



# Σχεδίαση και Υλοποίηση Μηχανισμού Κρυφής Μνήμης για Κατανεμημένο Σύστημα Αποθήκευσης σε Περιβάλλον Υπολογιστικού Νέφους

1. Καλημέρα σας, ονομάζομαι Αλέξιος Πυργιώτης  
Θα σας παρουσιάσω τη διπλωματική μου με τίτλο:..
2. Ακούγεται κάπως περίεργο στα ελληνικά...  
αυτό που πραγματεύεται είναι την δημιουργία ενός caching  
μηχανισμού για το Archipelago, ένα distributed, storage layer
3. Συγκεκριμένα, στην παρουσίαση αυτή θα μιλήσουμε για τον cached,  
δηλαδή τον caching μηχανισμό μας, αλλά και για το synapsed, ένα  
συμπληρωματικό εργαλείο που στόχος του είναι να δώσει στον  
cached δικτυακές δυνατότητες



Αλέξιος Πυργιώτης

Εθνικό Μετσόβιο Πολυτεχνείο

October 15, 2013

## 1. Ο κορμός της παρουσίασης είναι ο εξής:

- Αρχικά, παρουσιάζουμε κάποια εισαγωγικά που αφορούν το background της εργασίας μας. Αναφέρουμε τι είναι το Synnefo, τι είναι η υπηρεσία okeanos και τι είναι το Αρχιπέλαγο
- Έπειτα, δείχνουμε τον τρόπο με τον οποίο η υποδομή μας χειρίζεται αιτήματα δεδομένων από ένα VM.

# Contents

[Introduction](#)[Request handling](#)[Caching](#)[Cached design](#)[Cached evaluation](#)

# Table of Contents

Introduction

Request handling

Caching

Cached design

Cached evaluation



Ας ξεκινήσουμε με την παρούσα κατάσταση.  
Το software που τα ξεκίνησε όλα είναι το Synnefo

..by GRNET -> Και φυσικά τα παιδιά που βλέπετε εδώ

- Compute service, είναι η υπηρεσία η οποία προμηθεύει τους χρήστες με VMs και επιτρέπει το χειρισμό τους
- Network service, είναι η υπηρεσία η οποία δίνει τη δυνατότητα στους χρήστες να δημιουργήσουν ιδιωτικά δίκτυα και να συνδέσουν τα VMs τους σε αυτά.
- Storage service, που κοινώς αποθηκεύει τα αρχεία των χρηστών. Στην περίπτωση του Synnefo όμως, έχουμε ένα κοινό σημείο για τα πάντα: είτε είναι αρχεία, είτε δίσκοι των VMS, είτε images
- Image Service, υπεύθυνο για το deployment ενός VM από ένα image. Επίσης, κάνει και παραμετροποιήσεις (παράδειγμα ssh κλειδιά)

## Synnefo



Open source, production-ready, cloud software.  
Designed since 2010 by GRNET.

Synnefo, as most cloud software, has the following services:

- Compute Service
- Network Service
- Storage Service
- Image Service
- Identity Service



## okeanos



- IaaS είναι πρακτικά η παροχή εικονικής υποδομής σε χρήστες (δηλαδή πάρε υπολογιστή (VM), δίκτυα, αρχεία κτλ)
- Δωρεάν για τους Ακαδημαϊκούς σκοπούς, ήδη γίνονται εργαστήρια στο EMP και απ' αυτό το εξάμηνο σε άλλες σχολές

- IaaS service
- Targeted at the Greek Academic and Research Community
- Designed by GRNET
- In production since 2011



## okeanos



- IaaS είναι πρακτικά η παροχή εικονικής υποδομής σε χρήστες (δηλαδή πάρε υπολογιστή (VM), δίκτυα, αρχεία κτλ)
- Δωρεάν για τους Ακαδημαϊκούς σκοπούς, ήδη γίνονται εργαστήρια στο EMP και απ' αυτό το εξάμηνο σε άλλες σχολές

- IaaS service
- Targeted at the Greek Academic and Research Community
- Designed by GRNET
- In production since 2011
- ...and of course powered by Synnefo.



# Table of Contents

[Introduction](#)[Request handling](#)[Caching](#)[Cached design](#)[Cached evaluation](#)

## What is request handling?

Τι είναι η διαχείριση των αιτημάτων ενός VM?

Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>

... το storage μας <κλικ>

και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>

Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>

Όχι, χρειαζόμαστε επίσης **FIXME:**







## What is request handling?

Τι είναι η διαχείριση των αιτημάτων ενός VM?

Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>

... το storage μας <κλικ>

και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>

Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>

Όχι, χρειαζόμαστε επίσης **FIXME:**





## What is request handling?

Τι είναι η διαχείριση των αιτημάτων ενός VM?

Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

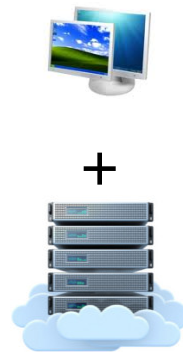
Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>

... το storage μας <κλικ>

και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>

Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>

Όχι, χρειαζόμαστε επίσης **FIXME:**





## What is request handling?

Τι είναι η διαχείριση των αιτημάτων ενός VM?

Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>

... το storage μας <κλικ>

και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>

Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>

Όχι, χρειαζόμαστε επίσης **FIXME:**





= ?

- Policy enforcement?
- Storage agnosticity?

## What is request handling?

Τι είναι η διαχείριση των αιτημάτων ενός VM?

Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>

... το storage μας <κλικ>

και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>

Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>

Όχι, χρειαζόμαστε επίσης **FIXME:**



+



= ?

- Policy enforcement?
- Storage agnosticity?

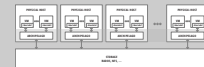


Request handling

Our solution

Our solution

Archipelago



Key features: 1) Software-defined 2) Distributed ) Modular  
Copy-On-Write Storage agnostic

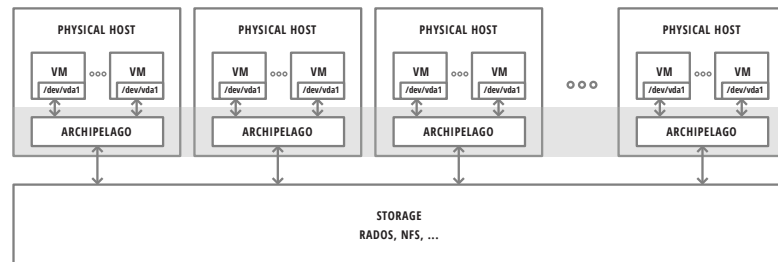
Η λύση που χρησιμοποιήσαμε είναι το Archipelago

- Software-defined: αν και είναι ένα όρος μαρκετινγκ, εμείς κανονικά. Σημαίνει με το software ΟΡΙΖΕΙΣ το storage (εφαρμογή policy, αλλαγή πορείας του request)
- τρέχει σε πολλούς κόμβους
- αποτελείται από διακριτά κομμάτια
- κάνει CoW (εξήγησε ότι τα images είναι λίγα, τα VMs πολλά, όπως όταν ένα process κάνει fork)
- μπορούμε χρησιμοποιήσουμε ότι θέλουμε

## Request handling

### Our solution

### Archipelago



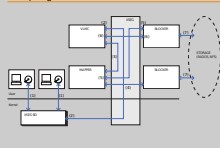
Key features: 1) Software-defined 2) Distributed ) Modular  
Copy-On-Write Storage agnostic



Request handling

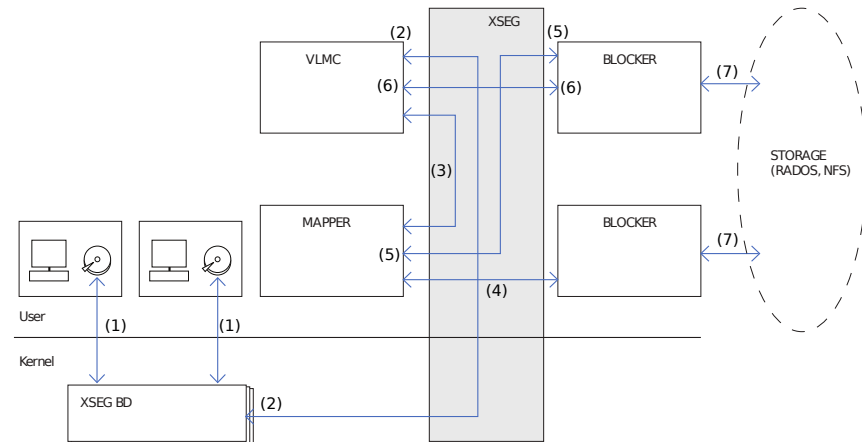
Archipelago Architecture

Archipelago Architecture



- Το VM στέλνει αίτημα στο δίσκο του, ο δίσκος είναι εικονικός, θα το δει ο hypervisor (εξήγησε τι είναι ο hypervisor) και θα το στείλει στον δίσκο που το έχουμε πει. (xsegbd)
- 2) **FIXME:**

## Archipelago Architecture



└ Request handling

└ RADOS

#### RADOS

The object store component of Ceph filesystem.

Key features:

- Replication
- Fault tolerance
- Self-management
- Scalability

## Request handling

# RADOS

The object store component of Ceph filesystem.

Key features:

- Replication
- Fault tolerance
- Self-management
- Scalability



#### RADOS

The object store component of Ceph filesystem.

##### Key features:

- Replication
- Fault tolerance
- Self-management
- Scalability

##### Speed issues:

VM with page-cache: > 90MB/s, < 1ms  
VM without page-cache: < 7MB/s, 10ms

# RADOS

The object store component of Ceph filesystem.

Key features:

- Replication
- Fault tolerance
- Self-management
- Scalability

Speed issues:

VM with page-cache: > 90MB/s, < 1ms

VM without page-cache: < 7MB/s, 10ms





#### RADOS

The object store component of Ceph filesystem.

##### Key features:

- Replication
- Fault tolerance
- Self-management
- Scalability

##### Speed issues:

VM with page-cache: > 90MB/s, < 1ms

VM without page-cache: < 7MB/s, 10ms

Thesis goal: make this faster.

# RADOS

The object store component of Ceph filesystem.

Key features:

- Replication
- Fault tolerance
- Self-management
- Scalability

Speed issues:

VM with page-cache: > 90MB/s, < 1ms

VM without page-cache: < 7MB/s, 10ms

Thesis goal: make this faster.



## Table of Contents

Introduction

Request handling

Caching

Cached design

Cached evaluation



- We have a slow medium
- Add a fast medium in a data path
- Transparently store the data that are intended for the slower medium.
- Profit: later accesses to the same data are faster

## Intro

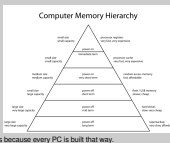
### Solution: Caching

#### Caching is:

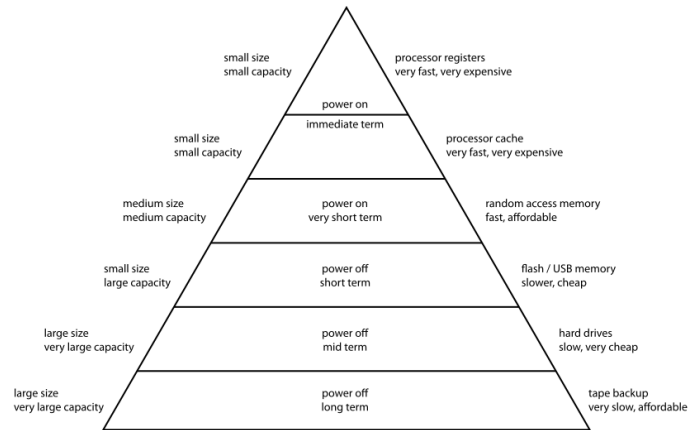
- We have a slow medium
- Add a fast medium in a data path
- Transparently store the data that are intended for the slower medium.
- Profit: later accesses to the same data are faster

#### Sounds familiar?





## Computer Memory Hierarchy



That's because every PC is built that way.



Is there anything to help us?

We are not the first to have speed issues

Facebook, Twitter, Dropbox, every one has hit and surpassed their limits.

There are solutions separated in two categories:

- Block store
- Key-value store

## Caching

Is there anything to help us?

We are not the first to have speed issues

Facebook, Twitter, Dropbox, every one has hit and surpassed their limits.

There are solutions separated in two categories:

- Block store
- Key-value store



- Bcache
- Flashcache
- EnhanceIO

## Block-store caching solutions

Most notable examples:

- Bcache
- Flashcache
- EnhanceIO

Typically scale-up solutions.

Pros: Simple, scale-up

Cons: Unaware of CoW, kernel solutions



- Memcached
- Couchbase

## Key-value caching solutions

Most notable examples:

- Memcached
- Couchbase

Typically scale-out solutions

Pros: Distributed with no SPOF, can utilize unneeded RAM

Cons: Memcached has no persistence, Couchbase cannot use  
RADOS as its backend, more suitable for databases



## Page-cache

What if we used the page-cache?

Pros: Easy to activate, tested, very fast

Cons: Unaware of CoW, no control over it, practically kernel solution





- Most solutions far from Archipelago's logic
- Block store might be good for the storage backend
- Must implement our own solution

## Conclusions

- Most solutions far from Archipelago's logic
- Block store might be good for the storage backend
- Must implement our own solution



# Table of Contents

Introduction

Request handling

Caching

**Cached design**

Cached evaluation



- Create something close to the Archipelago logic
- Measure the best possible performance we can get

- Nativity
- Pluggability
- In-memory
- Low indexing overhead

## Requirements

Design goals for cached:

- Create something close to the Archipelago logic
- Measure the best possible performance we can get

Stricter requirements for cached:

- Nativity
- Pluggability
- In-memory
- Low indexing overhead



└─ Cached design

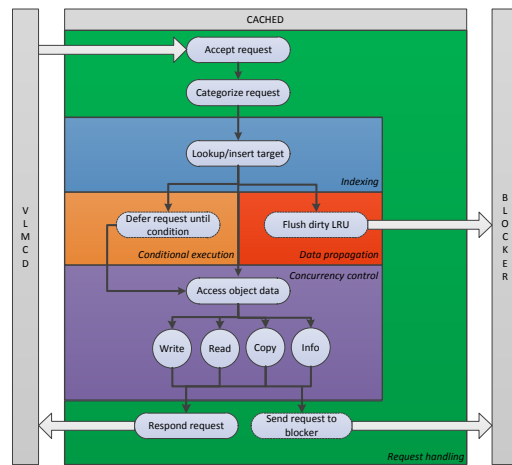
└─ Cached design

Cached design



## Cached design

## Cached design



└─ Cached design

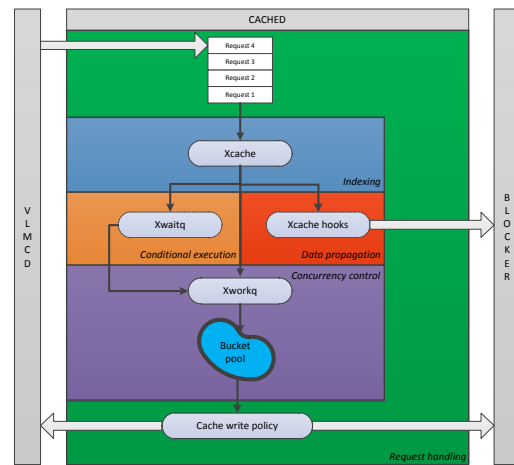
└─ Cached design

Cached design



## Cached design

## Cached design



└─ Cached design

└─ Xcache design

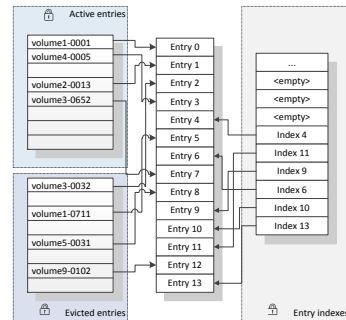
Xcache design



Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks

Cached design

## Xcache design



Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks



└─ Cached design

└─ Xcache design

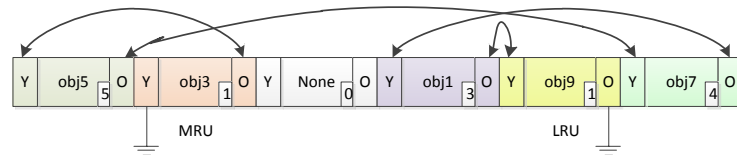
#### Xcache design



Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks

## Cached design

### Xcache design



Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks



└─ Cached design

└─ Xworkq design

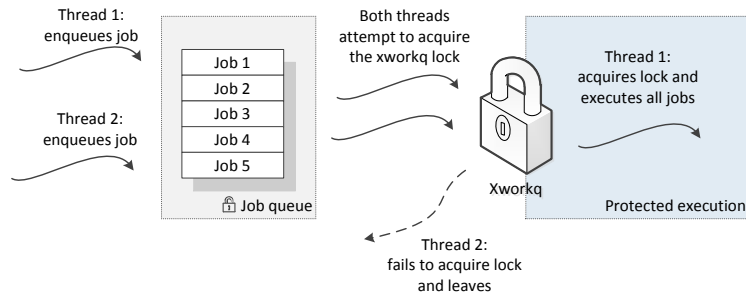
Xworkq design



Xworkq is responsible for concurrency control

Cached design

## Xworkq design



Xworkq is responsible for concurrency control

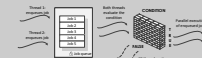




└─ Cached design

└─ Xwaitq design

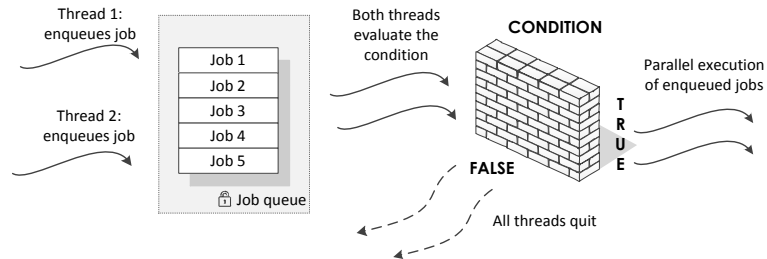
Xwaitq design



Xwaitq is responsible for deferred execution

Cached design

## Xwaitq design



Xwaitq is responsible for deferred execution



└─ Cached design

└─ Bucket pool

#### Bucket pool

When an object is indexed, it does not have immediate access to 4MB size of data because:

- RAM is limited
- Leads to small number of entries

Ideally, we want to:

- Decouple the objects from their data
- Cache unlimited objects but put a limit on their data

Solution:

- Preallocated data space
- Every object request a bucket (typically 4KB)
- When an object is evicted, its buckets are reclaimed

## Cached design

### Bucket pool

When an object is indexed, it does not have immediate access to 4MB size of data because:

- RAM is limited
- Leads to small number of entries

Ideally, we want to:

- Decouple the objects from their data
- Cache unlimited objects but put a limit on their data

Solution:

- Preallocated data space
- Every object request a bucket (typically 4KB)
- When an object is evicted, its buckets are reclaimed



└─ Cached design

└─ Other important cached tasks

#### Other important cached tasks

Several other key tasks are:

- Book-keeping
- Cache write policy
- Asynchronous task execution
- Data propagation

## Cached design

### Other important cached tasks

Several other key-tasks are:

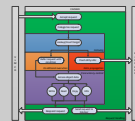
- Book-keeping
- Cache write policy
- Asynchronous task execution
- Data propagation



└─ Cached design

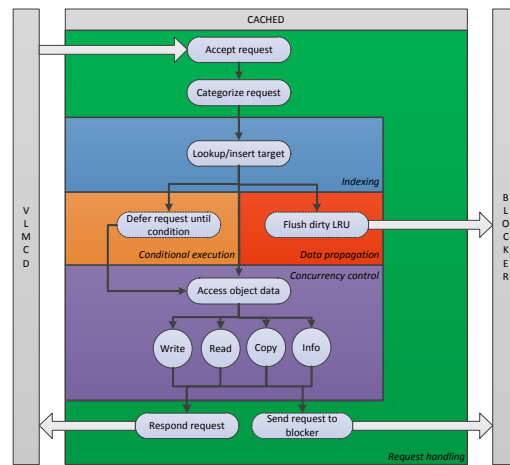
└─ Cached flow

Cached flow



## Cached design

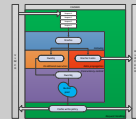
### Cached flow



└─ Cached design

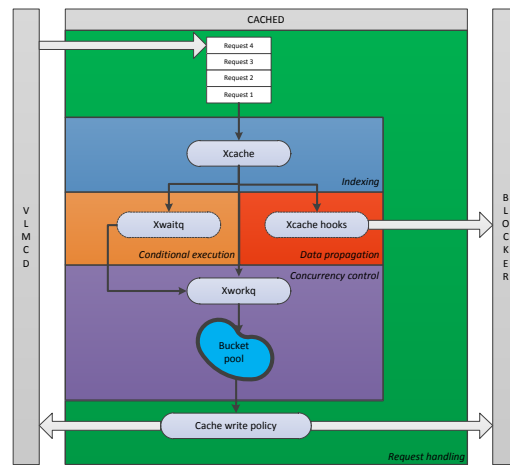
└─ Cached flow

Cached flow



## Cached design

### Cached flow



## Table of Contents

Introduction

Request handling

Caching

Cached design

Cached evaluation



## Benchmark methodology

We have conducted many benchmarks.  
They are separated in three categories:

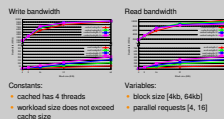
- Comparison between cached and sosd
  - Peak behavior
  - Sustained behavior
- Internal comparison of cached
  - Multithreading overhead
  - Indexing mechanism overhead
- Evaluation under a VM



└─ Cached evaluation

└─ Cached/sosd comparison - peak behavior

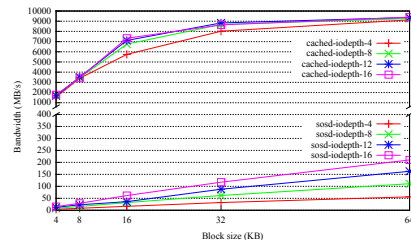
Cached/sosd comparison - peak behavior



## Cached evaluation

### Cached/sosd comparison - peak behavior

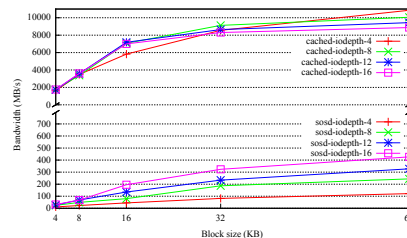
#### Write bandwidth



#### Constants:

- cached has 4 threads
- workload size does not exceed cache size

#### Read bandwidth



#### Variables:

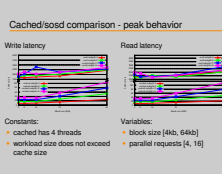
- block size [4kb, 64kb]
- parallel requests [4, 16]





└ Cached evaluation

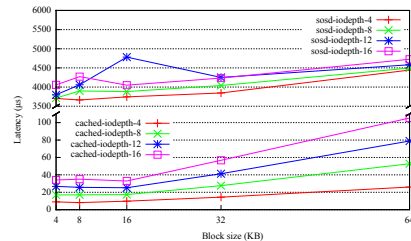
└ Cached/sosd comparison - peak behavior



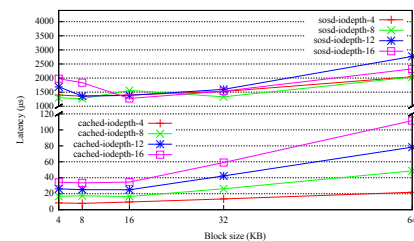
## Cached evaluation

### Cached/sosd comparison - peak behavior

#### Write latency



#### Read latency



#### Constants:

- cached has 4 threads
- workload size does not exceed cache size

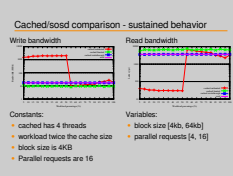
#### Variables:

- block size [4kb, 64kb]
- parallel requests [4, 16]



└─ Cached evaluation

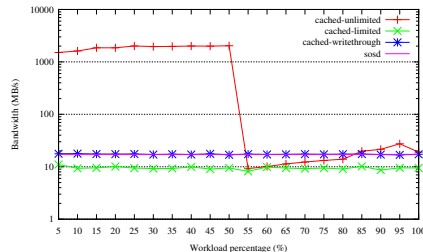
└─ Cached/sosd comparison - sustained behavior



## Cached evaluation

### Cached/sosd comparison - sustained behavior

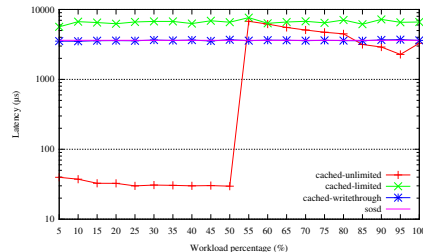
#### Write bandwidth



#### Constants:

- cached has 4 threads
- workload twice the cache size
- block size is 4KB
- Parallel requests are 16

#### Read bandwidth



#### Variables:

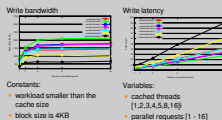
- block size [4kb, 64kb]
- parallel requests [4, 16]



└─ Cached evaluation

└─ Cached internals - multithreading

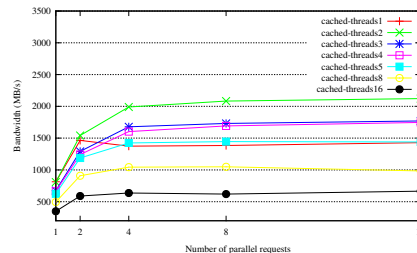
Cached internals - multithreading



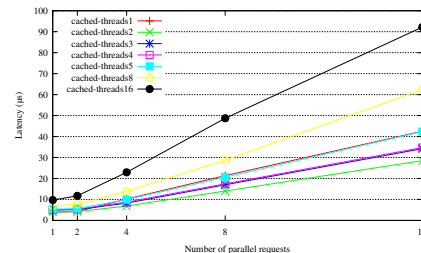
## Cached evaluation

### Cached internals - multithreading

#### Write bandwidth



#### Write latency



#### Constants:

- workload smaller than the cache size
- block size is 4KB

#### Variables:

- cached threads [1, 2, 3, 4, 5, 8, 16]
- parallel requests [1 - 16]



└─ Cached evaluation

└─ Cached internals - indexing

#### Cached internals - indexing

Latency of cold cache vs. warm cache

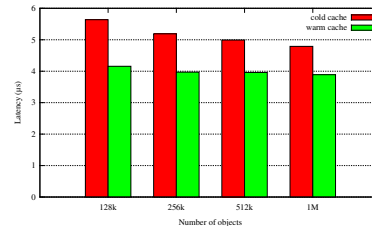


- Constants:
- workload smaller than the cache size
  - block size is 4KB
  - parallel requests are 1
- Variables:
- number of objects

## Cached evaluation

### Cached internals - indexing

Latency of cold cache vs. warm cache



Constants:

- workload smaller than the cache size
- block size is 4KB
- parallel requests are 1

Variables:

- number of objects

