



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
4. Σημείωση: Για οικονομία του λόγου, δε θα προβώ σε εξήγηση
βασικών όρων όπως VMs, storage. Για κάθε άγνωστη έννοια,
παρακαλώ να με διακόψετε.



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 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](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)[Synapsed](#)[Conclusion](#)

Table of Contents

Introduction

VM Volume storage

Caching

Cached design

Cached evaluation

Synapsed

Conclusion



- 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 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 παρέχει αποθηκευτικό χώρο στους χρήστες.
- 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
- **CLICK!**
- Και φυσικά παίζει πάνω σε Synnefo...

- 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 Ακαδημαϊκούς σκοπούς, ήδη γίνονται εργαστήρια στο EMP και απ' αυτό το εξάμηνο σε άλλες σχολές
- **CLICK!**
- Και φυσικά παίζει πάνω σε Synnefo...

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](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)[Synapsed](#)[Conclusion](#)

└─ VM Volume storage

└─ Intro

Intro

1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:
2. **CLICK!**
3. Το VM που τρέχει σε ένα host server
4. **CLICK!**
5. Και τους storage servers
6. **CLICK!**
7. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του;
Υπάρχουν διάφορες επιλογές όπως το DRBD, το RBD που χρησιμοποιούνται ευρέως και γεφυρώνουν τον αποθηκευτικό χώρο του VM με το χώρο όπου λειτουργεί.
8. Τι γίνεται όμως στην περίπτωση που θέλουμε *και* τα εξής;
9. **CLICK!**
10. Εξήγησε τους όρους





Intro

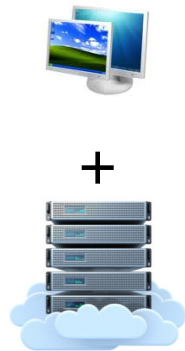
1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:
2. **CLICK!**
3. Το VM που τρέχει σε ένα host server
4. **CLICK!**
5. Και τους storage servers
6. **CLICK!**
7. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του;
Υπάρχουν διάφορες επιλογές όπως το DRBD, το RBD που χρησιμοποιούνται ευρέως και γεφυρώνουν τον αποθηκευτικό χώρο του VM με το χώρο όπου λειτουργεί.
8. Τι γίνεται όμως στην περίπτωση που θέλουμε *και* τα εξής;
9. **CLICK!**
10. Εξήγησε τους όρους





Intro

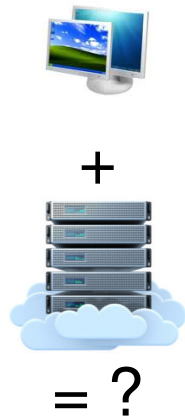
1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:
2. **CLICK!**
3. Το VM που τρέχει σε ένα host server
4. **CLICK!**
5. Και τους storage servers
6. **CLICK!**
7. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του;
Υπάρχουν διάφορες επιλογές όπως το DRBD, το RBD που χρησιμοποιούνται ευρέως και γεφυρώνουν τον αποθηκευτικό χώρο του VM με το χώρο όπου λειτουργεί.
8. Τι γίνεται όμως στην περίπτωση που θέλουμε *και* τα εξής;
9. **CLICK!**
10. Εξήγησε τους όρους





Intro

1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:
2. **CLICK!**
3. Το VM που τρέχει σε ένα host server
4. **CLICK!**
5. Και τους storage servers
6. **CLICK!**
7. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του;
Υπάρχουν διάφορες επιλογές όπως το DRBD, το RBD που χρησιμοποιούνται ευρέως και γεφυρώνουν τον αποθηκευτικό χώρο του VM με το χώρο όπου λειτουργεί.
8. Τι γίνεται όμως στην περίπτωση που θέλουμε *και* τα εξής;
9. **CLICK!**
10. Εξήγησε τους όρους





Intro

1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:

2. **CLICK!**

3. Το VM που τρέχει σε ένα host server

4. **CLICK!**

5. Και τους storage servers

6. **CLICK!**

7. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του;

Υπάρχουν διάφορες επιλογές όπως το DRBD, το RBD που χρησιμοποιούνται ευρέως και γεφυρώνουν τον αποθηκευτικό χώρο του VM με το χώρο όπου λειτουργεί.

8. Τι γίνεται όμως στην περίπτωση που θέλουμε *και* τα εξής;

9. **CLICK!**

10. Εξήγησε τους όρους



+



= ?

- Policy enforcement?
- Storage agnosticity?

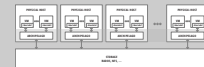


└ VM Volume storage

└ Our solution

Our solution

Archipelago



Key features: 1) Software-defined 2) Distributed 3) Modular 4) Copy-On-Write 5) Storage agnostic

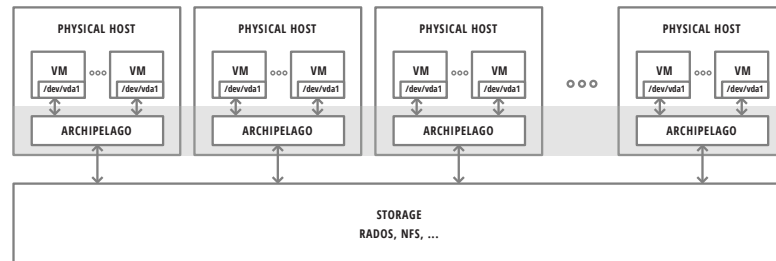
Η λύση που χρησιμοποιήσαμε είναι το Archipelago

- 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 storage θέλουμε

VM Volume storage

Our solution

Archipelago



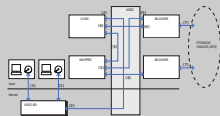
Key features: 1) Software-defined 2) Distributed 3) Modular 4) Copy-On-Write 5) Storage agnostic



└ VM Volume storage

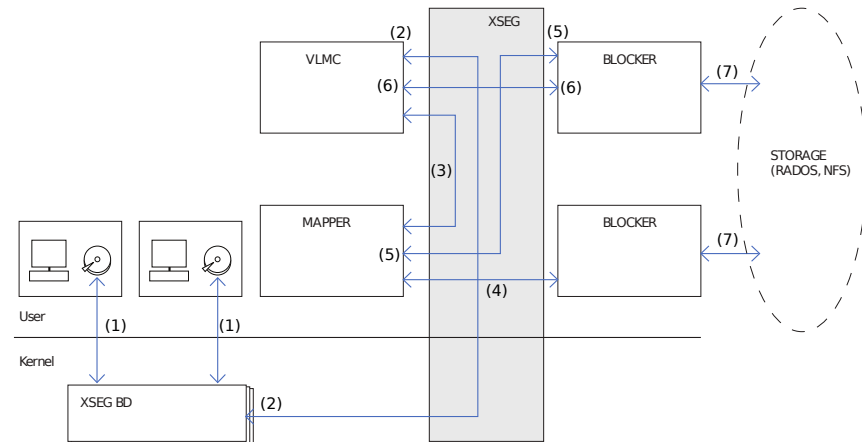
└ Archipelago Architecture

Archipelago Architecture



1. 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. Ο xsegbd στέλνει το αίτημα στο userspace κομμάτι του Αρχιπελάγους το οποίο αποφαινεται για τα αντικείμενα τα οποία αντιστοιχούν στο αίτημα. Συνοπτικά πως γίνεται αυτό:
 - Ο xsegbd στέλνει το αίτημα στο vmlmc
 - Ο vmlmc συμβουλευεται τα mappings του
3. μετά τα ζητάει από το storage μέσω των blockers

Archipelago Architecture



- Replication
- Fault tolerance
- Self-management
- Scalability

Αν και είμαστε storage agnostic, χρησιμοποιούμε ένα σημαντικό storage backend, το RADOS, που μας δίνει τα εξής:

- Αντίγραφα ασφαλείας των δεδομένων
- Ανοχή στο χάσιμο αποθηκευτικών κόμβων
- Load balancing και αυτοδιαχείριση
- και τέλος είναι επεκτάσιμο

CLICK! Αυτό το περίπλοκο σύστημα όμως δεν είναι αρκετά ταχύ. Παράδειγμα...

CLICK!

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

Αν και είμαστε storage agnostic, χρησιμοποιούμε ένα σημαντικό storage backend, το RADOS, που μας δίνει τα εξής:

- Αντίγραφα ασφαλείας των δεδομένων
- Ανοχή στο χάσιμο αποθηκευτικών κόμβων
- Load balancing και αυτοδιαχείριση
- και τέλος είναι επεκτάσιμο

CLICK! Αυτό το περίπλοκο σύστημα όμως δεν είναι αρκετά ταχύ. Παράδειγμα...

CLICK!



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](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)[Synapsed](#)[Conclusion](#)

- 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

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

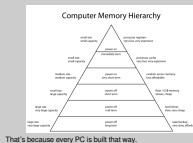
Solution: Caching

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

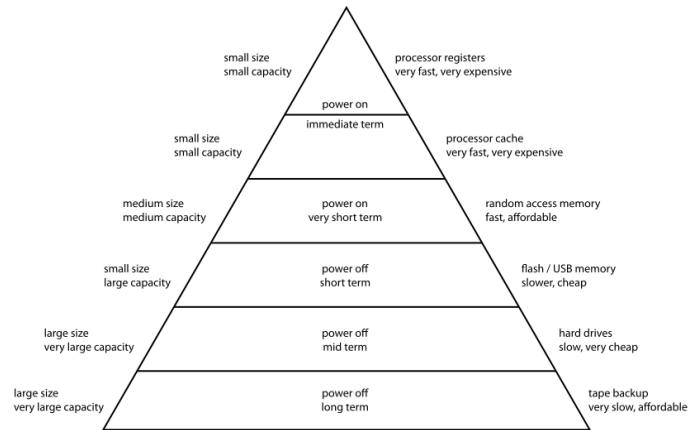
Sounds familiar?





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

Computer Memory Hierarchy



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:**

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



- Bcache
- Flashcache
- EnhanceIO

Block-store caching 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)

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](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)[Synapsed](#)[Conclusion](#)

- 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:

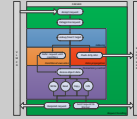
- Nativity
- Pluggability
- In-memory
- Low indexing overhead



└─ Cached design

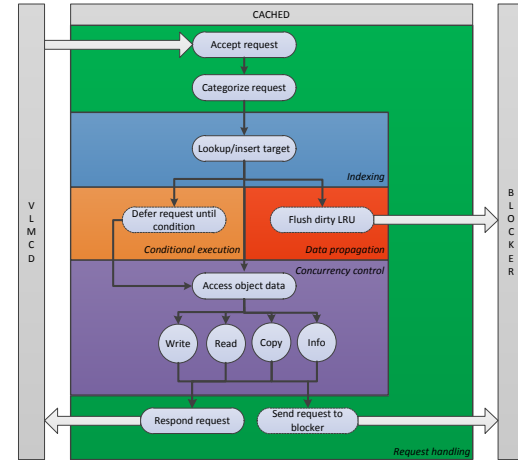
└─ Cached design

Cached design



Cached design

Cached design



└─ Cached design

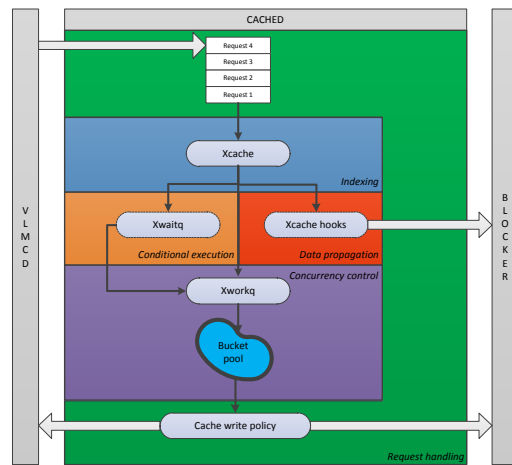
└─ Cached design

Cached design



Cached design

Cached design



└─ Cached design

└─ Xcache design

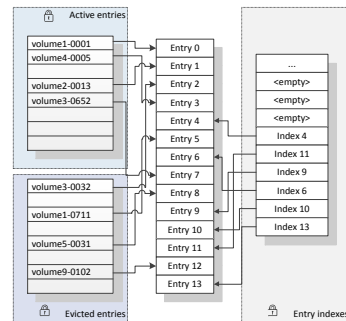
Xcache design



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

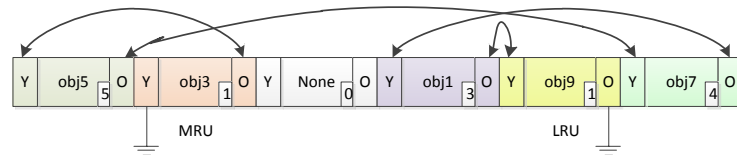
Xcache design



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

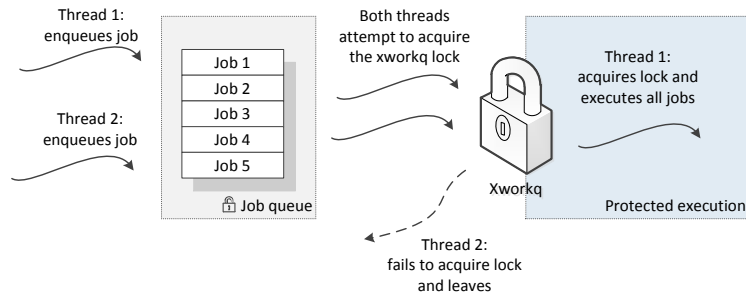
Xworkq design



Xworkq is responsible for concurrency control

Cached design

Xworkq design



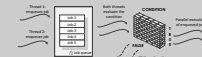
Xworkq is responsible for concurrency control



└─ Cached design

└─ Xwaitq design

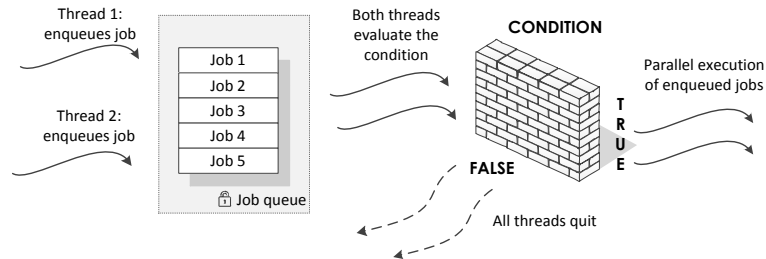
Xwaitq design



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:

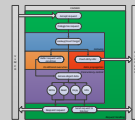
- Book-keeping
- Cache write policy
- Asynchronous task execution
- Data propagation



└─ Cached design

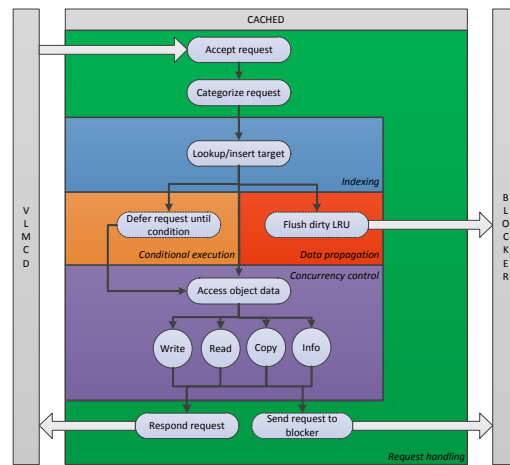
└─ Cached flow

Cached flow



Cached design

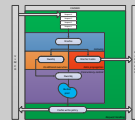
Cached flow



└─ Cached design

└─ Cached flow

Cached flow



Cached design

Cached flow

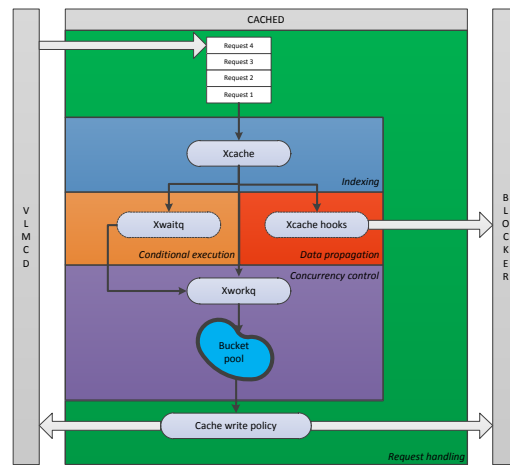


Table of Contents

[Introduction](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)[Synapsed](#)[Conclusion](#)

- 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:

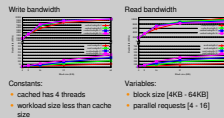
- Comparison between cached and sosd
 - Peak behavior
 - Sustained behavior
- Internal comparison of cached
 - Multithreading overhead
 - Indexing mechanism overhead
- Evaluation of VM/Archipelago



└─ Cached evaluation

└─ Cached/sosd comparison - peak behavior

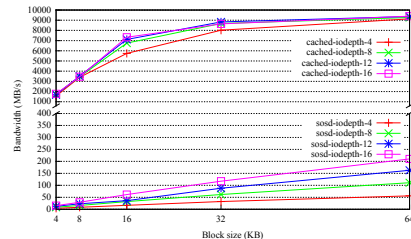
Cached/sosd comparison - peak behavior



Cached evaluation

Cached/sosd comparison - peak behavior

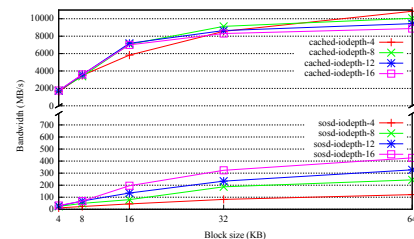
Write bandwidth



Constants:

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

Read bandwidth



Variables:

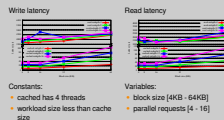
- block size [4KB - 64KB]
- parallel requests [4 - 16]



└─ Cached evaluation

└─ Cached/sosd comparison - peak behavior

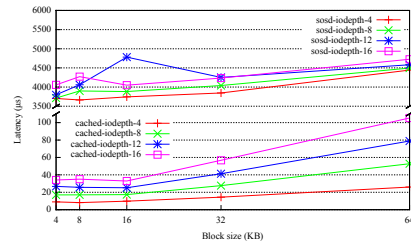
Cached/sosd comparison - peak behavior



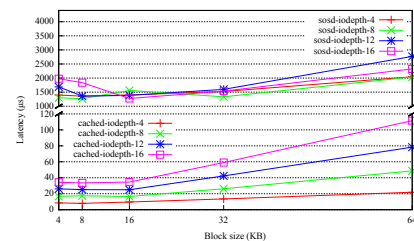
Cached evaluation

Cached/sosd comparison - peak behavior

Write latency



Read latency



Constants:

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

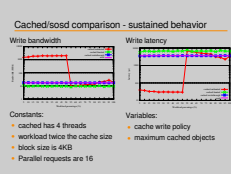
Variables:

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



└─ Cached evaluation

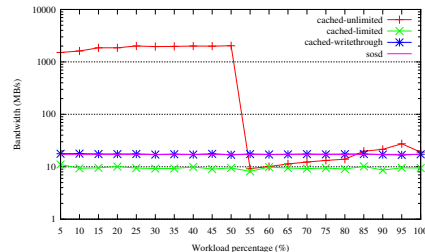
└─ Cached/sosd comparison - sustained behavior



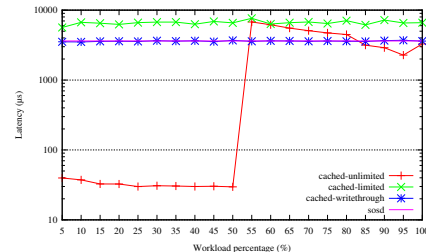
Cached evaluation

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:

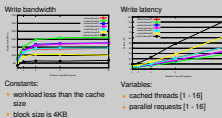
- cache write policy
- maximum cached objects



└─ Cached evaluation

└─ Cached internals - multithreading

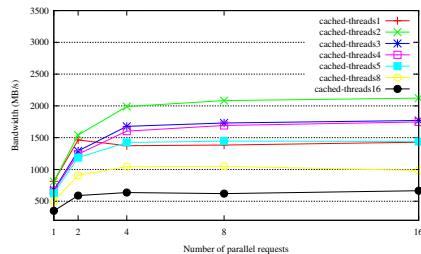
Cached internals - multithreading



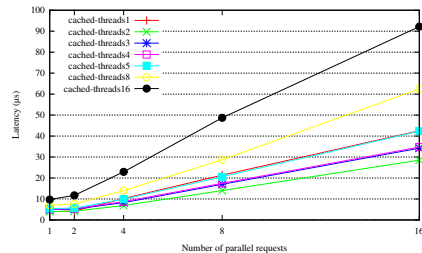
Cached evaluation

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 evaluation

└─ Cached internals - indexing

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

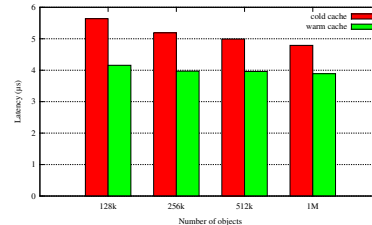
Variables:

- number of objects [128k - 1G]

Cached evaluation

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:

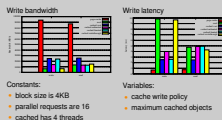
- number of objects [128k - 1G]



└─ Cached evaluation

└─ VM/Archipelago evaluation

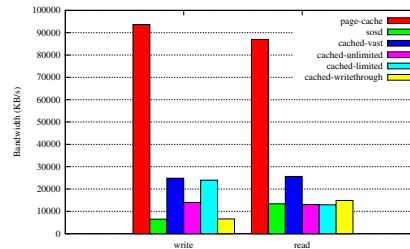
VM/Archipelago evaluation



Cached evaluation

VM/Archipelago evaluation

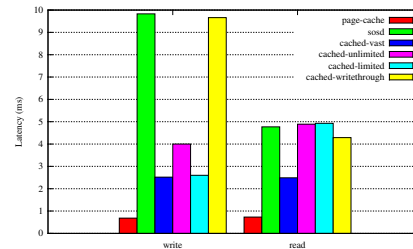
Write bandwidth



Constants:

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

Write latency



Variables:

- cache write policy
- maximum cached objects



Table of Contents

[Introduction](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)**[Synapsed](#)**[Conclusion](#)

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

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

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.

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.



- 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

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.



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.

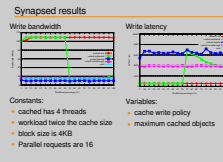
We will attempt to run most of the previous scenarios using synapsed this time.

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



└ Synapsed

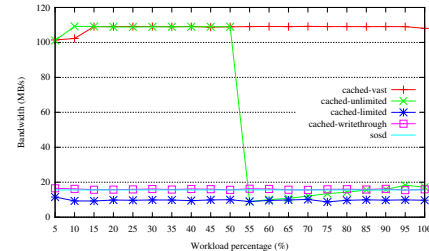
└ Synapsed results



Synapsed

Synapsed results

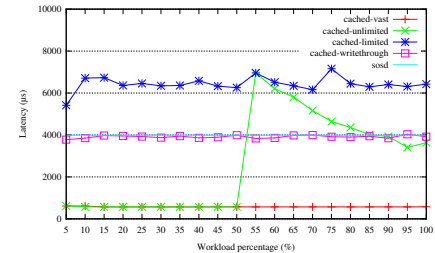
Write bandwidth



Constants:

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

Write latency



Variables:

- cache write policy
- maximum cached objects



Table of Contents

[Introduction](#)[VM Volume storage](#)[Caching](#)[Cached design](#)[Cached evaluation](#)[Synapsed](#)[Conclusion](#)

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



└ Conclusion

└ Future work

Future work

Future work is happening as we speak:

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

Conclusion

Future work

Future work is happening as we speak:

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

