One Optimized I/O Configuration per HPC Application

Leveraging I/O Configurability of Amazon EC2 Cloud

Mingliang Liu, Jidong Zhai, Yan Zhai
Tsinghua University

Xiaosong Ma
North Carolina State University
Oak Ridge National Laboratory

Wenguang Chen
Tsinghua University

APSys 2011, July 12

Outline

Background

- I/O becomes the bottleneck for many HPC applications
  - Intensive I/O operations and concurrency
  - One-size-fits-all I/O configuration
- There is a trend to migrate the HPC applications from traditional platforms to cloud
- Cloud provides tremendous flexibility in configuring I/O system
  - Fully controlled virtual machines
  - Easily deployed user scripts
  - Multiple types of low-level devices
  - Online device acquisition and migration

Motivation

The Problem

Can we employ I/O configurability of cloud for HPC apps?

Motivation

The Problem

Can we employ I/O configurability of cloud for HPC apps?

☺ Configurability lies in:

- Set up the specific file system at start up
- Explore different types of low-level devices
- Tune the file system inherent parameters

Motivation

The Problem

Can we employ I/O configurability of cloud for HPC apps?

☺ Configurability lies in:

- Set up the specific file system at start up
- Explore different types of low-level devices
- Tune the file system inherent parameters

☹ Challenges:

- The feasibility is highly workload-dependent
- Tradeoff between efficiency vs. cost-effectiveness

Amazon EC2 CCI

CCI: Cluster Computing Instance

Amazon's solution to HPC in Cloud

- Quad-core Intel Xeon X5570 CPU, with 23GB memory
- Interconnected by 10 Gigabit Ethernet
- Amazon Linux AMI, RedHat family OS with Intel MPI
- local block storage (Ephemeral) with 2*800 GB
- Elastic Block Store (EBS), attached as block storage devices
- Simple Storage Service (S3), key-value based object storage

Outline

File System Selection

- Different I/O access pattern and concurrency require different kinds of file system
  - shared vs. parallel
- NFS is enough for low I/O demands, simple to deploy
- A parallel file system (eg. PVFS, Lustre) can be employed to
  - support large and shared file writes
  - scale up well

File System Selection

- Different I/O access pattern and concurrency require different kinds of file system
  - shared vs. parallel
- NFS is enough for low I/O demands, simple to deploy
- A parallel file system (eg. PVFS, Lustre) can be employed to
  - support large and shared file writes
  - scale up well

Easy to choose and setup

118 LOC bash for NFS, and 173 LOC bash for PVFS

Device Selection Considerations

Devices differ in levels of abstraction and access interfaces.

Storage	Pros.	Cons.
S3	off-shelf, designed for Internet and database apps	no POSIX
Ephemeral	Free	non-persistent, only two disks
EBS	- persistent beyond instances - more disks available	Charged

Device Selection Considerations

Devices differ in levels of abstraction and access interfaces.

Storage	Pros.	Cons.
S3	off-shelf, designed for Internet and database apps	no POSIX
Ephemeral	Free	non-persistent, only two disks
EBS	- persistent beyond instances - more disks available	Charged

The choice depends on the needs of individual **applications**

File System Internal Parameters

Options	Description
Sync mode	NFS `sync` vs `async` write modes
Device number	Combining multiple disks into a software RAID0
I/O server number	NFS single-server bottleneck, PVFS can employ many I/O servers
Meta-data distribution	PVFS can distribute metadata to multiple servers
I/O Server Placement	Part-time vs. dedicated I/O servers
Data Striping	striping factor, unit size

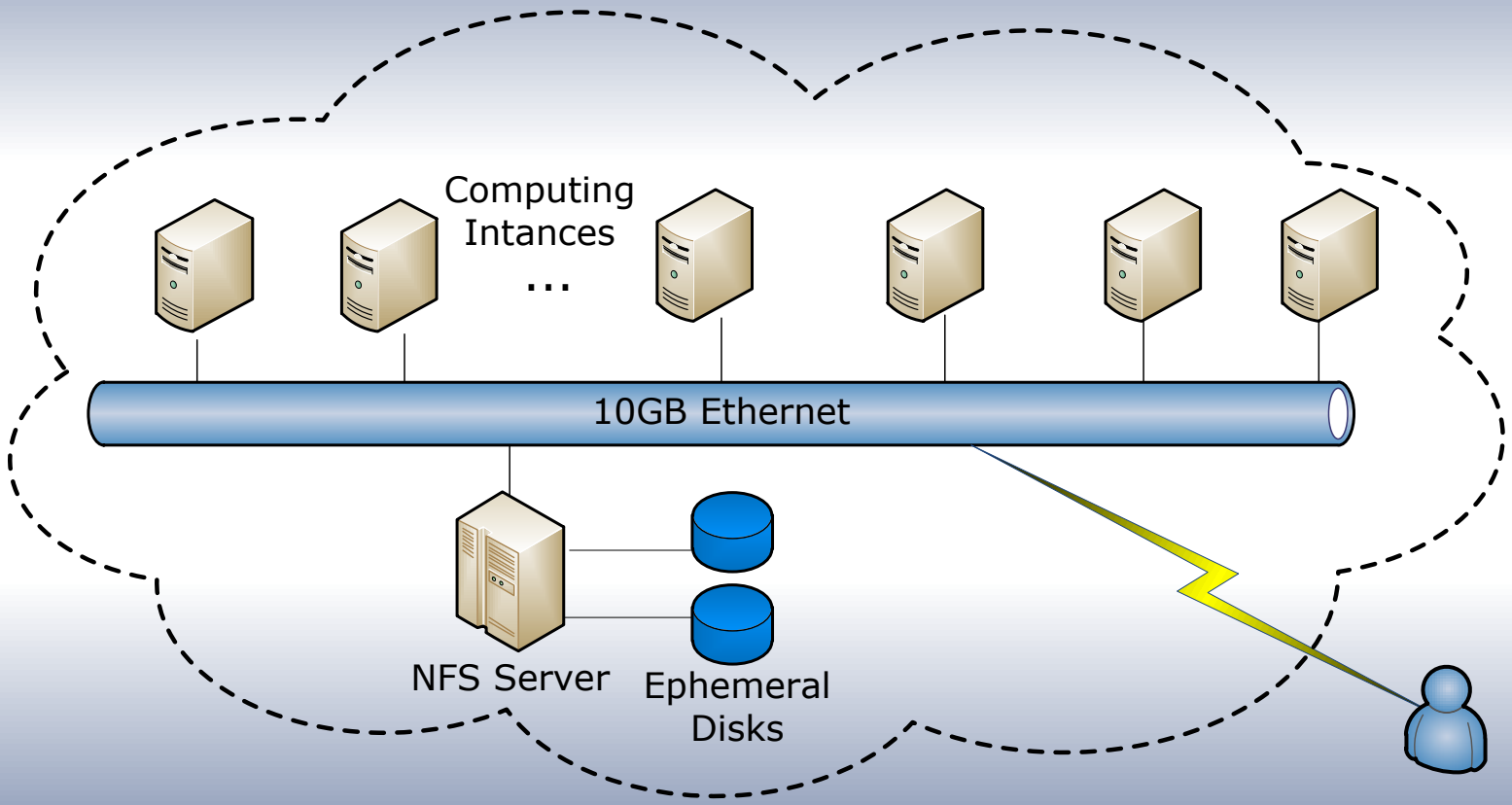
File System Internal Parameters

Options	Description
Sync mode	NFS `sync` vs `async` write modes
Device number	Combining multiple disks into a software RAID0
I/O server number	NFS single-server bottleneck, PVFS can employ many I/O servers
Meta-data distribution	PVFS can distribute metadata to multiple servers
I/O Server Placement	Part-time vs. dedicated I/O servers
Data Striping	striping factor, unit size

Again

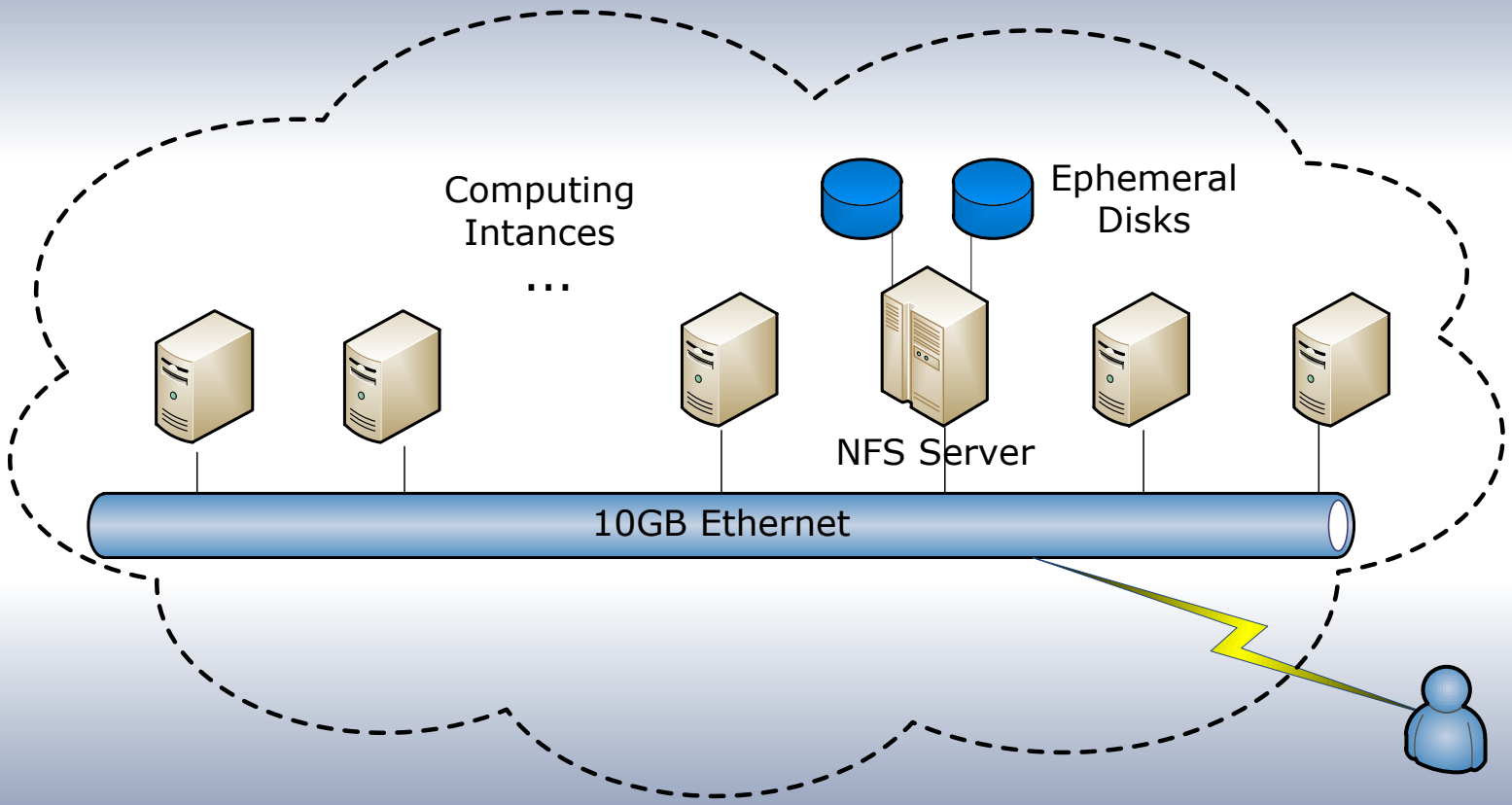
The choice depends on the needs of individual **applications**

Dedicated NFS, Ephemeral Disks



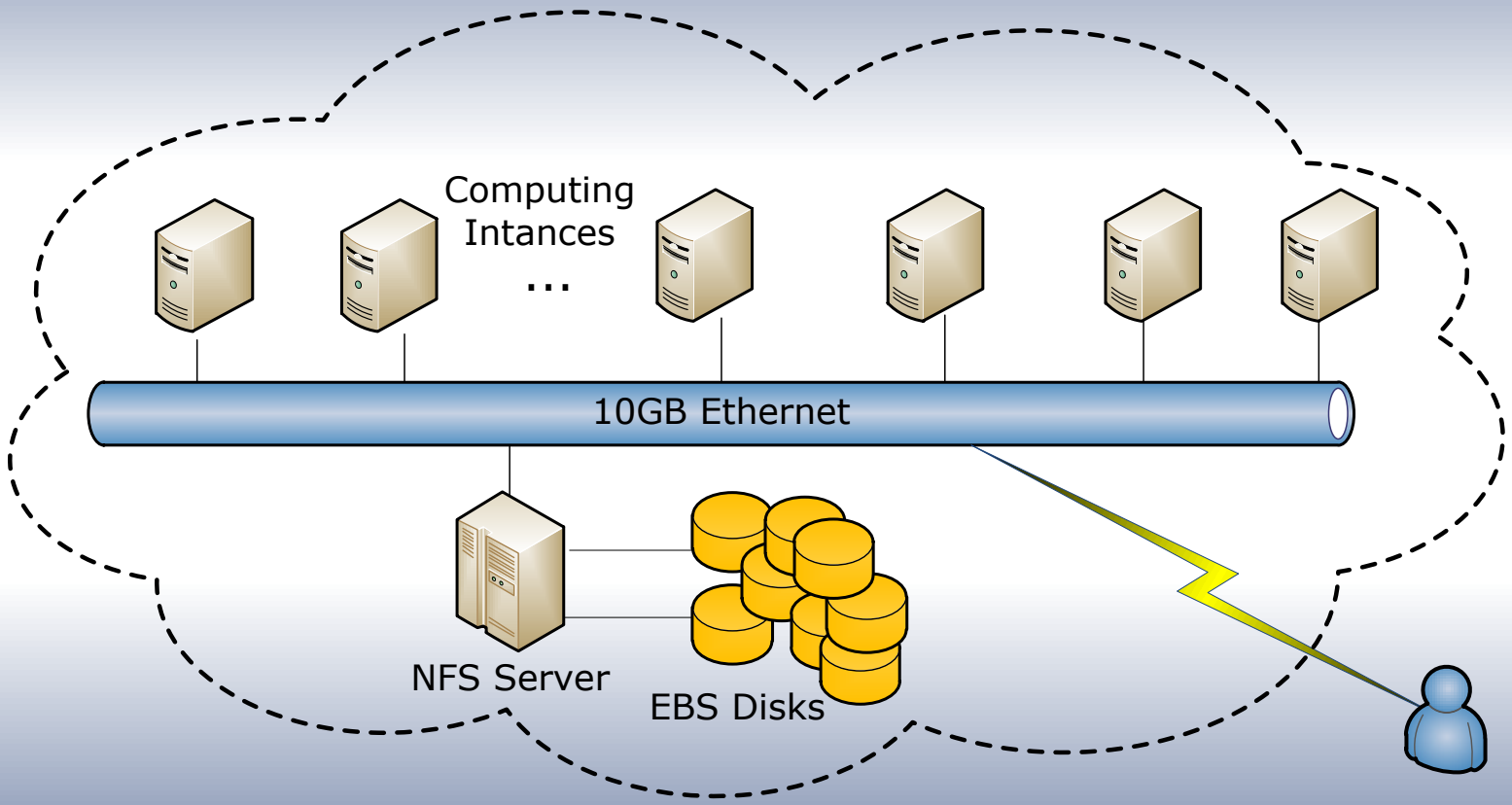
There is 1 NFS I/O server deployed in one dedicated instance, mounting two ephemeral disks.

Parttime NFS Mounting Ephemeral



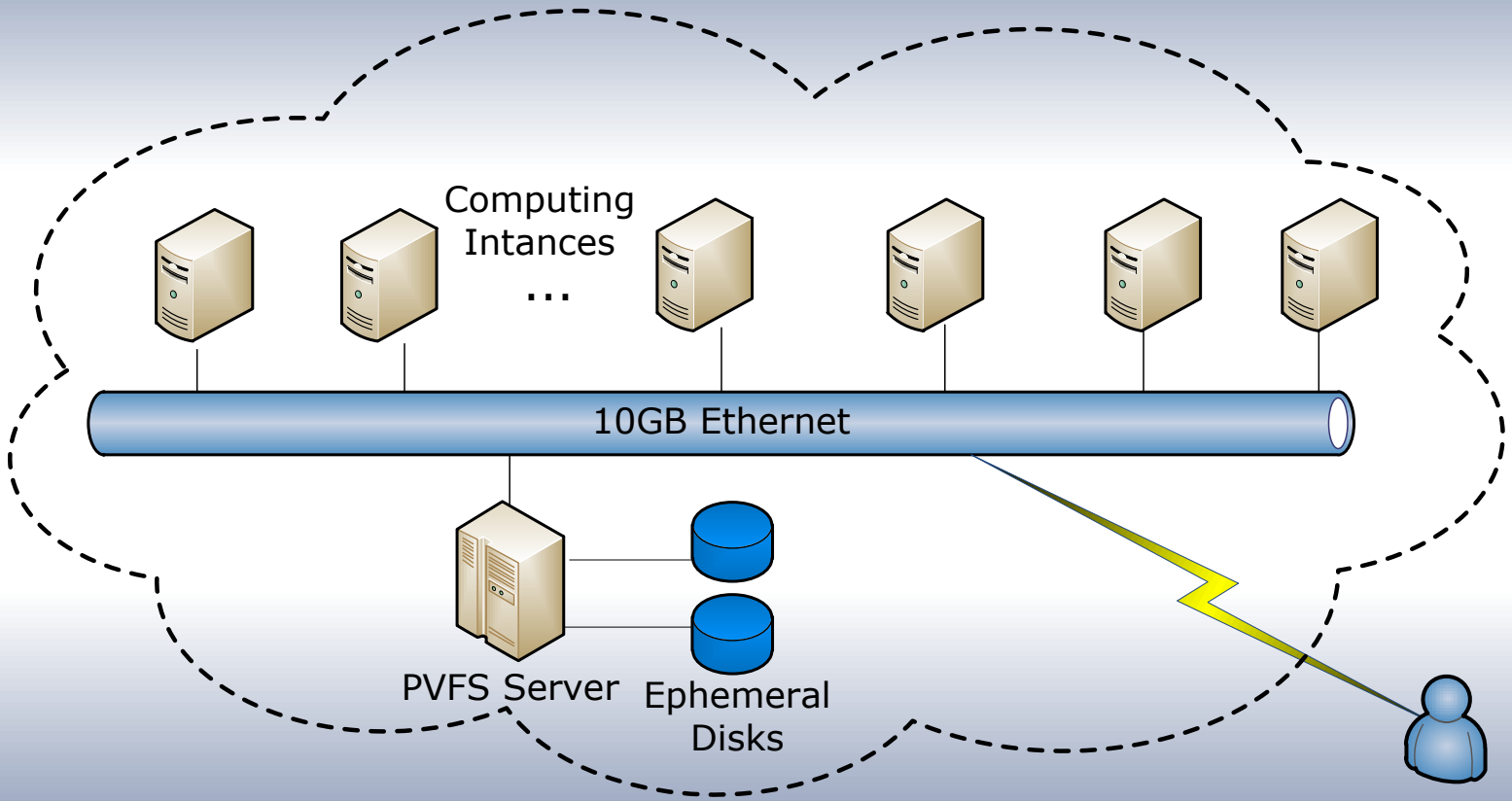
There is 1 NFS I/O server deployed in one parttime instance, mounting two ephemeral disks.

Dedicated NFS Mounting EBS Disks



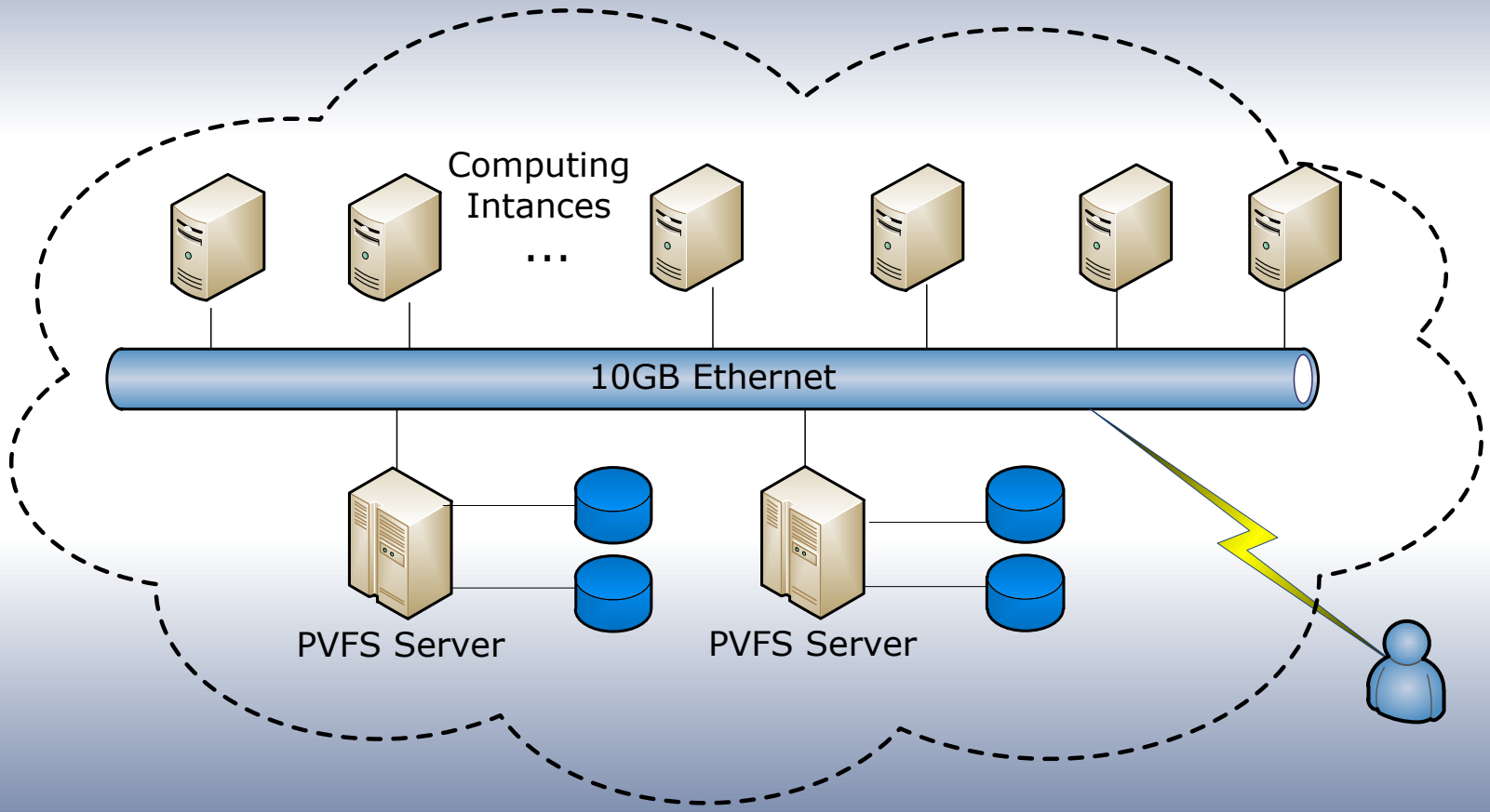
There is 1 NFS I/O server deployed in one dedicated instance, mounting 8 EBS disks into RAID0.

1 Dedicated PVFS I/O server



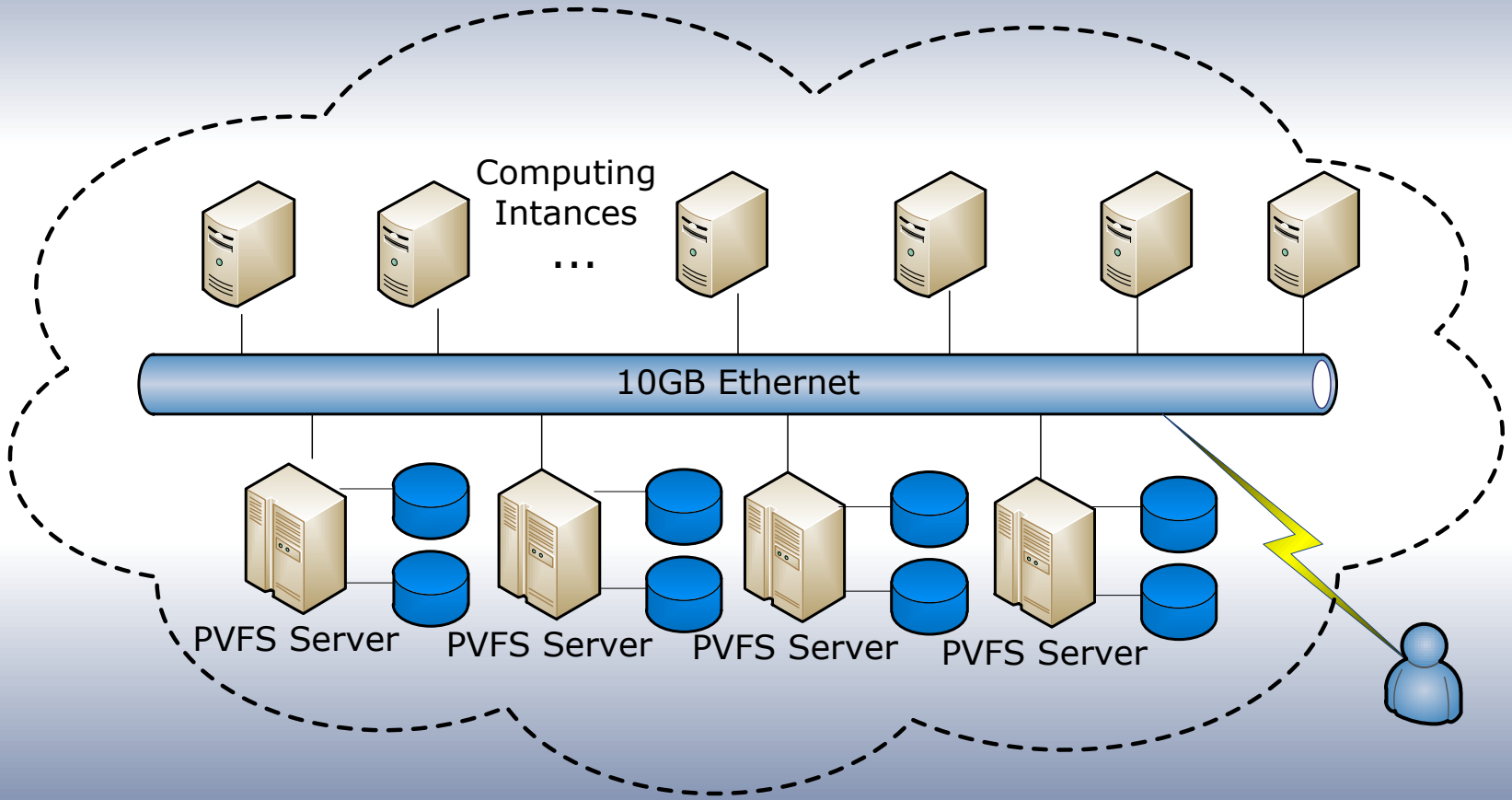
There is 1 PVFS I/O server deployed in one dedicated instance.

2 Dedicated PVFS I/O Servers



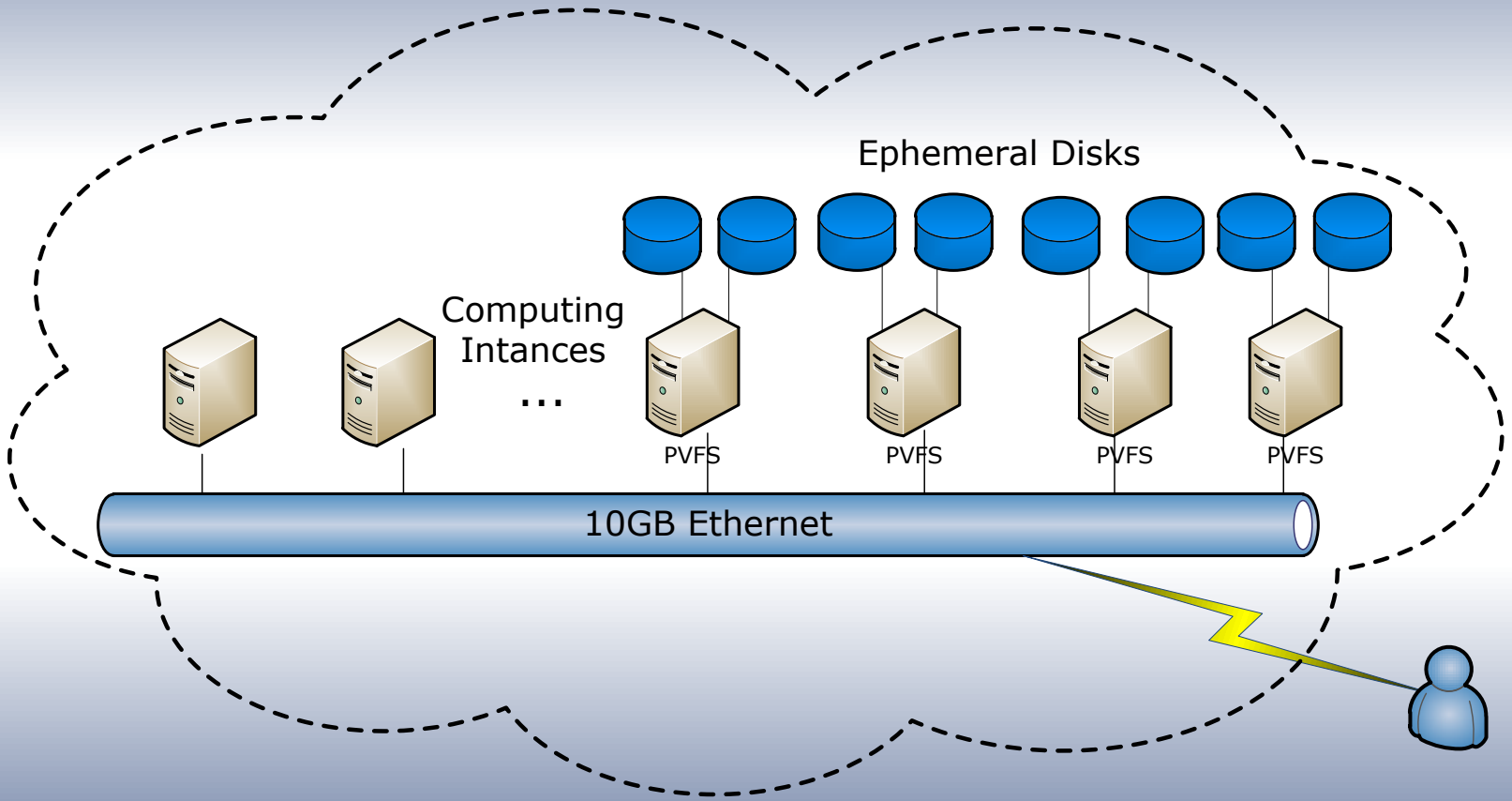
There are 2 PVFS I/O servers deployed in two dedicated instances.

4 Dedicated PVFS I/O Servers



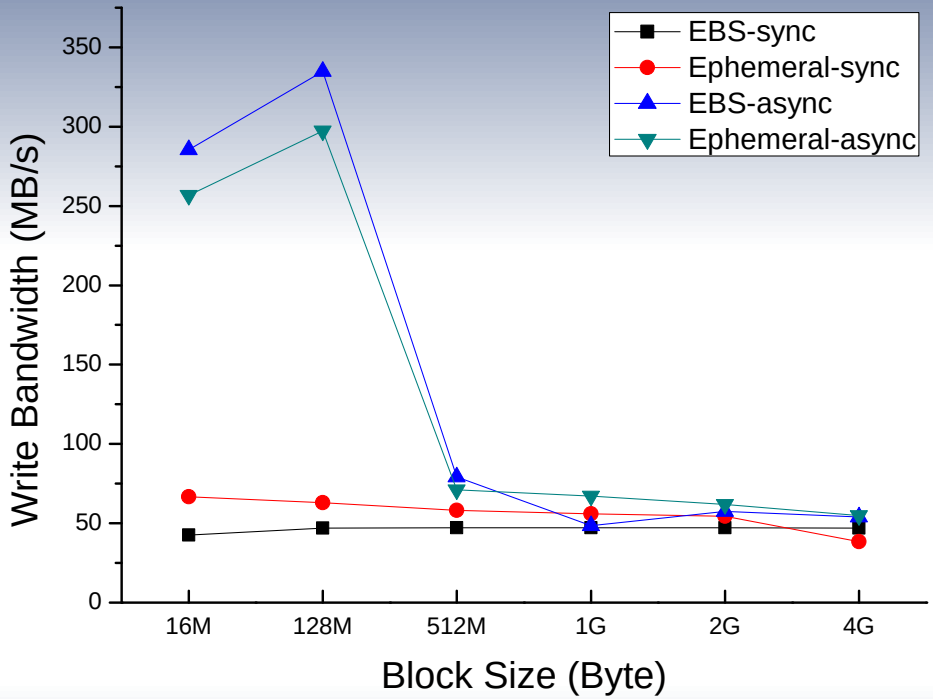
There are 4 PVFS I/O servers deployed in the dedicated instances, low untilized.

4 Part-time PVFS I/O servers



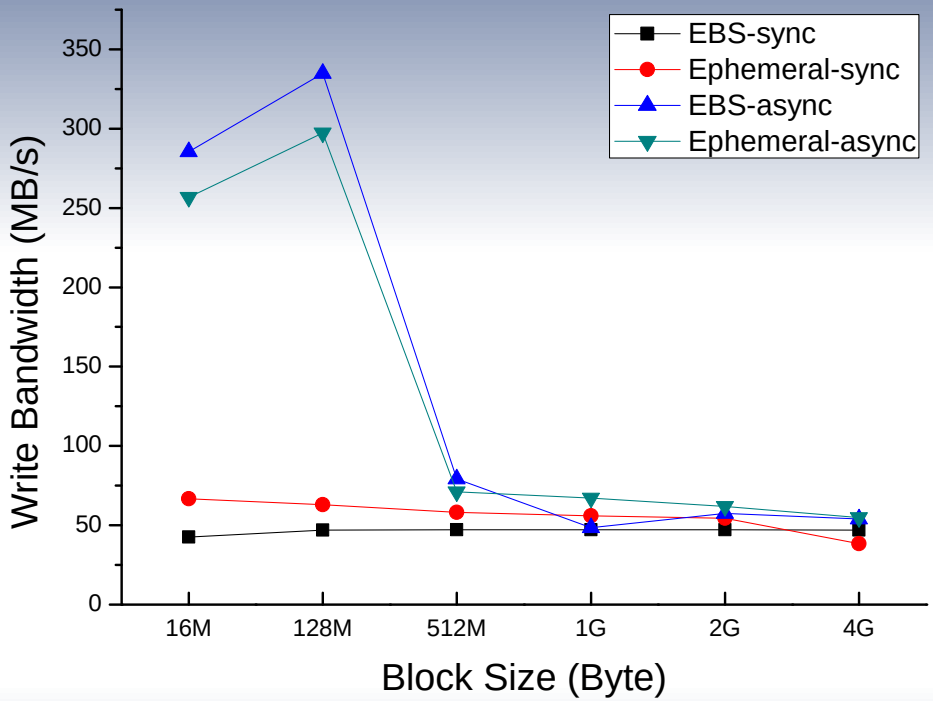
There are 4 PVFS I/O servers deployed in the computing instances, working part-time.

Options: sync mode and device



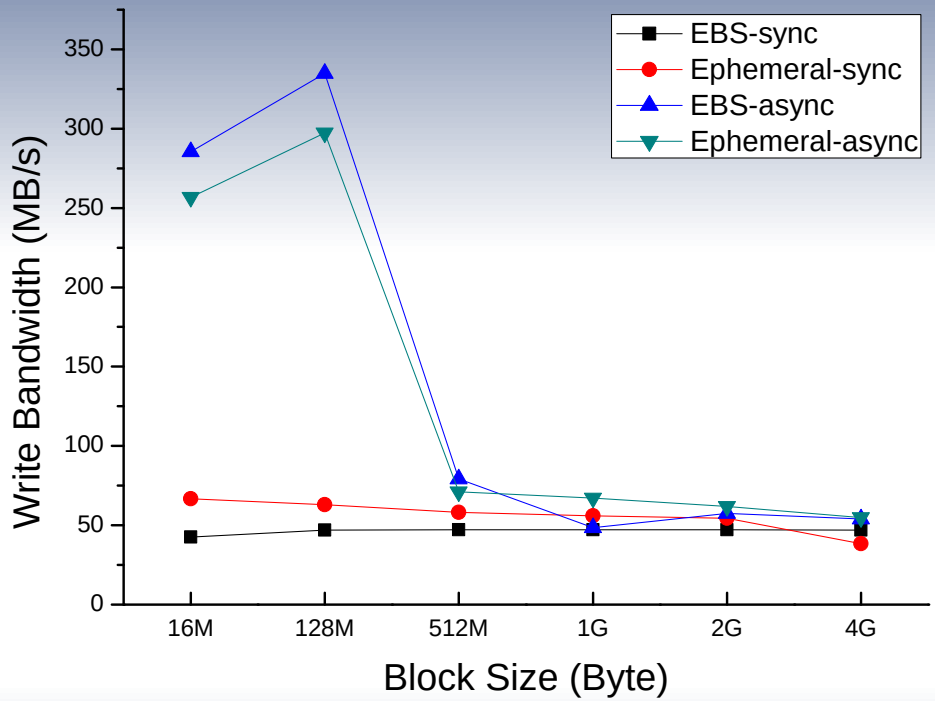
- Test by IOR benchmark in 16 processes at 2 nodes.

Options: sync mode and device



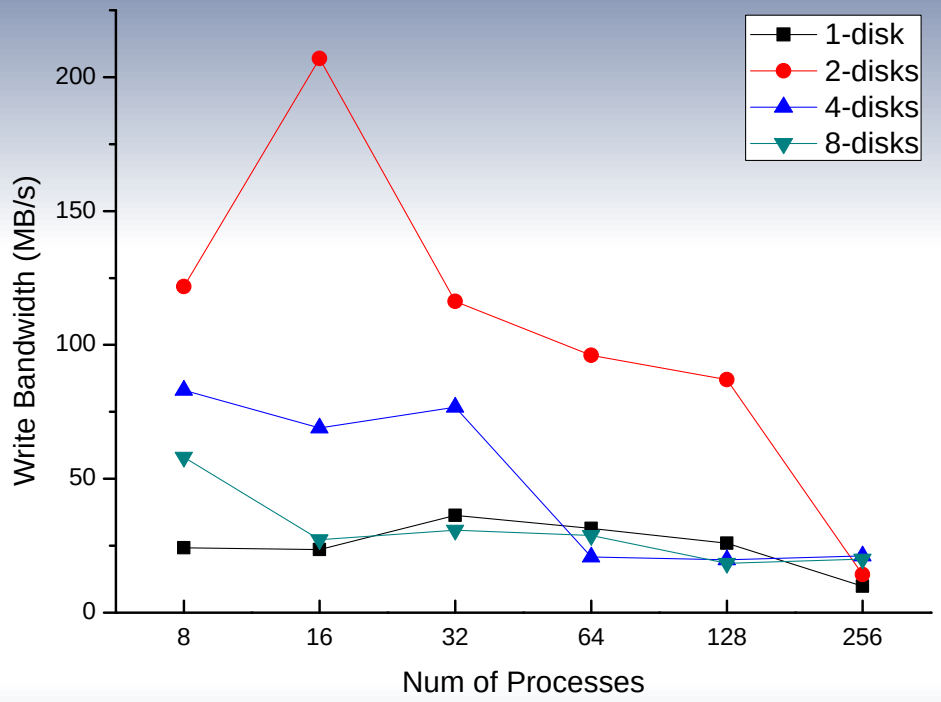
- Test by IOR benchmark in 16 processes at 2 nodes.
- Observations:
  - No obvious difference between EBS and ephemeral

Options: sync mode and device



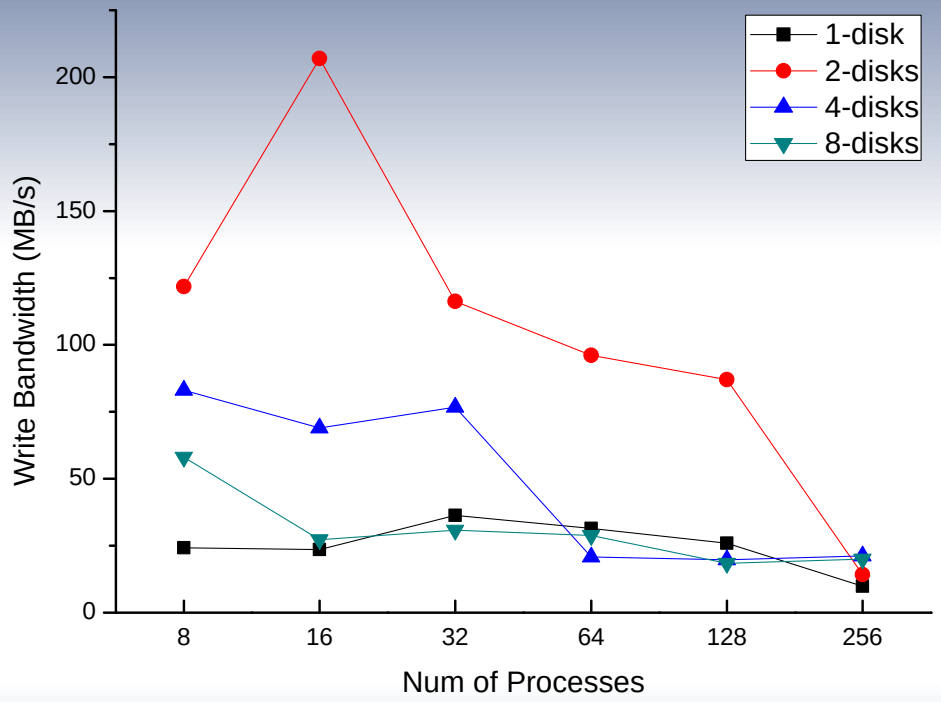
- Test by IOR benchmark in 16 processes at 2 nodes.
- Observations:
  - No obvious difference between EBS and ephemeral
  - `Async` mode prefers small block sizes

NFS Write Bandwidth



- Combining 1,2,4,8 EBS disks into a software RAID0

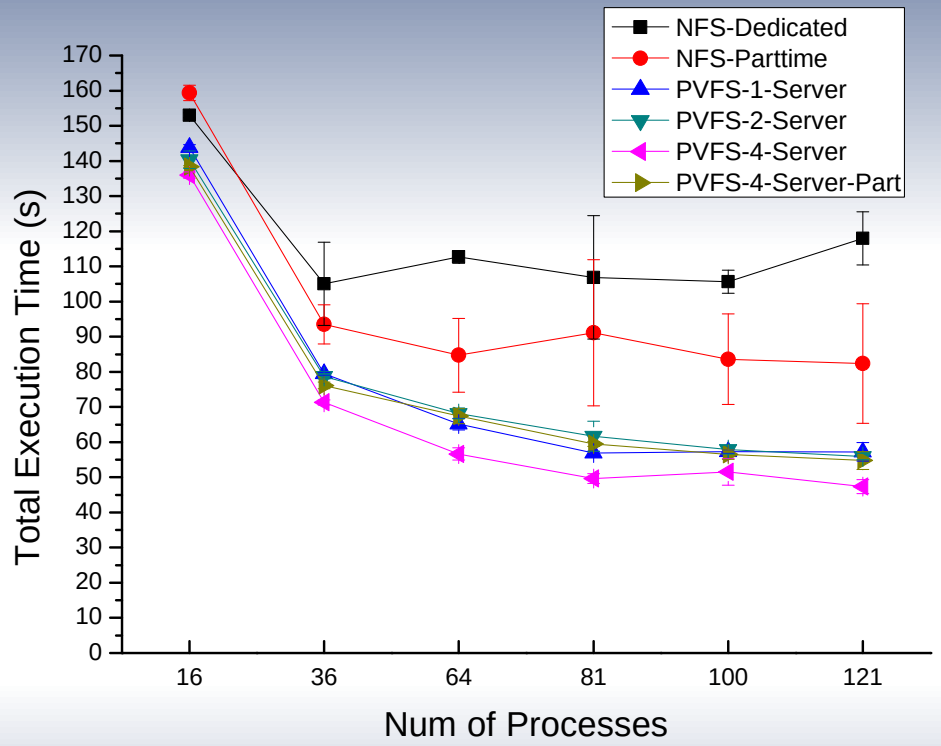
NFS Write Bandwidth



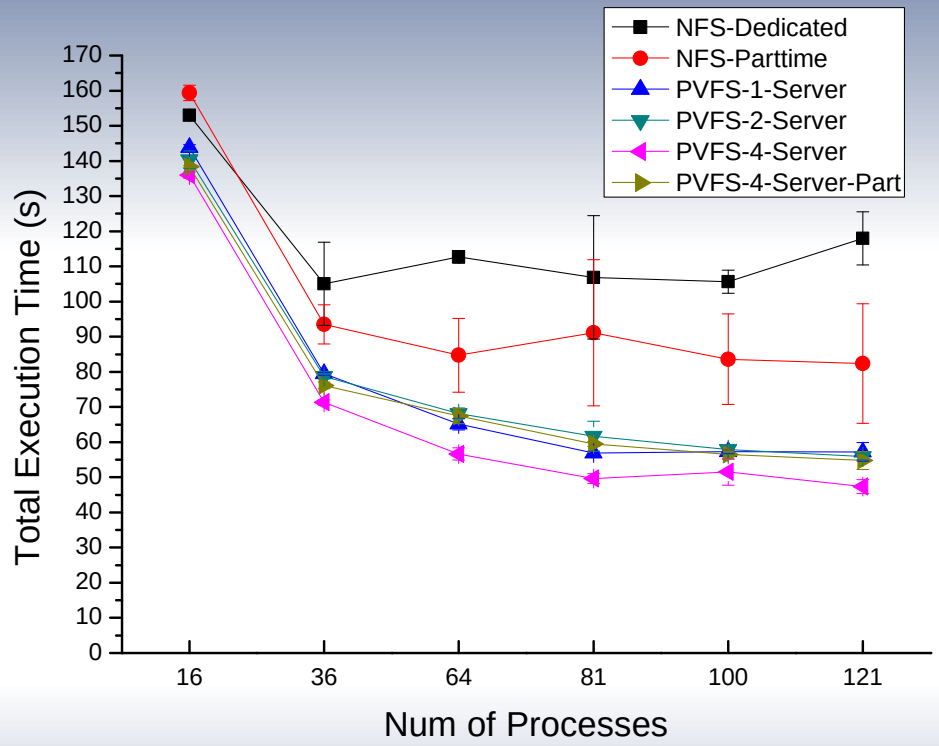
- Combining 1,2,4,8 EBS disks into a software RAID0
- The RAID0 doesn't scale well
  - possibly because of the virtualized layer

Outline

BTIO: CLASS=C, SUBTYPE=full

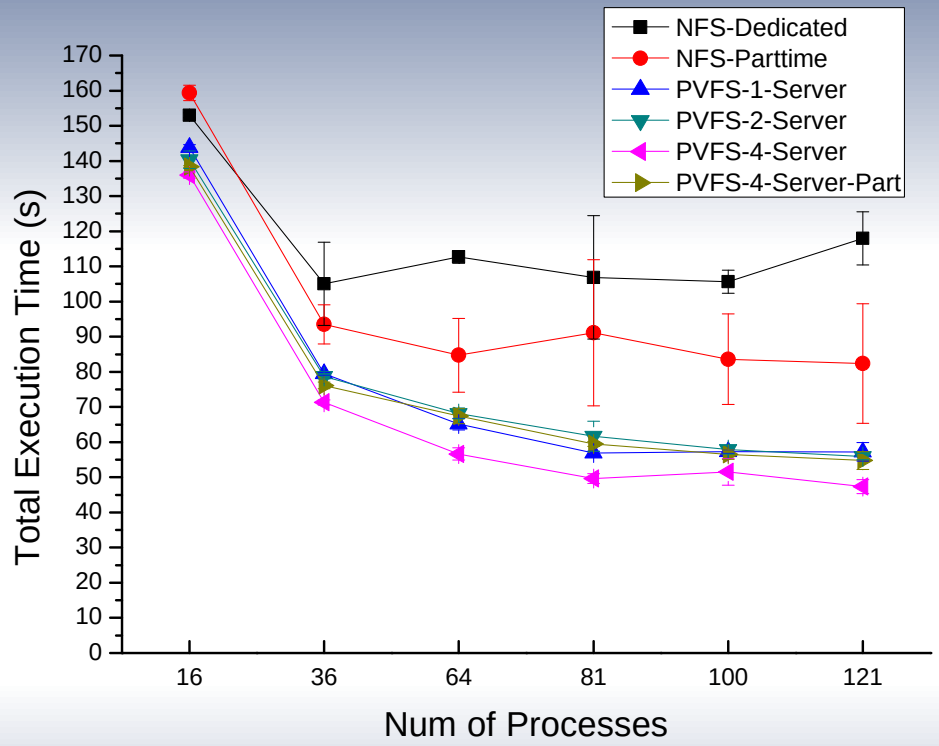


BTIO: CLASS=C, SUBTYPE=full



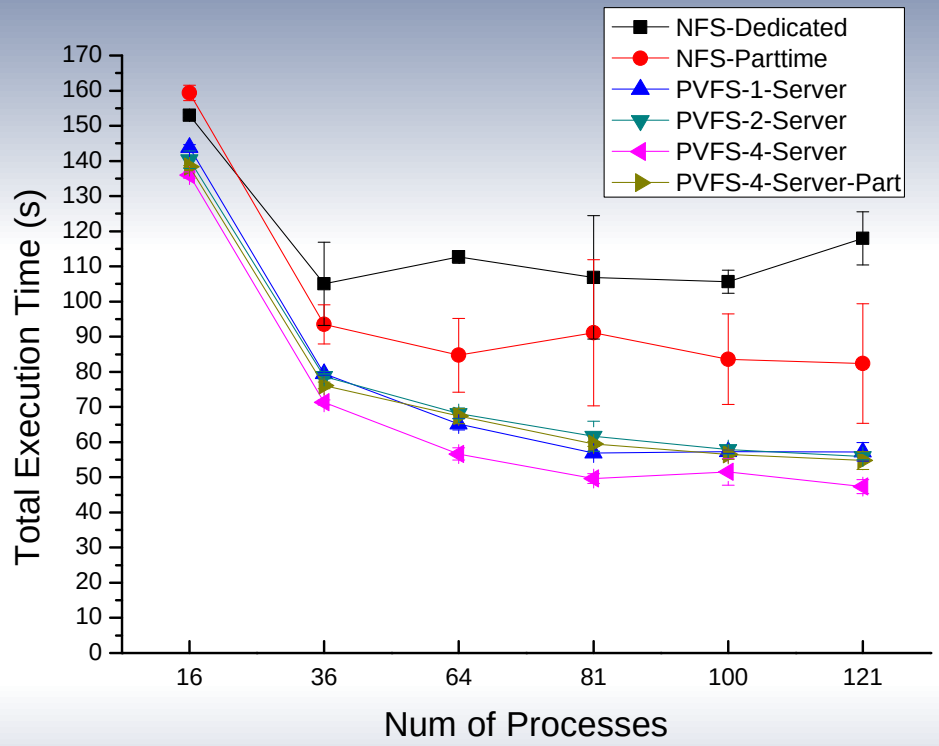
- PVFS outperforms NFS configurations all the time

BTIO: CLASS=C, SUBTYPE=full



- PVFS outperforms NFS configurations all the time
- PVFS scales up by adding more I/O servers

BTIO: CLASS=C, SUBTYPE=full

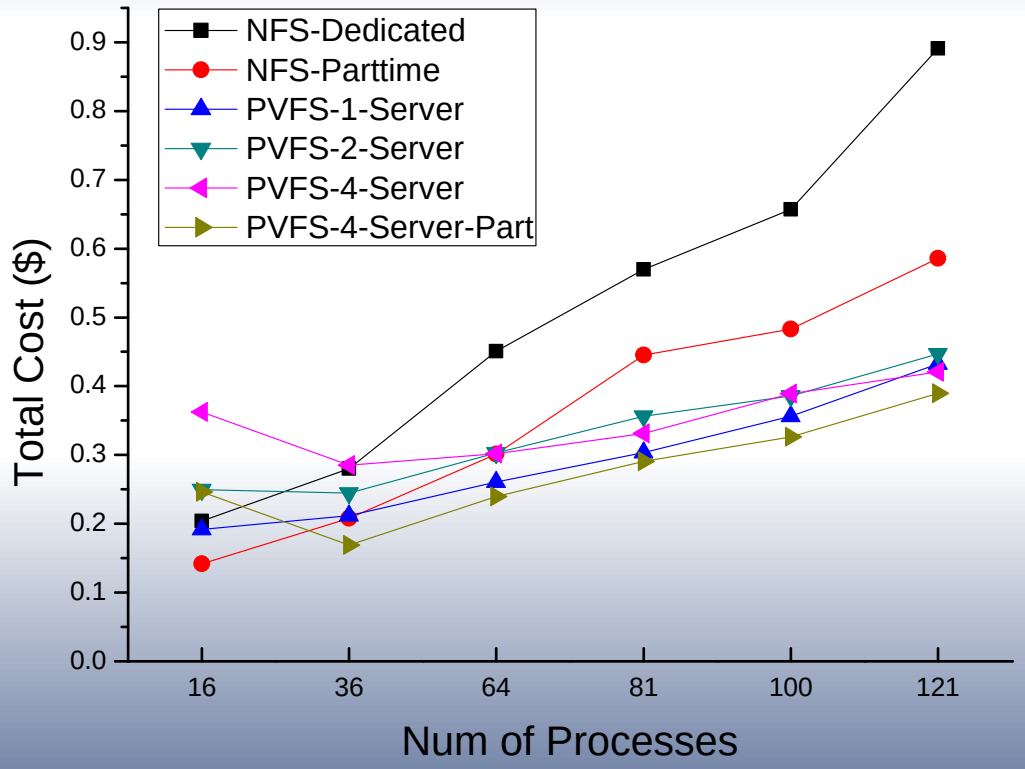


- PVFS outperforms NFS configurations all the time
- PVFS scales up by adding more I/O servers
- Parttime I/O servers provide pretty good performance

BTIO: Total Cost Analysis

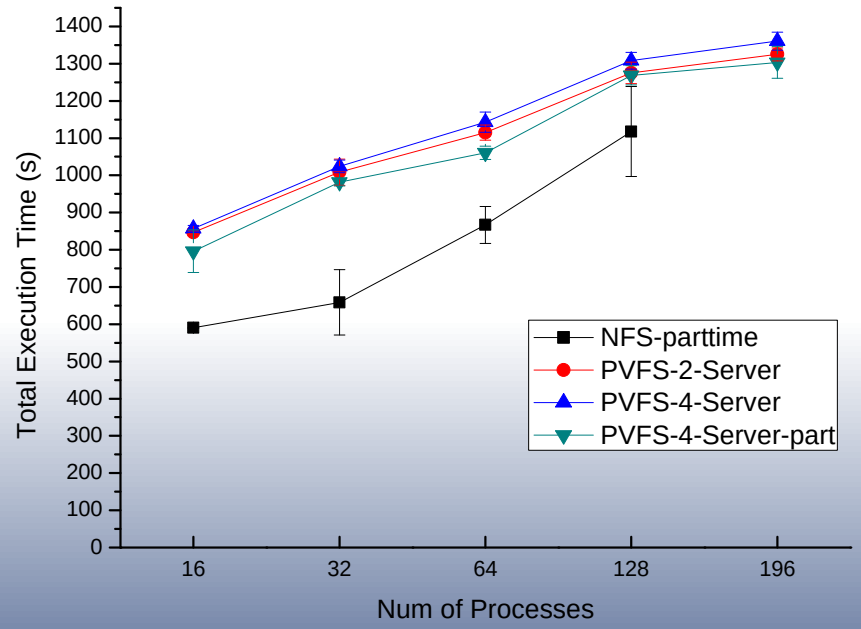
Cost calculation

$$Cost(\$) = Num_{computing\ instances} * Run\_time * 1.6/3600$$



POP: Parallel Ocean Program

- Process 0 carries out all I/O tasks via POSIX interface, which is very different from BTIO
- POP does not scale on EC2, due to its heavy communication with small messages

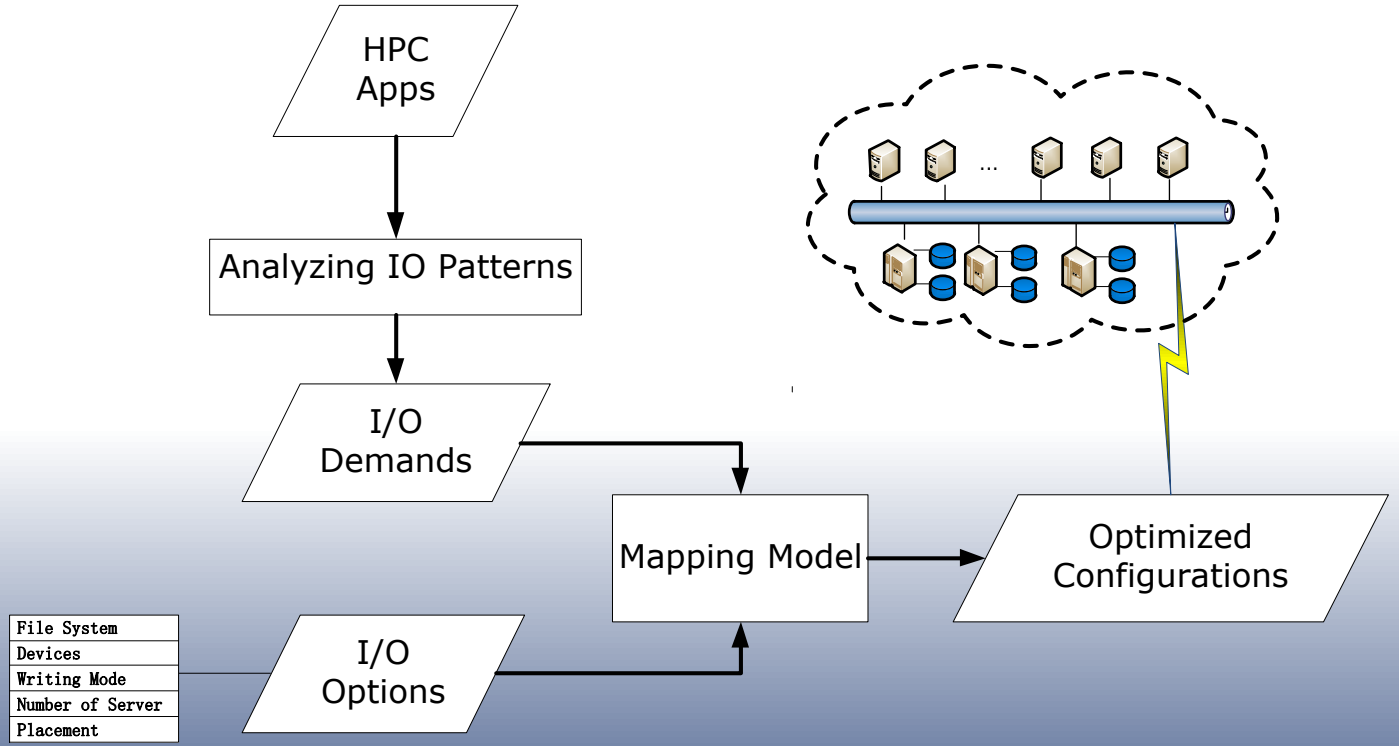


Outline

Future Work

The Problem

Configure the I/O system for a HPC app **automatically**



Conclusion

- Cloud enables users to build per-application I/O systems
- Our preliminary results hint that
  - HPC app behaves differently with different I/O system configurations in cloud
  - Configuration per app depends on its I/O access pattern and concurrency
- Tradeoff: cost vs. efficiency

Conclusion

- Cloud enables users to build per-application I/O systems
- Our preliminary results hint that
  - HPC app behaves differently with different I/O system configurations in cloud
  - Configuration per app depends on its I/O access pattern and concurrency
- Tradeoff: cost vs. efficiency

Thank you!