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

Αλέξιος Πυργιώτης
Εθνικό Μετσόβιο Πολυτεχνείο
October 16, 2013

1. Καλημέρα σας, ονομάζομαι Αλέξιος Πυργιώτης
   Θα σας παρουσιάσω τη διπλωματική μου με τίτλο:..
2. Ακούγεται κάπως περίεργο στα ελληνικά...
   αυτό που πραγματεύται είναι την δημιουργία ενός caching μηχανισμού για το Archipelago, ένα distibuted, 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

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

Αλέξιος Πυργιώτης

Εθνικό Μετσόβιο Πολυτεχνείο

October 16, 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 υπηρεσια okeanos και τι είναι το Αρχιπέλαγο
   - Έπειτα, δείχνουμε τον τρόπο με τον οποίο η υποδομή μας χειριζεται αιτήματα δεδομένων από ένα VM.

Contents

Table of Contents
Introduction
Request handling
Caching
Cached design
Cached evaluation
Synapsed
Conclusion

Table of Contents


Synnefo

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

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

..by GRNET -> Και φυσικά τα παιδιά που βλεπετε 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

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

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

okeanos

okeanos

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

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

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

okeanos

okeanos

- 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
Synapsed
Conclusion

Table of Contents

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?

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.

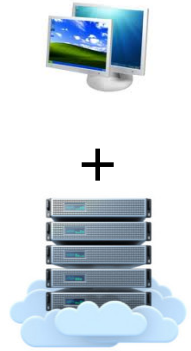
Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>
... το storage μας <κλικ>
και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>
Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>
Όχι, χρειαζόμαστε επίσης **FIXME:**

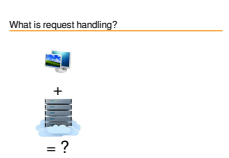
What is request handling?

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?

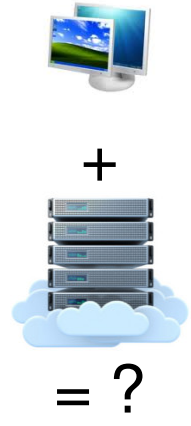


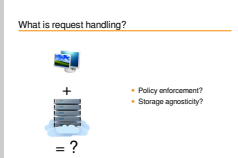


Τι είναι η διαχείριση των αιτημάτων ενός VM?
Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

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?

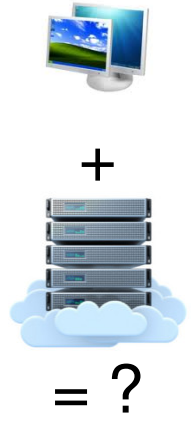




Τι είναι η διαχείριση των αιτημάτων ενός VM?
Είναι η εφαρμογή πολιτικών και επεξεργασία των αιτημάτων σε όλη την πορεία τους μέχρι το να φτάσουν στο storage.

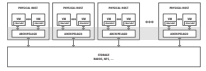
Δηλαδή έχουμε ένα εικονικό μηχάνημα <κλικ>
... το storage μας <κλικ>
και πρέπει με κάποιο τρόπο τα δεδομένα του μηχανήματος να φτάσουν σε εμάς <κλικ>
Ένας απλός τρόπος θα ήταν να τα συνδέσουμε. Άλλωστε όταν τρέχει VM, ο hypervisor κοιτάει block device. Θα μπορούσε να ήταν κομμάτι του storage Είναι αυτό αρκετό; <κλικ>
Όχι, χρειαζόμαστε επίσης **FIXME:**

What is request handling?



- Policy enforcement?
- Storage agnosticity?

Our solution
Archipelago



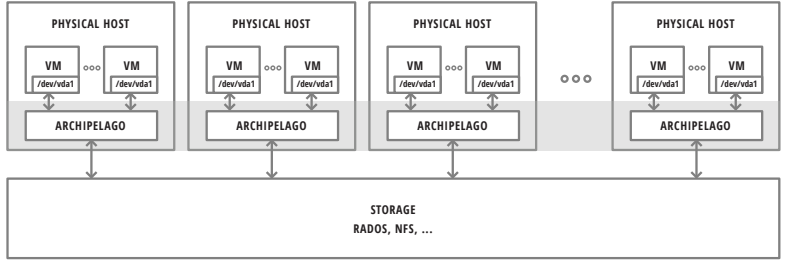
Key features: 1) Software-defined 2) Distributed) Modular Copy-On-Write Storage agnostic

Η λύση που χρησιμοποιήσαμε είναι το Archipelago

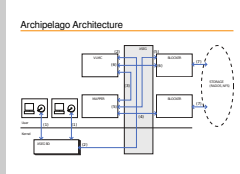
- Software-defined: αν και είναι ένα όρος μαρκετινγκ, εμείς κανονικά. Σημαίνει με το software ΟΡΙΖΕΙΣ το storage (εφαρμογή policy, αλλαγή πορείας του request)
- τρέχει σε πολλούς κόμβους
- αποτελείται από διακριτά κομμάτια
- κάνει CoW (εξήγησε ότι τα images είναι λίγα, τα VMs πολλά, όπως όταν ένα process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

Our solution

Archipelago

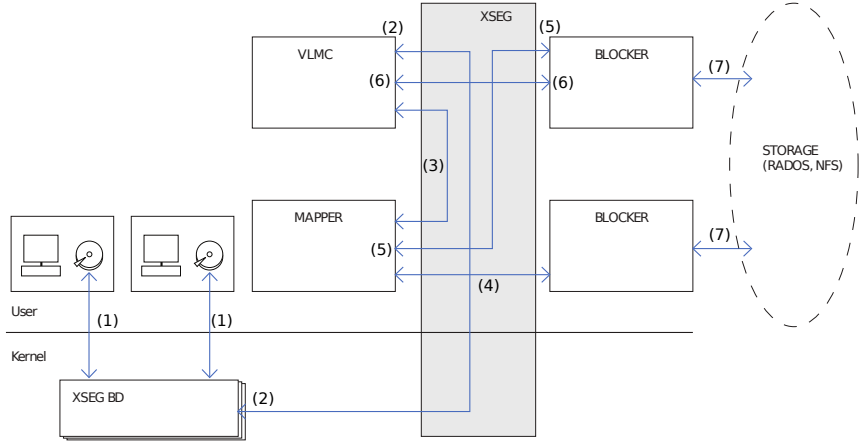


Key features: 1) Software-defined 2) Distributed) Modular Copy-On-Write Storage agnostic



- 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



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

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
Synapsed
Conclusion

Table of Contents

1. Η κλασσική λύση σε τέτοια προβλήματα είναι η χρήση ενός γρηγορότερου αποθηκευτικού μέσου για caching.
2. Για όσους δεν ξέρουν τι σημαίνει caching, θα το εξηγήσουμε συνοπτικά: έχεις ροή, αργό μέσο, βάζεις ένα γρήγορο, speedup
3. Προφανώς αυτό το concept είναι γνωστό. Από που;

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?

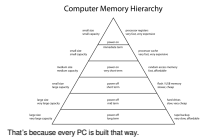
Intro

Solution: Caching

Caching is:

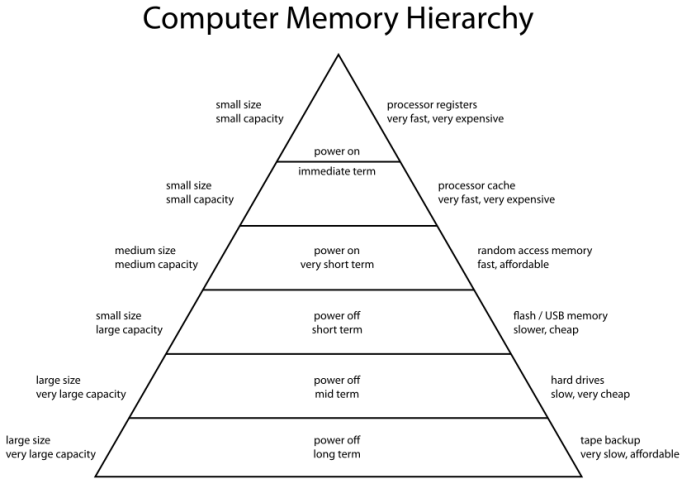
- We have a slow medium
- Add a fast medium in a data path
- Transparently store the data that are intended for the slower medium.
- Profit: later accesses to the same data are faster

Sounds familiar?



1. Από το ιεραρχικό μοντέλο του υπολογιστή: cpu cache vs ram, ram vs disk

Caching



That's because every PC is built that way.

Is there anything to help us?

FACT: 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

1. Τώρα που ξέρουμε τη λύση, υπάρχει κάτι που μπορούμε να κάνουμε;
2. Υπάρχει κάτι έτοιμο; ΝΑΙ
3. Δυο κατηγορίες:
   - βλέπουν τα δεδομένα σαν blocks ενός δίσκου
   - βλέπουν τα δεδομένα σαν κομμάτια αντικειμένων

   Διαφορά: TODO:

Caching

Is there anything to help us?

FACT: 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

Block-store caching solutions

Most notable examples:
- Bcache
- Flashcache
- EnhanceIO

Typically scale-up solutions.

Pros: Simple, scale-up
Cons: Unaware of CoW, kernel solutions

1. Πρώτη κατηγορία που εξετάσαμε έχει αρκετά ενδιαφέρουσες λύσεις.
2. Δε θα επεκταθούμε γιατί έχουν κάποια βασικά κοινά:
   - Kernel modules
   - expose εικονικά block devices
3. Εξήγησε που μπαίνουν (xsegbd)

Block-store caching solutions

Most notable examples:

- Bcache
- Flashcache
- EnhanceIO

Typically scale-up solutions.

Pros: Simple, scale-up
Cons: Unaware of CoW, kernel solutions

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

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

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

Conclusions

- 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
Synapsed
Conclusion

Table of Contents

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

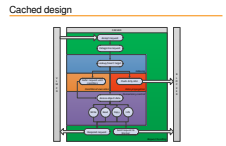
Requirements

Design goals for cached:

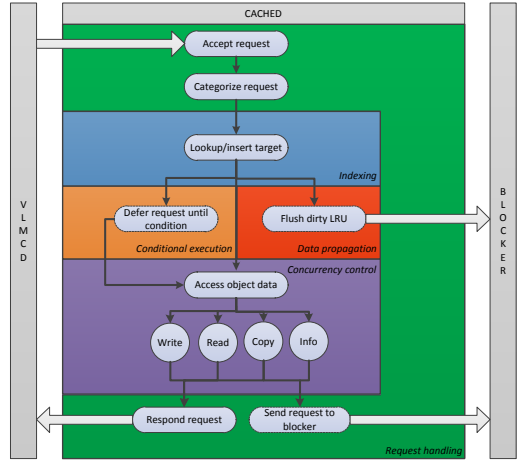
- Create something close to the Archipelago logic
- Measure the best possible performance we can get

Stricter requirements for cached:

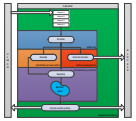
- Nativity
- Pluggability
- In-memory
- Low indexing overhead



Cached design

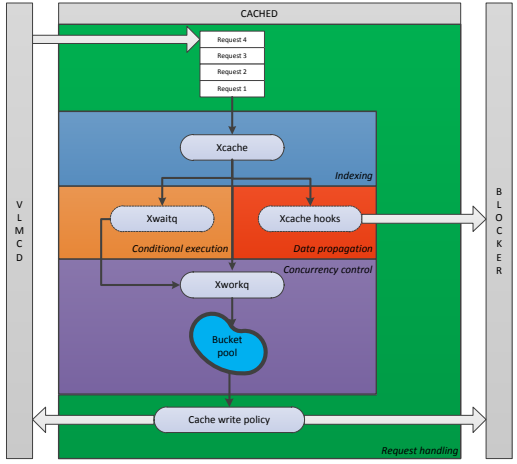


Cached design



Cached design

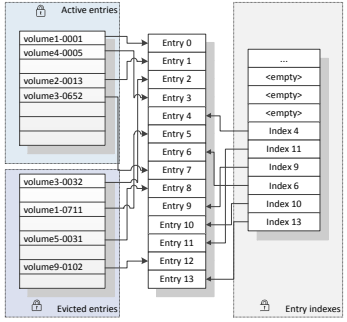
Cached design



Xcache design

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

Xcache design

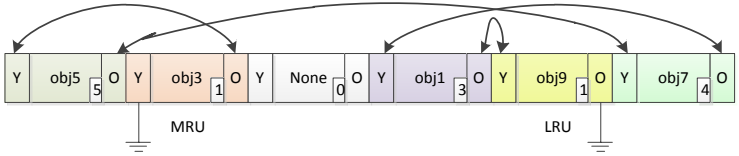


Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks

Xcache design

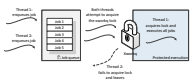
Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks

Xcache design



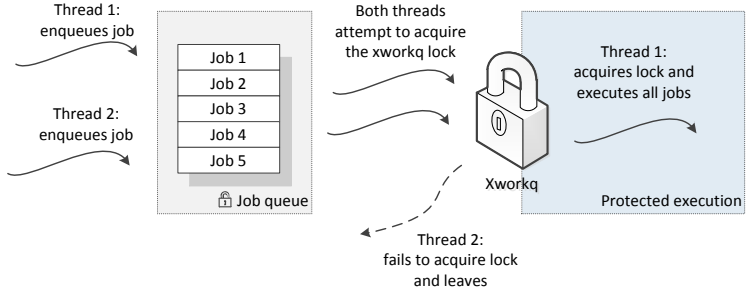
Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control, 4) notification via event hooks

Xworkq design



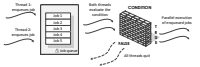
Xworkq is responsible for concurrency control

Xworkq design



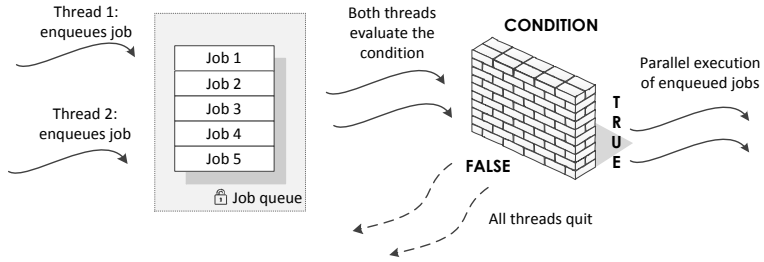
Xworkq is responsible for concurrency control

Xwaitq design



Xwaitq is responsible for deferred execution

Xwaitq design



Xwaitq is responsible for deferred execution

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

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

Other important cached tasks

Several other key-tasks are:
- Book-keeping
- Cache write policy
- Asynchronous task execution
- Data propagation

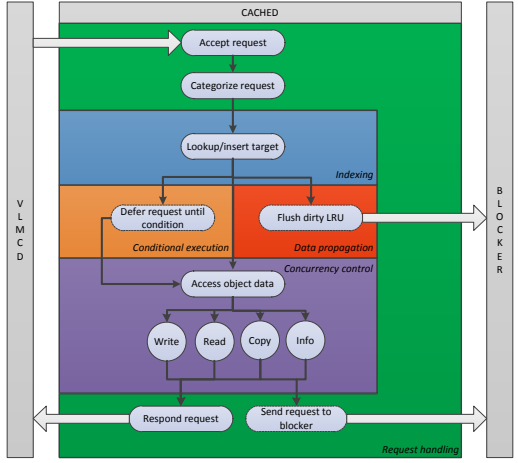
Other important cached tasks

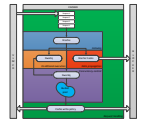
Several other key-tasks are:

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


Cached flow

Cached flow


CACHED
Accept request
Categorize request
Lookup/insert target
Indexing
Defer request until condition
Flush dirty LRU
Conditional execution
Data propagation
Concurrency control
Access object data
Write
Read
Copy
Info
Respond request
Send request to blocker
Request handling
VLMCD
BLOCKER


Cached flow

Cached flow

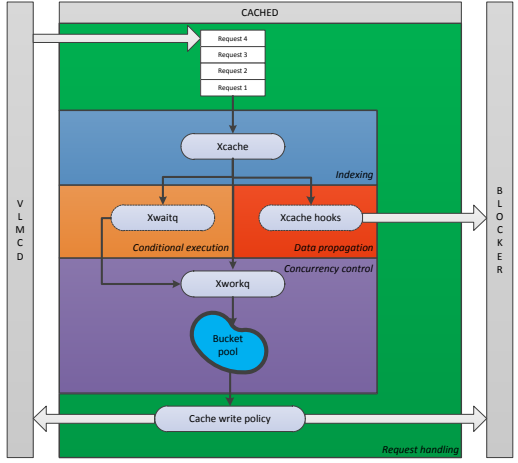

CACHED
Request 4
Request 3
Request 2
Request 1
Xcache
Indexing
Xwaitq
Xcache hooks
Conditional execution
Data propagation
Concurrency control
Xworkq
Bucket pool
Cache write policy
Request handling
VLMCD
BLOCKER

Table of Contents
Introduction
Request handling
Caching
Cached design
Cached evaluation
Synapsed
Conclusion

Table of Contents

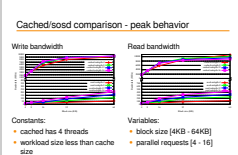
Benchmark methodology

We have conducted exhaustive 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 of VM/Archipelago

Benchmark methodology

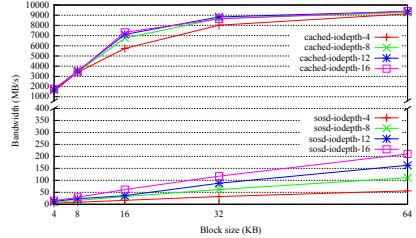
We have conducted exhaustive benchmarks.
They are separated in three categories:

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

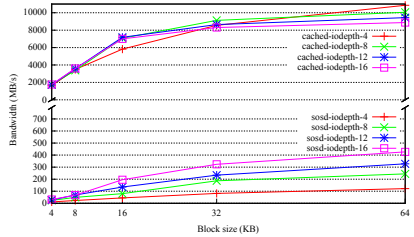


Cached/sosd comparison - peak behavior

Write bandwidth



Read bandwidth



Constants:

- cached has 4 threads
- workload size less than cache size

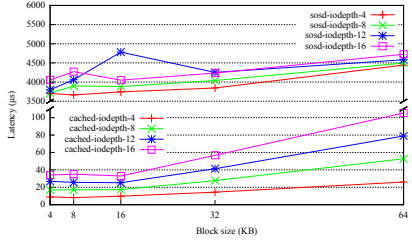
Variables:

- block size [4KB - 64KB]
- parallel requests [4 - 16]

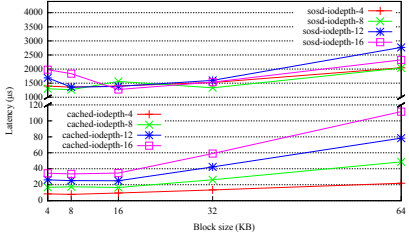
Cached/sosd comparison - peak behavior
Write latency
Read latency
Constants:
cached has 4 threads
workload size less than cache size
Variables:
block size [4KB - 64KB]
parallel requests [4 - 16]

Cached/sosd comparison - peak behavior

Write latency



Read latency

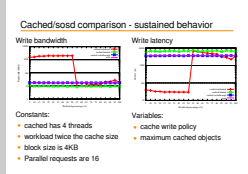


Constants:

- cached has 4 threads
- workload size less than cache size

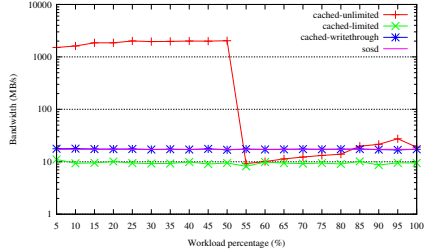
Variables:

- block size [4KB - 64KB]
- parallel requests [4 - 16]

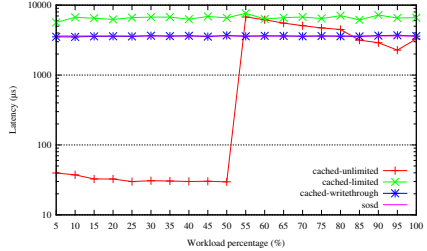


Cached/sosd comparison - sustained behavior

Write bandwidth



Write latency

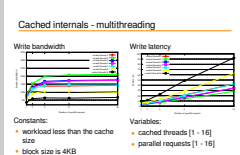


Constants:

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

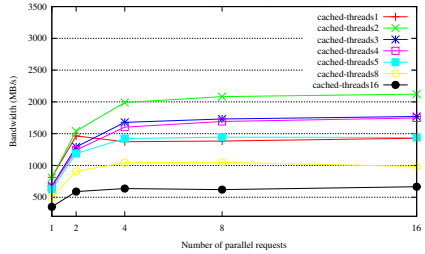
Variables:

- cache write policy
- maximum cached objects

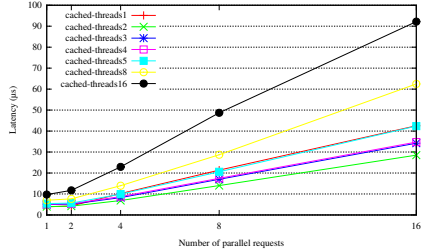


Cached internals - multithreading

Write bandwidth



Write latency



Constants:

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

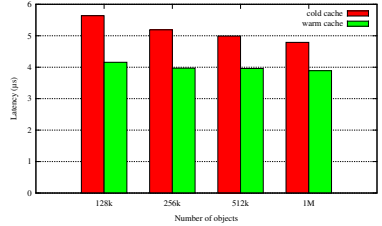
Variables:

- cached threads [1 - 16]
- parallel requests [1 - 16]

Cached internals - indexing
Latency of cold cache vs. warm cache
Constants:
- workload less than the cache size
- block size is 4KB
- no threads or parallel requests
Variables:
- number of objects [128k - 1G]

Cached internals - indexing

Latency of cold cache vs. warm cache

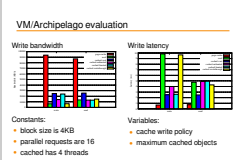


Constants:

- workload less than the cache size
- block size is 4KB
- no threads or parallel requests

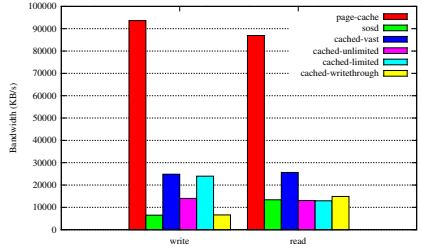
Variables:

- number of objects [128k - 1G]

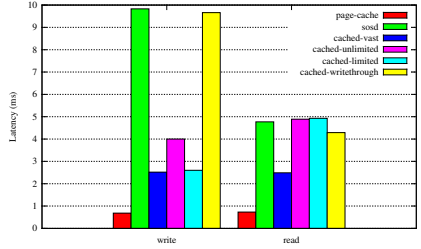


VM/Archipelago evaluation

Write bandwidth



Write latency



Constants:

- block size is 4KB
- parallel requests are 16
- cached has 4 threads

Variables:

- cache write policy
- maximum cached objects

Table of Contents
Introduction
Request handling
Caching
Cached design
Cached evaluation
Synapsed
Conclusion

Table of Contents

Introduction

Previous results show that:
- There is high lock contention
- The amount of RAM is important

If cached remains at the host, it will:
- Compete for CPU time
- Use a fraction of the host's RAM

Idea: what if cached ran on storage nodes?

Introduction

Previous results show that:

- There is high lock contention
- The amount of RAM is important

If cached remains at the host, it will:

- Compete for CPU time
- Use a fraction of the host's RAM

Idea: what if cached ran on storage nodes?

If cached was on storage nodes, the pros would be:
- Access to more RAM
- Major step towards a distributed cache

On the other hand, the cons would be:
- Network bottleneck
- Bigger complexity

Archipelago is network-unaware. Must create a proof-of-concept network peer to help us in this task.

Synapsed

If cached was on storage nodes, the pros would be:

- Access to more RAM
- Major step towards a distributed cache

On the other hand, the cons would be:

- Network bottleneck
- Bigger complexity

Archipelago is network-unaware. Must create a proof-of-concept network peer to help us in this task.

Synapsed design

Synapsed is designed to do the following:
- Connect two Archipelago peers over network
- Forward read/write XSEG requests
- Use the TCP protocol
- Integrate with the Archipelago signaling mechanism
- Use zero-copy methods

Replication should be trivial to implement, but it is currently missing.

Synapsed design

Synapsed is designed to do the following:

- Connect two Archipelago peers over network
- Forward read/write XSEG requests
- Use the TCP protocol
- Integrate with the Archipelago signaling mechanism
- Use zero-copy methods

Replication should be trivial to implement, but it is currently missing.

Benchmark preamble

The most important part is that synapsed works. We are **now** able to run cached or part of Archipelago in the storage nodes.

However, let's check its performance.
We will attempt to run most of the previous scenarios using synapsed this time.

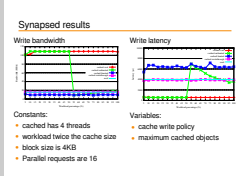
Note, synapsed is proof-of-concept and not performance-tuned.
Also, the tested configuration uses a 1Gbit connection.

Benchmark preamble

The most important part is that synapsed works. We are **now** able to run cached or part of Archipelago in the storage nodes.

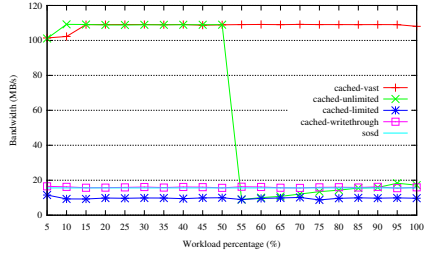
However, let's check its performance.
We will attempt to run most of the previous scenarios using synapsed this time.

Note, synapsed is proof-of-concept and not performance-tuned.
Also, the tested configuration uses a 1Gbit connection.

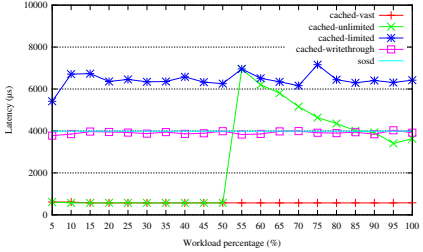


Synapsed results

Write bandwidth



Write latency



Constants:

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

Variables:

- cache write policy
- maximum cached objects

Table of Contents
Introduction
Request handling
Caching
Cached design
Cached evaluation
Synapsed
Conclusion

Table of Contents

Concluding remarks

We close this presentation with the following remarks:
- Cached and synapsed have covered important Archipelago needs
- Synthetic benchmarks show that cached can achieve 200x better performance than sosd
- In more real-life scenarios, cached speeds Archipelago up to 400%
- Synapsed can bridge two peers over network with minimum latency

Concluding remarks

We close this presentation with the following remarks:

- Cached and synapsed have covered important Archipelago needs
- Synthetic benchmarks show that cached can achieve 200x better performance than sosd
- In more real-life scenarios, cached speeds Archipelago up to 400%
- Synapsed can bridge two peers over network with minimum latency

Future work

Future work is happening as we speak:
- Full CoW support
- Namespace support
- Support for different policies and limits per volume

Future work

Future work is happening as we speak:

- Full CoW support
- Namespace support
- Support for different policies and limits per volume