# The Design and Implementation of AQuA: An Adaptive Quality of Service Aware Object-Based Storage Device

Joel Wu and Scott Brandt
Department of Computer Science
University of California Santa Cruz

MSST2006

May 17, 2006


# What is Storage QoS?

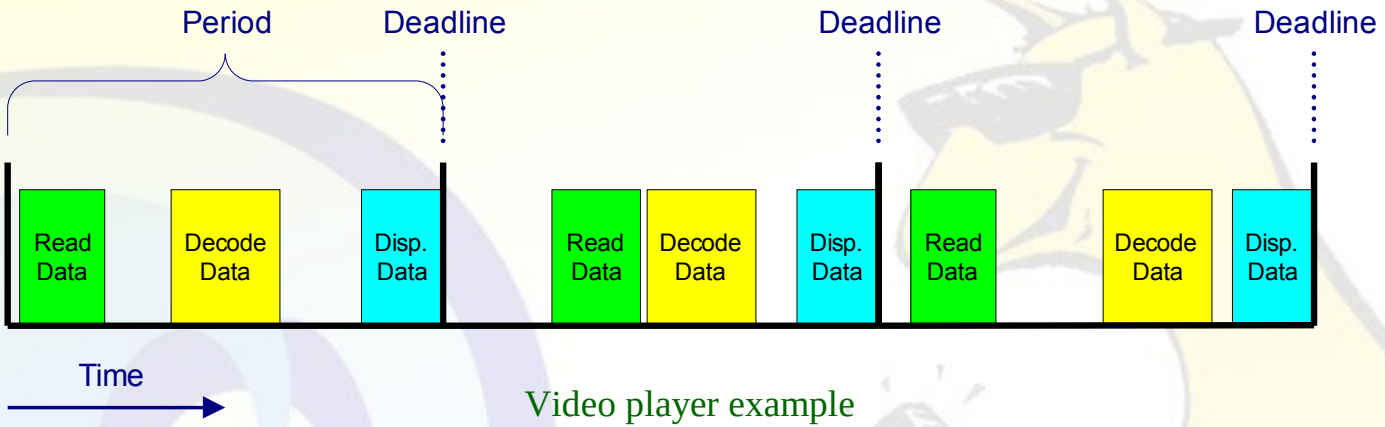
- Storage system quality of service can have multiple dimensions: availability, performance, reliability, and even security

- We define QoS in term of performance (data rate)

  QoS support: Ability to assure a certain level of performance (data rate) that users can obtain from the storage system.


# Why storage QoS?

- Storage-bound applications with timing constraints




Video player example

- Consolidation of storage results in larger and more complex shared storage systems
  - Unrelated workload and multiple organizations/users sharing the same storage resource
  - During overload, applications must compete for access and interfere with each other


# How to achieve QoS?

- Provisioning: Ensure enough resource to go around
  - Automated design tools

- Limitations of provisioning:
  - Require detailed knowledge on expected workloads
  - Slow to react
  - Storage workloads are transient and bursty, provisioning for worst-case scenario can be prohibitively expensive

Adequate provisioning is necessary but not enough

Need additional solution to provide QoS assurance


# Object-Based Storage System (Ceph)

**Object-Based Model**

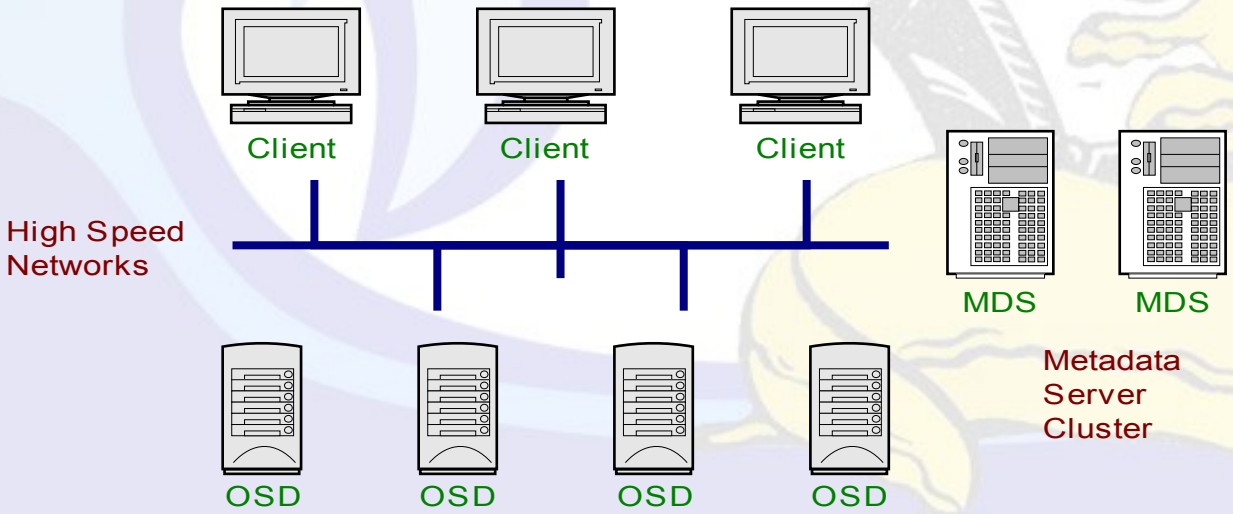
- Offloads handling of low-level storage details to the storage devices
- Storage device accessed through object interface
- Metadata management decoupled from data management

Three major components in Ceph:

- Clients
- Metadata server cluster
- Object-Based Storage Devices (OSD)

Ceph OSD:

- Intelligent and autonomous storage device with P2P capability
- Consists of CPU, memory, NIC, and block-based disk(s)



# QoS support in Ceph

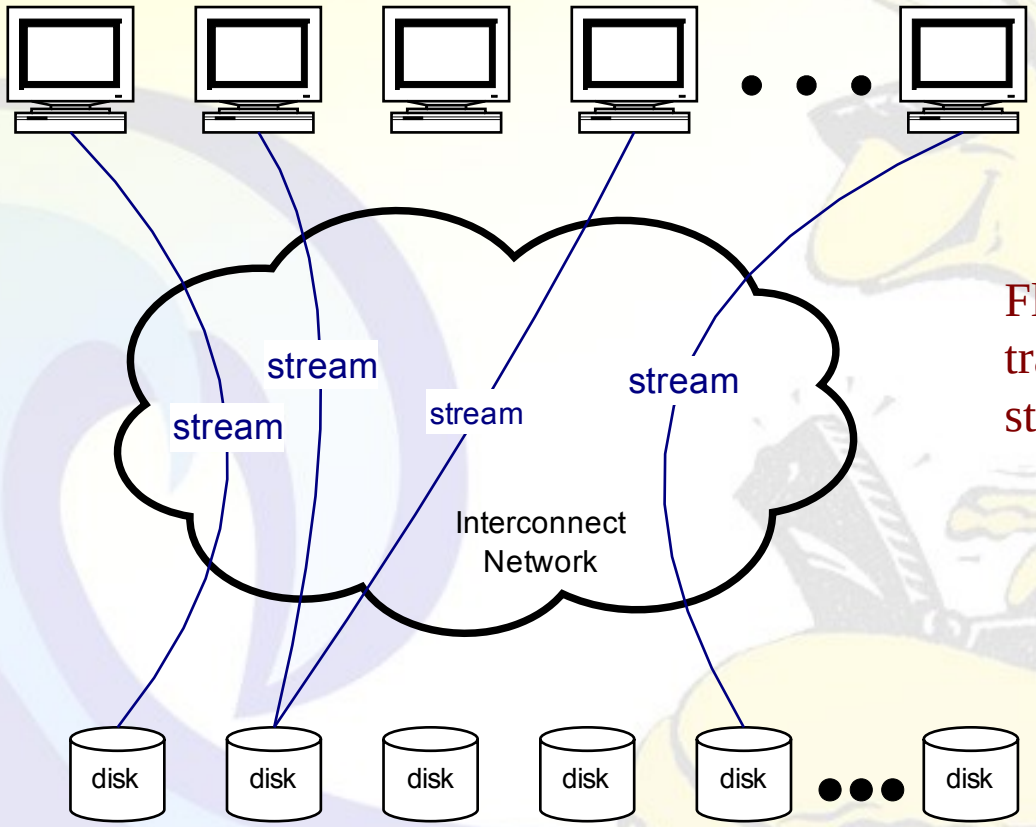
***QoS-capable:*** Ability to provide storage bandwidth assurance

**QoS framework for Ceph**

- Provide underlying framework that can be utilized to achieve higher level QoS goals
- Build from QoS-aware OSDs
- Provide assurance for different *classes*
  - Class: Generic term referring to an aggregate of storage traffic sharing the same QoS goal
  - Semantic and granularity defined by administrator
  - Limit interference between different classes


# Streams in non-striped distributed storage

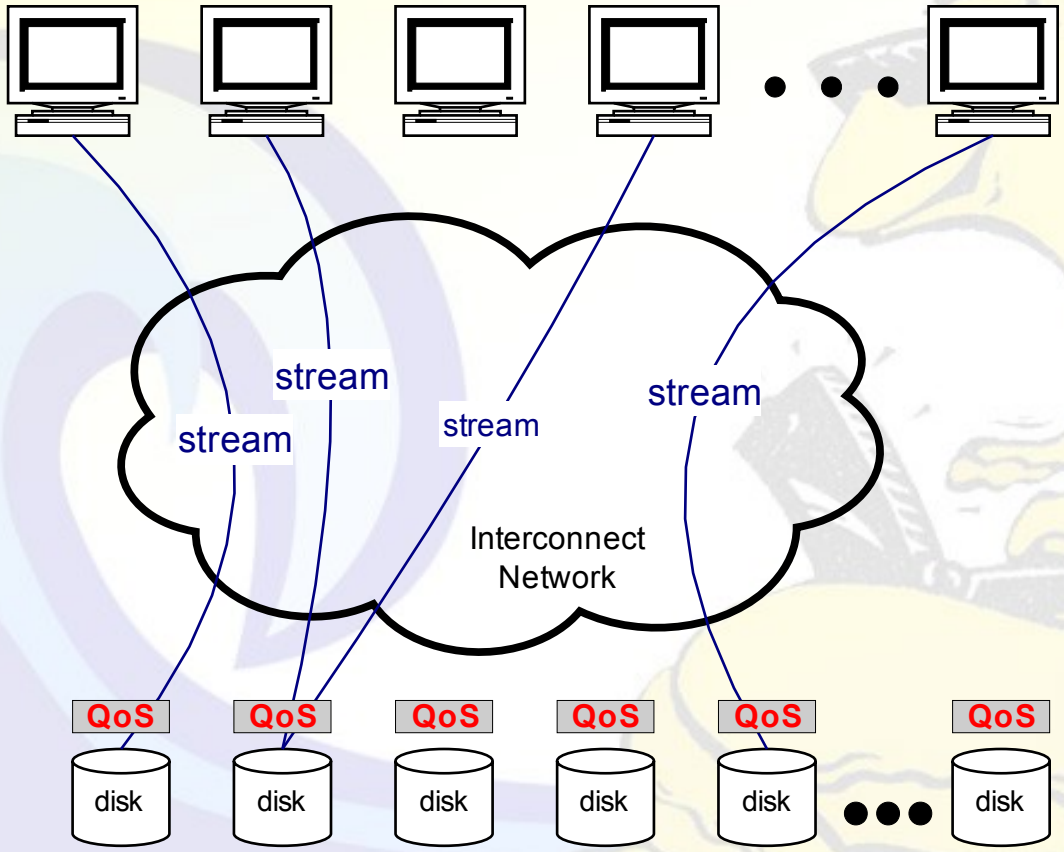
- Stream: An end-to-end data path




Flows in non-striped traditional distributed storage system

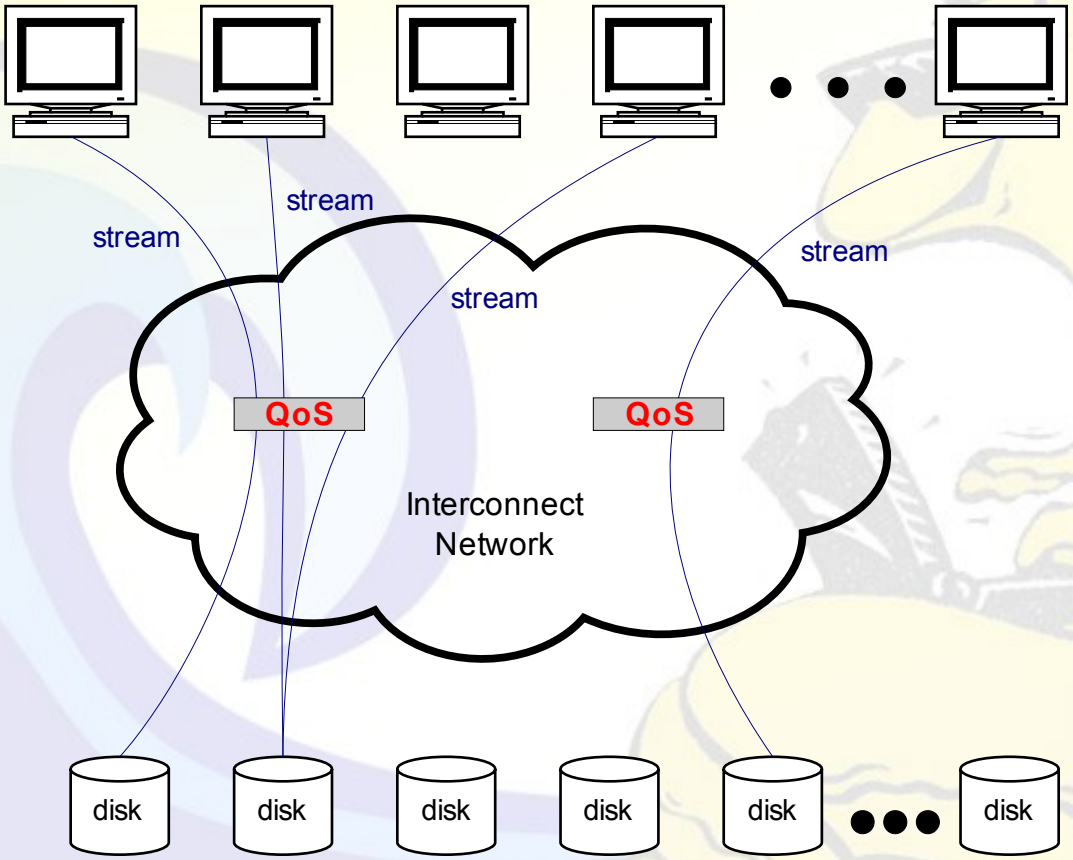

# QoS mechanism for individual device

- QoS mechanism per device: Façade, Zygaria, Sundaram03, XFS GRIO2



# QoS mechanism in SLEDS

- SLEDS [Chambliss'03]

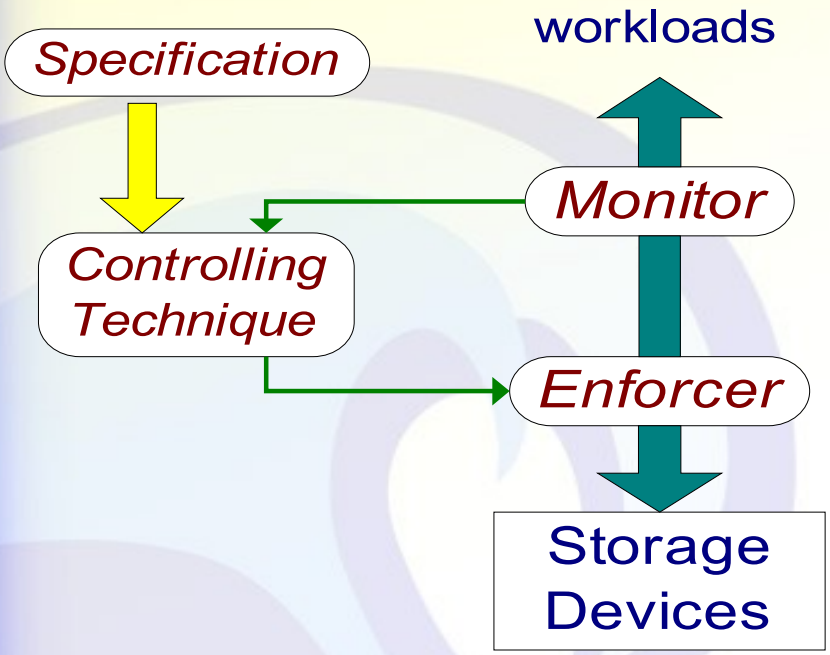


# Throttling model

Generalization of QoS: A resource has a capacity and is redistributed among a number of consumers to satisfy QoS goals

- Specification
  - Allows the desired *quality level* to be specified (IOPS, bytes/sec, latency, request rate).
  - Can be associated with different entities (clients/hosts, groups of clients, application class)
- Monitor
  - Monitoring the rate different entities are receiving data.
  - Can monitor different parameters of the system (queue length, average completion time, response time, throughput, bandwidth)
- Enforcer
  - Mechanism that shapes bandwidth by throttling (I/O scheduler, leaky bucket)
- Controlling technique
  - Decides when and how much to throttle bandwidth (heuristics, control-theoretic)


# Throttling model

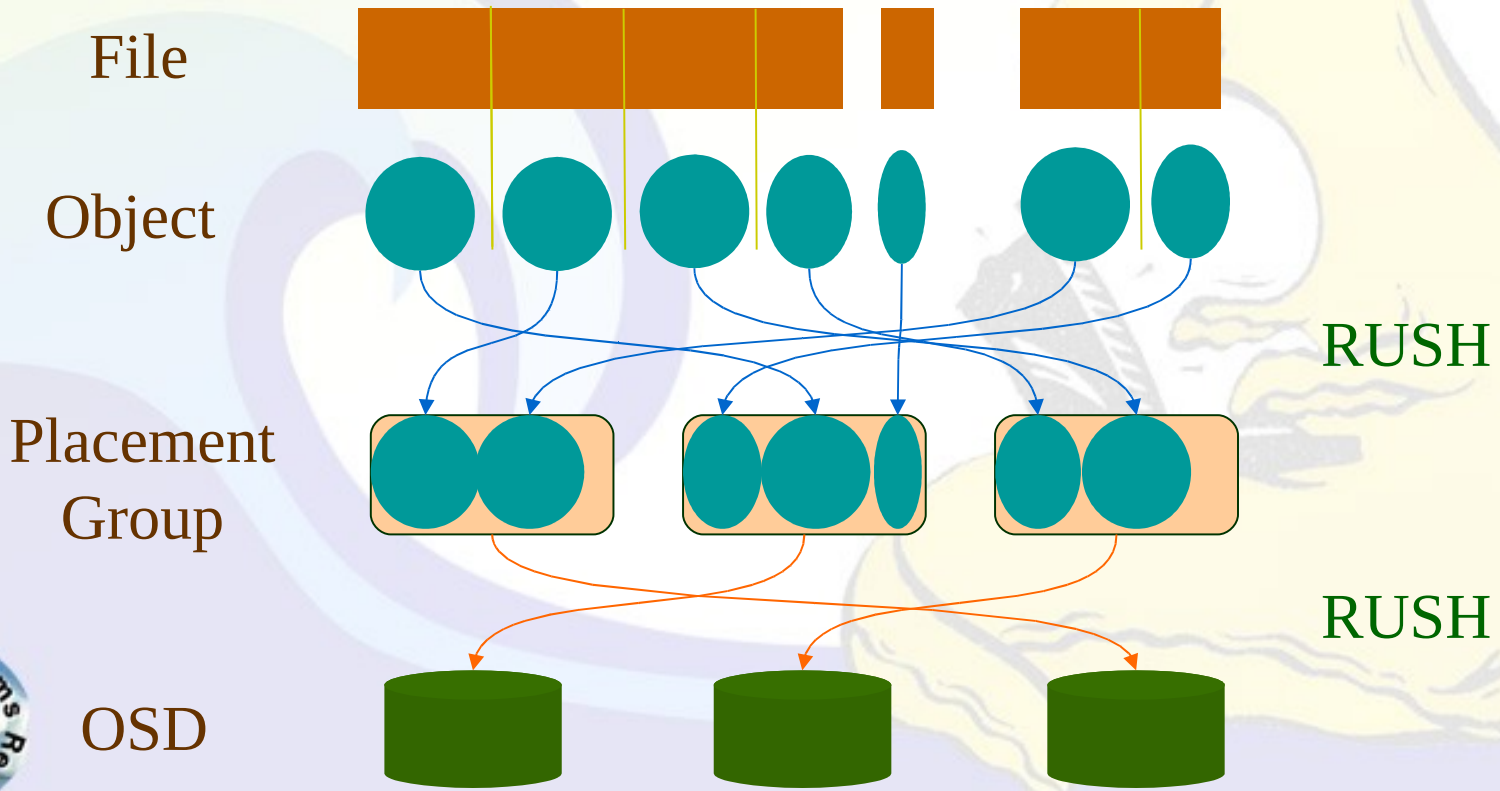



1. Specify the desired rate. May involve admission control.
2. Monitor the actual rate received
3. If actual rate received is less than desired rate, throttle competing traffic
4. Ease throttling if actual rate received is satisfactory.


# Striping of Objects in Ceph

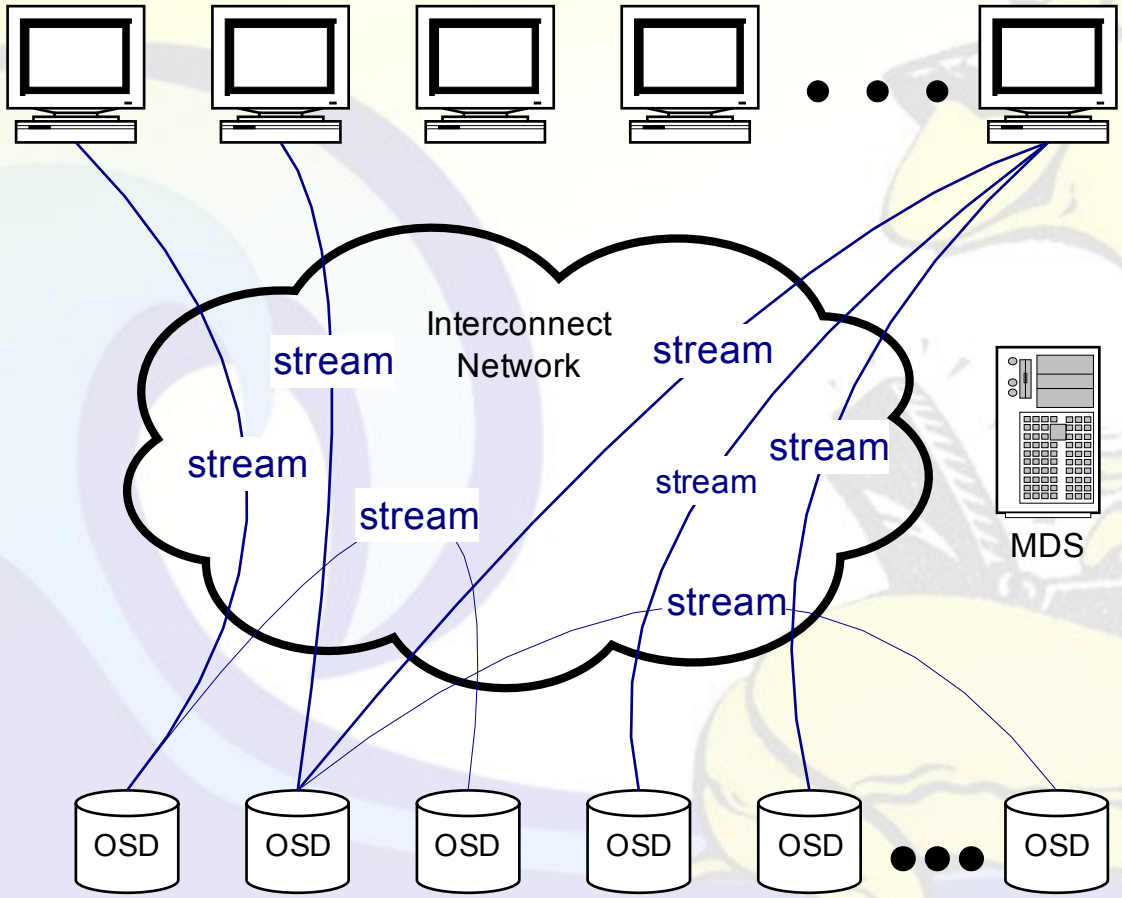
Files are broken up into objects and striped across OSDs

- Mapping by pseudo-random RUSH algorithm
- Goal is load balancing



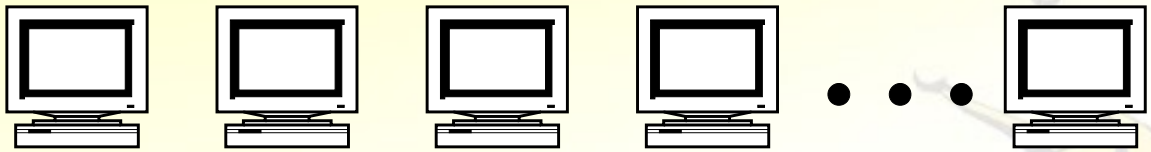
# Streams in Ceph

- Striping - accessing a single file may involve multiple streams
- Peer-to-peer replication/recovery traffic between OSDs




Data flows in Ceph


# Ceph QoS Architecture



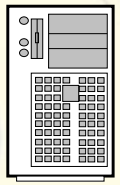
clients

QoS

administrator

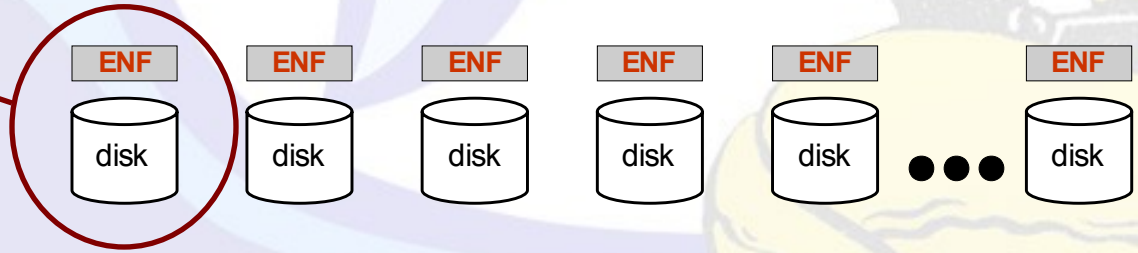
spec

Interconnect Network



MDS

AQuA OSD





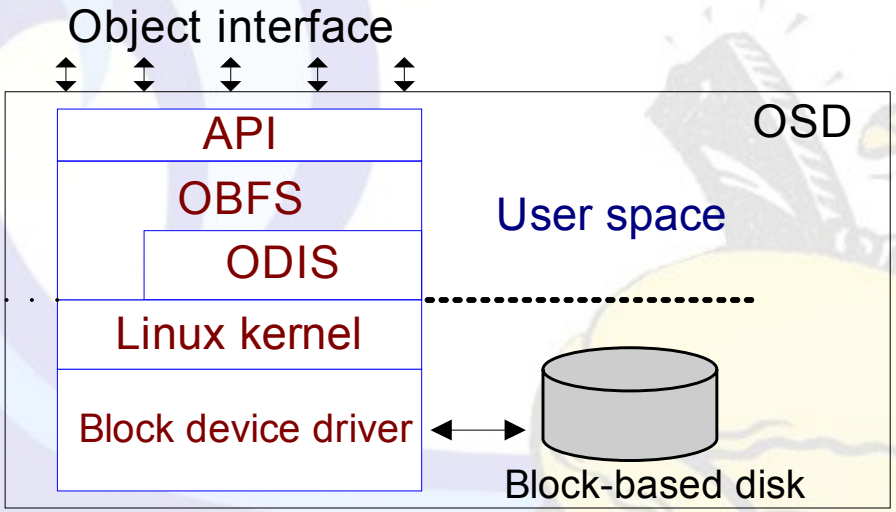
# Basic QoS support in OSD

- Fundamental capability that all QoS-aware systems possess is the ability to *shape* disk bandwidth – to redistribute resource
- Higher level goals can be decomposed into why, when, how, which, and how much to shape disk traffic (enforcer component)
- Giving OSD this capability - Push and encapsulate complexity into OSD

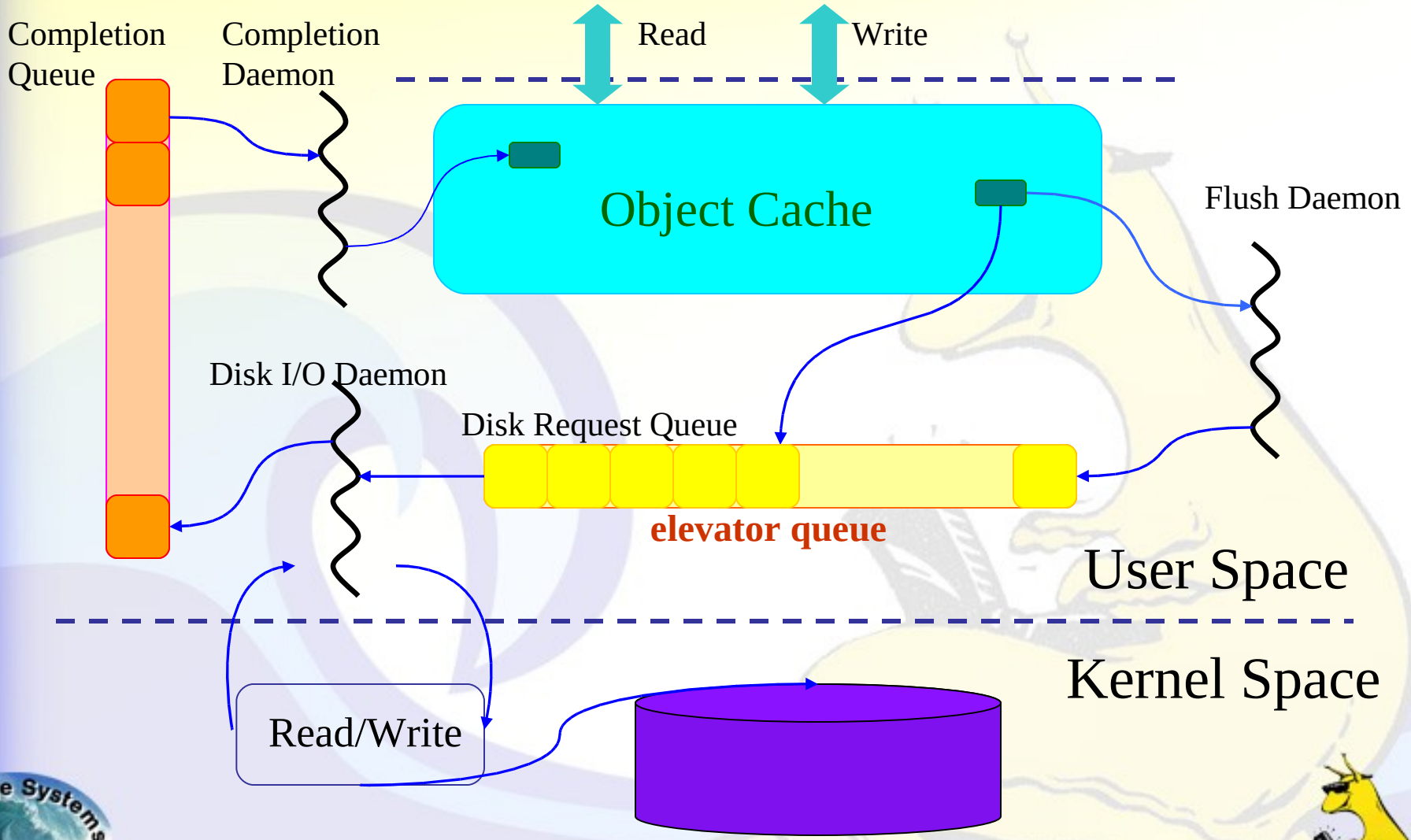

# AQuA OSD:

**A**daptive **Qu**alify of service **A**ware **O**bject-based **S**torage **D**evice

- Enforce bandwidth allocation among classes
- Based on Object-based file system (OBFS) – Small and efficient file system for managing block-based hard disks [Wang 2004]
- Object Disk I/O Scheduler (ODIS) – QoS-aware disk scheduler incorporated into object-based file system
- AQuA = OBFS + ODIS + Bandwidth Maximizer

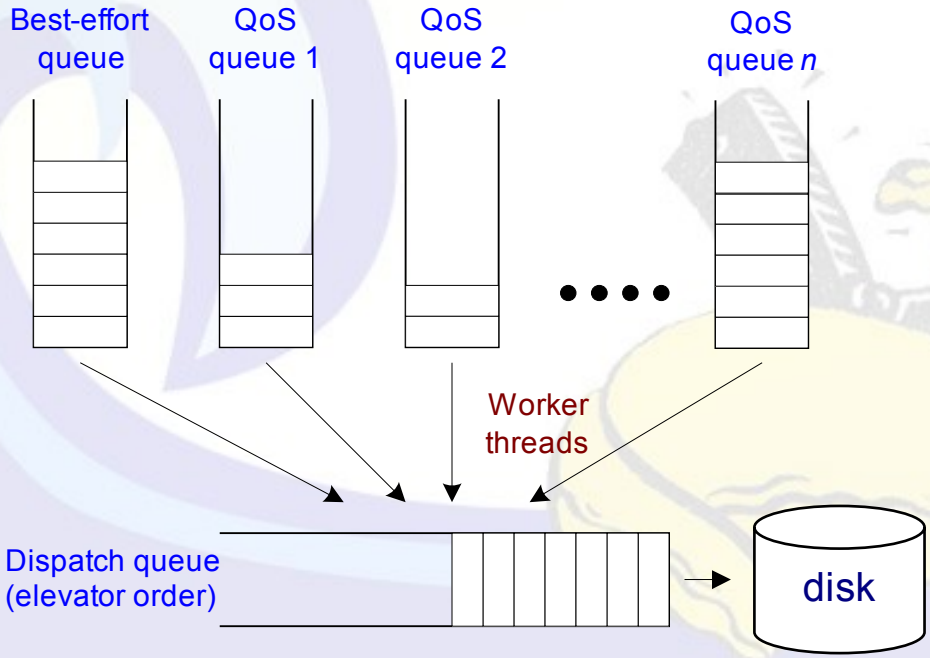


# OBFS Internal Structure [Wang 2004]

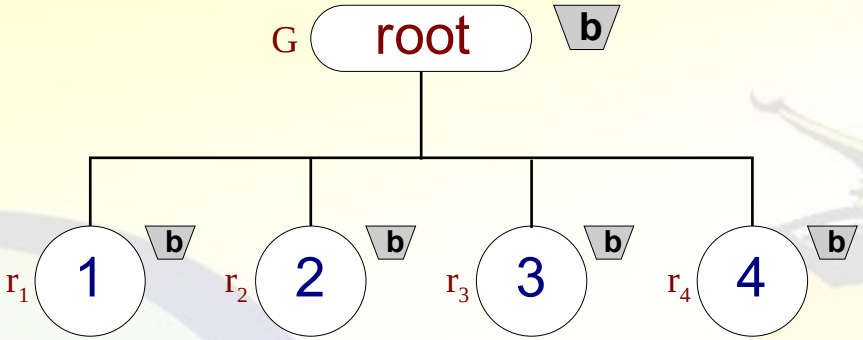


# Object Disk I/O Scheduler (ODIS)

- Limit interference between classes
- Allows reservations and reclaims unused bandwidth
- Replaces standard elevator scheduler in OBFS
- *n* QoS queues, 1 best-effort queue, and 1 dispatch queue
- Number of QoS queues is dynamic
- Each queue has an associated worker thread
- Specification of a class determines the rate requests are moved to the dispatch queue

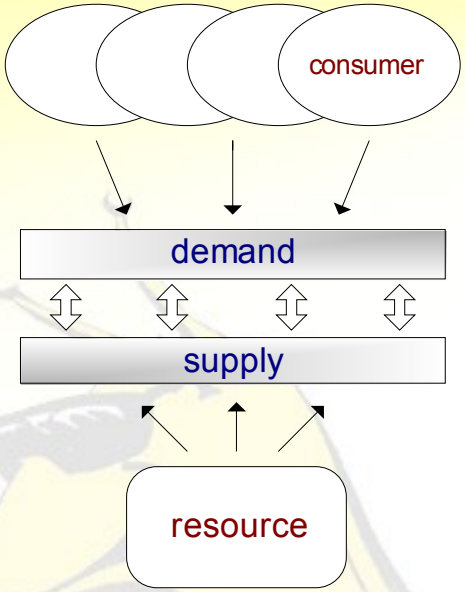


# HTB Implementation




- Implemented with hierarchical token buckets (HTB) [NOSSDAV'05]
  - Each node has an associated bucket and token rate
  - Root node represents the aggregate bandwidth (token rate) of the disk (G)
- Each token represents 1 KB of bandwidth
  - Leaf node tokens replenished at a rate corresponding to reservation
  - Root node tokens replenished with a fixed rate
- Root node facilitates sharing and reclamation of unused bandwidth
- HTB can be of more than two level


# Stateful disk




- Disk drive is stateful
- Total resource (available bandwidth) is not fixed
  - Dependent on the workload
- How to assure resource allocation when the amount of available resource varies?
  1. Disk model (RT disk schedulers)
  2. Proportional allocation (Sundaram'03, Cello, YFQ)
  3. Assume fixed resource (DFS, XFS GRIO2, Zygaria)
  4. Adaptation - Throttling model (Façade, SLEDS)


# QoS assurance vs. total throughput

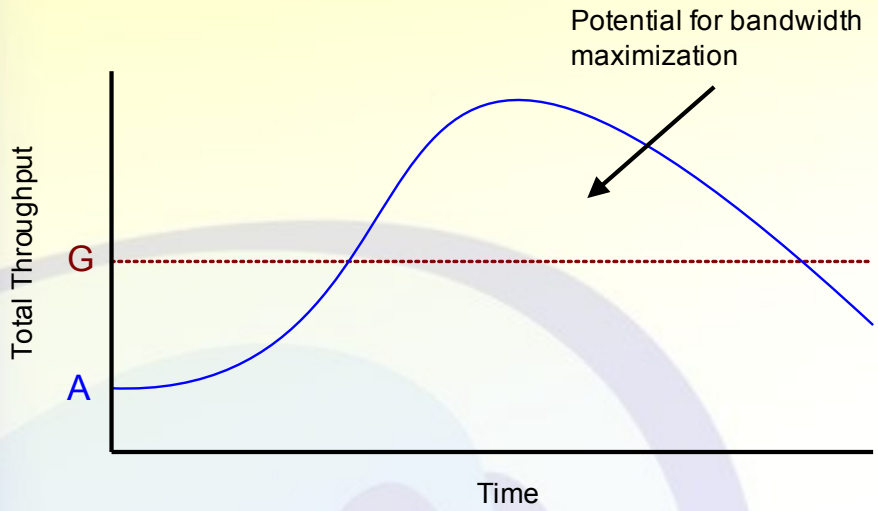
- Estimation of total bandwidth: tradeoff between “tightness” of QoS assurance and total throughput

Aggressive
- Looser QoS assurance
- Over-commitment
- Higher utilization

Conservative
- Tighter QoS assurance
- Underutilization
- Reduced total throughput

- AQuA:
  1. Conservative estimate of total bandwidth to ensure stringent QoS assurance
     - Bandwidth allocated by ODIS
  2. Minimize underutilization of disk with dynamic adaptation
     - Bandwidth maximizer attempts to maximize total throughput


# Bandwidth Maximizer




**A**: Achievable Throughput

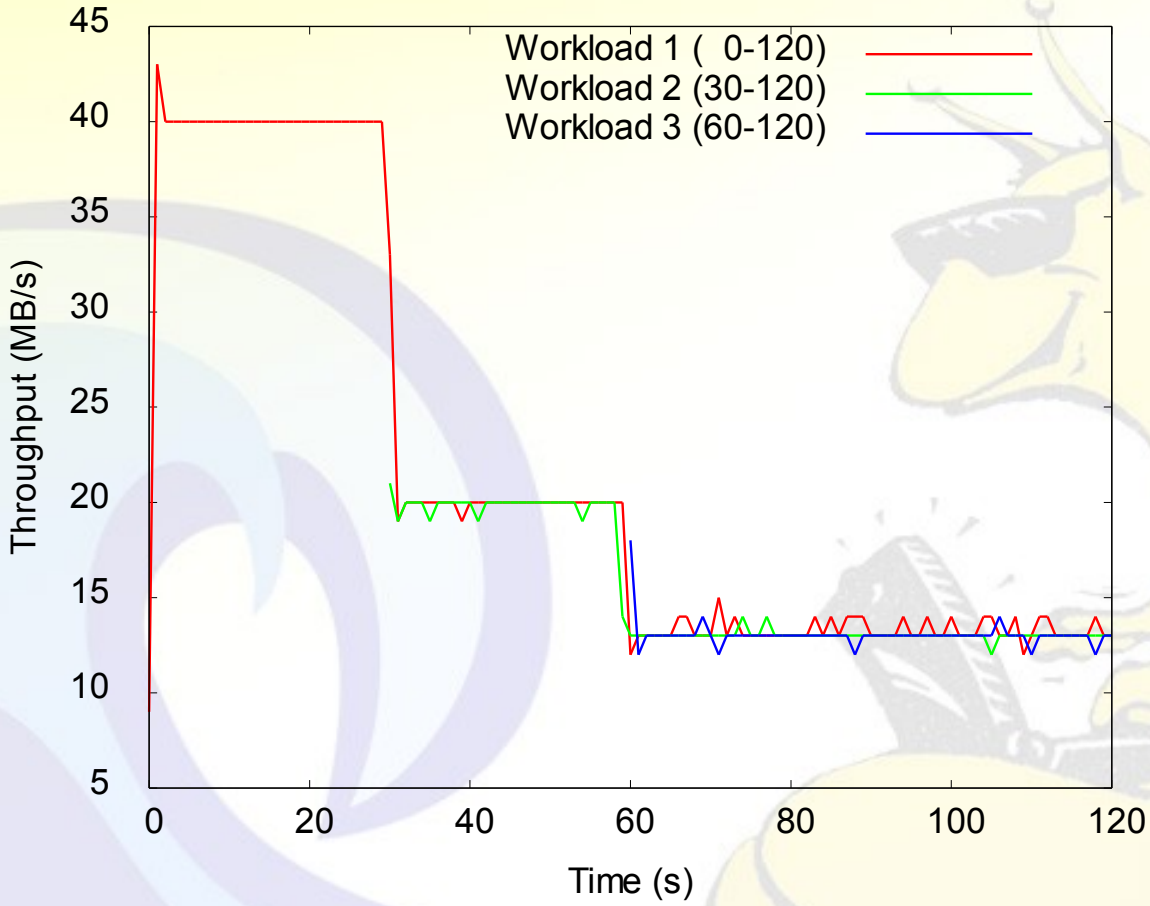
**G**: Global Token Rate (Estimation of total disk bandwidth)

1. When the demand is not capped by either A or G. $O < A$ and $O < G$
2. When the demand is capped by $A$, and $A < G$
3. When the demand is capped by $G$, $G < A$.

O: Observed (actual) rate

- Proof-of-concept implementation: The heuristic adjusts $G$ by monitoring the status:
  - If disk throughput is capped by $G$ and no QoS commitments are violated, it increase $G$.
  - If disk throughput is capped by $G$, $G$ has been increased to greater than its original value, and some QoS commitments are violated, decrease $G$.
  - $G$ will not drop below its original value.

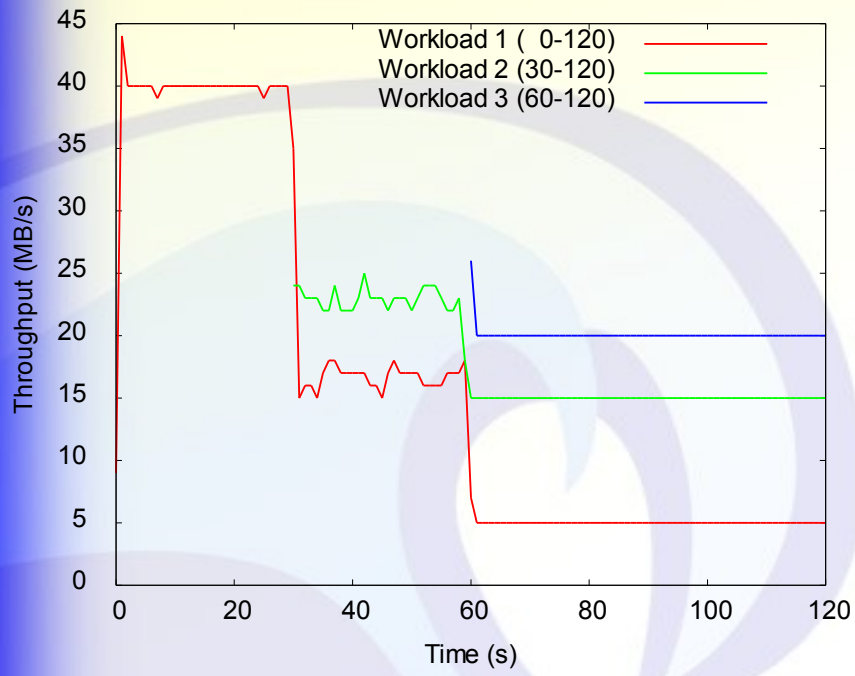

# AQuA: Results



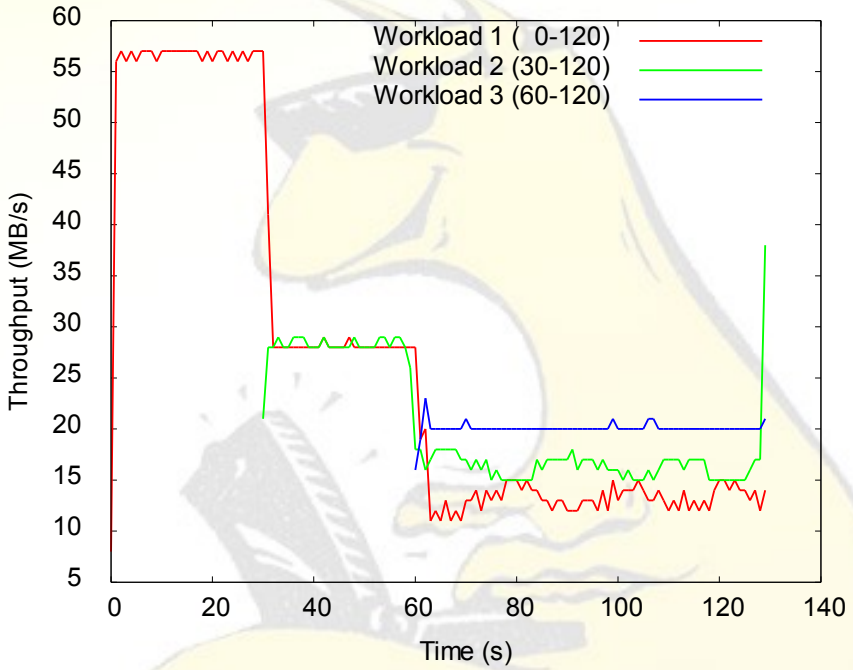

Without reservation and assurance


# AQuA: Results

With ODIS




With ODSI and Bandwidth Maximizer




Reservation at 5, 15, and 20 MB/s


# Conclusion and Future Work

- QoS-aware OSD
  - Encapsulating bandwidth shaping mechanism within OSD by combining OBFS with QoS-aware disk scheduler
  - Adaptive heuristic minimizes underutilization
  - Basic building block of the overall QoS framework
- Future Works will shift from using OBFS to EBOFS (Extend-Based Object File System)
- More intelligent adaptation method
- Global QoS framework