CNSBench: A Cloud Native Storage Benchmark

19th USENIX Conference on File and Storage Technologies (FAST 2021)

Alex Merenstein[1], Vasily Tarasov[2], Ali Anwar[2], Deepavali Bhagwat[2], Julie Lee[1], Lukas Rupprecht[2], Dimitris Skourtis[2], Yang Yang[1], and Erez Zadok[1]

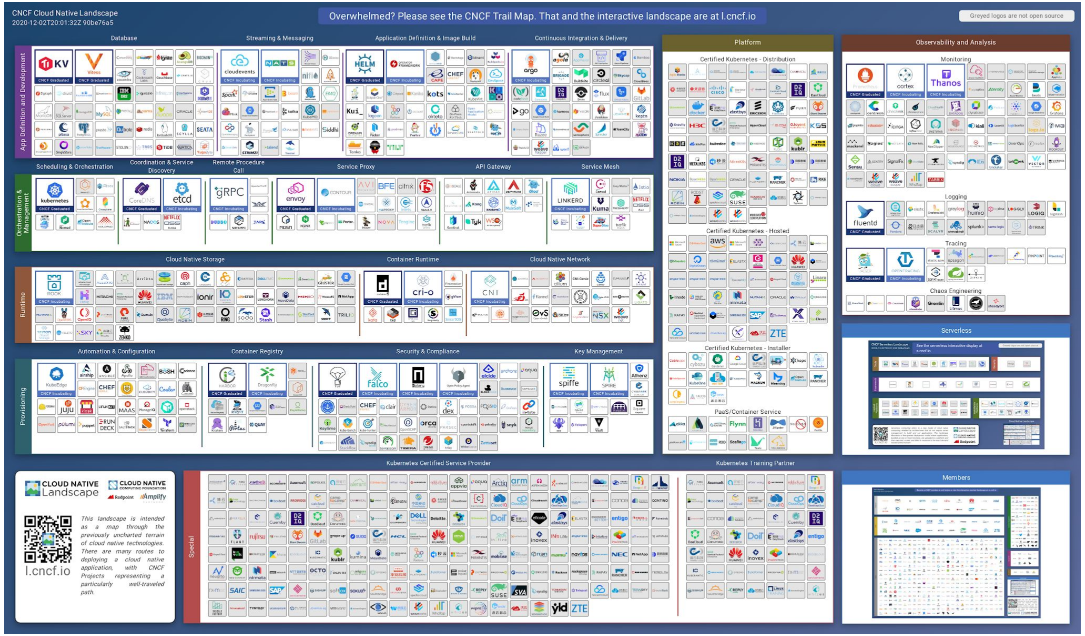
[1]Stony Brook University; [2]IBM Research - Almaden

Outline

- **Background & Motivation**
- Design & Implementation
- Evaluation
- Conclusion

New Trends in Cloud Computing

- Cloud native software
  - Container based
  - Microservice architectures
  - Highly dynamic
  - Frequent deployments
  - Automated management



The cloud native community is large and growing

https://landscape.cncf.io

Motivation 1: More Storage Control Operations

- Storage control operations
  - Creating volumes, attaching volumes, snapshotting, resizing, etc.
  - Volumes: single unit of storage provisioned by a storage provider
- More frequent in cloud native environments
- Existing benchmarks do not generate storage control operations

Motivation 1: More Storage Control Operations

- On one platform, 54% of containers ran for ≤5 minutes and hosts ran a median of 30 containers[2]
  - On a 20 nodes cluster, that results in a rate of one container creation per second
- Companies run 600+ services, deploy 100-1,000+ updates each day[1]
- Users, not administrators create containers

1. https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/definition
2. https://sysdig.com/blog/sysdig-2019-container-usage-report/

Motivation 2: Diversity and Specialization

- Projects such as Docker make cloud native computing widely available
  - Containers for bioinformatics, data science, HPC, ML, etc. available on Docker Hub
- Microservice and serverless architectures
  - Highly specialized workloads
  - Higher density of workloads per node/cluster
- Workloads on hosts and clusters more diverse

Motivation 3: Elasticity and Dynamicity

- Scale to meet spikes in demand
- Increased deployment velocity[1]
  - Netflix: 600+ services, 100+ deployments/day
  - Uber: 1,000+ services, 1,000+ deployments/week
  - WeChat: 3,000+ services, 1,000+ deployments/day

1. https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/definition

Motivation Summary

- Many more storage control operations, current benchmarks can not generate control operations
- Workloads running on each host are more complex, infeasible to manually reproduce
- Applications are elastic and dynamic, infeasible to manually simulate

Outline

- Background & Motivation
- **Design & Implementation**
- Evaluation
- Conclusion

CNSBench Design Requirements

- Separate I/O workloads and control workloads
- Use existing tools to generate I/O workloads
- Specify and create realistic control workloads
- Easy to define and run benchmarks

CNSBench Design Requirements

- Separate I/O workloads and control workloads
- Use existing tools to generate I/O workloads
- Specify and create realistic control workloads
- Easy to define and run benchmarks

CNSBench Design Requirements

- Separate I/O workloads and control workloads
- Use existing tools to generate I/O workloads
- Specify and create realistic control workloads
- Easy to define and run benchmarks

CNSBench Design Requirements

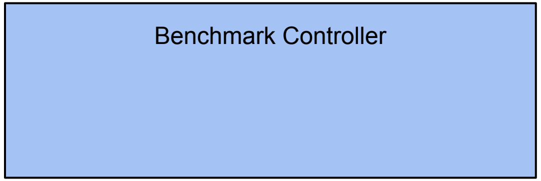
- Separate I/O workloads and control workloads
- Use existing tools to generate I/O workloads
- Specify and create realistic control workloads
- Easy to define and run benchmarks

CNSBench Design Requirements

- Separate I/O workloads and control workloads
- Use existing tools to generate I/O workloads
- Specify and create realistic control workloads
- Easy to define and run benchmarks

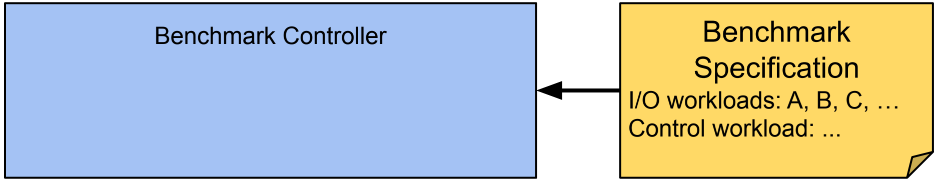
Design

- Controller instantiates I/O workloads & runs control workload



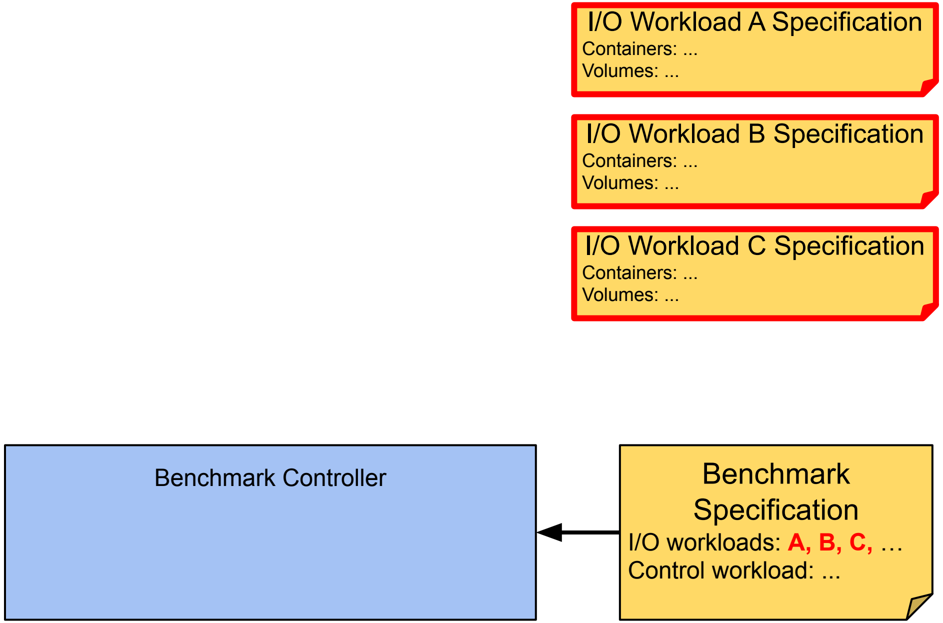
Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification



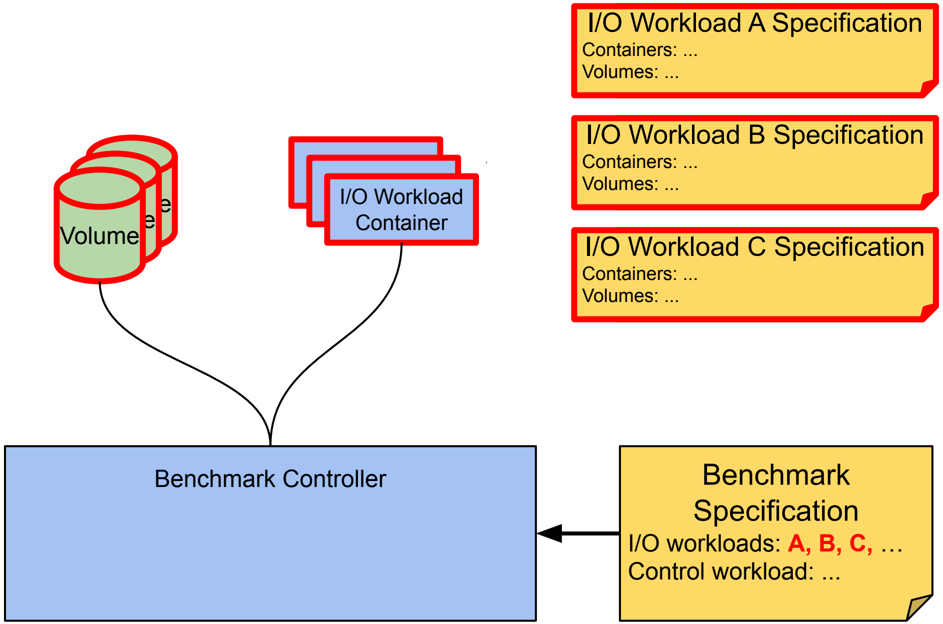
Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification
- Workload specification tells controller how to instantiate I/O workload



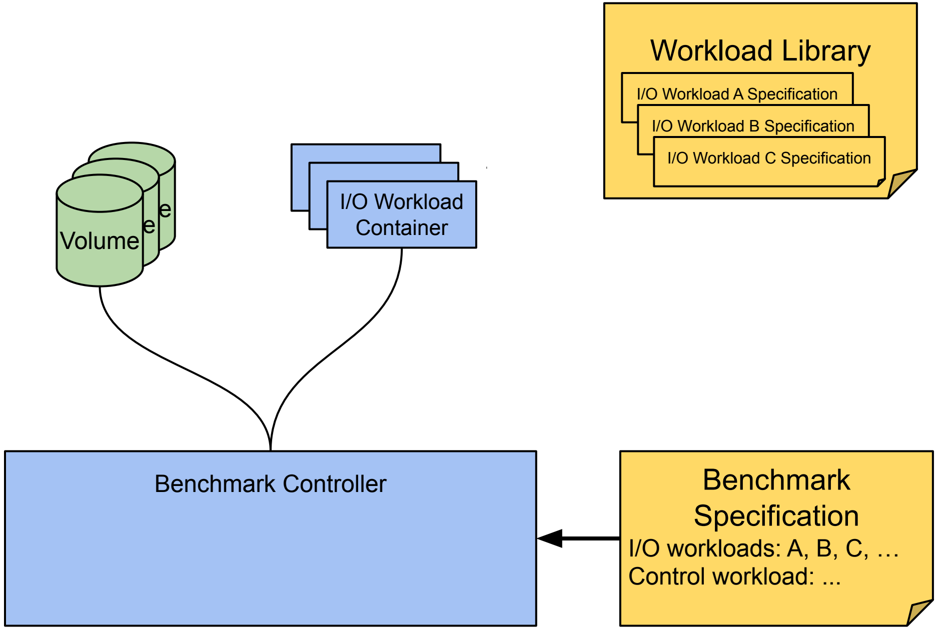
Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification
- Workload specification tells controller how to instantiate I/O workload



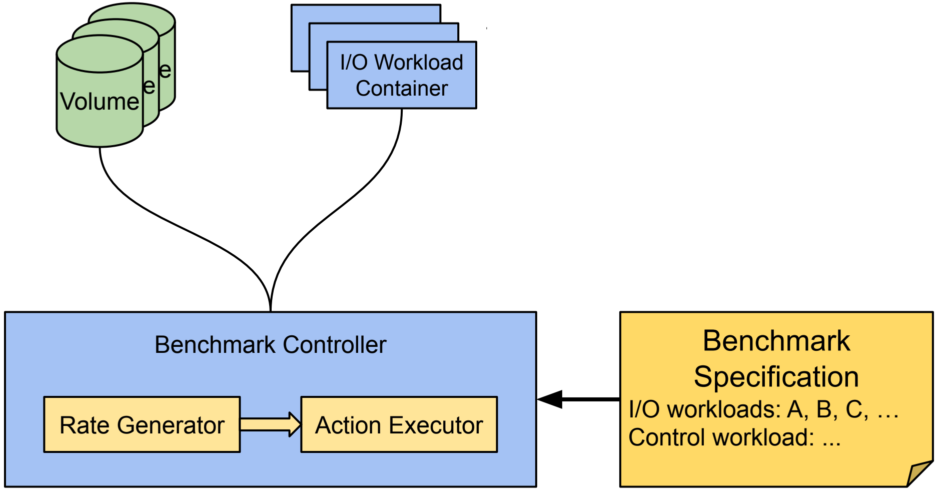
Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification
- Workload specification tells controller how to instantiate I/O workload
- Workload Library contains workload specifications



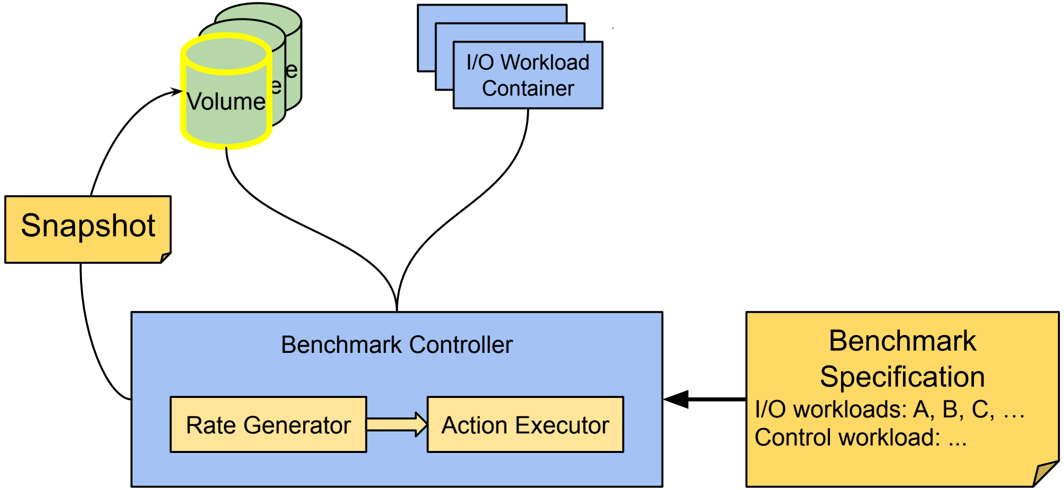
Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification
- Control workload specifies *rates* and *actions*



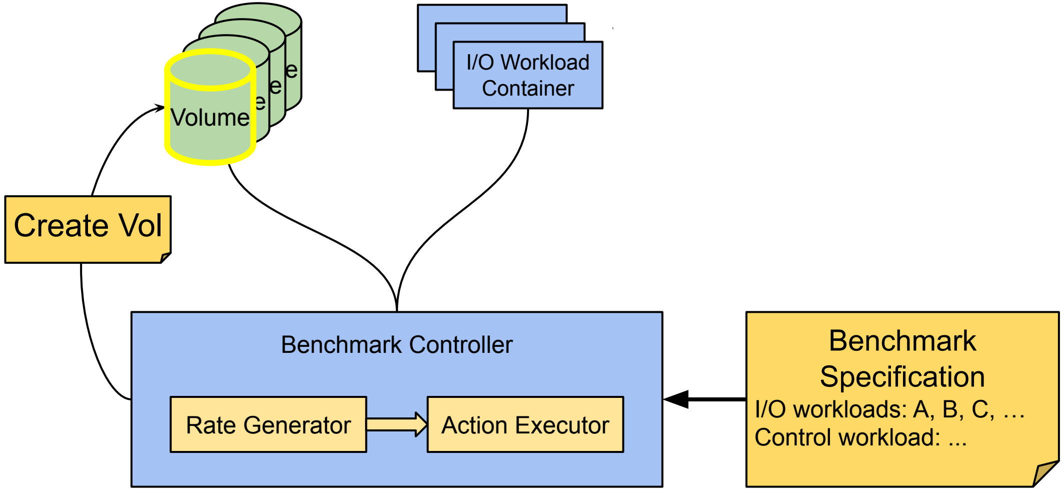
Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification
- Control workload specifies *rates* and *actions*



Design

- Controller instantiates I/O workloads & runs control workload
- User writes Benchmark specification
- Control workload specifies *rates* and *actions*



IBM Stony Brook University

Outline

- Background & Motivation
- Design & Implementation
- **Evaluation**
- Conclusion

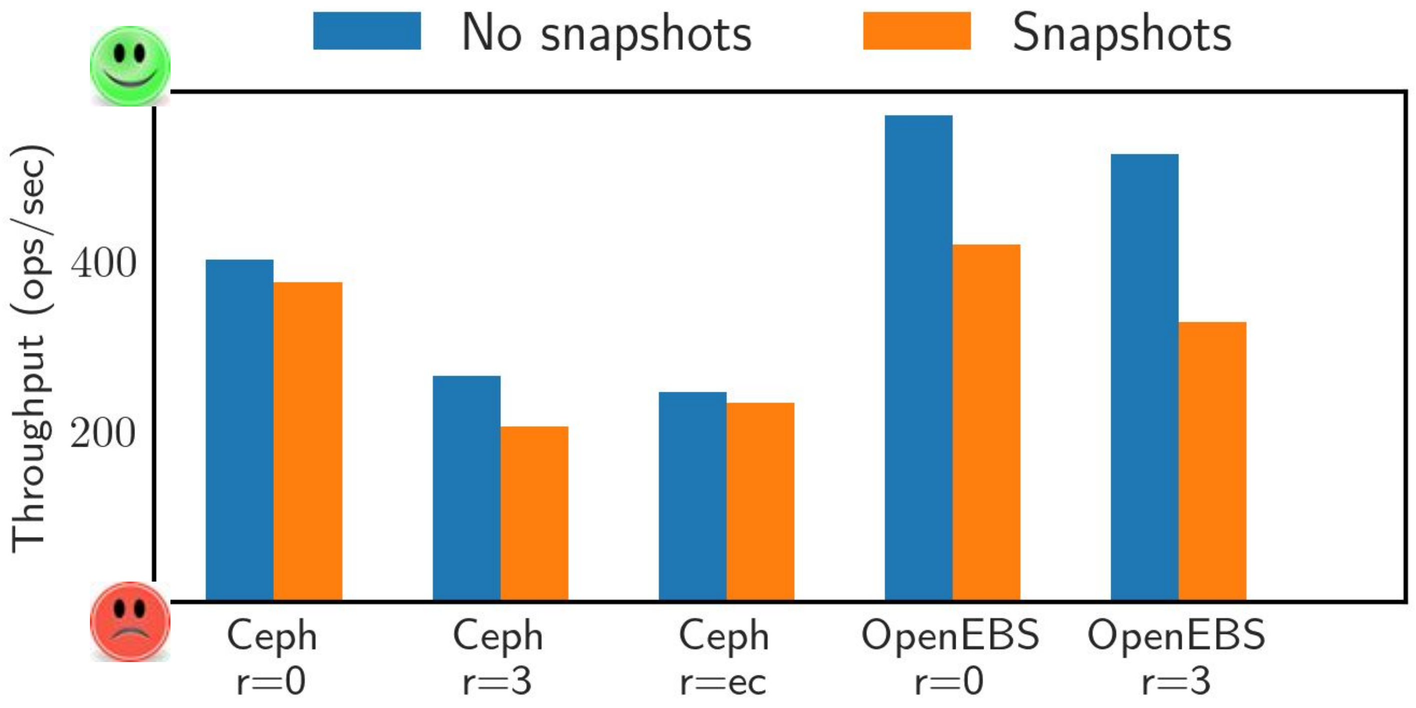
Experimental Setup

- Kubernetes cluster: 10 workers + 1 control plane
- Storage providers
  - Ceph, no replication
  - Ceph, three copies of data
  - Ceph, erasure coding (two copies of data, one coding chunk)
  - OpenEBS, no replication
  - OpenEBS, three copies of data

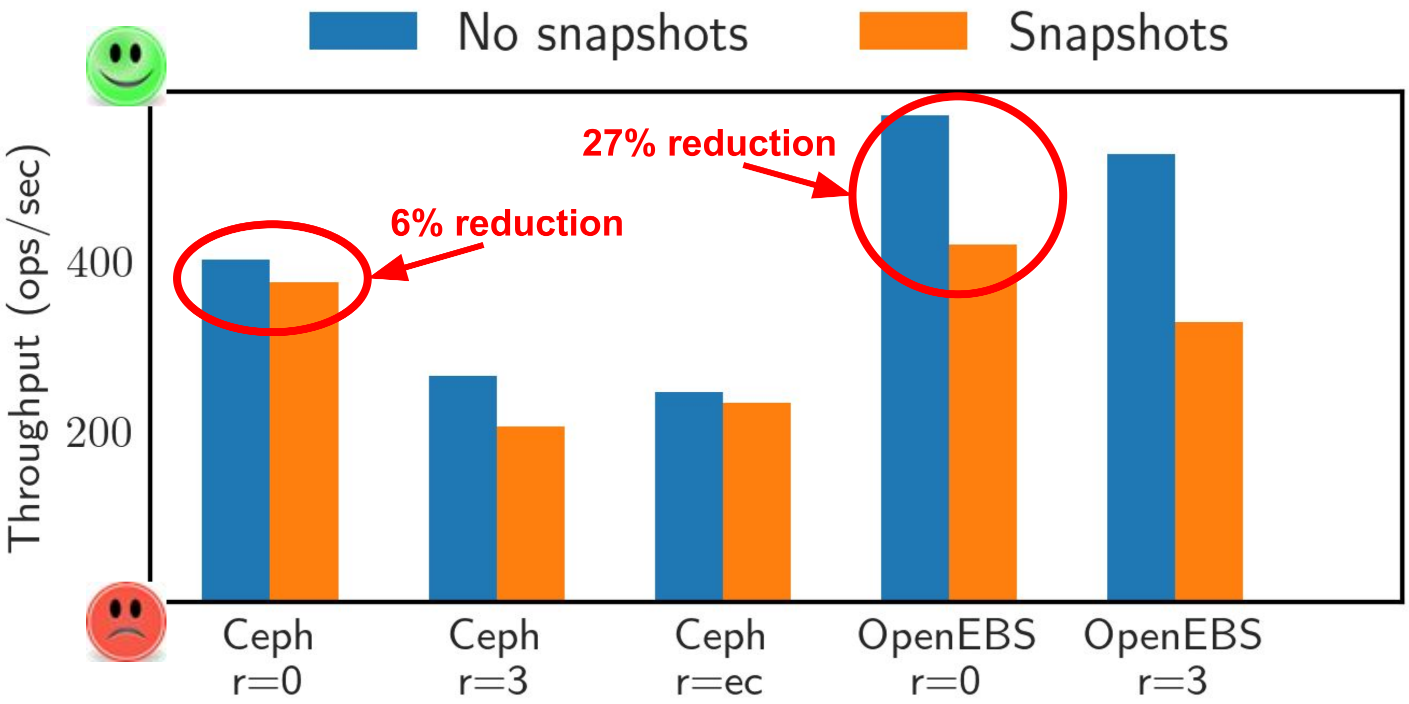
Impacts on I/O Workloads

- Questions
  - Do control operations have an impact on I/O workloads?
  - Is that impact different across storage configurations?
- Setup
  - MongoDB evaluated with YCSB, with & without snapshots

Impacts on I/O Workloads

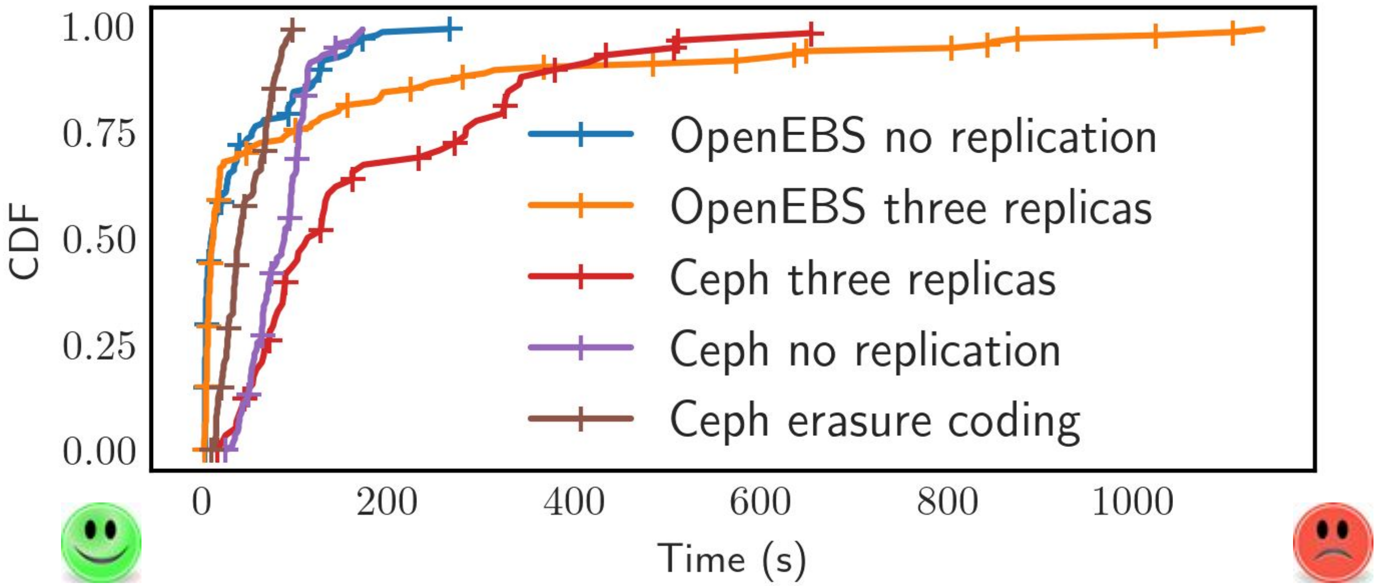


Impacts on I/O Workloads

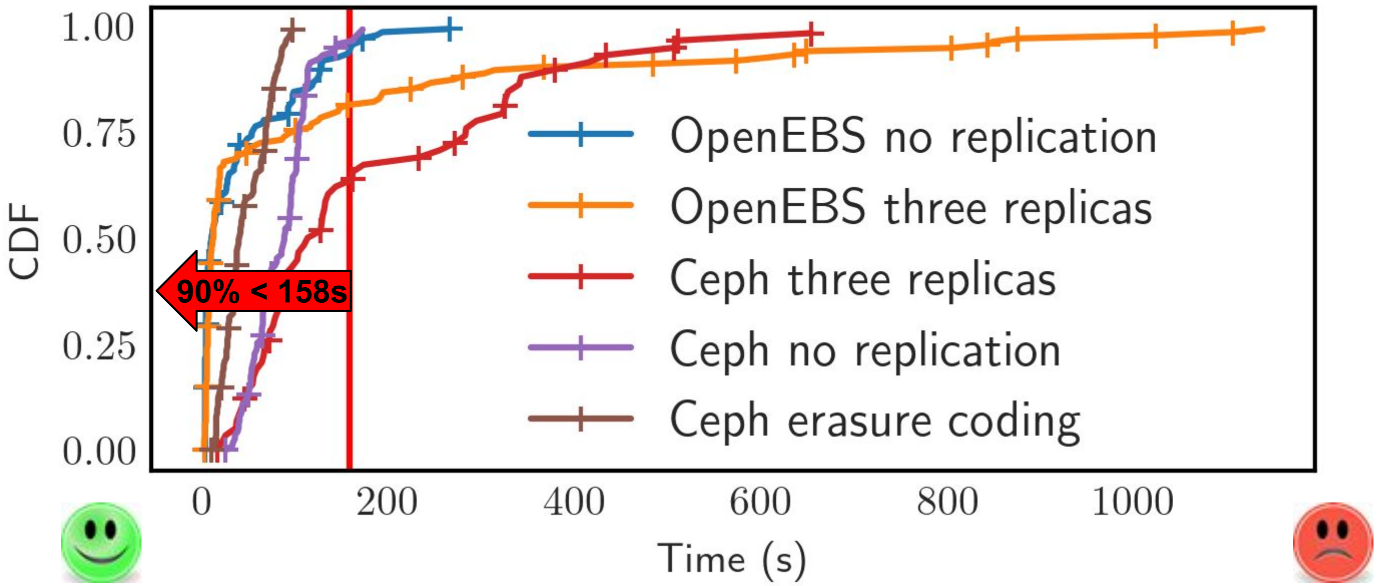


Stony Brook University

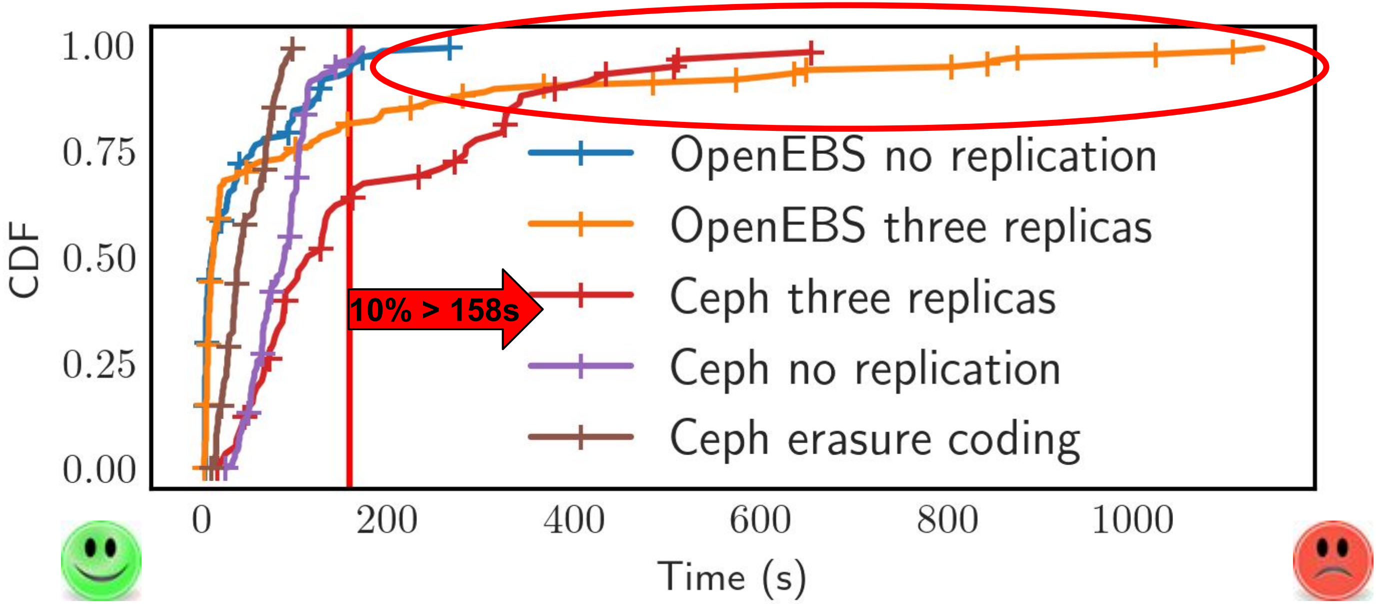
Impacts on I/O Workloads



Impacts on I/O Workloads



Impacts on I/O Workloads



Outline

- Background & Motivation
- Design & Implementation
- Evaluation
- **Conclusion**

Future Work

- Building out a Workloads Library
- Collecting real world traces for analysis
- Improving analysis of results
- Use CNSBench to improve storage systems

Conclusion

- New benchmark is needed to support cloud native environments
- Presented requirements, design, and implementation of CNSBench
- Demonstrated utility of CNSBench through three evaluations

CNSBench: A Cloud Native Storage Benchmark

Thank You

Q&A

mmerenstein@cs.stonybrook.edu

Get and contribute to CNSBench at:
https://github.com/CNSBench

IBM
Stony Brook University

Example Benchmark Specification

```yaml
kind: Benchmark
metadata:
  name: fio-benchmark
spec:
  workloads:
    - name: fio-rw
      workload: fio
      count: 3
      vars:
        storageClass: obs-r1
        config: fio-config
      outputs:
        outputName: es
  actions:
    - name: snapshots
      rateName: const-rate
      snapshotSpec:
        snapshotClass: obs-csi
        workloadName: fio-rw
  rates:
    - name: const-rate
      constantRateSpec:
        interval: 60
  outputs:
    - name: es
      httpPostSpec:
        url: http://es:9200/fio/_doc/
```

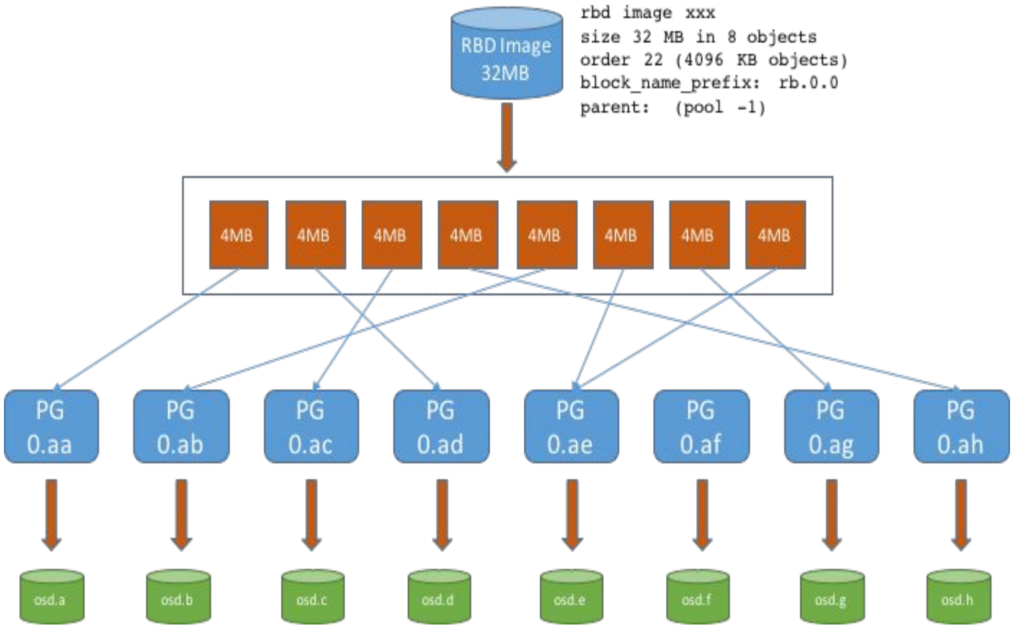
Metrics Collection

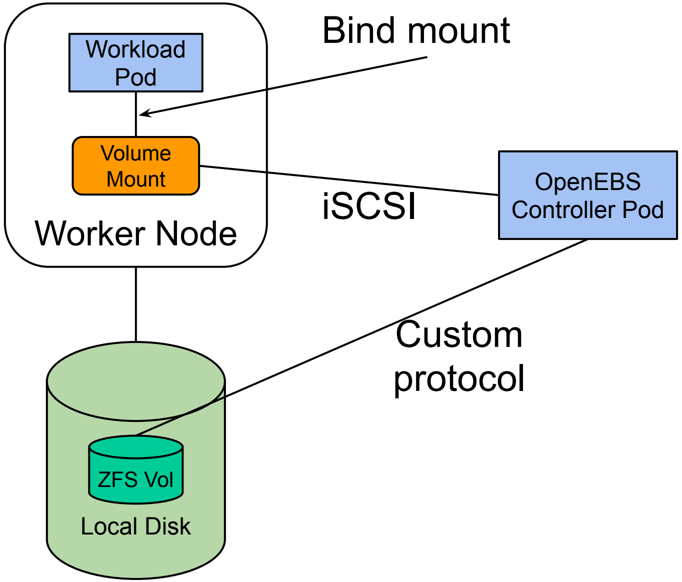
- Many tools available for collecting metrics from Kubernetes clusters
- Documentation on Github provides examples of how to “merge” CNSBench results with these 3rd party metrics
- Future work to investigate automatically gathering metrics & doing analysis

Real World Data

- No publically available data on rates of control operations, rates of storage use, etc.
  - In many cases data not being collected
- Ephemeral volume support in Kubernetes
- Anecdotally know volume creation is a bottleneck in some use cases

Ceph vs OpenEBS





https://www.openshift.com/blog/openshift-container-storage-4-introduction-to-ceph

Performance Explanations

- More experimentation needed to fully explain results
  - Also working to determine what metrics/information should be collected during benchmarking to help with this
- Snapshots: some storage providers quiesce the filesystem before taking a snapshot, others don’t
- Polling architecture: introduces variability in performance, also some polling intervals might be long

Benchmark Initialization

- As part of benchmark spec, should allow user to define a workload that runs before the rest of the benchmark
- Can make use of Kubernetes' "readiness" condition to wait to start executing benchmark operations

Use Cases

- Sysadmin/application architect
  - Evaluating different storage providers
  - Evaluating different settings
  - Testing how a cluster would respond to different scenarios
    - Spike in traffic
    - Failed node
  - Other questions
    - "How many snapshots can I take per hour without impacting my app?"
    - "How many volumes can I provision at a time"?
  - App mobility: does moving an app to a new platform impact performance?
- Storage provider developer: improving their product

Why Not Extend Existing Benchmark?

- Cloud native benchmark requires “orchestration” capabilities to manage cluster resources
  - Would be difficult to add to existing benchmarks
  - Does not naturally extend existing benchmarks’ capabilities
- Cloud native benchmark needs to leverage existing benchmarks

Supporting Reproducibility

- Declarative model of infrastructure management helps reproduce cluster configuration
- Experimentation needed to determine what data should be collected during benchmarking