

# Skyhook:

Programmable storage for databases

*Vault'19*

*Facebook, April 2019*

**Jeff LeFevre** [jlefevre@ucsc.edu](mailto:jlefevre@ucsc.edu), Noah Watkins [nwatkins@redhat.com](mailto:nwatkins@redhat.com),  
Michael Sevilla [msevilla@ucsc.edu](mailto:msevilla@ucsc.edu), Carlos Maltzahn [carlosm@ucsc.edu](mailto:carlosm@ucsc.edu)

- Bridges gap between student research & open source projects
- Funded by Sage Weil endowment & corporate memberships
- Goals
  - Leverage OSS culture in university research
  - Incubate work beyond graduation to reach critical mass
- [cross.ucsc.edu](http://cross.ucsc.edu)



# What is programmable storage?

- For Skyhook - **Pushdown** some data management tasks into the storage layer
  - Transformations (process/format data)
  - Indexing, statistics, re-sharding
- Skyhook uses Ceph object storage
  - Open source, extensible, originated at UCSC
- See Programmability.us for more info

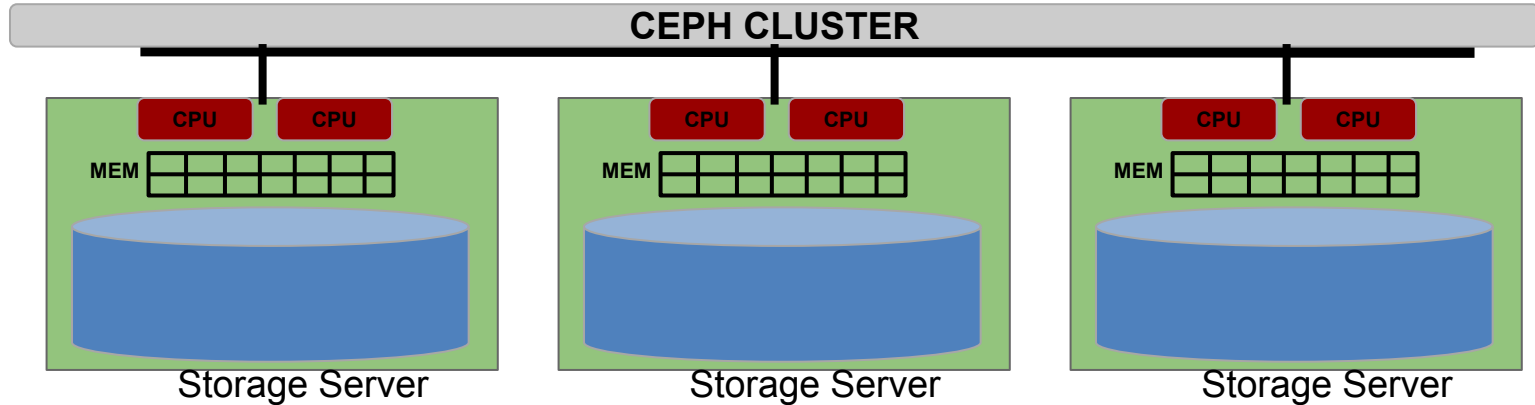
# Ceph Distributed Object Storage

- Distributed, scalable, fault-tolerant
  - Widely available in the cloud
- Objects are the core entity (read/write/replicate)
  - Other API wrappers on top: file, block, S3
- LIBRADOS object library
  - Users can interact directly with objects
  - Create user-defined object classes (read/write)

# Ceph Distributed Object Storage

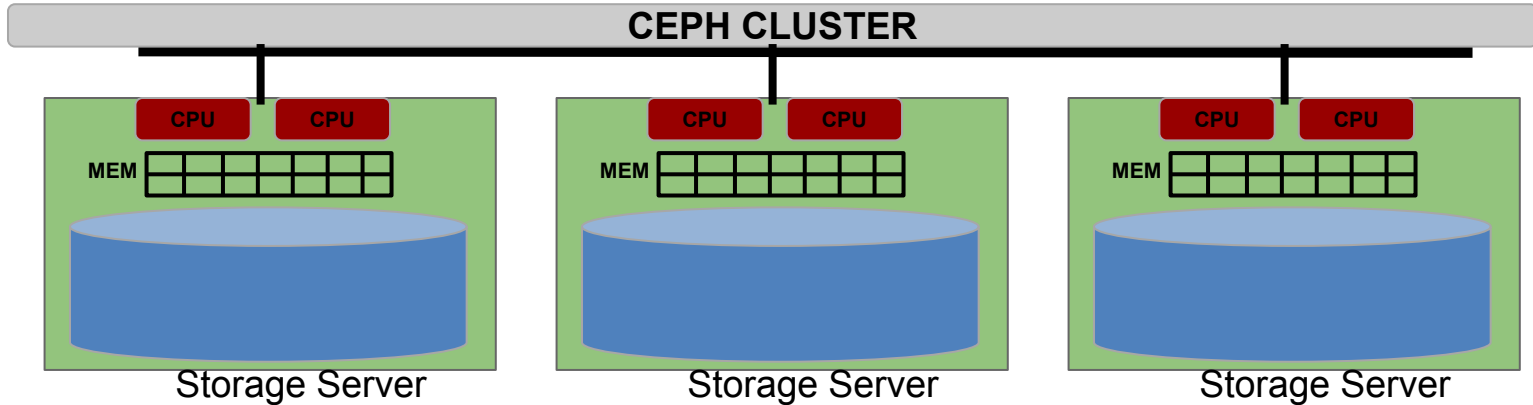
- Distributed, scalable, fault-tolerant
  - Widely available in the cloud
- Objects are the core entity (read/write/replicate)
  - Other API wrappers on top: file, block, S3
- LIBRADOS object library
  - Users can interact directly with objects
  - Create user-defined object classes (read/write)

# Data Storage in Ceph Cluster

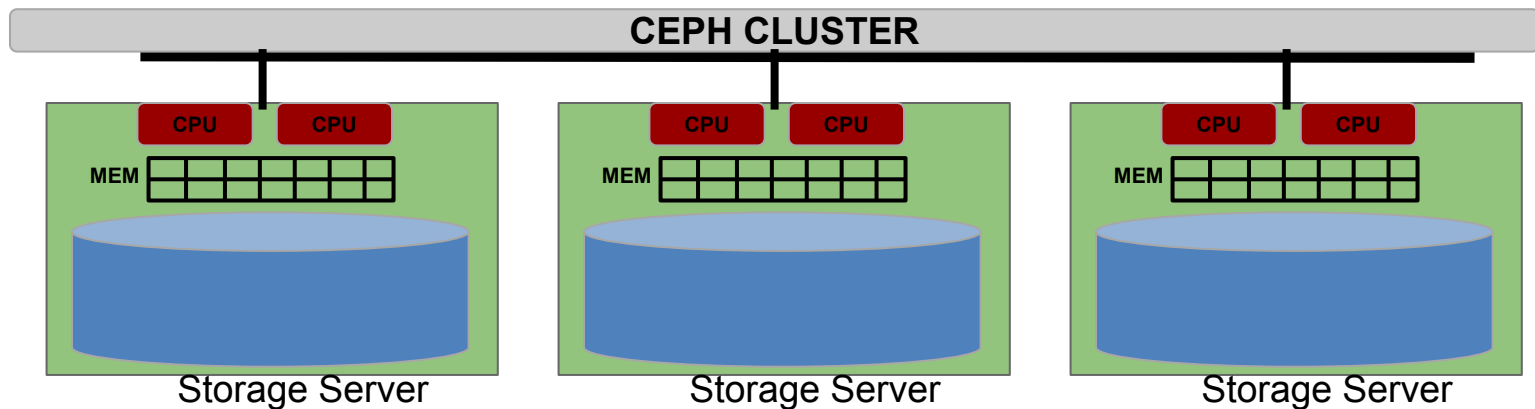
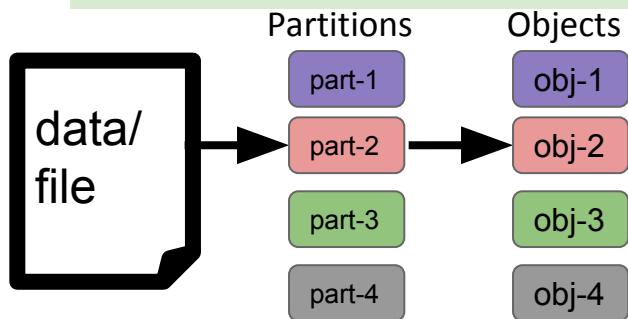


# Data Storage in Ceph Cluster

Partitions

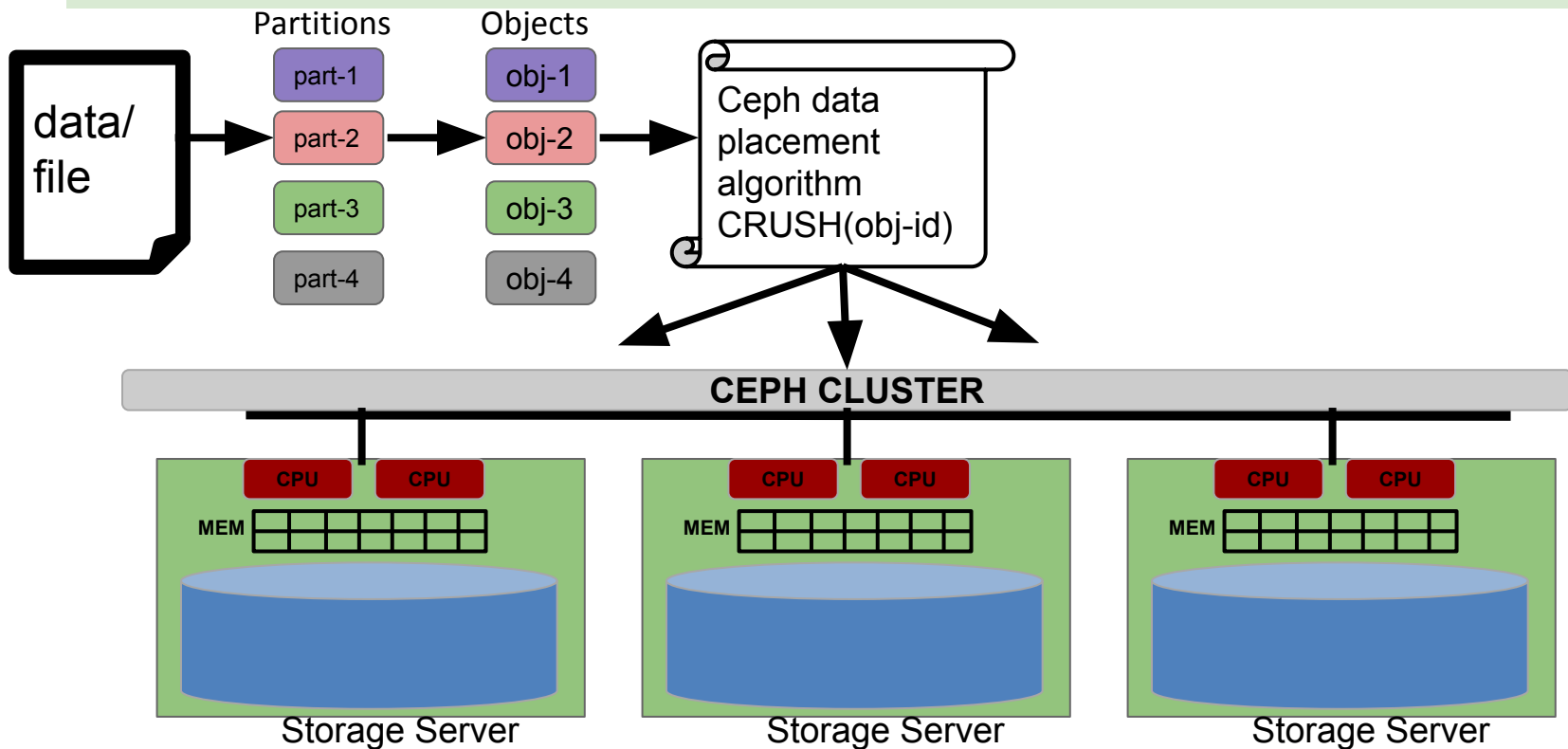


# Data Storage in Ceph Cluster

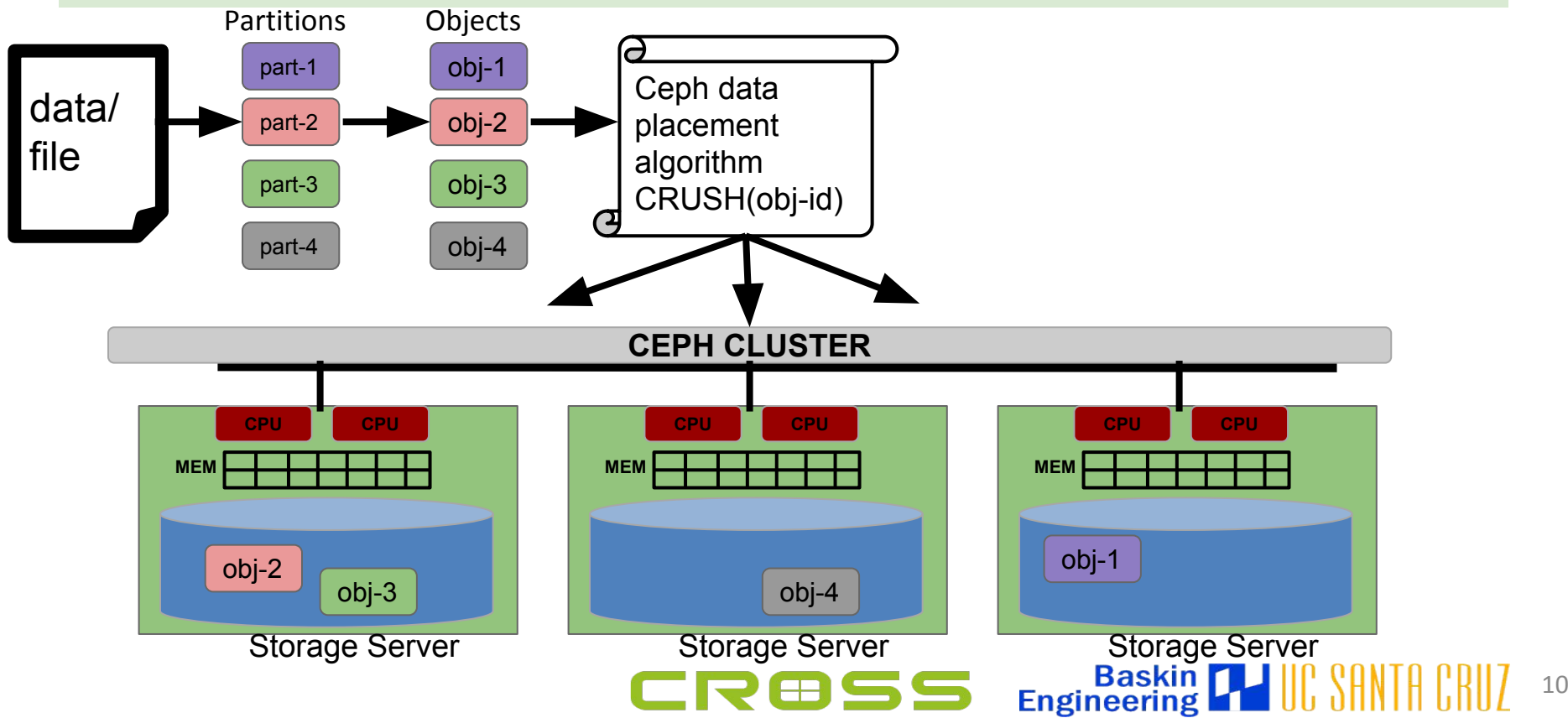




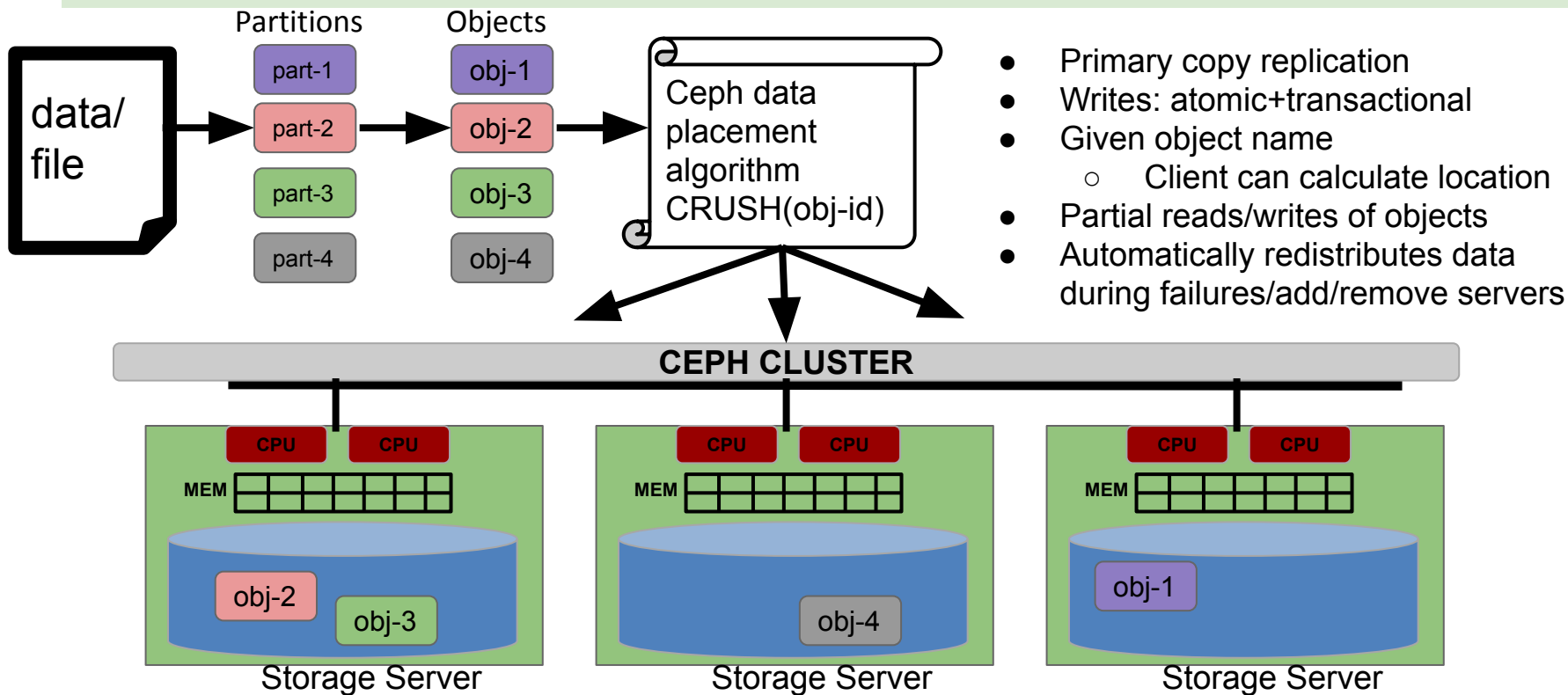
# Data Storage in Ceph Cluster



# Data Storage in Ceph Cluster



# Data Storage in Ceph Cluster



- Primary copy replication
- Writes: atomic+transactional
- Given object name
  - Client can calculate location
- Partial reads/writes of objects
- Automatically redistributes data during failures/add/remove servers

# How does this help us?

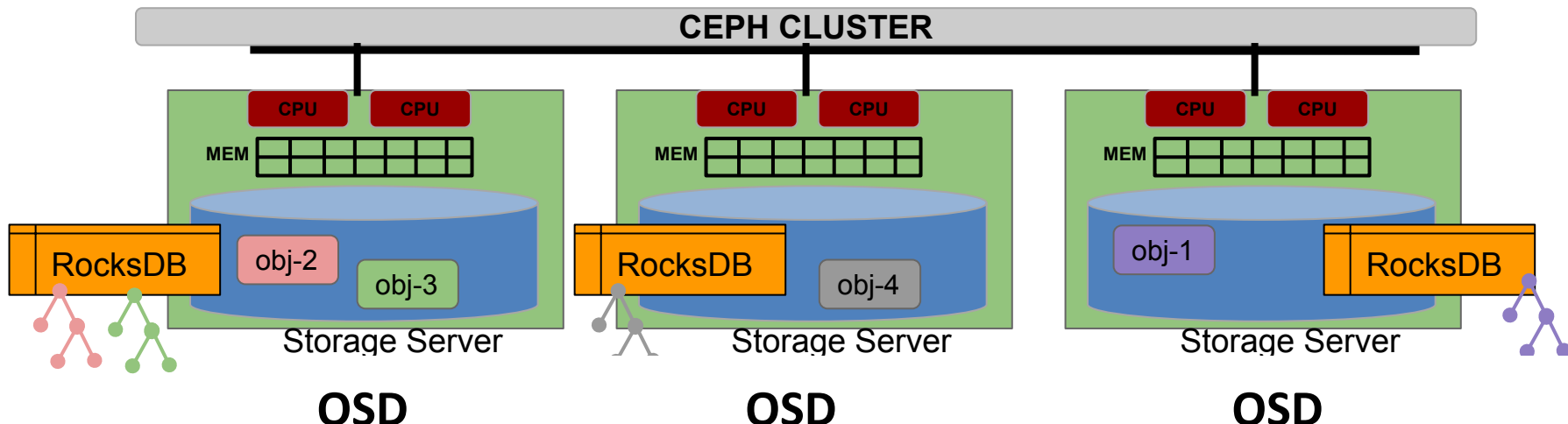
- Ceph storage provides transparent
  - Data distribution/scaling
  - Fault tolerance/recovery
- Remote processing on storage servers
  - **via custom object classes**
- Query-able metadata on storage servers
  - **via local indexing mechanism (RocksDB)**

# How does this help us?

- Ceph storage provides transparent
  - Data distribution/scaling
  - Fault tolerance/recovery
- Remote processing on storage servers
  - **via custom object classes**
- Query-able metadata on storage servers
  - **via local indexing mechanism (RocksDB)**

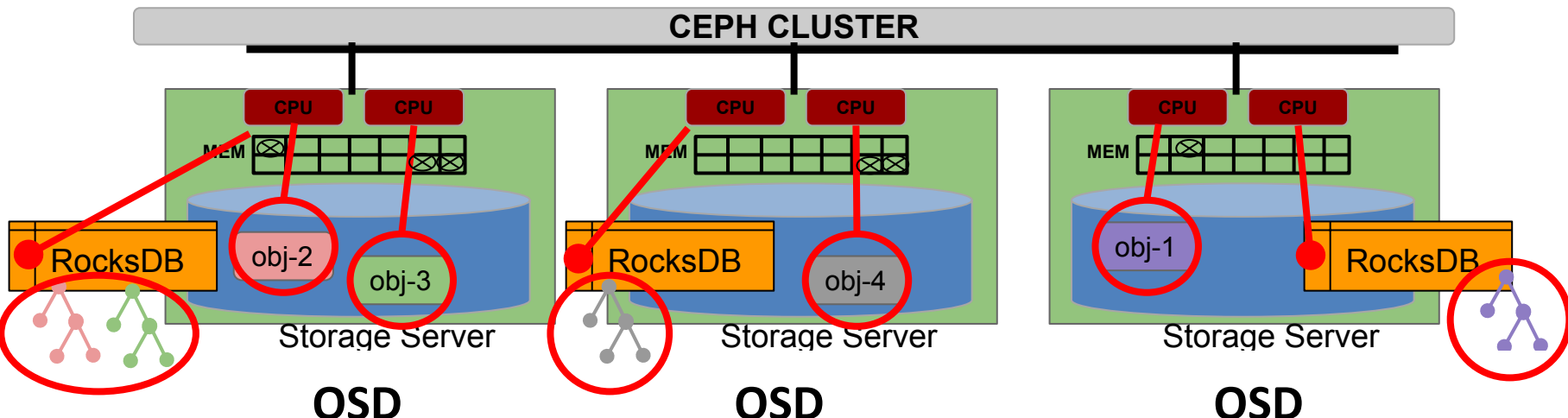
# Remote Processing + Indexing

- Execution of custom object classes and indexing is performed by Ceph storage servers (OSDs)



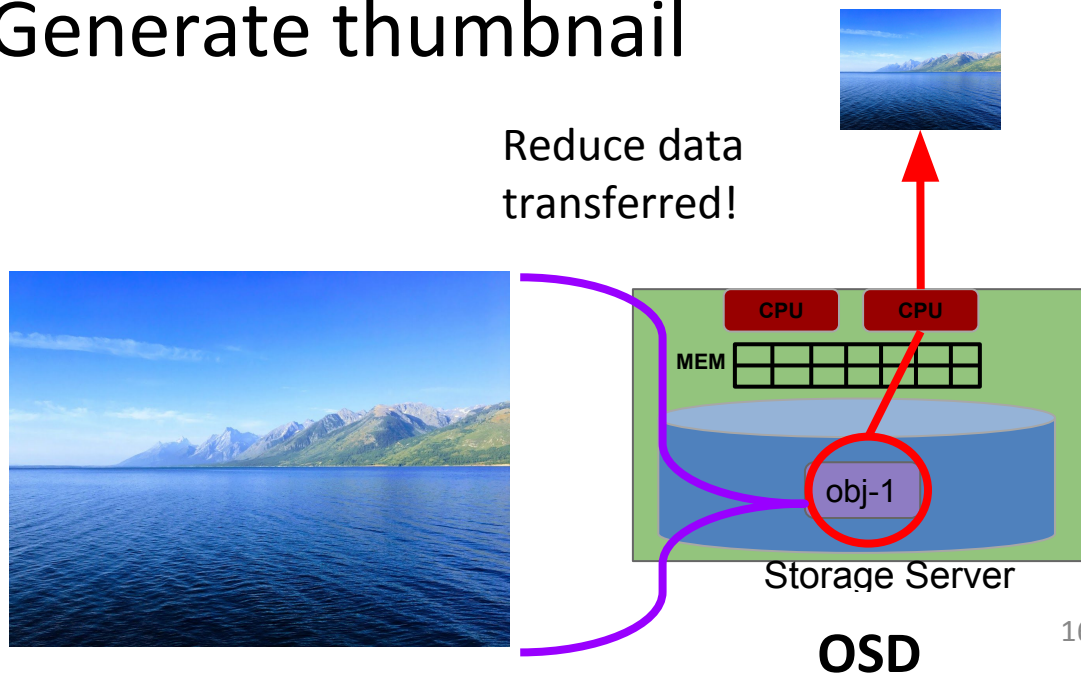
# Remote Processing + Indexing

- Execution of custom object classes and indexing is performed by Ceph storage servers (OSDs)
- Utilizes remote resources



# Remote Processing Example

- Full size image stored in object
  - Custom read: Generate thumbnail
- Other examples
  - Checksum
  - Filter/Regex
  - Aggregate
  - Transform
  - Reorg data





# Ceph Custom Object Classes (CLS)

## C++ interface

```
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;
}
```

## Lua interface

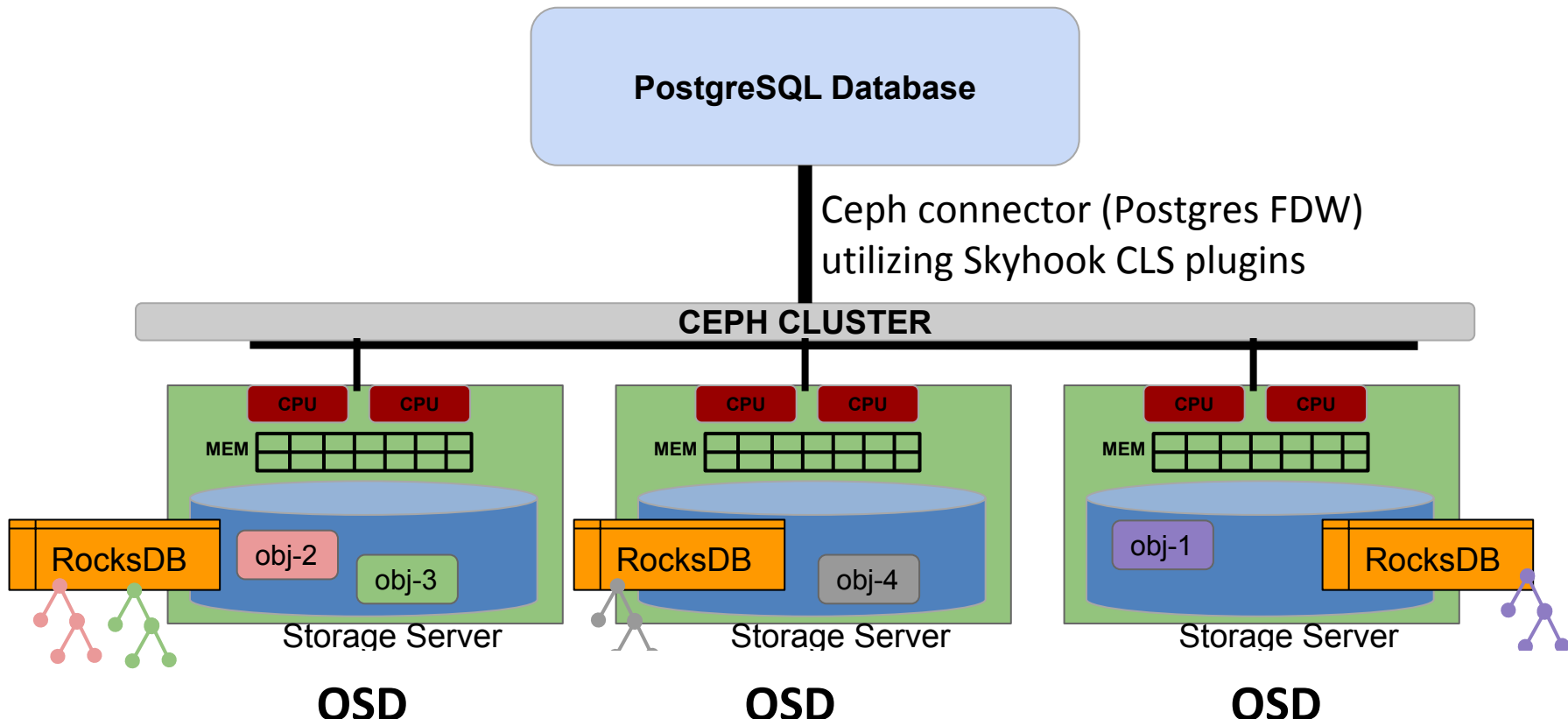
```
local md5 = require 'md5'

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

# Putting it All Together - Skyhook

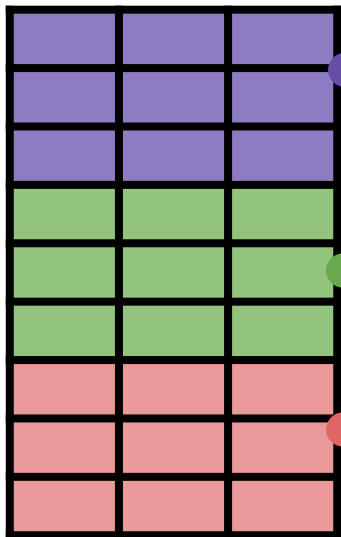
- ***Data partitioning***
  - Physical data layout and format
- ***Remote processing***
  - Custom object classes
- ***Remote indexing***
  - Query-able metadata (data vals, stats)

# Skyhook Architecture



# Skyhook - *Data partitioning + format*

Table data



Row partitions\*

part-1

Formatted data\*\*

Obj-1

part-2

Obj-2

part-3

Obj-3

1:1

- Format retains data's semantics (table schema)
- Object names are generated
- Objects are distributed by Ceph based on name
- Object location not stored by Skyhook

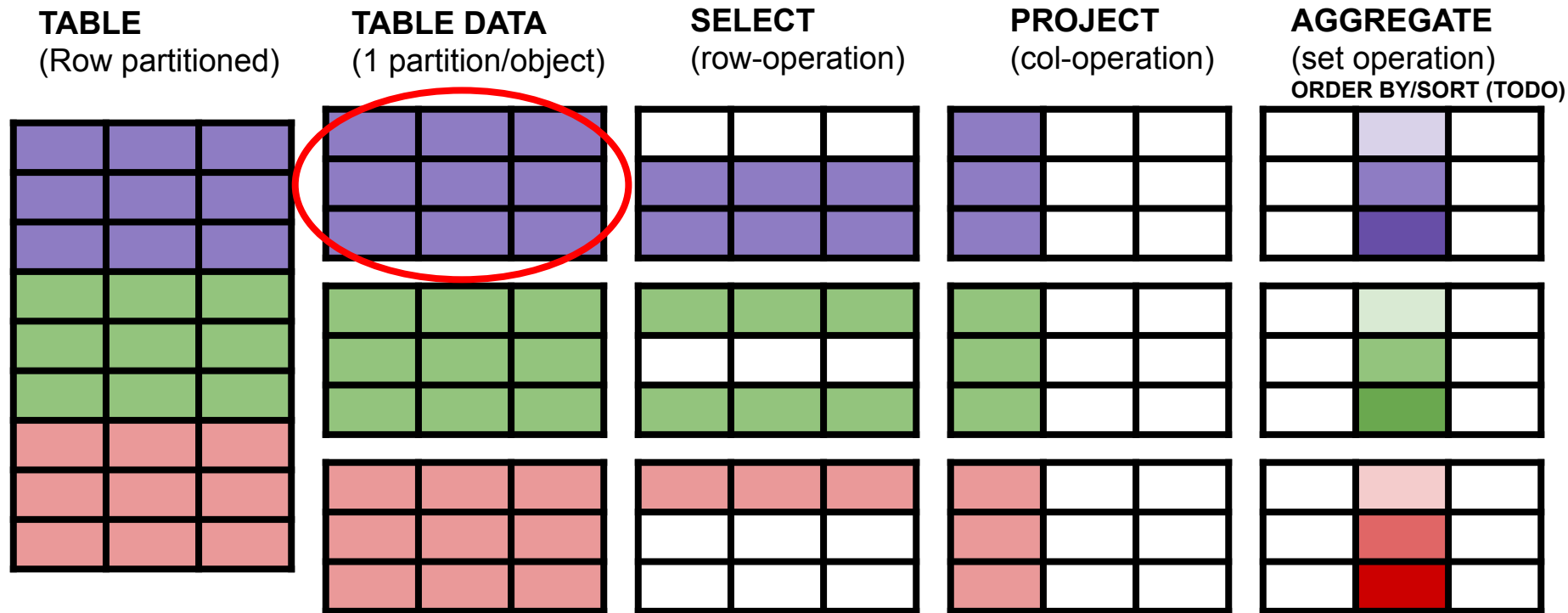
\*Partition rows with  
[JumpConsistentHash](#)

\*\*Partitions formatted as  
[Google Flatbuffers](#)

# Skyhook - *Remote Processing*

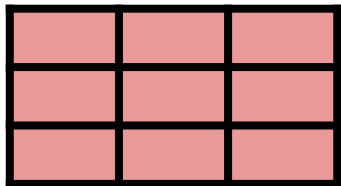
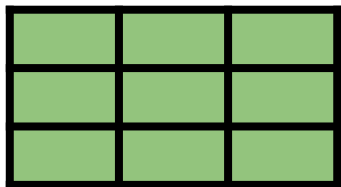
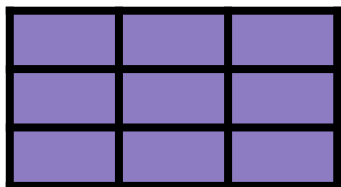


# Skyhook - *Remote Processing*

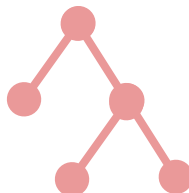
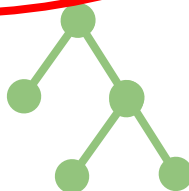
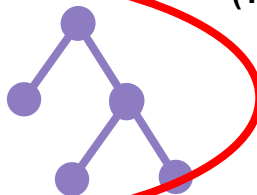
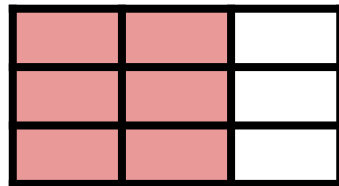
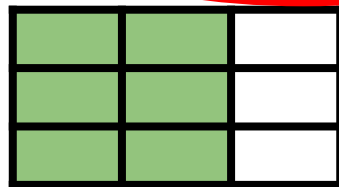
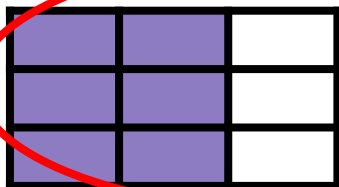


# Skyhook - *Remote Indexing*

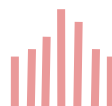
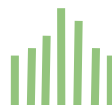
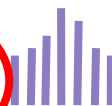
**TABLE DATA**  
(1 partition/object)



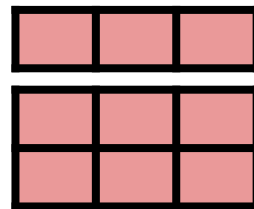
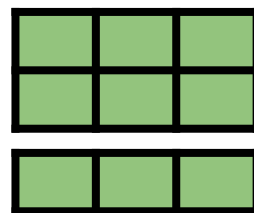
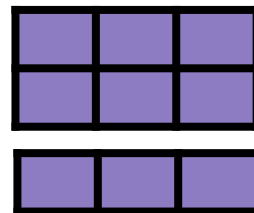
**CREATE LOCAL INDEX on**  
**Subset of columns**



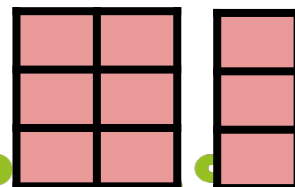
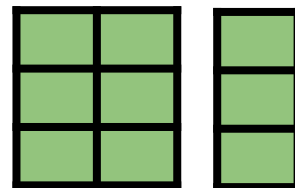
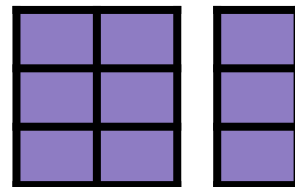
**COMPUTE**  
**STATS**  
(TODO)



**SPLIT**  
(TODO)



**REFORMAT**  
(TODO)



**RESULTS**

# Skyhook - Ceph Extensions developed

- Custom object classes ([cls](#))
  - SELECT (SELECT \* from T WHERE a>5)
  - PROJECT (SELECT a,b from T)
  - AGGREGATE (SELECT min(a)from T WHERE b>5)
- Query-able Metadata
  - index(a), index(a,b,...), stats(a), min(a), count(a)



# Skyhook - Ceph Extensions developed

- Custom object classes ([cls](#))
  - SELECT (SELECT \* from T WHERE a>5)
  - PROJECT (SELECT a,b from T)
  - AGGREGATE (SELECT min(a) from T WHERE b>5)
- Query-able Metadata
  - index(a), index(a,b,...), stats(a), min(a), count(a)

# Experimental Results

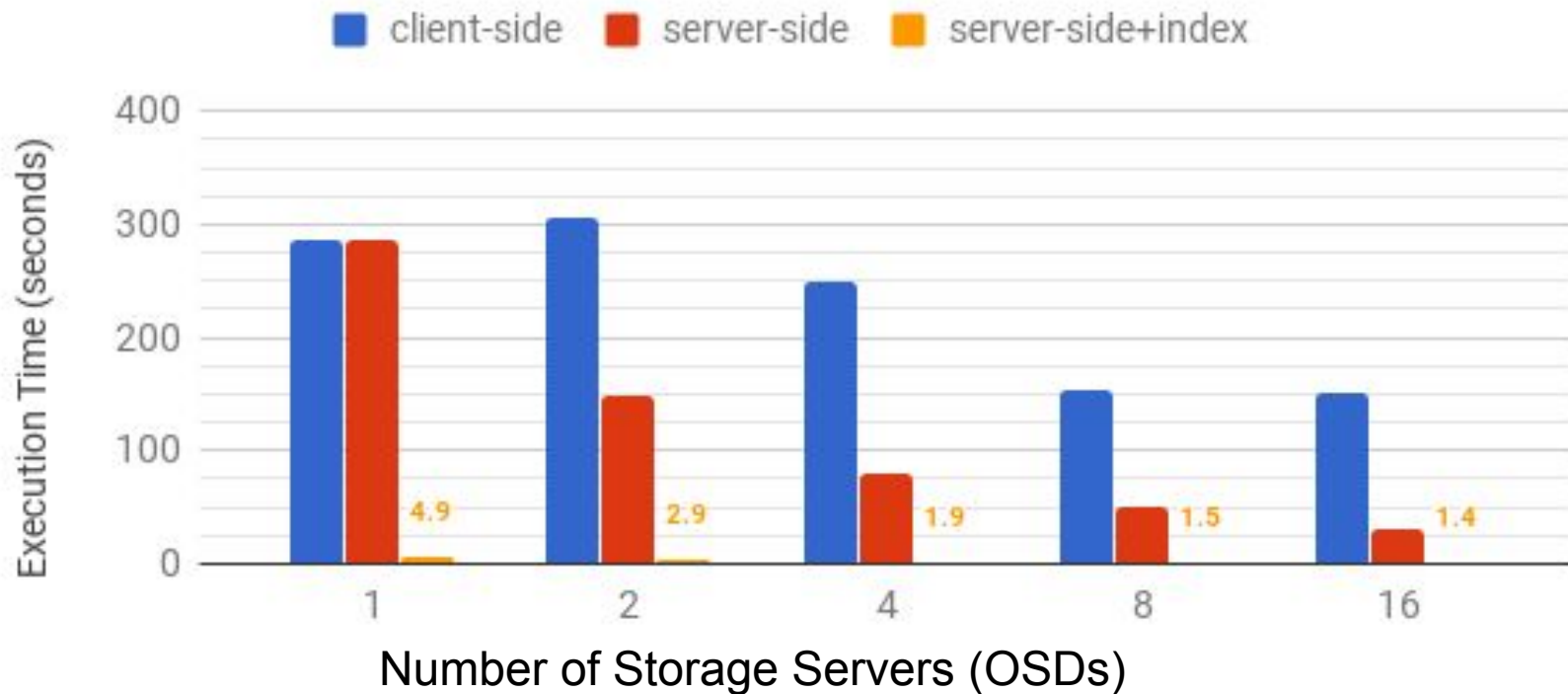
- Dataset: TPC-H lineitem table, 1 billion rows, ~140GB
- Objects: 10K objects ~14MB each (each with local index)
- Queries: Point, range
- Machines: 1 Client node; 1--16 Storage nodes
  - CPU=20 cores (Xeon), MEM=160GB, Net=10GbE, Intel SSDs
- Compare (report average of 3 runs each experiment)
  - Skyhook approach (query processing done in storage servers)
  - Standard approach (query processing done by client/database)<sup>26</sup>

# Experimental Results

- Dataset: TPC-H lineitem table, 1 billion rows, ~140GB
- Objects: 10K objects ~14MB each (each with local index)
- Queries: Point, range, regex
- Machines: 1 Client node; 1--16 Storage nodes
  - CPU=20 cores (Xeon), MEM=160GB, Net=10GbE, Intel SSDs
- Compare (report average of 3 runs each experiment)
  - Skyhook approach (query processing done in storage servers)
  - Standard approach (query processing done by client/database)<sup>27</sup>

# Point Query (single matching row)

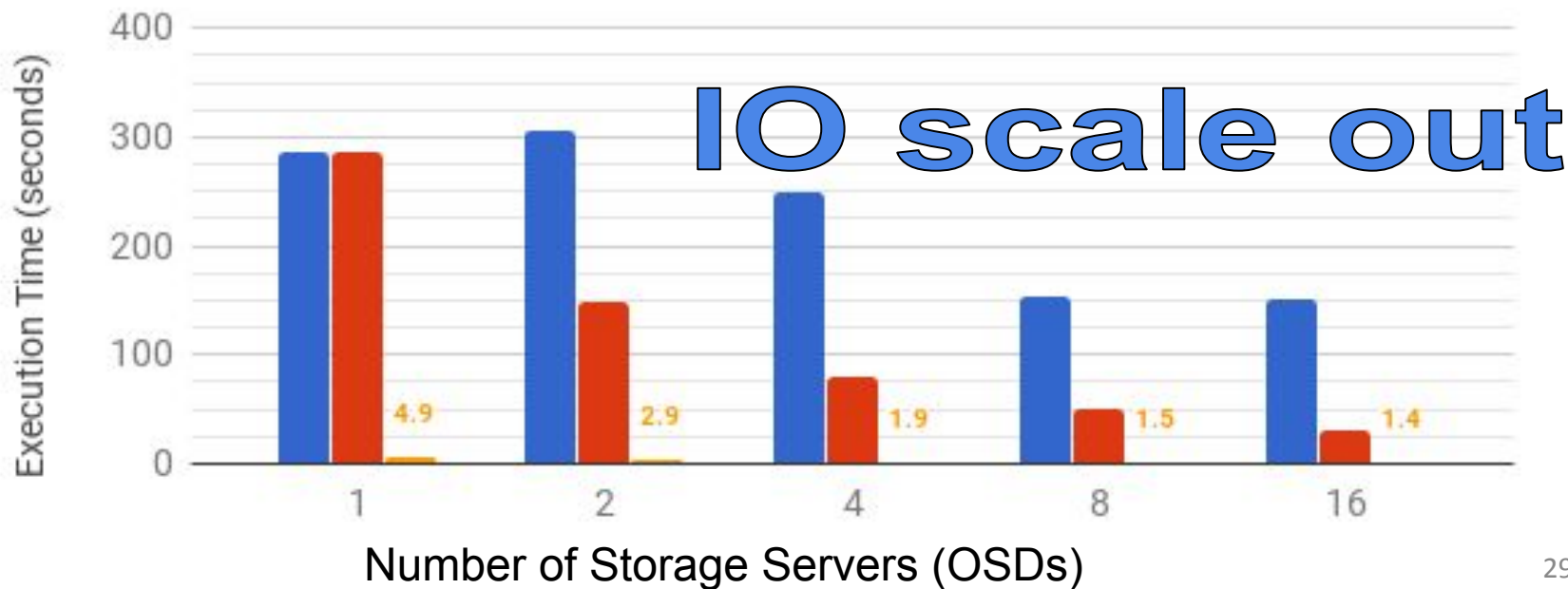
SELECT \* WHERE l\_orderkey=5 and l\_linenumber=3



# Point Query (single matching row)

SELECT \* WHERE l\_orderkey=5 and l\_linenumber=3

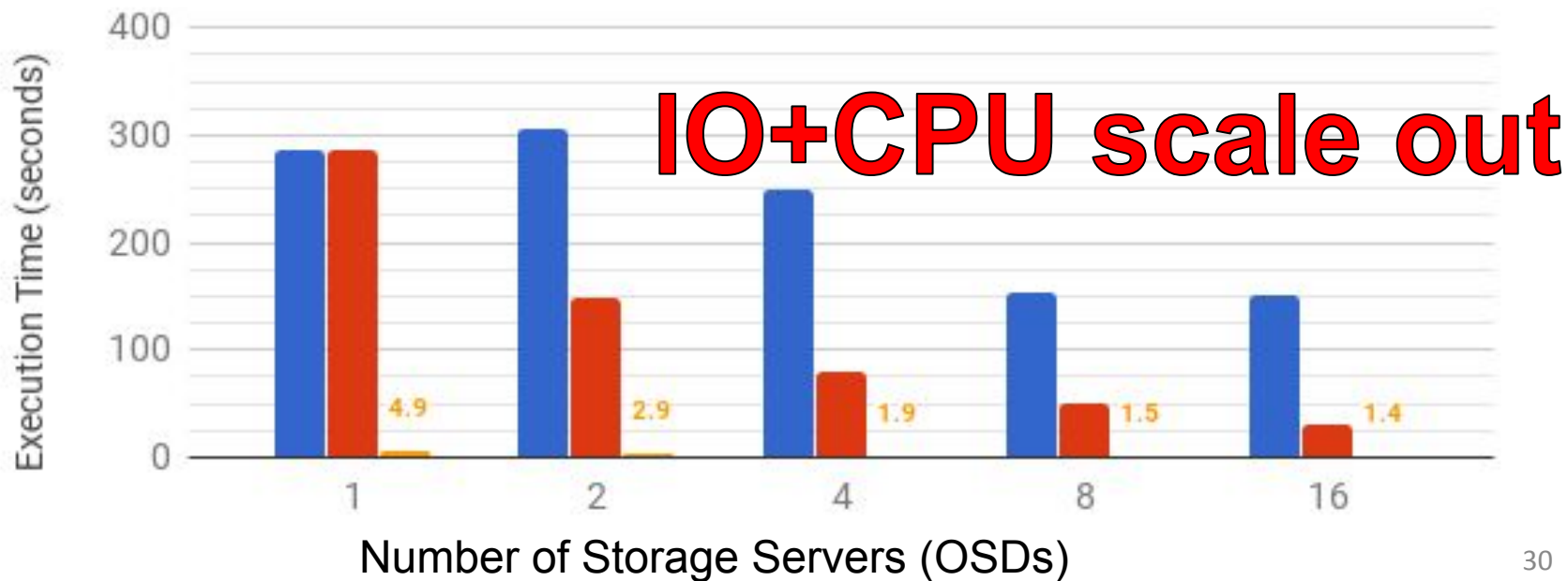
client-side server-side server-side+index



# Point Query (single matching row)

SELECT \* WHERE l\_orderkey=5 and l\_linenumber=3

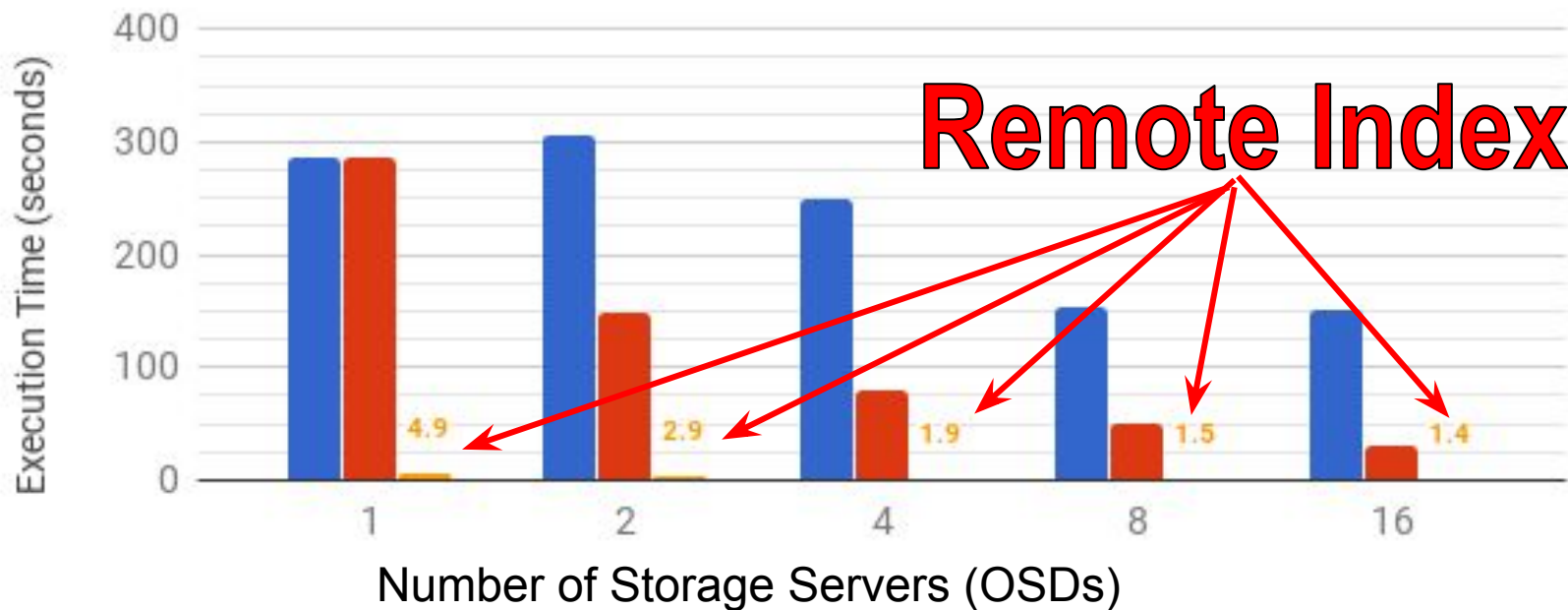
client-side server-side server-side+index



# Point Query (single matching row)

SELECT \* WHERE l\_orderkey=5 and l\_linenumber=3

client-side server-side server-side+index



# Range Query (selectivity=10%)

SELECT \* WHERE `l_extendedprice > 71000`



Number of Storage Servers (OSDs)

- Remote processing done:
  - Integer comparison**

SELECT \* WHERE `comment like %uriously%`



Number of Storage Servers (OSDs)

- Remote processing done:
  - Regex on text**
  - Others?**



# Skyhook - Key Takeaways

- Store data partitions in objects, *retain the data's semantics*
  - For RDBMS - a subset of a table (row/col partition)
- Utilize remote resources on storage servers
  - Ceph's custom object classes (read/write)
  - Ceph's local indexing mechanism (RocksDB)

# Thank You

Funding provided by

- NSF-OAC-1836650, NSF-CNS-1764102, NSF-CNS-1705021, DOE DE-SC0016074
- Center for Research in Open Source Software

More info:

- <https://cross.ucsc.edu>
- [SkyhookDM.com](https://skyhookdm.com)

# References (1)

[Oracle 2016a] [Oracle database appliance](#), Nov 2016.

[Oracle 2012] [A Technical overview of the Oracle Exadata database machine and Exadata Storage Server](#), Jun 2012.

[Oracle 2016b] [Oracle database appliance evolves as an enterprise private cloud building block](#), Jan 2016.

[Kee 1998] K. Keeton, D.Patterson, J.Hellerstein, *A case for intelligent disks (iDISKS)*, SIGMOD Record, Sep 1998.

[Bor 1983] H.Boral, D.DeWitt, *Database machines, an idea whose time has passed?*, Workshop on Database Machines 1983.

[Woo 2014] L. Woods, Z.Istvan, G.Alonso, *Ibex: an intelligent storage engine with support for advanced SQL offloading*, VLDB 2014.

[Teradata 2016] Marketing statement: “*Multi-dimensional scalability through independent scaling of processing power and storage capacity via our next generation massively parallel processing (MPP) architecture*”

[Sevilla+Watkins 2017] M. Sevilla, N. Watkins, I. Jimenez, P. Alvaro, S. Finkelstein, J. LeFevre, C. Maltzahn, “*Malacology: A Programmable Storage System*”, in [EuroSys 2017](#).

[LeFevre 2014] J. LeFevre, J. Sankaranarayanan, H. Hacigumus, J. Tatemura, N. Polyzotis, M.J. Carey, “*MISO: Souping Up Big Data Query Processing with a Multistore System*”, in [SIGMOD 2014](#).

[LeFevre 2016] J. LeFevre, R. Liu, C. Inigo, M. Castellanos, L. Paz, E. Ma, M. Hsu, “*Building the Enterprise Fabric for Big Data with Vertica and Spark*”, in [SIGMOD 2016](#).

# References (2)

[Anantharayanan 2011] Ganesh Anantharayanan, Ali Ghodsi, Scott Shenker, Ion Stoica, "[\*Disk-Locality in Datacenter Computing Considered Irrelevant\*](#)", in USENIX HotOS, May 2011.

[Dageville 2016] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, et. al, "[\*The Snowflake Elastic Data Warehouse\*](#)", in SIGMOD 2016.

[FlashGrid 2017] White paper on Mission-Critical Databases in the Cloud. "[\*Oracle RAC on Amazon EC2 Enabled by FlashGrid® Software\*](#)". rev. Jan 6, 2017.

[Gupta 2015] Anurag Gupta, Deepak Agarwal, Derek Tan, et.al, "[\*Amazon Redshift and the Case for Simpler Data Warehouses\*](#)", in SIGMOD 2015.

[Srinivasan 2016] Vidhya Srinivasan, "[\*What's New with Amazon Redshift\*](#)", AWS re-invent, Nov 2016.

[Dynamo 2017] Amazon's DynamoDB, <https://aws.amazon.com/dynamodb/>, accessed Jan 2017.

[Brantner 2008] Matthias Brantner, Daniela Florescu, David Graf, Donald Kossmann, Tim Kraska, "[\*Building a database on S3\*](#)", in SIGMOD 2008.

[Shuler 2015] Bill Shuler, "[\*CitusDB Architecture for Real-time Big Data\*](#)", Jun 2015.

[Citusdata 2017] Citus Data, <https://www.citusdata.com>, accessed Jan 2017.

[Pavlo 2016] Andrew Pavlo, Matthew Aslett, "[\*What's Really New with NewSQL?\*](#)", in SIGMOD Record Jun 2016.

# References (3)

[Xi 2015] Sam (Likun) Xi, Oreoluwa Babarinsa, Manos Athanassoulis, Stratos Idreos, "[Beyond the Wall: Near-Data Processing for Databases](#)", in DAMON 2015.

[Kim 2015] Sungchan Kim, Hyunok Ohb, Chanik Parkc, Sangyeun Choc, Sang-Won Leed, Bongki Moone, "[In-storage processing of database scans and joins](#)", Information Sciences, 2015.

[Do 2013] Jaeyoung Do, Yang-Suk Kee, Jignesh M. Patel, Chanik Park, Kwanghyun Park, David J. DeWitt, "[Query Processing on Smart SSDs: Opportunities and Challenges](#)", in SIGMOD 2013.

[Gkantsidis 2013] Christos Gkantsidis, Dimitrios Vytiniotis, Orion Hodson, Dushyanth Narayanan, Florin Dinu, Antony Rowstron, "[Rhea: automatic filtering for unstructured cloud storage](#)", in NSDI 2013.

[Balasubramonian 2014] Rajeev Balasubramonian, Jichuan Chang, Troy Manning, Jaime H. Moreno, Richard Murphy, Ravi Nair, Steven Swanson, "[NEAR-DATA PROCESSING: INSIGHTS FROM A MICRO-46 WORKSHOP](#)", IEEE Micro, Aug 2014.

[Kudu 2017] Apache Kudu, [Kudu: New Apache Hadoop Storage for Fast Analytics on Fast Data](#)

[Kudu 2015] Lipcon et. al "[Kudu: Storage for Fast Analytics on Fast Data](#)", white paper.

[Sevilla 2018] M. Sevilla, R. Nasirigerdeh, C. Maltzahn, **J. LeFevre**, N. Watkins, P. Alvaro, M. Lawson, J. Lofstead, J. Pivarski, "[Tintenfisch: File System Namespace Schemas and Generators](#)", Hot Storage 2018.