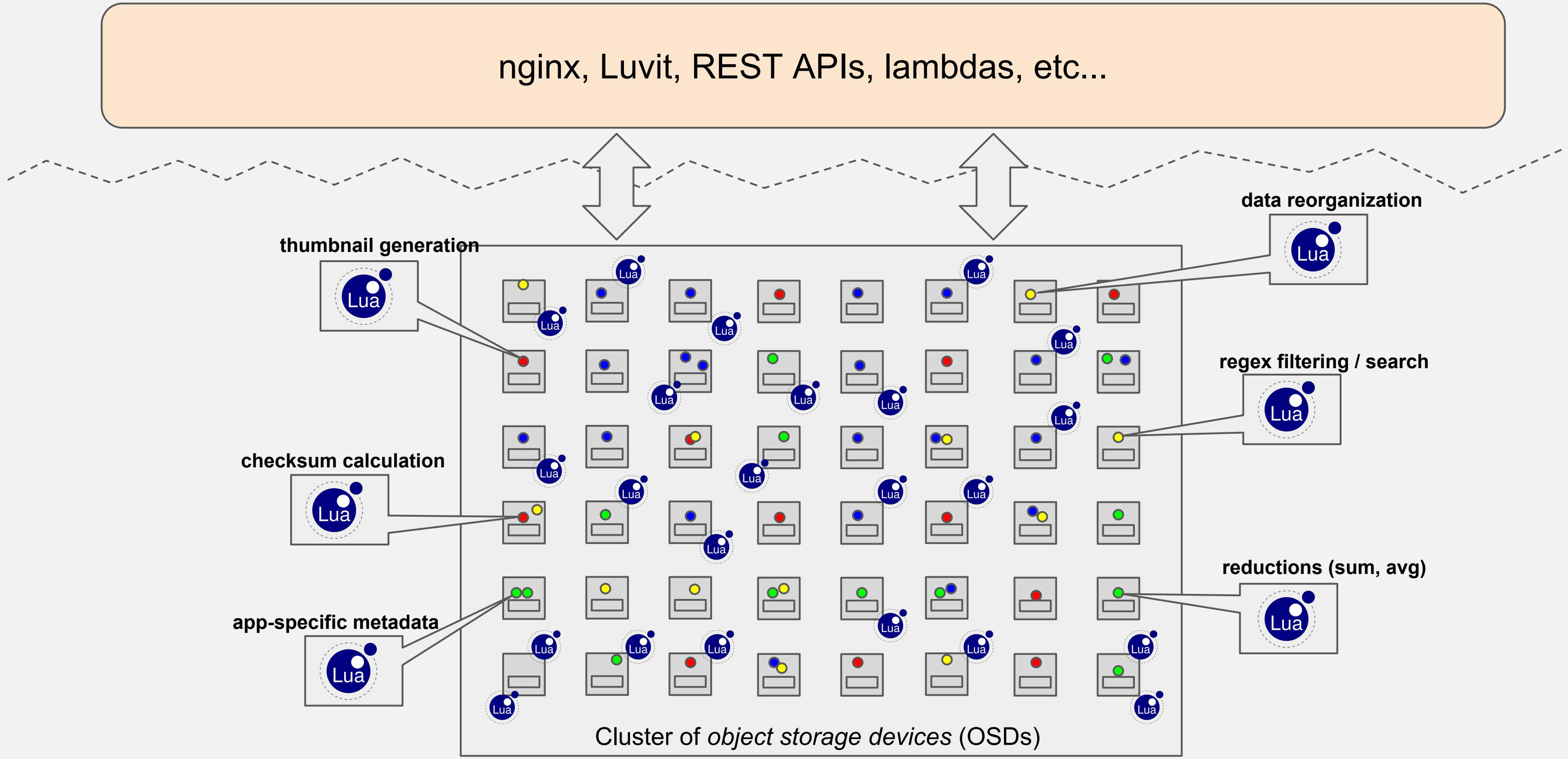
Control billions of objects with Lua



Control billions of objects with Lua



Control billions of objects with Lua



Shipping in upstream Ceph as of March ‘17 (Kraken)

- Available in official packages
  - >= Kraken release
- Expand API coverage as needed
- Use your favorite Ceph clients
  - C/C++, Python, Lua
  - Java, Rust, Go
- Input methods (Lua + Input Data)
  - C/C++ has efficient data handling
    - Native encoding
  - JSON for compatability

API:

Enumeration	I/O hints	Atomic compounds
Watch/Notify	Extended attr.	Locking
Sync, Async	Key-value DB	Read
Caching	Snapshots	Create
Write	Stat	
Remove		

```
{
  "script": "function echo(input, output) output:append(input:str()); end
              objclass.register(json_echo)",
  "input": "omg it works",
  "handler": "echo"
}
```

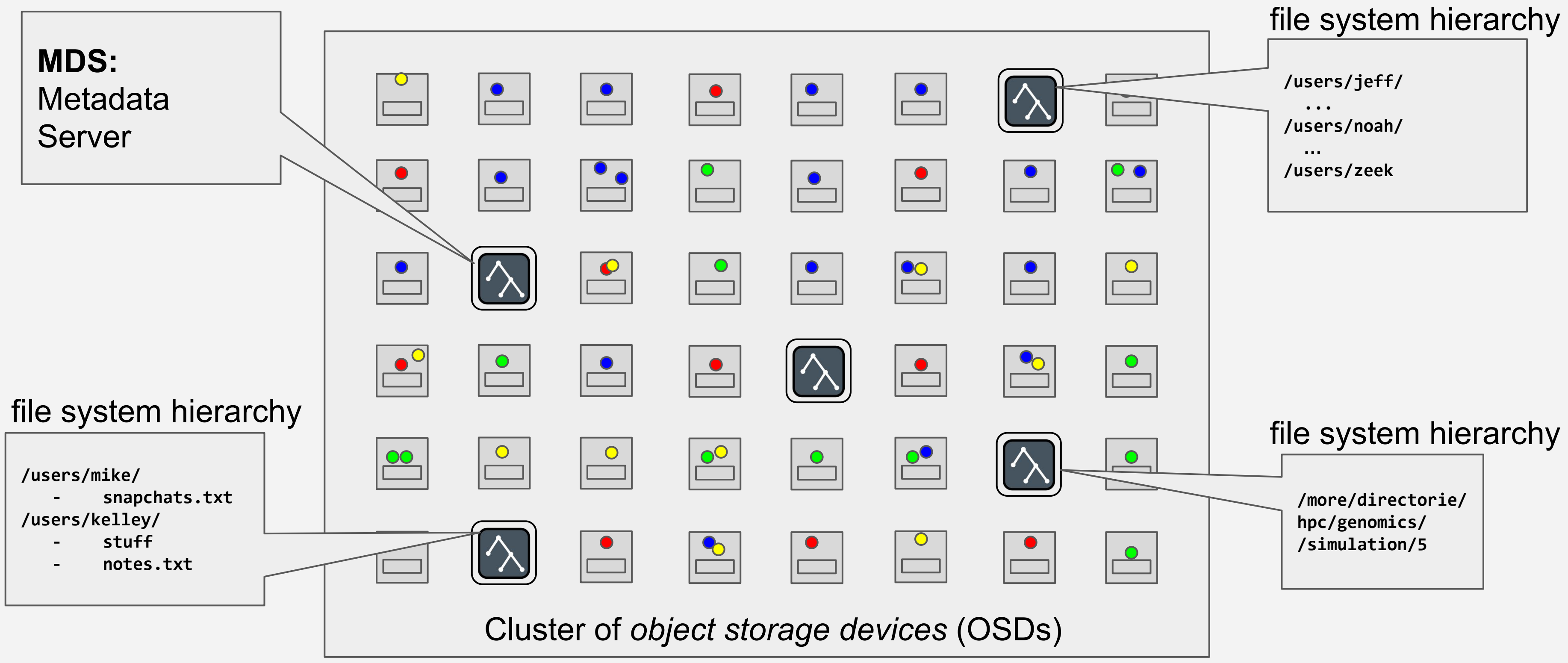
Lua in your file systems

Mantle: programmable metadata load balancer

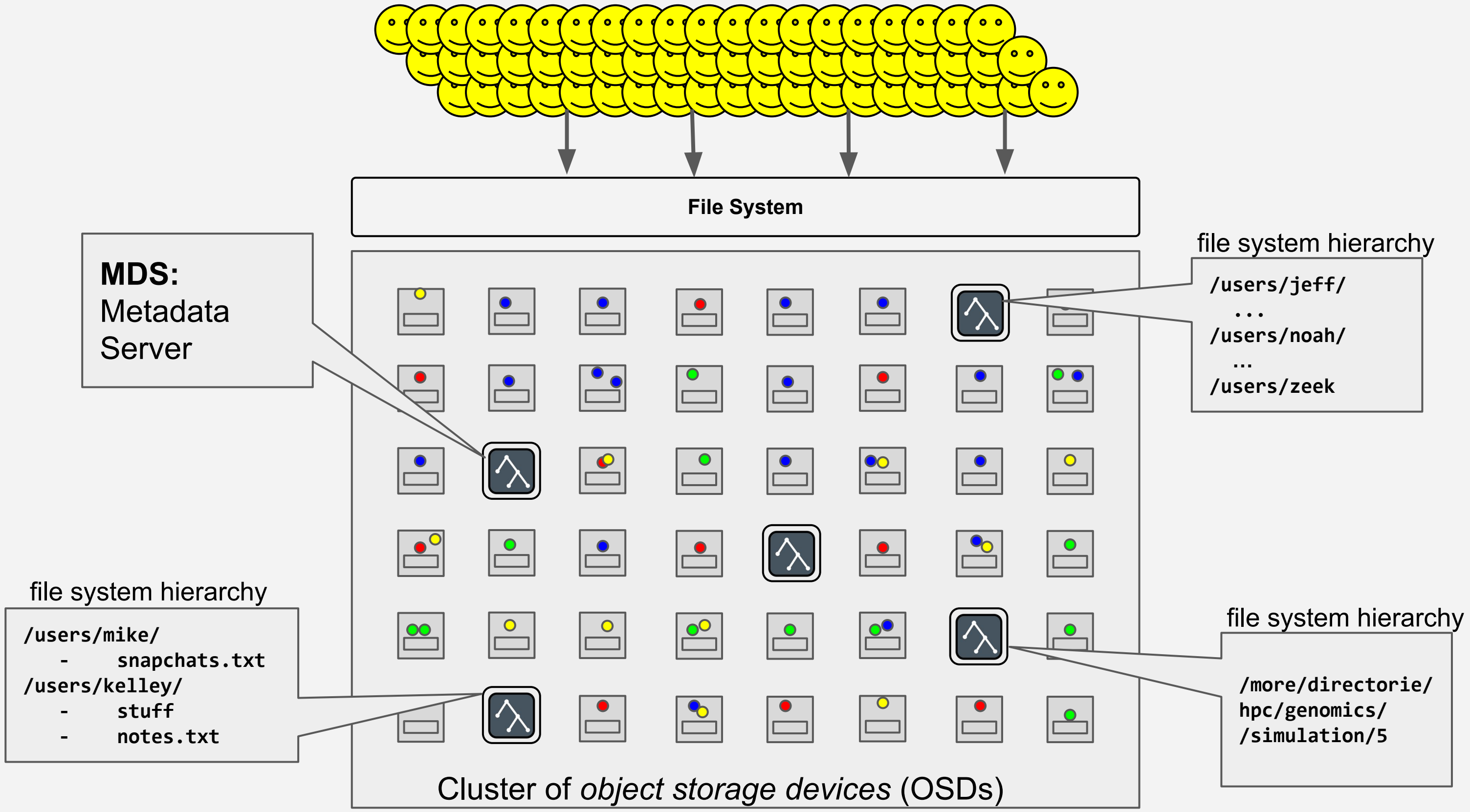
Michael Sevilla

– http://programmability.us

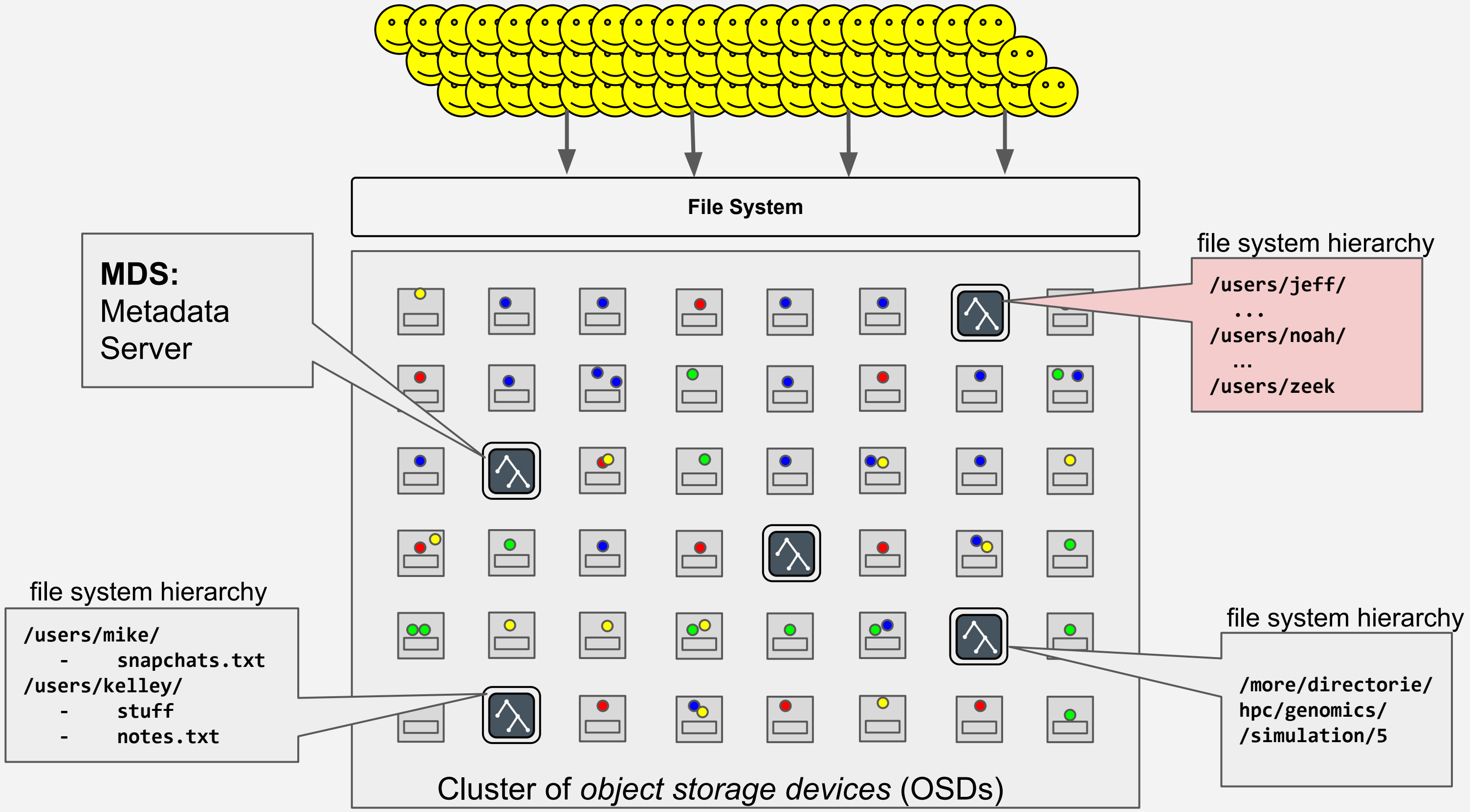
Ceph file system metadata load balancing



Ceph file system metadata load balancing



Ceph file system metadata load balancing

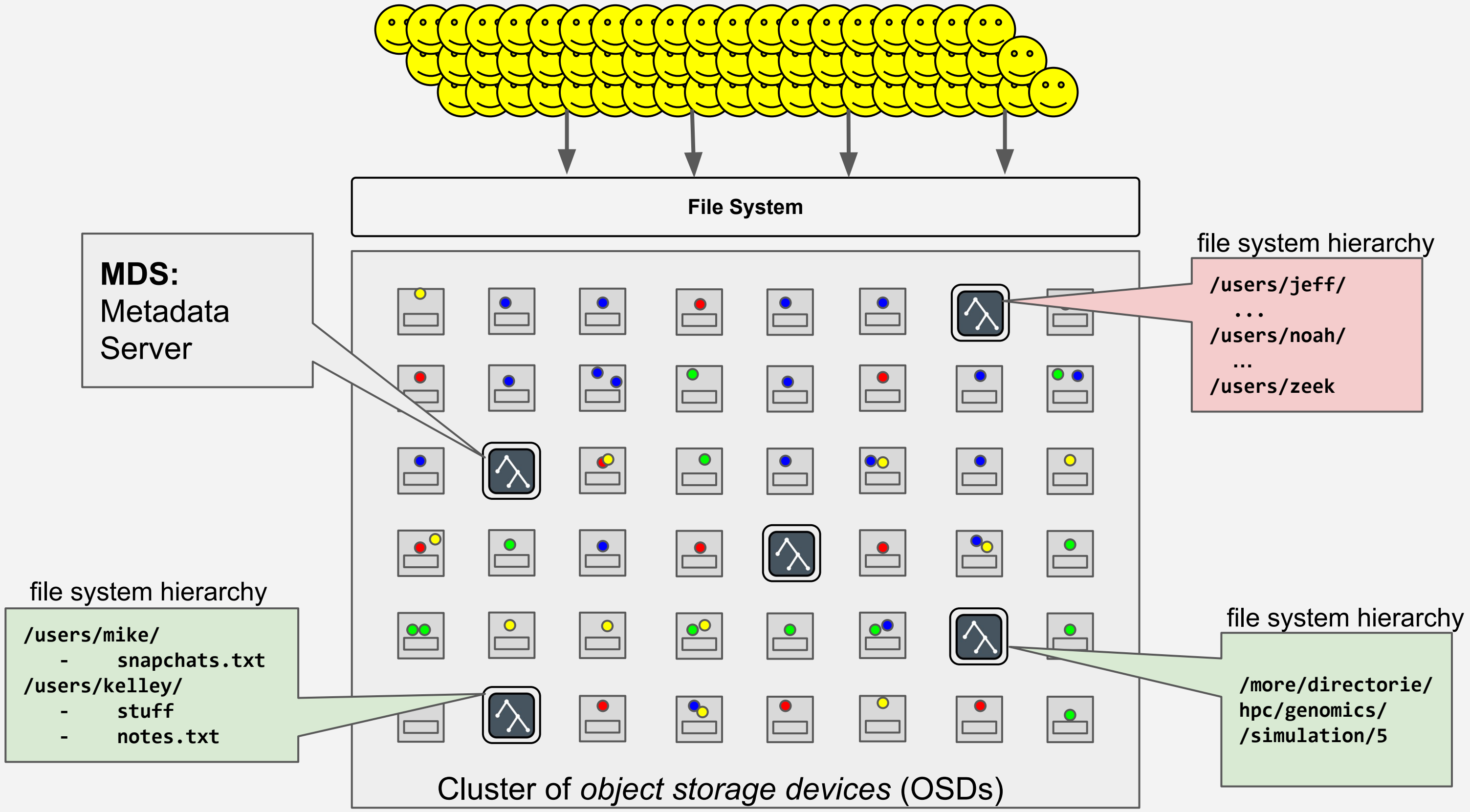


```
/users/jeff/
  ...
/users/noah/
  ...
/users/zeek
```

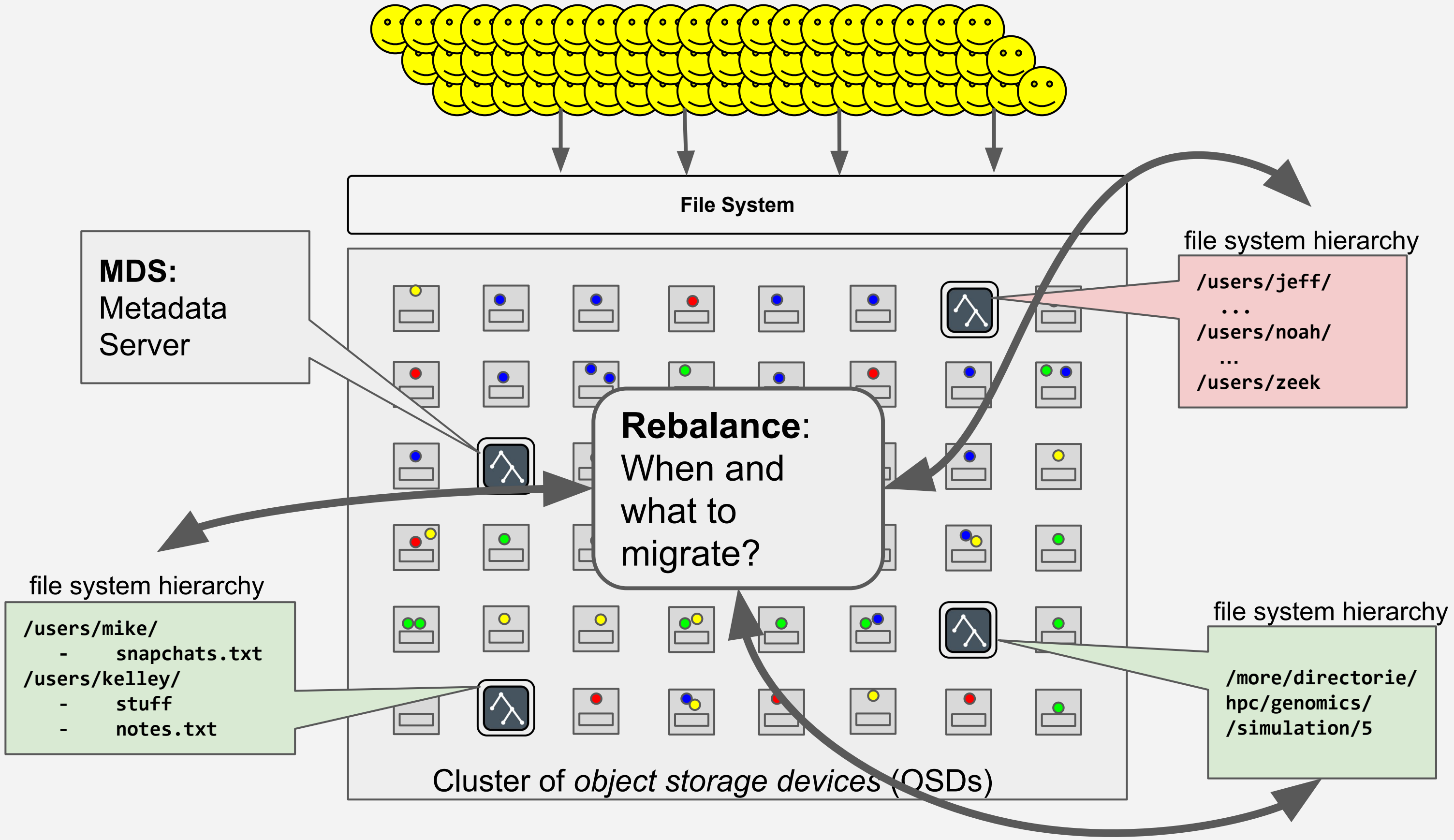
```
/users/mike/
   -    snapchats.txt
/users/kelley/
   -    stuff
   -    notes.txt
```

```
/more/directorie/
hpc/genomics/
/simulation/5
```

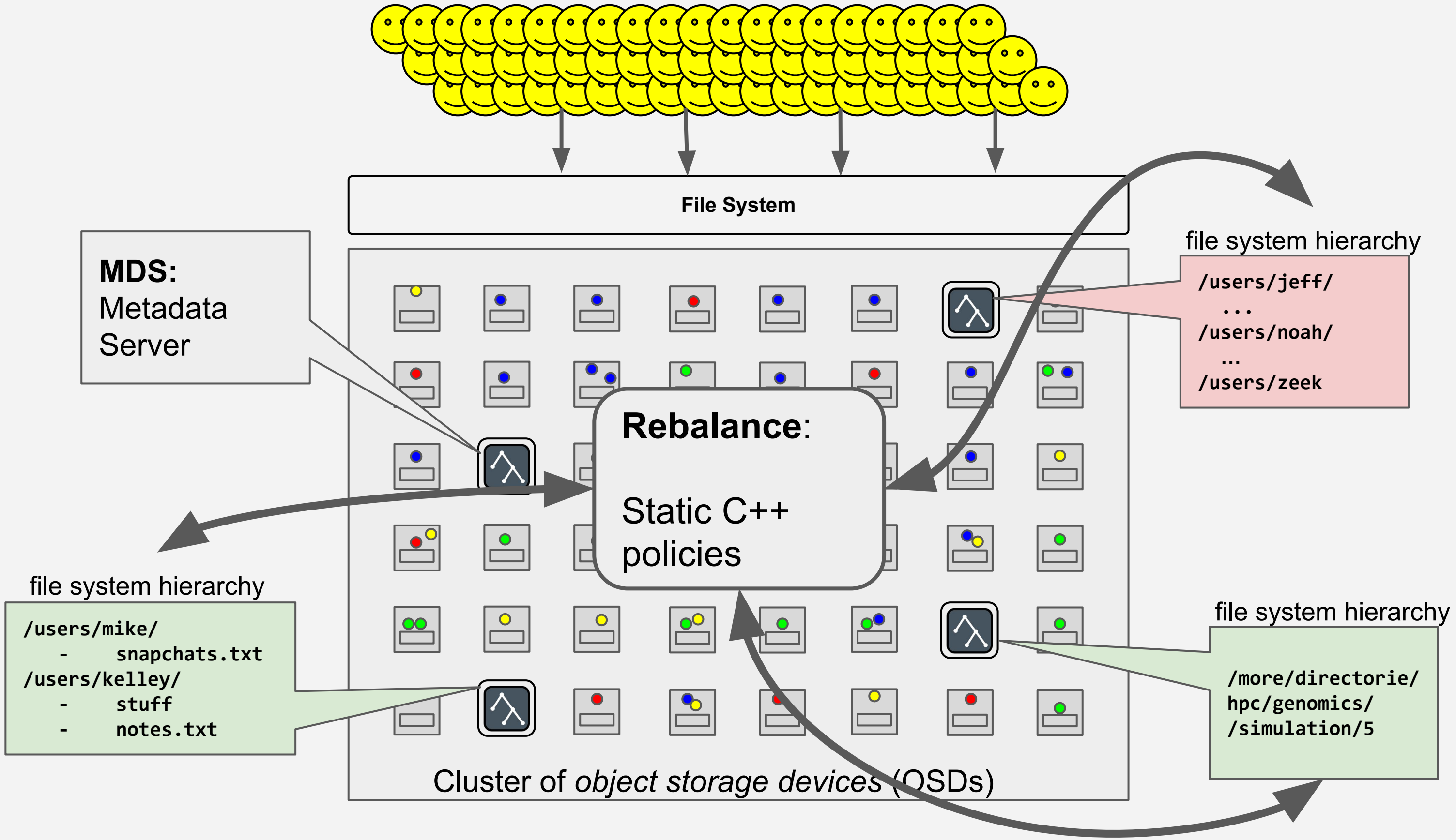
Ceph file system metadata load balancing



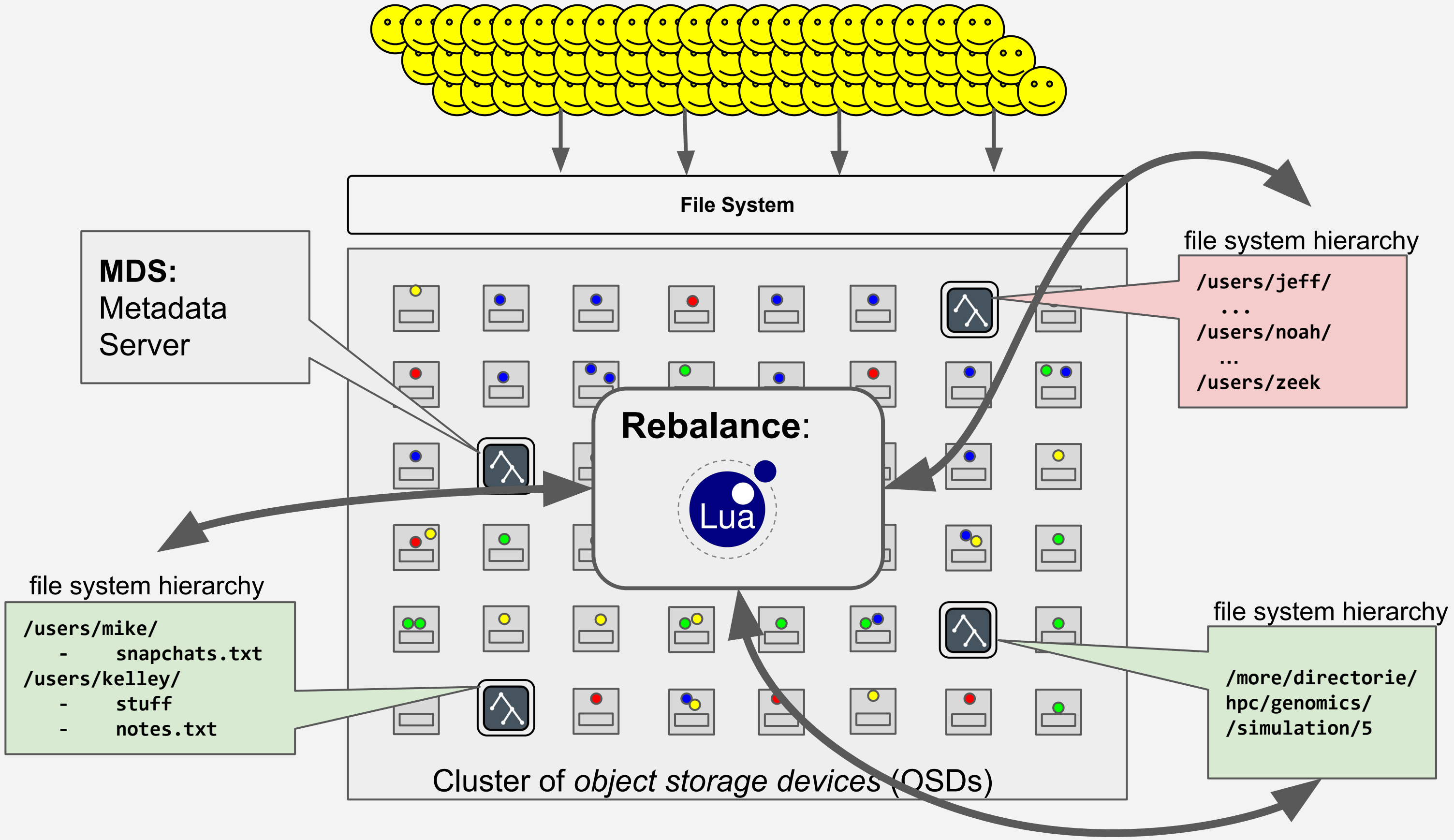
Ceph file system metadata load balancing



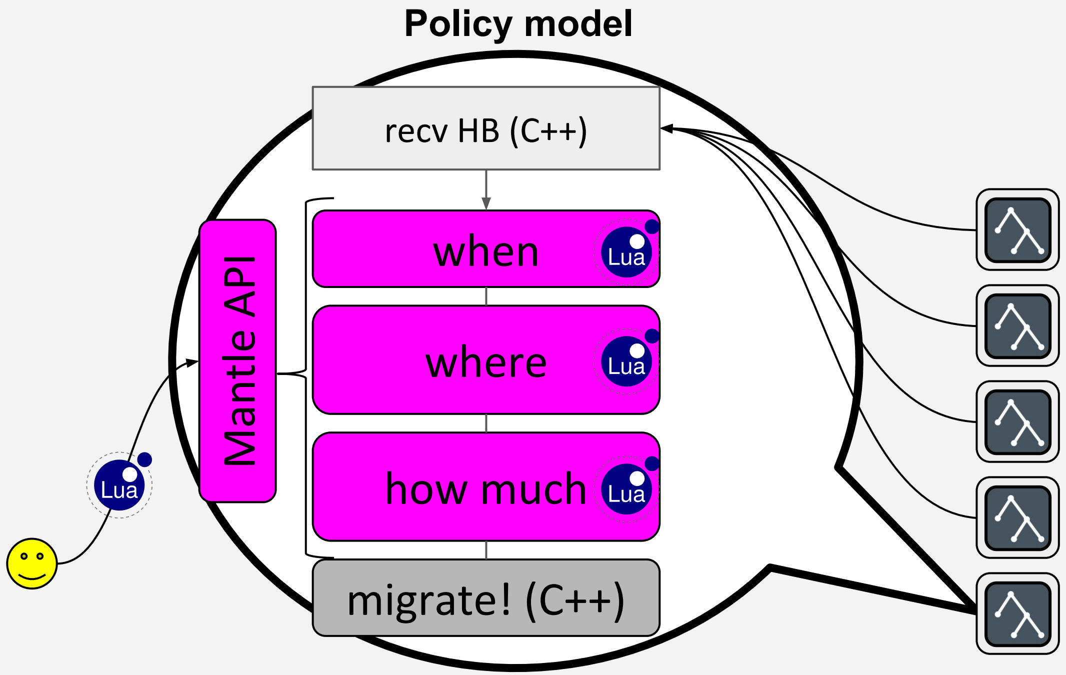
Ceph file system metadata load balancing



Ceph file system metadata load balancing



Mantle load balancing policy example



Example

```
local function when()
      if    server[whoami]['load']>0.1 and
            server[whoami + 1]['load']<0.1 then
            return true
      end
      return false
end
```

EXAMPLE!

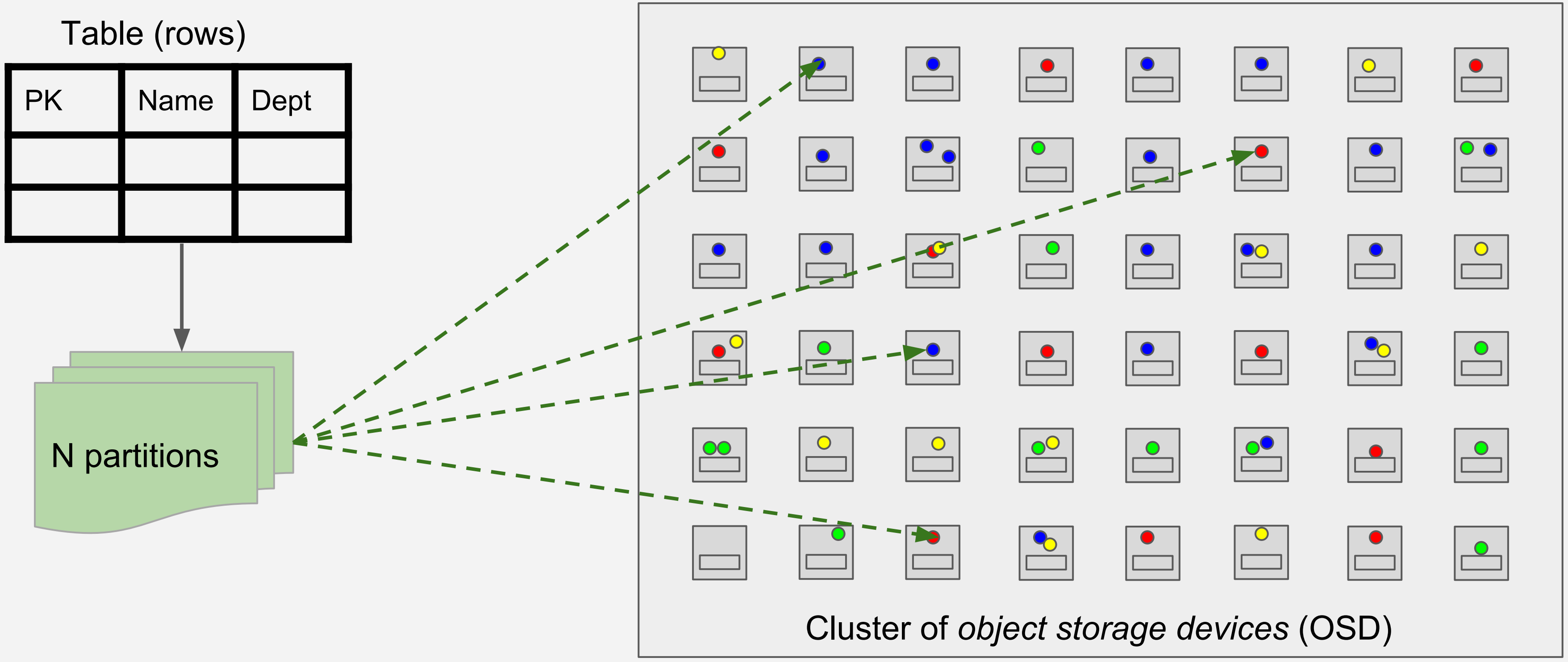
Lua in your databases

SkyhookDB: leveraging programmable storage toward DB elasticity

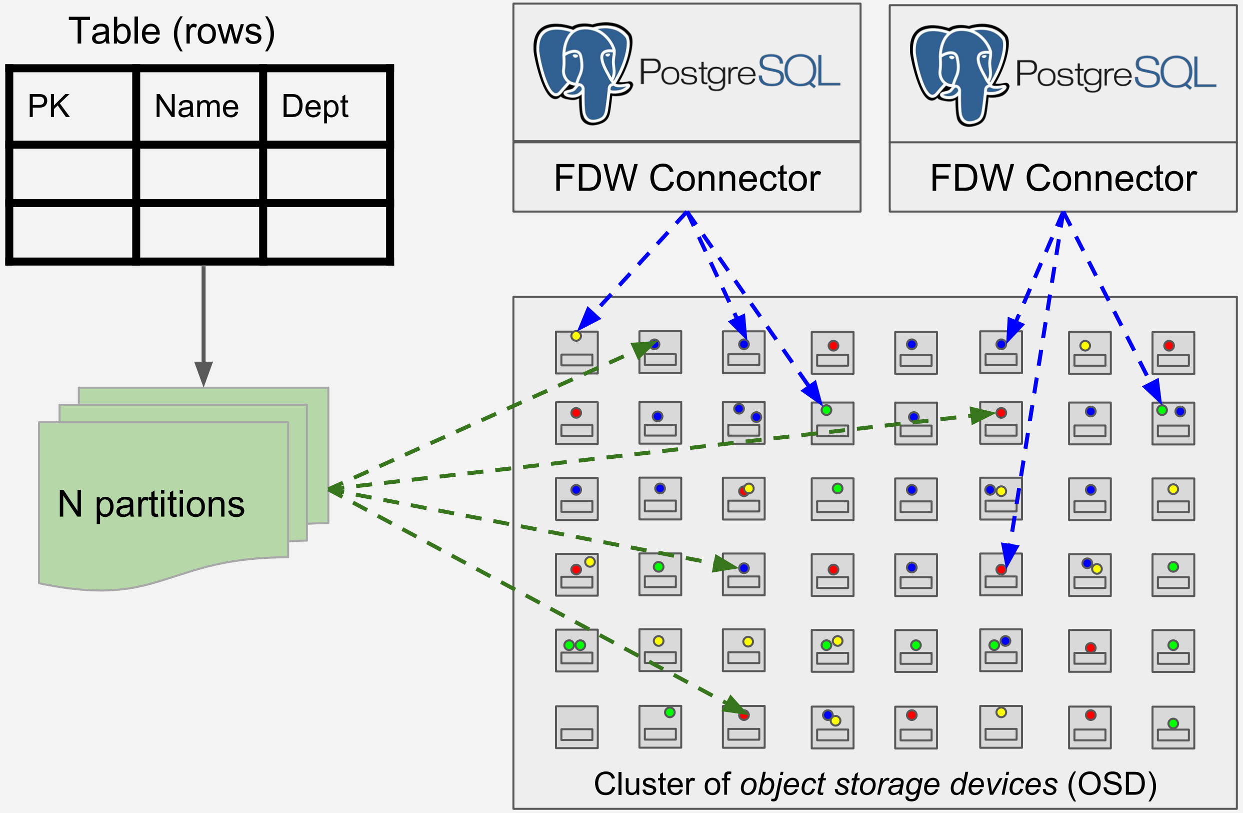
Jeff LeFevre and Noah Watkins

- https://sites.google.com/site/skyhookdb/
- https://cross.ucsc.edu
- jlefevre@ucsc.edu

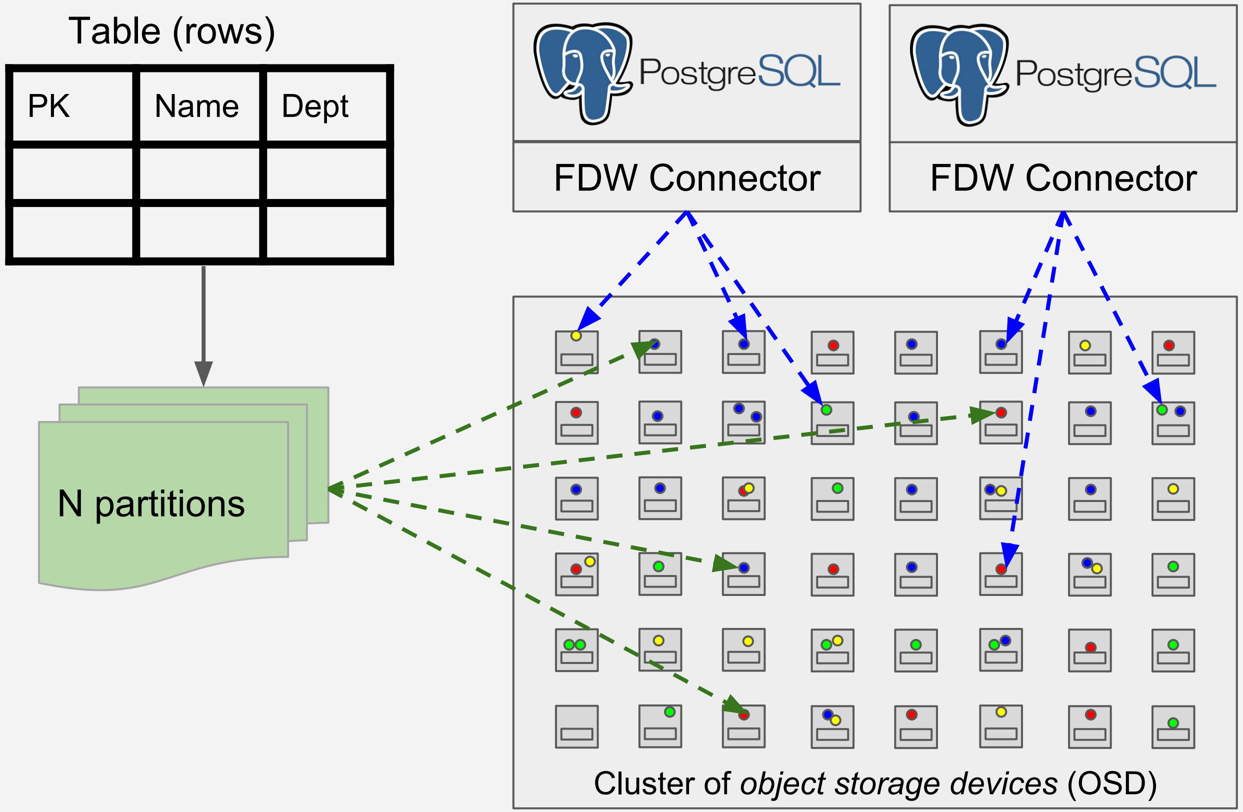
Skyhook Storage Engine (Database System)



Skyhook Storage Engine (Database System)

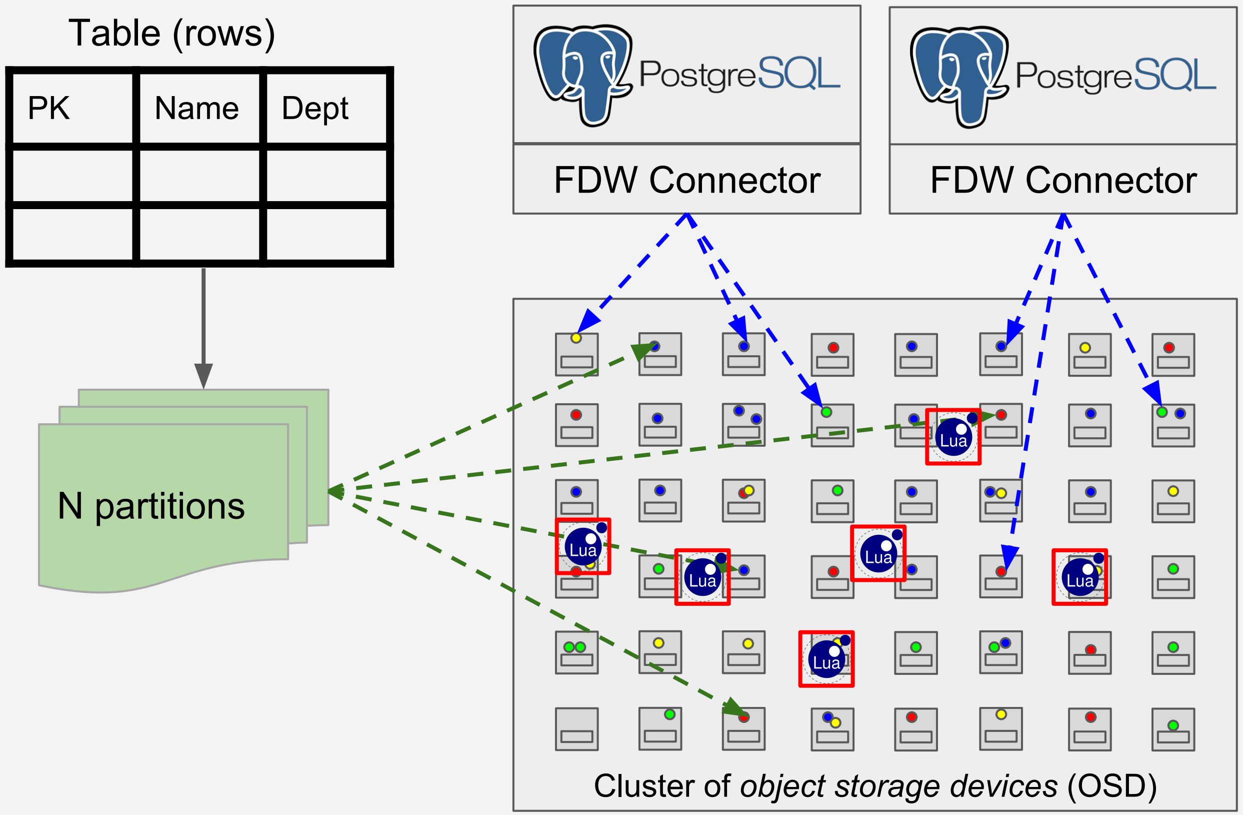


Skyhook Storage Engine (Database System)



- Scale-out
    - CPU
    - I/O
- Push-downs on steroids
- Predicate evaluation
- Regex search
- Projection

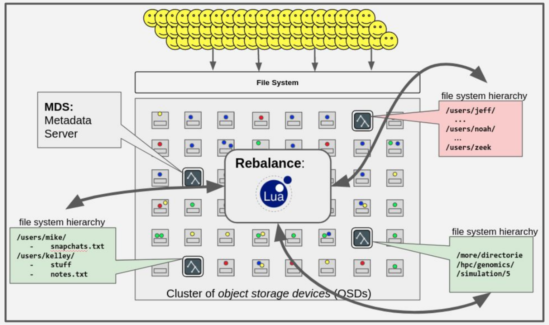
Skyhook Storage Engine (Database System)

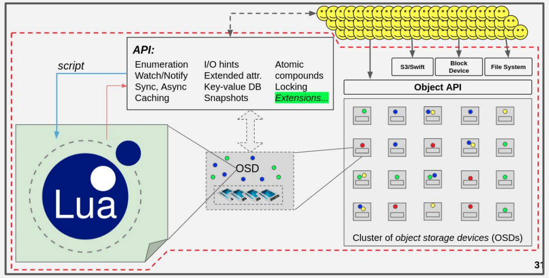


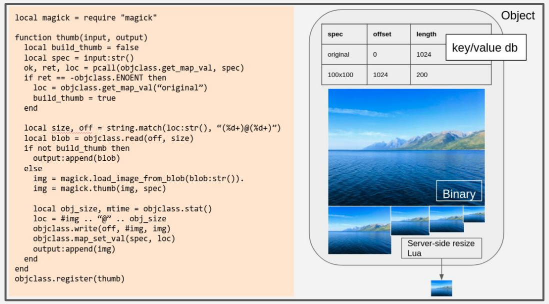
- Scale-out
  - CPU
  - I/O
- Push-downs on steroids
- Predicate evaluation
- Regex search
- Projection

Recapping

- Ceph distributed storage system
  - Massively scalable
  - Object storage system
- Embedding Lua
  - Flexible policy injection
  - Application I/O interfaces
  - Remote computation
- Resources
  - https://github.com/ceph/ceph
  - https://ceph.com/rados/dynamic-object-interfaces-with-lua/
    - Basic tutorial







```
local magick = require "magick"

function thumb(input, output)
  local build_thumb = false
  local spec = input:str()
  ok, ret, loc = pcall(objclass.get_map_val, spec)
  if ret == -objclass.ENOENT then
    loc = objclass.get_map_val(“original”)
    build_thumb = true
  end

  local size, off = string.match(loc:str(), “(%d+)@(%d+)”)
  local blob = objclass.read(off, size)
  if not build_thumb then
    output:append(blob)
  else
    img = magick.load_image_from_blob(blob:str()).
    img = magick.thumb(img, spec)

    local obj_size, mtime = objclass.stat()
    loc = #img .. “@” .. obj_size
    objclass.write(off, #img, img)
    objclass.map_set_val(spec, loc)
    output:append(img)
  end
end
objclass.register(thumb)
```

spec	offset	length
original	0	1024
100x100	1024	200

What we’d like to do next

- We aren’t Lua or programming language experts
  - Our APIs are simple and mimic the C APIs
  - What are better interfaces for our use cases?
- Script and dependency management (big issue)
  - Currently
    - Scripts sent with request
    - Dependencies at host level
  - Ideas
    - Install scripts “into” the cluster
    - Caching VMs to reduce latency

Thanks. And questions?

Noah Watkins
noahwatkins@gmail.com
@noahdesu

Michael Sevilla
mikesevilla3@gmail.com

Lua Workshop, October 2017, San Francisco