

Automated Sharded MongoDB Deployment and Benchmarking for Big Data Analysis

Gregor von Laszewski

Mark McCombe

laszewski@gmail.com

Indiana University

Intelligent Systems Engineering Department

Acknowledgement

- This study has been conducted as part of the I524 class with the topic Big Data and Software Projects
- The class used the following resources
 - Students computers
 - FutureSystems (DSC @ Indiana University) a continuation of the FutureGrid (NSF)
 - Chameleon Cloud (NSF): Project CH-818664, KVM
 - Jetstream (NSF)
- Some students also elected to use
 - AWS
 - Azure
- All resources as far as we know were provided to us for free.

Outline

- Motivation for the project
 - IU educates data scientists
- Sharded Mongo DB deployment
- Benchmarks
- Usage Observations
- Conclusion
 - Why we did not do a large scale study ...
 - Implication for future classes ...

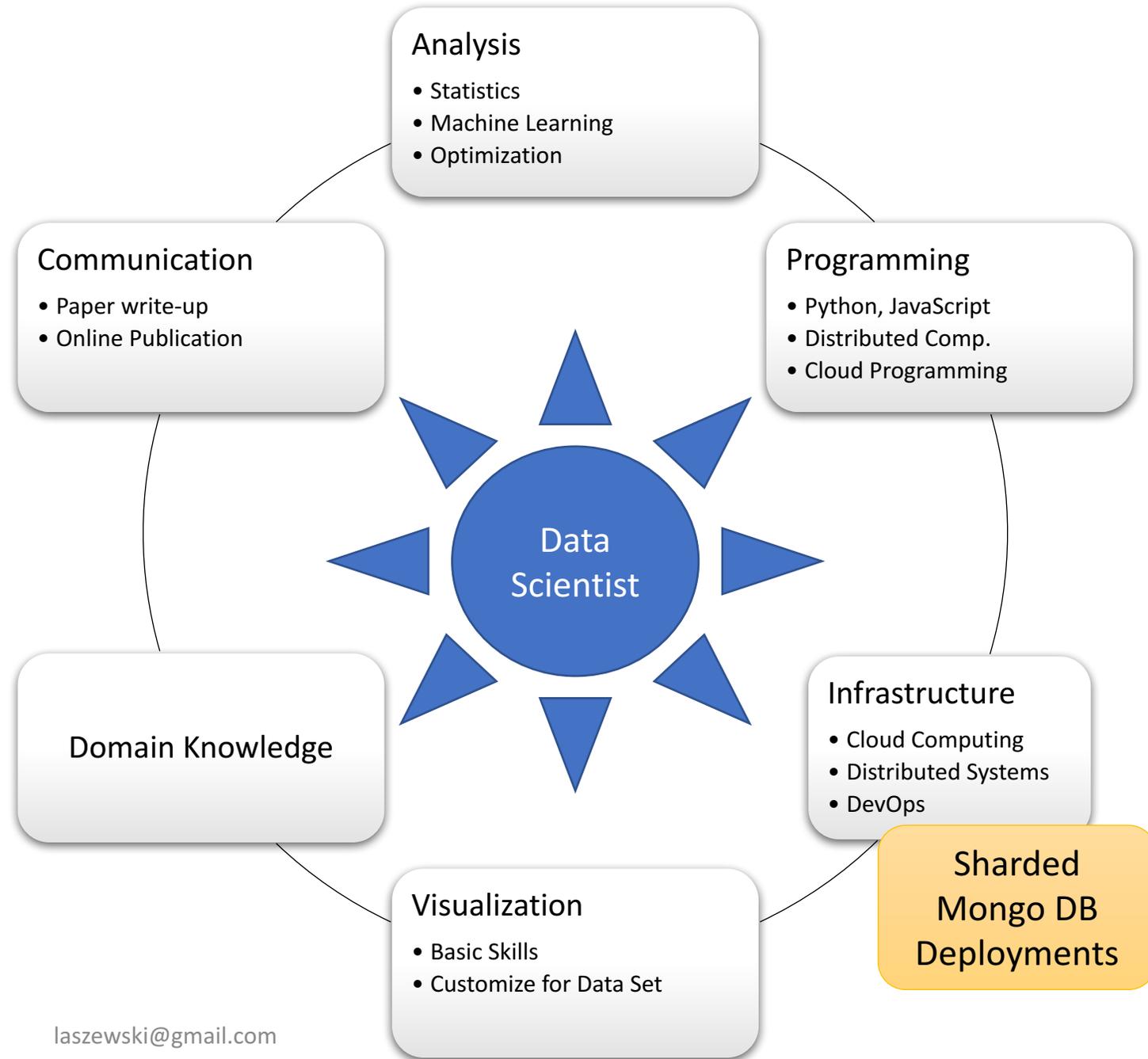
Data Scientist

- Requires integrated knowledge in several key areas. We use a project that addresses:

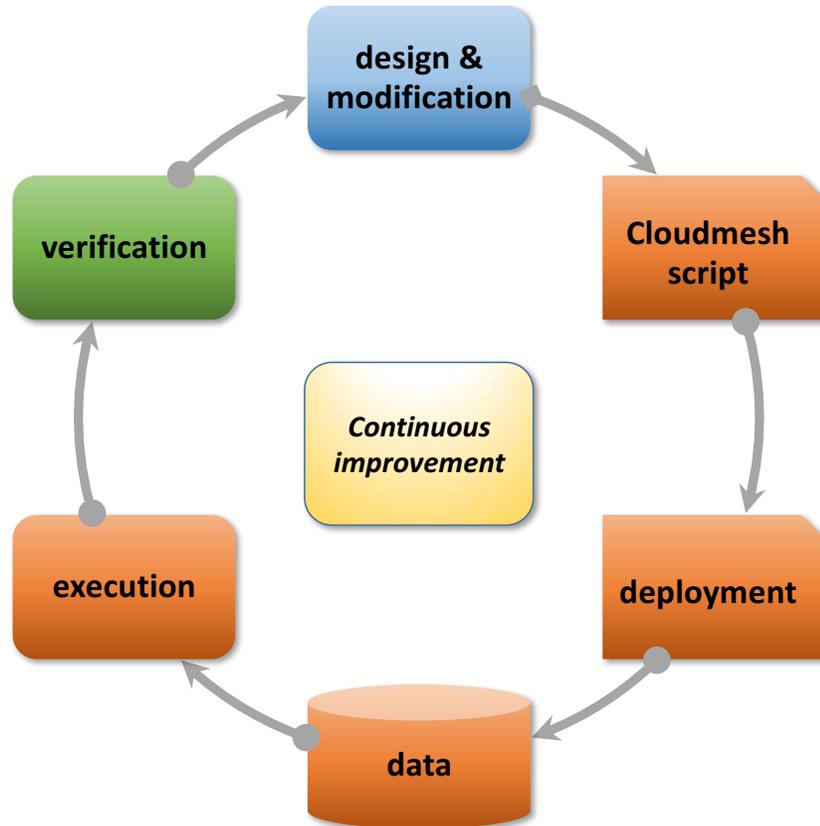
- Communication
- Analysis
- Visualization
- Programming
- Infrastructure
- Domain knowledge

Sharded Mongo DB
Deployments

- Education Programs need to address all of them



Continuous Improvement vs. Continuous Deployment via DevOps



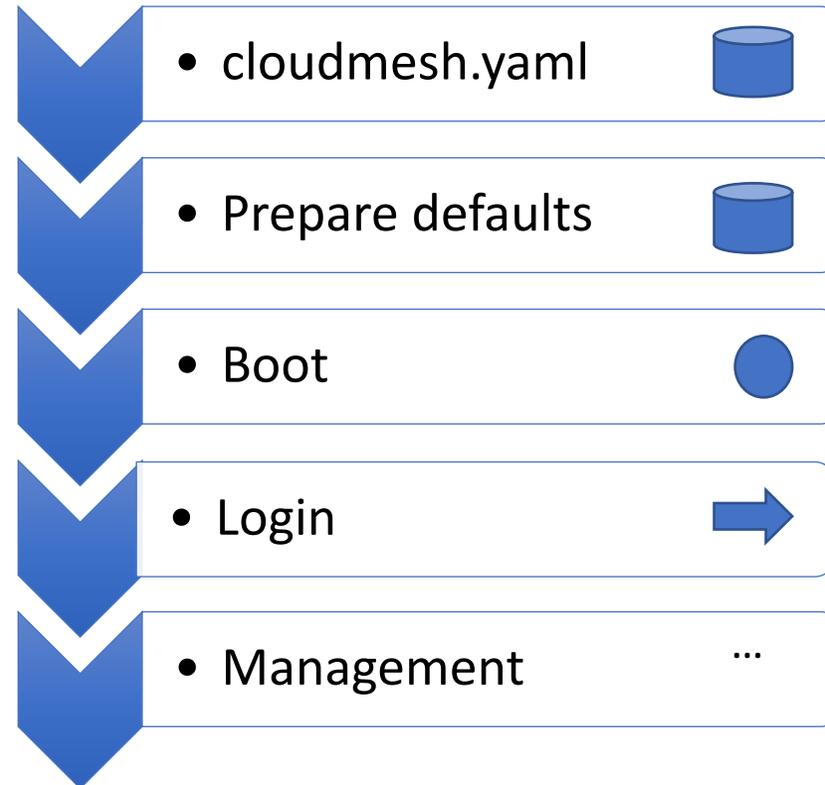
- DevOps is integrated
- Leads to improvement when not only targeting application but also deployment environment.

Cloudmesh Shell – Make Booting Simple

```
$ emacs cloudmesh.yaml
$ cms default cloud=NAME
$ cms default image=NAME
$ cms default flavor=NAME
$ cms vm boot

$ cms vm login

$ cms vm delete
```

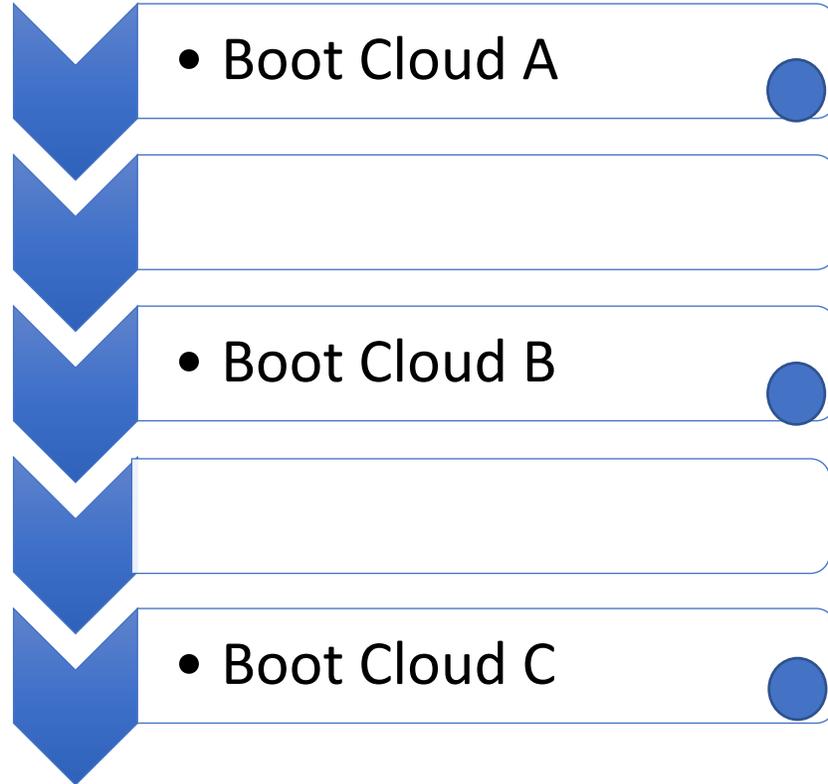


Cloudmesh Shell – Manage Hybrid Clouds

```
$ cms aws boot
$ cms vm boot

$ cms default cloud=chameleon
$ cms vm boot

$ cms default cloud=IUCloud
$ cms vm boot
```

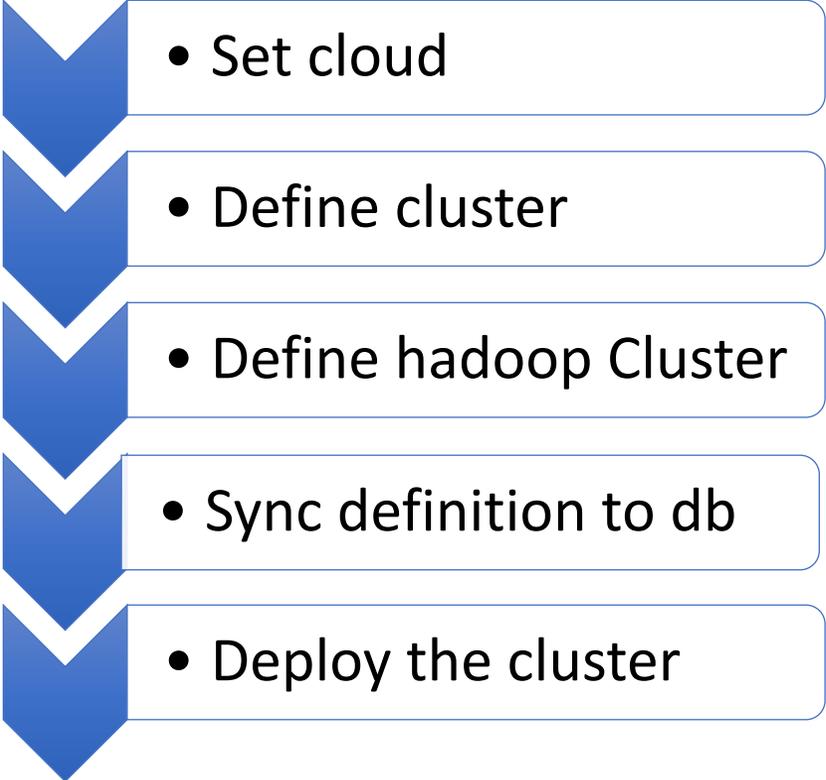


Cloudmesh Shell – Create a Hadoop Cluster

```
$ cm default cloud=chameleon
$ cm cluster define --count=10
  --flavor=m1.large
$ cm hadoop define spark

$ cm hadoop sync # ~30 sec

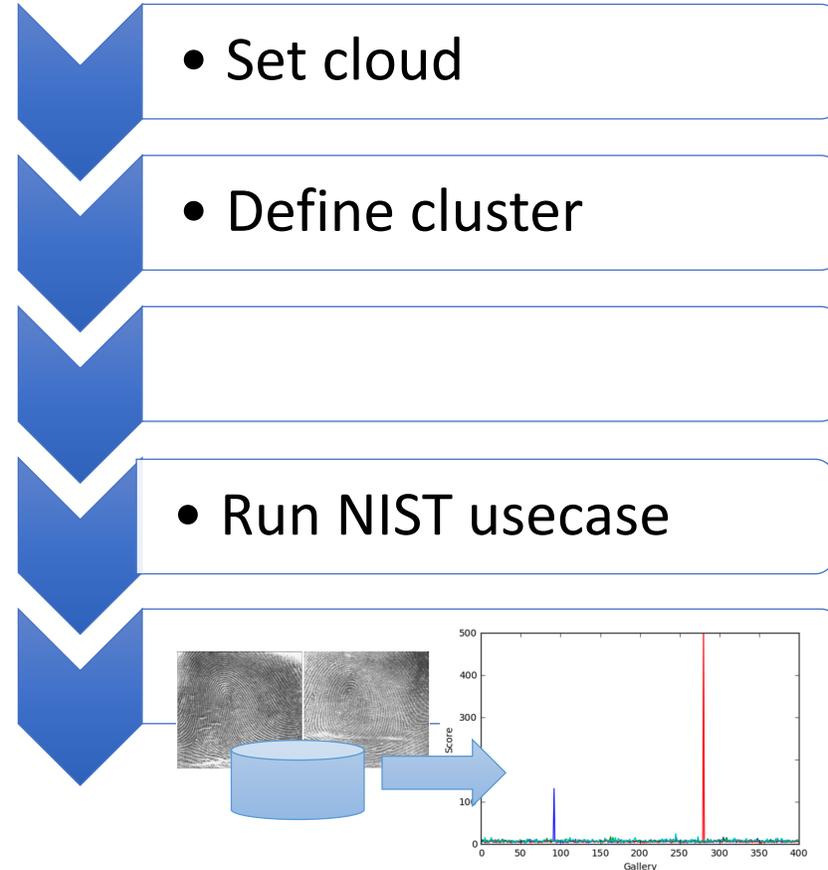
$ cm hadoop deploy # ~ 7 min
```

- 
- Set cloud
 - Define cluster
 - Define hadoop Cluster
 - Sync definition to db
 - Deploy the cluster

Cloudmesh Shell – Create a Hadoop Cluster

```
$ cm default cloud=IUCloud
$ cm cluster define --count=10
  --flavor=m1.large

$ cm nist fingerprint # ~ 30 min
```



Additional resources:

https://github.com/cloudmesh/classes/blob/master/docs/source/notebooks/fingerprint_matching.ipynb

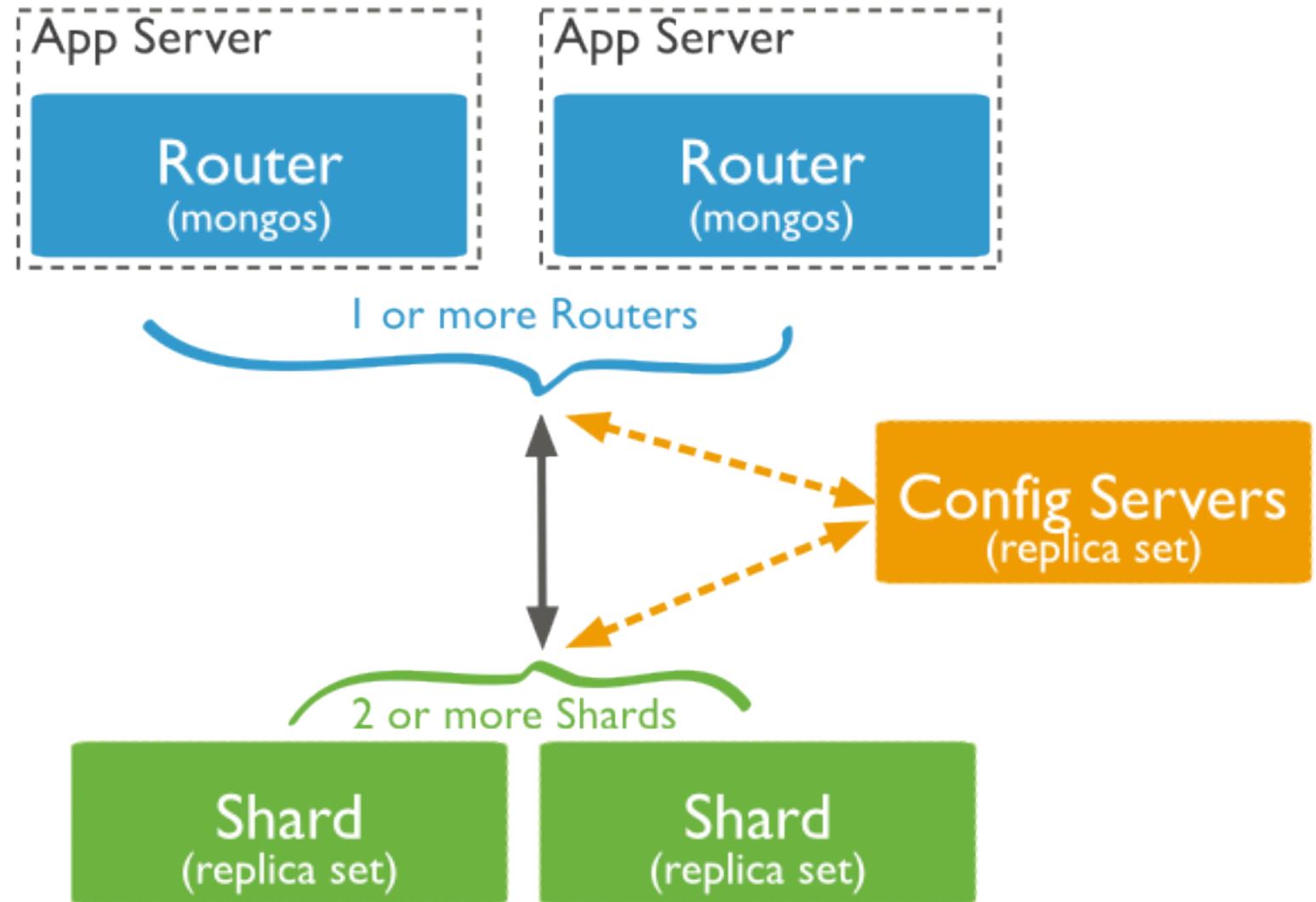
Mongo DB Features

- Document oriented NoSQL data base
 - JSON-like documents
 - Specified through schemas
- Cross-platform compatible
- Free open source
- NoSQL = data that is modeled in means other than the tabular relations used in relational databases.
- Ad-hoc queries
- Indexing
- Replication
- Load Balancing with Sharding
- File Storage
- Aggregation
- Server Side JavaScript
- Capped collections

Mongo DB - Sharding

- User selects **shard key** that determines how the data in a collection will be distributed.
 - data is split into ranges (based on the shard key)
 - distributed across multiple shards.
 - (a) a shard is a master with one or more slaves.
 - (b) or the shard key can be hashed to map to a shard allowing even data distribution.
- MongoDB can run over multiple servers,
 - balancing the load
 - duplicating data for fault tolerance

Architecture



Benchmarks on Clouds

- Three clouds were selected for deployment:
 - Chameleon Cloud
 - Futuresystems
 - Jetstream
- Goal
 - Compare within the allocation limitations of a class multiple cloud performances by varying a number of parameters.
- Scripted Deployments
 - We developed automated scripted deployment and benchmarking process
 - cloud name is passed as a parameter
 - Customization for the deployment of MongoDB is passed via commandline

Cloud Comparison

	FutureSystems	Chameleon	Jetstream
CPU	Xeon E5-2670	Xeon X5550	Haswell E-2680
Cores	1024	1008	7680
Speed	2.66GHz	2.3GHz	2.5GHz
RAM	3072GB	5376GB	40TBr
Storage	335TB	2TB	2 TB
Deployment year	2010	Early 2015	OS 2016

Flavor and OS

- Ubuntu 16.04 LTS (Xenial Xerus) operating system.
- Flavors – slightly different between clouds we use most alike
 - m1.medium Chameleon Cloud
 - m1.medium FutureSystems
 - m1.small was used on Jetstream
 - Flavors have more resources than Chameleon and FutureSystems
 - Storage is lower on jetstream

Cloud	Flavor	VCPU	RAM	Size
Chameleon	m1.medium	2	4	40
FutureSystems	m1.medium	2	4	40
Jetstream	m1.small	2	4	20

Requirements

- Resource Requirements
 - 60 users -> VM hours were limited.
- Capability Requirements
 - creation of VMs and the execution of our applications within these VMs
- Monitoring Requirements
 - Monitoring and benchmarking was conducted by hand without need for specialized services.
- New software created
 - improved the cloudmesh client software [5][6] [7], essential to the success of the class.
- Performance Comparison
 - We have conducted a significant performance comparison among all clouds.

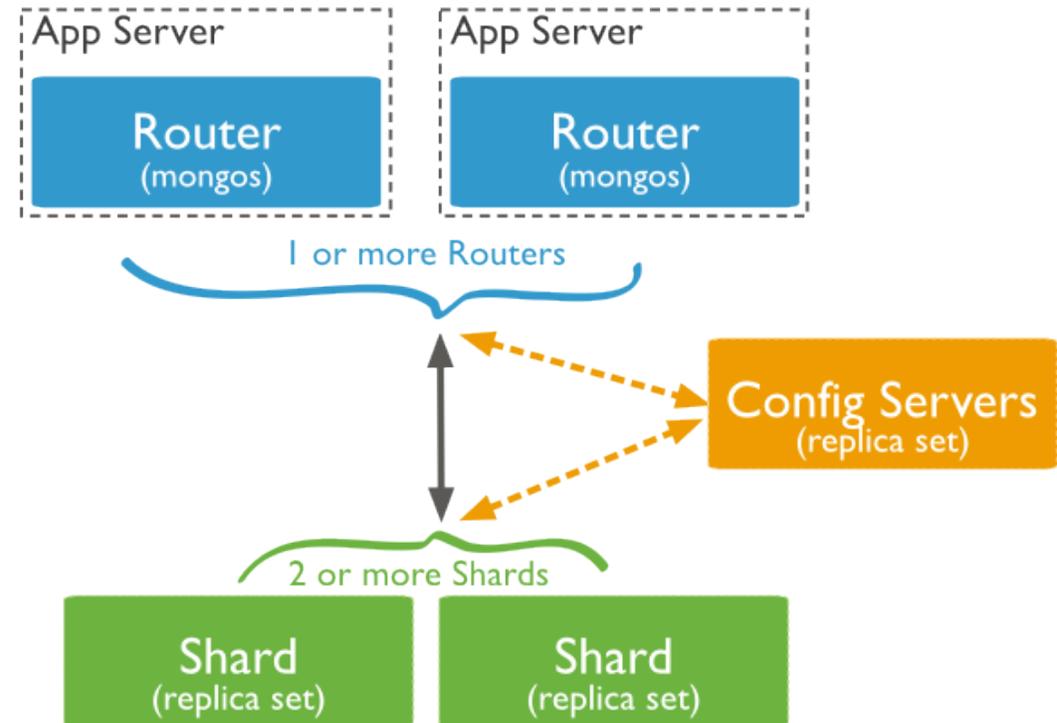
Benchmark

- Deployment times
- Comparing Mongo DB versions
- Comparing Clouds

Deployment times

Deployments

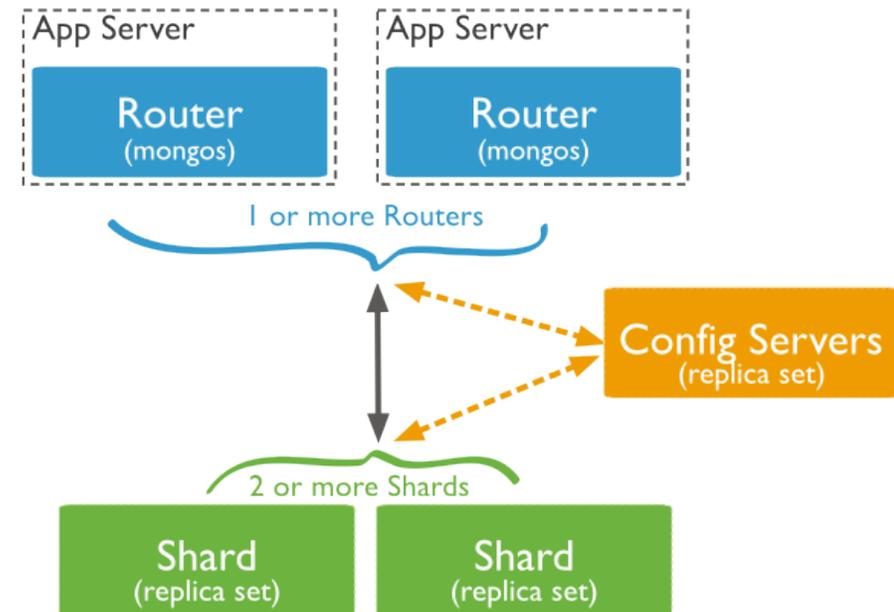
- Deployment A
 - a simple deployment with only one of each component being created..
- Deployment B
 - variation in config servers and shards and an additional Mongos instance.
- Deployment C
 - focus on high performance.
 - 9 shards no replication



	Config	Mongos	Shards	Replicas	Seconds
A	1	1	1	1	330
B	3	2	3	3	1059
C	1	1	9	1	719

Variing other Deployment times

Config Servers -c	Mongos -m	Shards -s	Replicas -r	Time in Seconds
5	1	1	1	534
1	5	1	1	556
1	1	5	1	607
1	1	1	5	524



Data

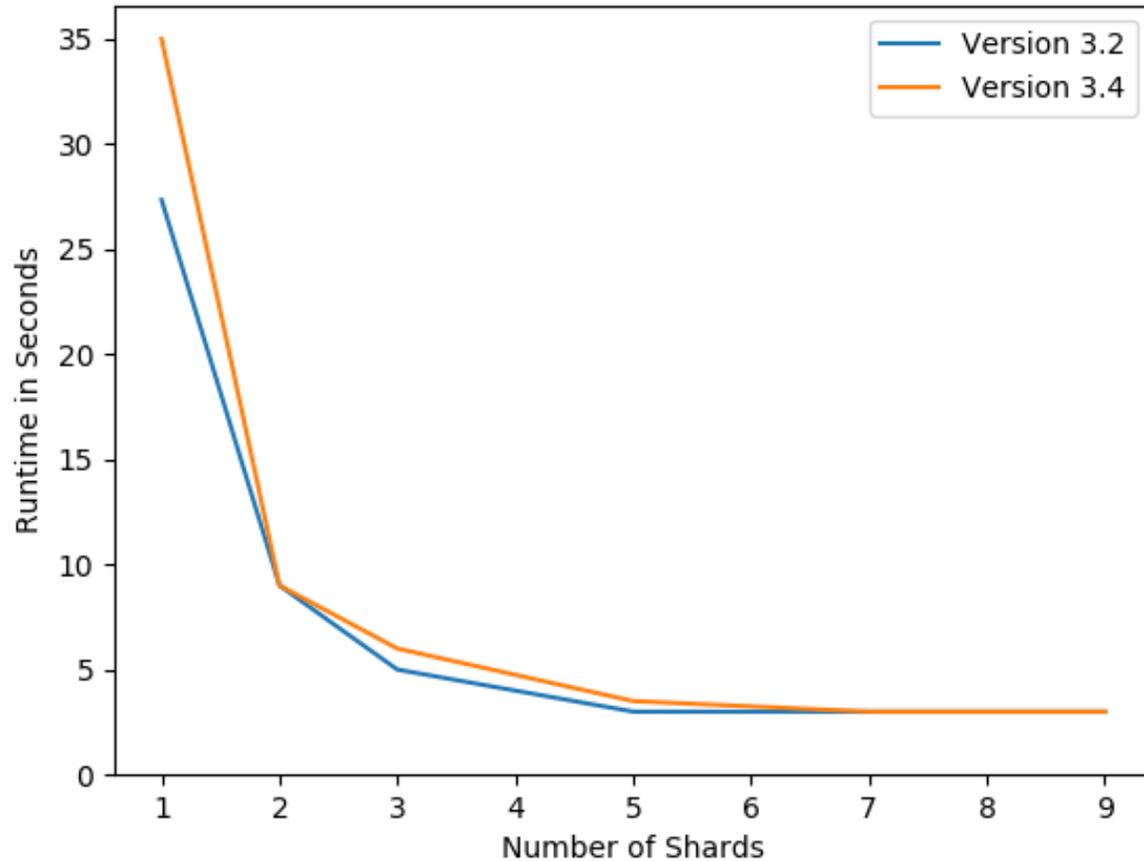
- Major League Baseball PITCH f/x data obtained by using the program Baseball on a Stick (BBOS).
- BBOS is a Python program created by "willkoky" and hosted on sourceforge.net which extracts data from mlb.com and loads it into a MySQL database.
- data was captured locally to the default MySQL database and then extracted to a CSV
- CSV file was imported
- Contains 5,508,014 rows and 61 columns. 1.58 GB in size uncompressed.

Version Comparison

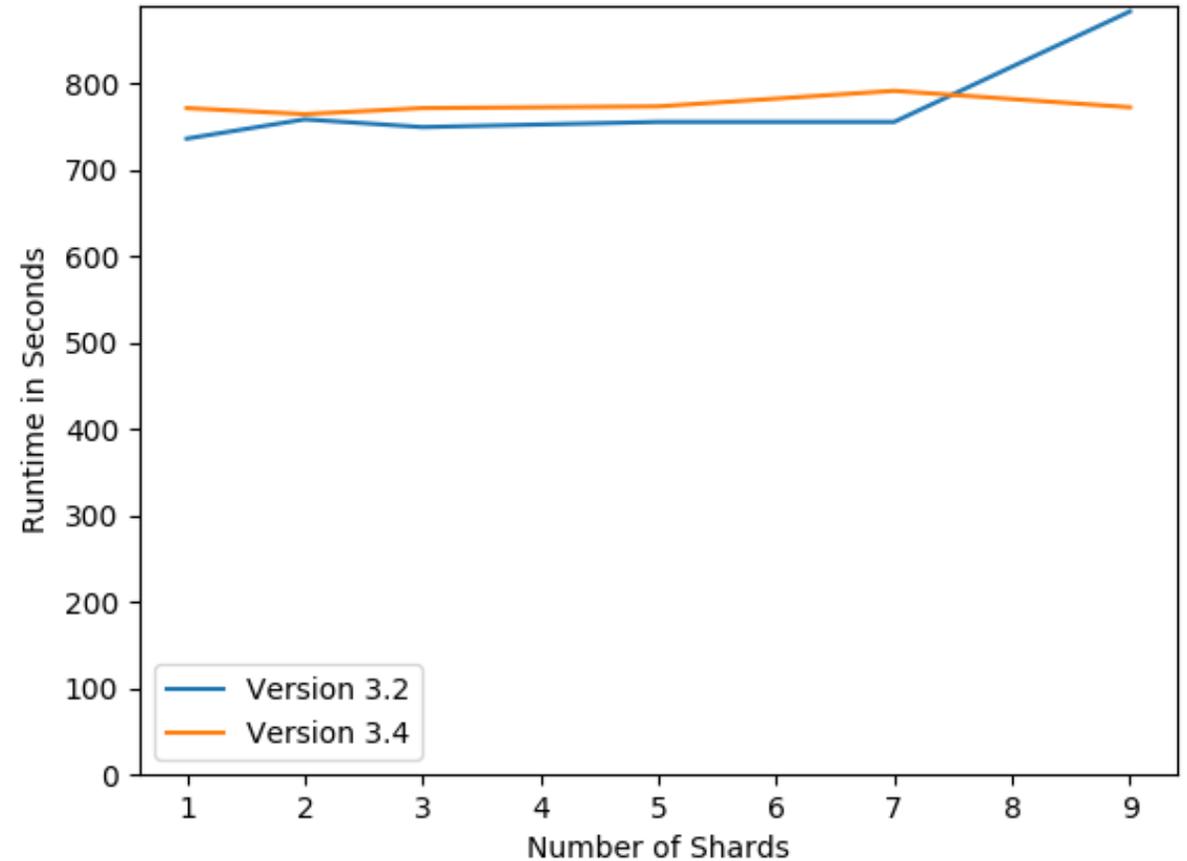
Version Comparison: 3.2 vs 3.4

(Chameleon Cloud)

Find Command



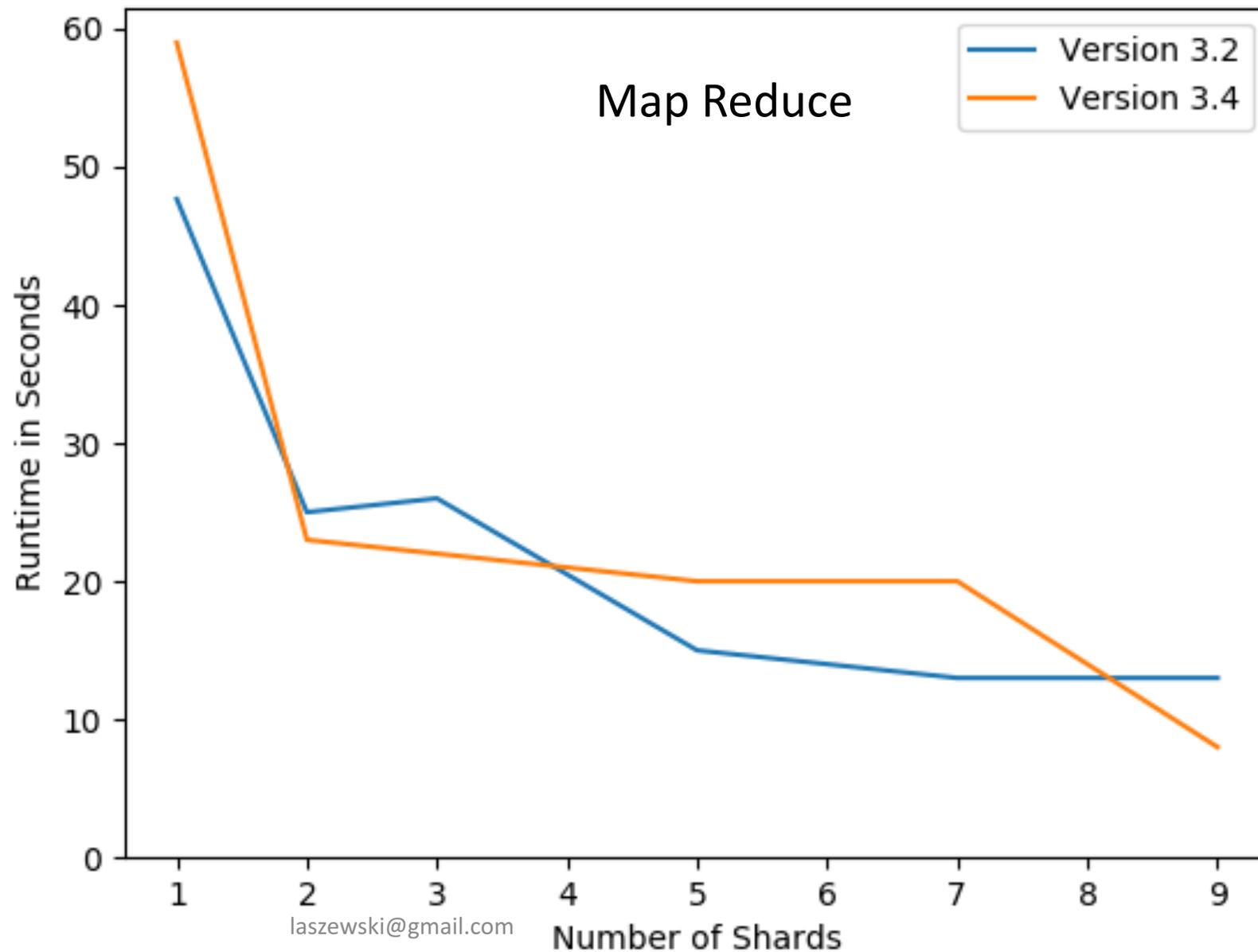
Mongoimport Command



Version Comparison: 3.2 vs 3.4 Map Reduce

(Chameleon Cloud)

Result: not too many changes



Cloud Comparison

Sharding Test

Figure 2: Mongoimport Command - Sharding Test

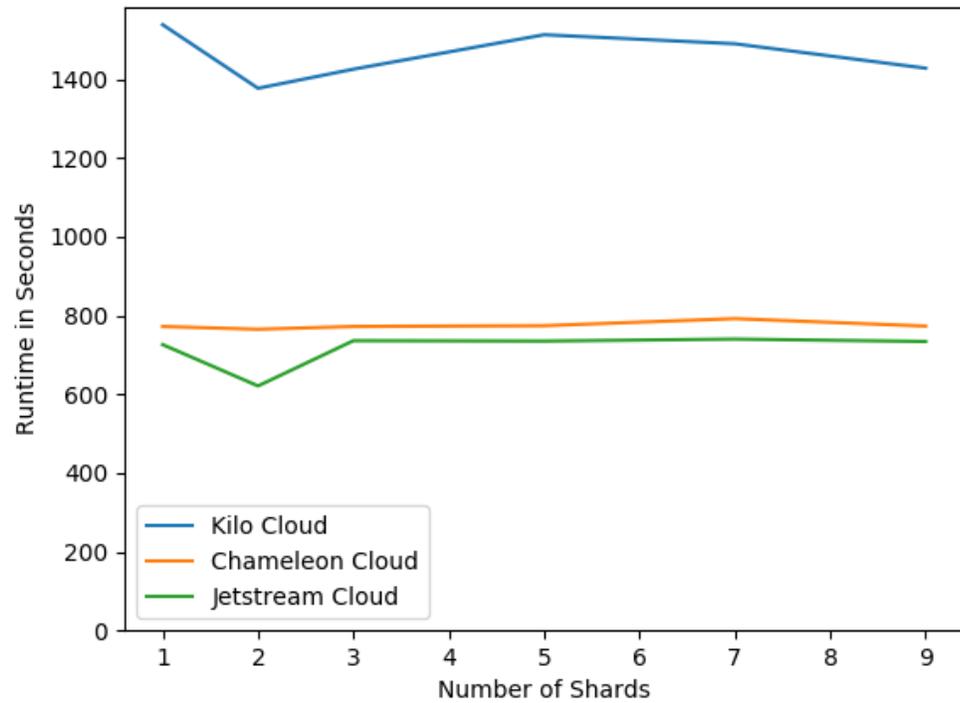
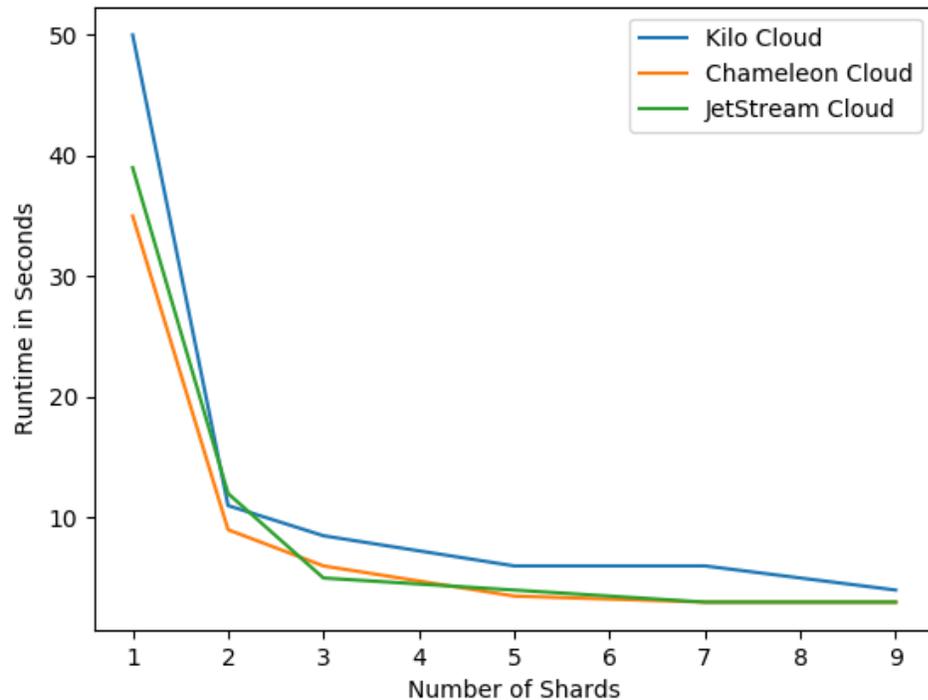
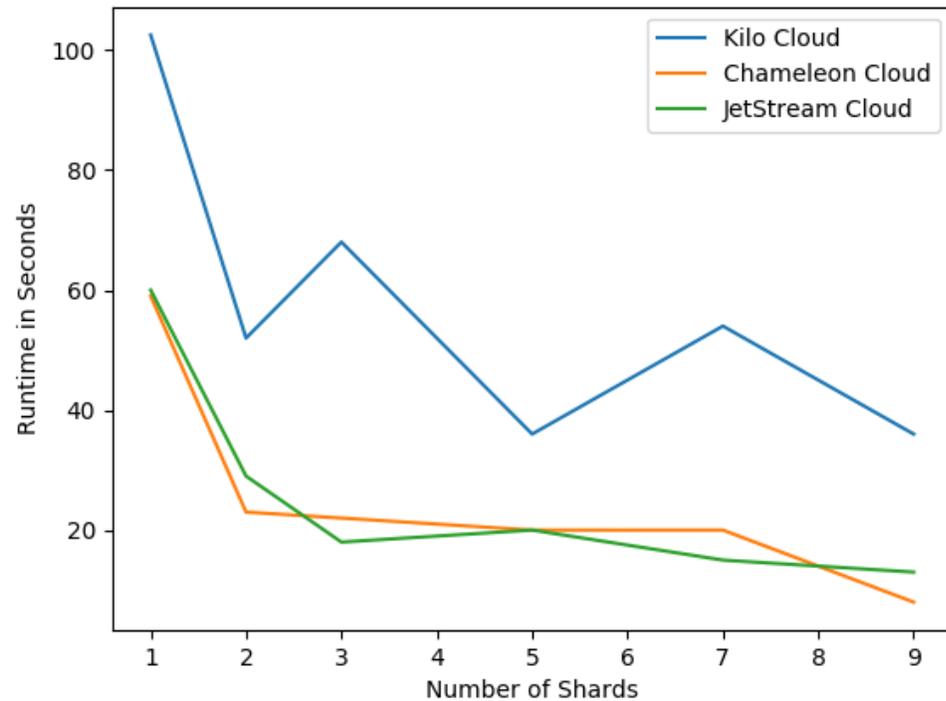


Figure 1: Find Command - Sharding Test



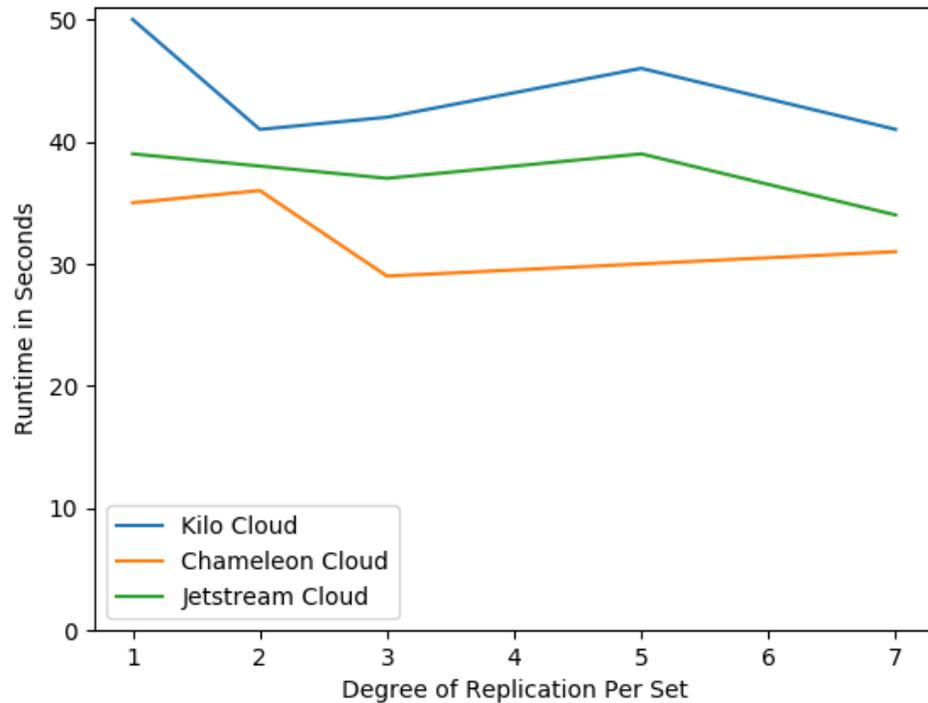
- Chameleon – Jetstream
 - Same
- FutureSystems
 - Acceptable results with higher number of shards

Figure 3: MapReduce - Sharding Test



- Chameleon – jetstream
 - Same
- Futuresystems
 - Significantly worse

Figure 4: Find Command - Replication Test



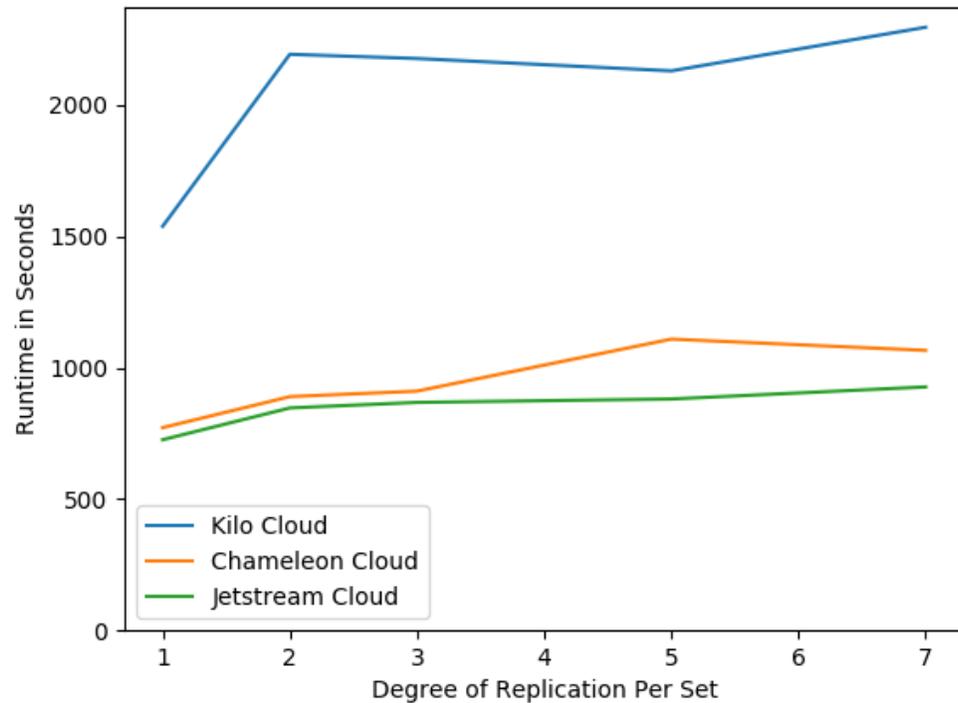
- Replication

- Chameleon cloud seems to perform slightly better
- Futuresystems performs surprisingly well

Cloud Comparison

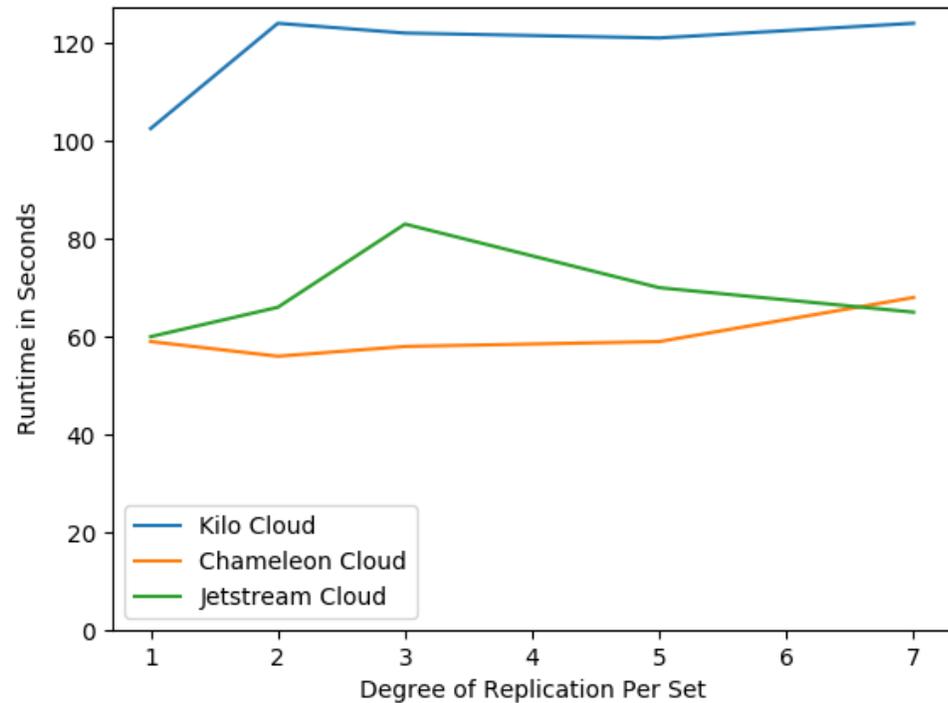
Replication Test

Figure 5: Mongoimport Command - Replication Test



- Chameleon – jetstream
 - Same
- Futuresystems
 - Significantly worse

Figure 6: MapReduce - Replication Test



- Chameleon
 - Slightly better than Jetstream
- Futuresystems
 - Significantly worse

Conclusion

- Jetstream and Chameleon Cloud are essentially the same.
- In some instances Chameleon Cloud performs slightly better
 - (disks/network ...)
- As expected FutureSystem is older machine and performs not as well
 - For some queries FutureSystem is surprisingly good
- Experiments were limited by number of node hours for 60 students in class.
- After class is over no time to run on larger examples
- Its not obvious for a teacher when to give larger allocations for a student that performs well.
- Allocation process broken
- Futuresystems allocation process is superior