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

Αλέξιος Πυργιώτης
Εθνικό Μετσόβιο Πολυτεχνείο
October 17, 2013

1. Καλημέρα σας, ονομάζομαι Αλέξιος Πυργιώτης
   Θα σας παρουσιάσω τη διπλωματική μου με τίτλο:..
2. Συγκεκριμένα, στο πρώτο σκέλος της παρουσίασης θα μιλήσουμε για το Archipelago, ένα distributed storage layer και το RADOS, ένα storage backend του Αρχιπέλαγο.
3. Στο δεύτερο σκέλος, θα μιλήσουμε για την προσφορά της διπλωματικής μας και συγκεκριμένα δυο επεκτάσεις του Αρχιπέλαγο:
   - τον cached, δηλαδή τον caching μηχανισμό μας και κύριο αντικείμενο της διπλωματικης
   - το synapsed, ένα συμπληρωματικό εργαλείο που στόχος του είναι να δώσει στον cached δικτυακές δυνατότητες
4. Αν κάποια έννοια σας είναι άγνωστη ή για όποια απορία κατά τη διάρκεια της παρουσίασης, μπορείτε ελεύθερα να με διακόψετε και να ρωτήσετε.

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

Αλέξιος Πυργιώτης

Εθνικό Μετσόβιο Πολυτεχνείο

October 17, 2013

Contents
Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion

1. Ο κορμός της παρουσίασης είναι ο εξής:
   - Αρχικά, παρουσιάζουμε κάποια εισαγωγικά που αφορούν το background της εργασίας μας. Αναφέρουμε τι είναι το Synnefo και τι είναι η υπηρεσια okeanos
   - Έπειτα, μιλαμε για το archipelago, το RADOS και τελικά τον στόχο της διπλωματικής.
   - Στη συνέχεια μιλάμε για το τι είναι caching και αναφέρουμε κάποιες σύγχρονες λύσεις για caching.
   - Τα επόμενα δυο κεφάλαια παρουσιάζουν τη σχεδίαση και αξιολόγηση του cached.
   - Αντίστοιχα παρουσιάζουμε το synapsed, τη σχεδίαση και την αξιολόγηση του
   - Τέλος, συνοψίζουμε όσα ειπώθηκαν παραπάνω και μιλάμε για μελλοντικές πιθανές μελοντικές εργασίες

## Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



Table of Contents
Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



# Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion




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

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

- IaaS είναι πρακτικά η παροχή εικονικής υποδομής σε χρήστες (δηλαδή πάρε υπολογιστή (VM), δίκτυα, αρχεία κτλ)
- Φτιαγμένο επίσης από την ΕΔΕΤ
- Δωρεάν για τους Ακαδημαϊκους σκοπούς, ήδη χρησιμοποιείται ως πλατφόρμα για εργαστήρια στο ΕΜΡ και απ' αυτό το εξάμηνο σε άλλες σχολές



## Thesis background

synnefo

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

okeanos

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



Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



## Table of Contents

Introduction

VM Volume storage

Caching

Cached design

Cached evaluation

Synapsed design

Synapsed evaluation

Conclusion



## Intro

Δηλαδή εφαρμογή δικών μας πολιτικών, χρήση οποιουδήποτε τύπου storage

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 φυσικούς κόμβους
4. CLICK!
5. τον εικονικό σκληρό του δίσκο, το κλασσικό /dev/sda/
6. CLICK!
7. Και τους storage servers
8. CLICK!
9. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του; Παράλληλα, τι γίνεται στην περίπτωση που θέλουμε *και* τα εξής;
10. CLICK!





## Intro

Δηλαδή εφαρμογή δικών μας πολιτικών, χρήση οποιουδήποτε τύπου storage

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 φυσικούς κόμβους
4. CLICK!
5. τον εικονικό σκληρό του δίσκο, το κλασσικό /dev/sda/
6. CLICK!
7. Και τους storage servers
8. CLICK!
9. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του; Παράλληλα, τι γίνεται στην περίπτωση που θέλουμε *και* τα εξής;
10. CLICK!





## Intro

Δηλαδή εφαρμογή δικών μας πολιτικών, χρήση οποιουδήποτε τύπου storage

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 φυσικούς κόμβους
4. CLICK!
5. τον εικονικό σκληρό του δίσκο, το κλασσικό /dev/sda/
6. CLICK!
7. Και τους storage servers
8. CLICK!
9. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του; Παράλληλα, τι γίνεται στην περίπτωση που θέλουμε *και* τα εξής;
10. CLICK!





## Intro

Δηλαδή εφαρμογή δικών μας πολιτικών, χρήση οποιουδήποτε τύπου storage

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 φυσικούς κόμβους
4. CLICK!
5. τον εικονικό σκληρό του δίσκο, το κλασσικό /dev/sda/
6. CLICK!
7. Και τους storage servers
8. CLICK!
9. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του; Παράλληλα, τι γίνεται στην περίπτωση που θέλουμε *και* τα εξής;
10. CLICK!



Intro
• Policy enforcement?
• Storage agnosticity?

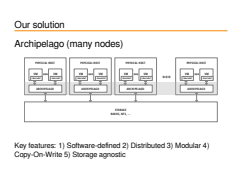
Δηλαδή εφαρμογή δικών μας πολιτικών, χρήση οποιουδήποτε τύπου storage

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 φυσικούς κόμβους
4. CLICK!
5. τον εικονικό σκληρό του δίσκο, το κλασσικό /dev/sda/
6. CLICK!
7. Και τους storage servers
8. CLICK!
9. Το ερώτημα είναι λοιπόν, πως το VM θα αποθηκεύει τα δεδομένα του; Παράλληλα, τι γίνεται στην περίπτωση που θέλουμε *και* τα εξής;
10. CLICK!



## Intro

- Policy enforcement?
- Storage agnosticity?



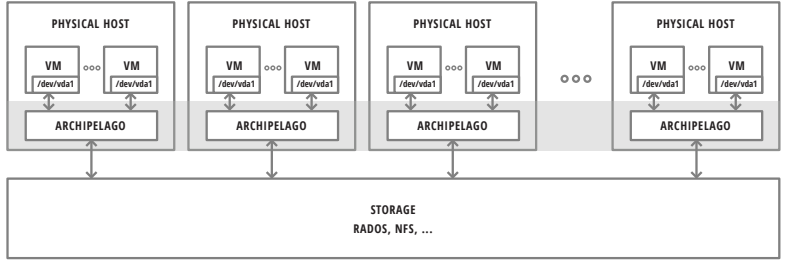

Η λύση που χρησιμοποιήσαμε είναι το Archipelago

1. Πιο συγκεκριμένα, το Αρχιπέλαγο τρέχει στον host κόμβο. Είναι ΑΚΡΙΒΩΣ κάτω από τον σκληρό δίσκο του VM, το /dev/vda1. Τα δεδομένα είναι στο storage
2. – Software-defined: με το software ΟΡΙΖΕΙΣ το storage (εφαρμογή policy, χρήση διαφορετικού storage για κάθε volume)
   – τρέχει σε πολλούς κόμβους
   – αποτελείται από διακριτά κομμάτια
   – χρησιμοποιεί Copy-on-Write policy. ΝΑ αναφέρουμε εδώ ότι το copy on write είναι μια πολιτική αντιγραφής ενός αντικειμένου. Το αντικείμενο σπάει σε κομμάτια τα οποία αντιγράφονται ΜΟΝΟ όταν πάει να γράψει σε κάποιο από αυτά.
   – μπορούμε χρησιμοποιήσουμε ότι storage θέλουμε

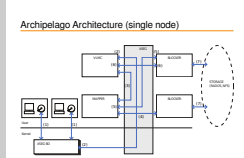


## Our solution

### Archipelago (many nodes)



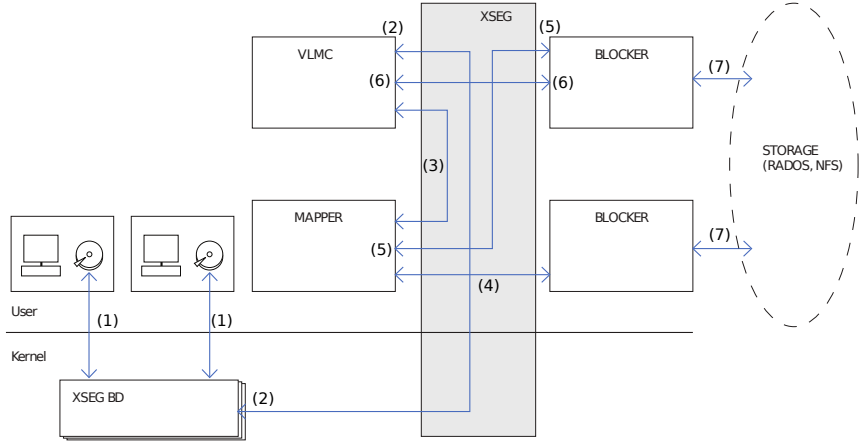

Key features: 1) Software-defined 2) Distributed 3) Modular 4) Copy-On-Write 5) Storage agnostic

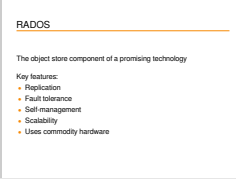



1. Βλέπουμε τώρα το Αρχιπέλαγο σε ένα κόμβο. Όλα είναι userspace διεργασίες, μόνο ο xsegbd είναι kernel driver (όλα είναι PEERS)
2. Ας δουμε τη διαδρομή ενός IO request. Το VM στέλνει αίτημα στο δίσκο του. Ποιον δίσκο; /dev/vda1
3. ο δίσκος είναι εικονικός, θα το δει ο hypervisor (ο υπεύθυνος για το virtualization του Virtual Machine) και θα το στείλει στον δίσκο που το έχουμε πει. (xsegbd)
4. Ο xsegbd είναι kernel driver και θα στέλνει το αίτημα στο userspace κομμάτι του Αρχιπελάγους το οποίο αποφαίνεται για τα αντικείμενα τα οποία αντιστοιχούν στο αίτημα. Στο αίτημα αυτό θα αναφέρεται ότι προέρχεται από το volume foo.
5. το παίρνει ο vlmc, ρωτάει τον mapper, από ποια αντικείμενα αποτελείται; λέει ότι είναι το foo
6. μετά τα ζητάει από το storage μέσω των blockers. ΤΩΡΑ ΜΙΛΑΜΕ για την έλλειψη στα δεξια



## Archipelago Architecture (single node)




RADOS

The object store component of a promising technology

Key features:
- Replication
- Fault tolerance
- Self-management
- Scalability
- Uses commodity hardware

Αν και είμαστε storage agnostic, χρησιμοποιούμε ένα σημαντικό storage backend, το RADOS. Το rados είναι το object store component μιας πολλά υποσχόμενης τεχνολογίας που μας δίνει τα εξής:

- Αντίγραφα ασφαλείας των δεδομένων
- Ανοχή στο χάσιμο αποθηκευτικών κόμβων
- Load balancing και αυτοδιαχείριση
- ειναι κλιμακώσιμο
- χρησιμοποιεί απλό hardware

υ.γ. αντικείμενα είναι ονοματοδοτούμενη ποσότητα πληροφορίας



## RADOS

The object store component of a promising technology

Key features:
- Replication
- Fault tolerance
- Self-management
- Scalability
- Uses commodity hardware



Thesis motivation
- RADOS has speed issues: Bandwidth < 7MB/s, Latency ~10ms
- Request type: Block size: 4KB, parallel requests: 16
- Qualitative benchmark that depicts a hard workload

Thesis goal: Mitigate RADOS speed issues.

Observation:
VM with page-cache of host enabled: Bandwidth > 90MB/s, Latency < 1ms

Page-cache is great but it has many limitations:
- KVM and Linux kernel dependent (we REALLY want to avoid this)
- Unaware of CoW policies
- Difficult to manage

1. Αυτό το περίπλοκο σύστημα όμως δεν είναι αρκετά ταχύ. Από πραγματικές μετρήσεις σε ένα VM έχουμε ότι:...
2. Αυτά τα αποτελέσματα δεν είναι καλά
3. Στόχος διπλωματικής, απάλειψη του αντίκτυπου της επίδοσης του RADOS στο Αρχιπέλαγο
4. Παρατηρήσαμε ότι αν ενεργοποιήσουμε την page cache έχουμε...
5. Η page-cache όμως έχει σοβαρούς περιορισμούς
6. Επίση, εμείς μπορεί να θέλουμε να cach-άρουμε σε ssd, να κάνουμε replication, να έχουμε επιλεγόμενα επίπεδα αξιοπιστίας για κάθε VM



## Thesis motivation

- RADOS has speed issues: Bandwidth < 7MB/s, Latency ~10ms
- Request type: Block size: 4KB, parallel requests: 16
- Qualitative benchmark that depicts a hard workload

Thesis goal: Mitigate RADOS speed issues.

Observation:
VM with page-cache of host enabled: Bandwidth > 90MB/s, Latency < 1ms

Page-cache is great but it has many limitations:
- KVM and Linux kernel dependent (we REALLY want to avoid this)
- Unaware of CoW policies
- Difficult to manage

Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



## Table of Contents

Introduction

VM Volume storage

Caching

Cached design

Cached evaluation

Synapsed design

Synapsed evaluation

Conclusion



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. 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 δεδομένων, καταλήγουν σε αργό μέσο, βάζεις μπροστά ένα μικρό αλλά γρήγορο μέσο, οι επόμενες προσβάσεις στη μνήμη θα είναι πιο γρήγορες
3. Γιατί δε βάζεις μόνο γρήγορο, γιατί είναι ακριβά
4. Προφανώς αυτό το concept είναι γνωστό. Από που;



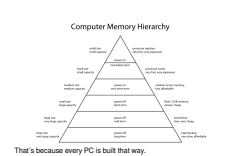
## Intro

Solution: Caching

Caching is:

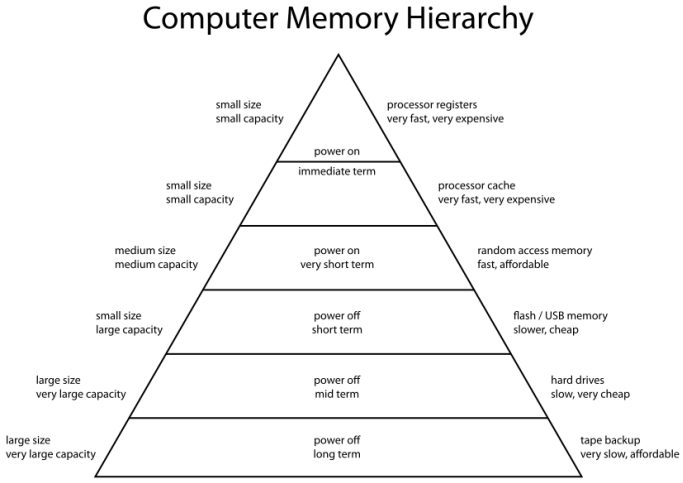
- We have a slow medium
- Add a fast medium in a data path
- Transparently store the data that are intended for the slower medium.
- Profit: later accesses to the same data are faster

Sounds familiar?




1. Τρέχα

## Caching




That's because every PC is built that way.



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-based caching
- Object-based caching

## 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-based caching
- Object-based caching



Block-based caching solutions

Most notable examples:
- Bcache
- Flashcache
- EnhanceIO

Typically scale-up solutions.

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

1. Δε θα επεκταθούμε γιατί έχουν κάποια βασικά κοινά:
   - Kernel modules
   - expose εικονικά block devices που δείχνουν σε γρήγορα μέσα
   - Καθαρά caching μηχανισμοί που παίζουν με writeback, writethrough κτλ.
2. Εξήγησε που μπαίνουν (xsegbd). ΠΗΓΑΙΝΕ στο archipelago
3. Παρότι είναι αρκετά απλοί, δε γνωρίζουν την CoW πολιτική άρα χάνουν χώρο και είναι στον kernel (που δε θέλουμε)



## Block-based caching solutions

Most notable examples:

- Bcache
- Flashcache
- EnhanceIO

Typically scale-up solutions.

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



Object-based caching solutions

Most notable examples:
• Memcached
• Couchbase

Typically scale-out solutions

Pros: Distributed with no SPOF, can utilize unneeded RAM, user-space solutions
Cons: Memcached has no persistence, Couchbase cannot use RADOS as its backend, more suitable for databases

Τα πράγματα γίνονται πιο ενδιαφέροντα εδώ:

- Κατανεμημένα συστήματα χωρίς ενιαίο σημείο αποτυχίας (SPOF), μπορούν να τρέχουν εκτός host οπότε δεν το επιβαρύνουν, είναι user-space λύσεις
- Memcached δε διατηρεί με ασφάλεια τα αντικείμενα που cachάρει, couchbase δεν μπορεί να μιλήσει με RADOS



## Object-based caching solutions

Most notable examples:

- Memcached
- Couchbase

Typically scale-out solutions

Pros: Distributed with no SPOF, can utilize unneeded RAM, user-space solutions
Cons: Memcached has no persistence, Couchbase cannot use RADOS as its backend, more suitable for databases



Conclusions

- Most solutions far from Archipelago's logic
- Others not suited for storage and more suited for databases
- Block-based caching might be good for the storage backend
- Must implement our own solution

Οι λύσεις αυτές είναι μακριά από τη λογική και το I/O pipeline του αρχιπελάγους. Είναι άσχετα συστήματα που πρέπει να επέμβουμε σε αυτά για να κάνουμε αυτά που θέλουμε. Άρα πρέπει να κάνουμε τη δική μας λύση



## Conclusions

- Most solutions far from Archipelago's logic
- Others not suited for storage and more suited for databases
- Block-based caching might be good for the storage backend
- Must implement our own solution



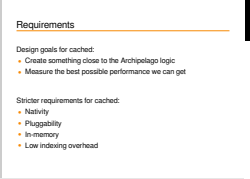
Table of Contents
Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



## Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion




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

Επιλέξαμε λοιπόν να δημιουργήσαμε τη δική μας λύση. Την ονομάσαμε cached από το cache daemonΟρίσαμε τους εξής γενικούς στόχους:

- Να είναι κοντά στη λογική του Archipelago
- Η υλοποίηση να είναι όσο το δυνατόν πιο γρήγορη για να δουμε αν μια Αρχιπελαγική λύση μας βοηθάει

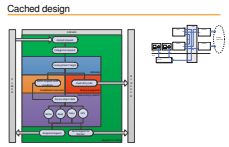
Ακόμα, θέσαμε κάποιες πιο αυστηρές απαιτήσεις για την υλοποίηση μας: 1)να είναι peer του Archipelago, 2) να μπορεί να ενεργοποιείται και να απενεργοποιείται σε ένα σύστημα που είναι εν λειτουργία, 3)να χρησιμοποιεί τη RAM, 4)ο indexing μηχανισμός να είναι γρήγορος



## 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

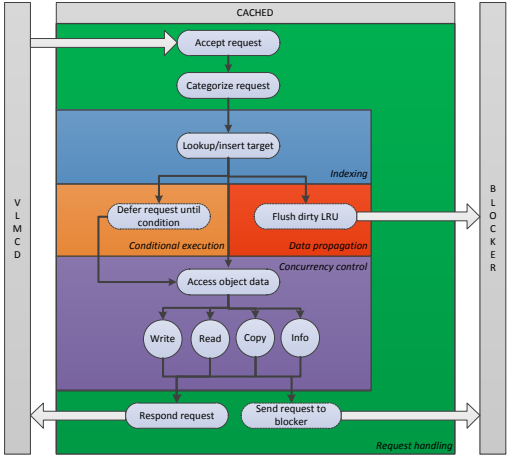


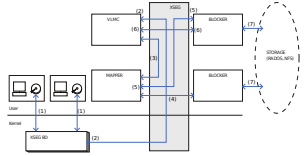

Εδώ βλέπουμε το design του cached. Ο cached μπαίνει ανάμεσα στον vlmc και στον blocker και cach-άρει ότι άιτημα για αντικείμενα πάει στο storageΟι εργασίες του cached χωρίζονται σε 5 κατηγορίες:

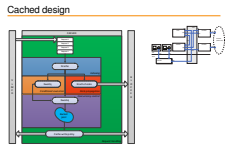
1. Στην διαχείριση των αιτημάτων από και προς vlmc, blocker
2. Στο indexing (εύρεση και καταχώρηση) των αντικειμένων
3. Στην υπο συνθήκες εκτέλεση εργασιών
4. Στην ασφαλή μετάδοση των cacharισμένων δεδομένων στο storage
5. Καθώς επίσης και στην ασφαλή επεξεργασία των cacharισμένων δεδομένων
6. CLICK!
7. Και εδώ βλέπουμε τα διακριτά κομμάτια κώδικα που υλοιποιούν τα παραπάνω και τα οποία θα συζητήσουμε ευθύς αμέσως



## Cached design



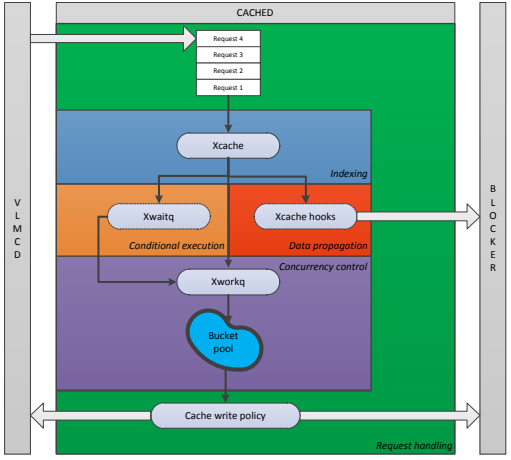


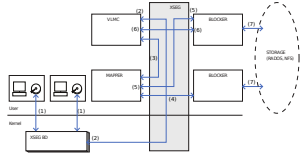

Εδώ βλέπουμε το design του cached. Ο cached μπαίνει ανάμεσα στον vlmc και στον blocker και cach-άρει ότι άιτημα για αντικείμενα πάει στο storageΟι εργασίες του cached χωρίζονται σε 5 κατηγορίες:

1. Στην διαχείριση των αιτημάτων από και προς vlmc, blocker
2. Στο indexing (εύρεση και καταχώρηση) των αντικειμένων
3. Στην υπο συνθήκες εκτέλεση εργασιών
4. Στην ασφαλή μετάδοση των cacharισμένων δεδομένων στο storage
5. Καθώς επίσης και στην ασφαλή επεξεργασία των cacharισμένων δεδομένων
6. CLICK!
7. Και εδώ βλέπουμε τα διακριτά κομμάτια κώδικα που υλοιποιούν τα παραπάνω και τα οποία θα συζητήσουμε ευθύς αμέσως

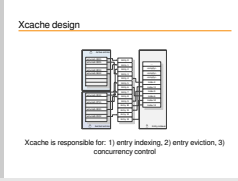


## Cached design



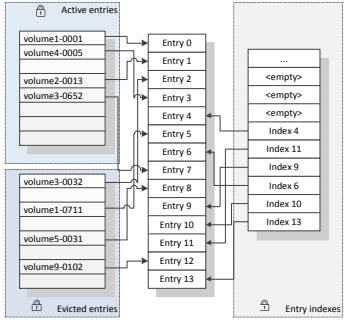





1. Αυτό είναι το xcache, που είναι υπεύθυνο για την
   - Indexing των αντικειμένων
   - Eviction αντικειμένων από την cache με τη χρήση LRU
   - Χειρισμό πολλαπλών threads
2. Έχουμε δυο hash table, το καθένα με το δικό του lock, ένα χώρο που αποθηκεύονται τα entries και μόνο (όχι τα δεδομένα, προσοχή) και μια στοίβα όπου κρατιούνται indexes των ελεύθερων entries.
3. Το ένα hash table <αυτό> που κρατάει τα ονόματα των cached αντικειμένων.
4. Ο αποθηκευτικός τους χώρος είναι preallocated και είναι <αυτό>. Σε αυτό το χώρο, η αναφορά γίνεται με δείκτες. Ο ελεύθερος χώρος είναι στη στοιβα αυτή. Τέλος, όταν ένα αντικείμενο φύγει, μένει σε αυτό το hash table μεχρι να το ξαναζητήσουν ή να το διωχτεί
5. CLICK!
6. Κάθε item έχει ένα reference counter για να ξέρουμε πόσοι το χρησιμοποιούν, όνομα, lru

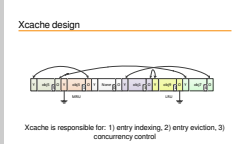


## Xcache design




Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control

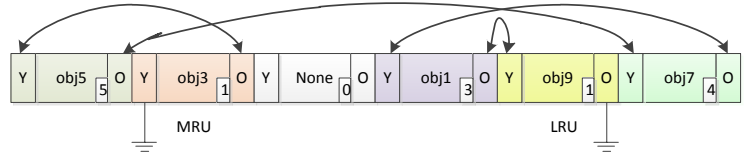





1. Αυτό είναι το xcache, που είναι υπεύθυνο για την
   - Indexing των αντικειμένων
   - Eviction αντικειμένων από την cache με τη χρήση LRU
   - Χειρισμό πολλαπλών threads
2. Έχουμε δυο hash table, το καθένα με το δικό του lock, ένα χώρο που αποθηκεύονται τα entries και μόνο (όχι τα δεδομένα, προσοχή) και μια στοίβα όπου κρατιούνται indexes των ελεύθερων entries.
3. Το ένα hash table <αυτό> που κρατάει τα ονόματα των cached αντικειμένων.
4. Ο αποθηκευτικός τους χώρος είναι preallocated και είναι <αυτό>. Σε αυτό το χώρο, η αναφορά γίνεται με δείκτες. Ο ελεύθερος χώρος είναι στη στοιβα αυτή. Τέλος, όταν ένα αντικείμενο φύγει, μένει σε αυτό το hash table μεχρι να το ξαναζητήσουν ή να το διωχτεί
5. CLICK!
6. Κάθε item έχει ένα reference counter για να ξέρουμε πόσοι το χρησιμοποιούν, όνομα, lru

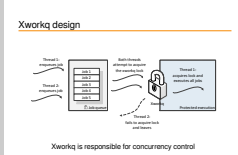


## Xcache design




Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control

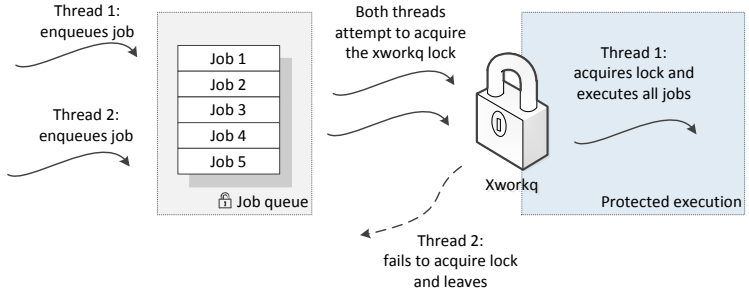





1. xworkq υπεύθυνο για την ασφαλή επεξεργασία των δεδομένων ενός αντικείμενου.
2. Το spinning είναι αργό, όλοι τοποθετούν μια δουλειά, ένας την εκτελεί.

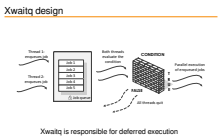


## Xworkq design




Xworkq is responsible for concurrency control

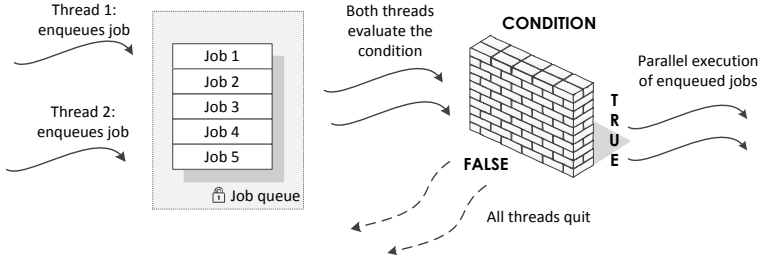





1. xwaitq υπεύθυνο για την κατά συνθήκη εκτελεση εργασιών
2. Αν π.χ. μας τελειώσει ο χώρος, δεν μπορούμε να περιμένουμε σύγχρονα. Το thread μπορεί να τοποθετήσει μια δουλειά και μετά να εκτελέσει κάτι άλλο



# 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

1. Το ότι κάνουμε index ένα object δε σημαίνει ότι κατ'ευθείαν μπορούμε να γράψουμε σε αυτό
2. Δεν υπάρχει τόση RAM και ακόμα και ως αποτέλεσμα, θα cacháραμε μικρό αριθμό από objects
3. Ιδανικά θέλουμε να διαχωρίσουμε την καταχώρη/όνομα του αντικειμένου από τα δεδομένα του. Δυνητικά θα μπορούμε να καταχωρούμε πάρα πολλά αντικείμενα αλλά θα έχουμε μικρότερο χώρο
4. Preallocated χώρος, όλοι παίρνουν indexes από αυτό (Θυμίζει xcache



## 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



Το cached είναι επίσης επιφορτισμένο και με άλλες δουλειές όπως:

- Κρατάει στατιστικά (πόσα entries είναι dirty, πόσα buckets έχει κάνει allocate ένα entry
- Εφαρμόζει writeback/writethrough πολιτική
- Φρόντίζει ώστε οι εργασίες να μπορούν να γίνουν ασύγχρονα
- Και φυσικά φροντίζει τα δεδομένα να γράφονται σωστά στο storage

Other important cached tasks

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

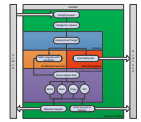


## Other important cached tasks

Several other key-tasks are:

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

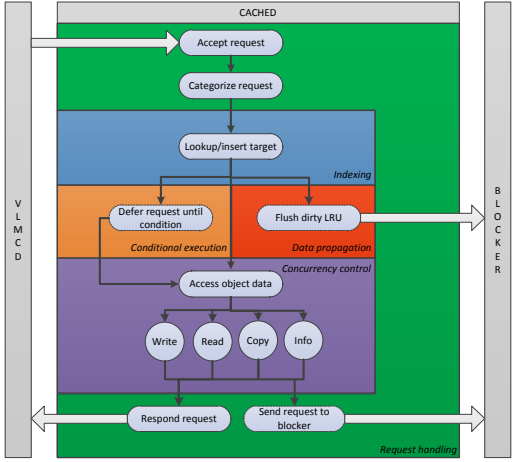


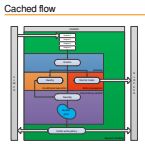



1. Εδώ παίζεις με τα slides
2. Θα παρουσιάσουμε πολύ γρήγορα τη ροή ενός αιτήματος στον cached
3. Έρχεται request, το κάνουμε index, μπαίνουμε στη workq και πειράζουμε τα δεδομένα του και ανάλογα το cache policy το γράφουμε πίσω στον blocker αλλιώς τελειώσαμε
4. Optional σεναρια:
   - Αν γίνει ένα eviction, πρέπει να γράψουμε τα δεδομένα του πίσω με ασφάλεια. Επειδή το αντικείμενο μπορει να καταχωρηθεί, να μπει και να ξαναβγεί, πρέπει να είμαστε προσεκτικοί
   - Αν ξεμείνουμε από πόρους (χώρο στο hash table, buckets κτλ, πρέπει να συνεχίσουμε μονο όταν μπορούμε



## Cached flow






1. Εδώ παίζεις με τα slides
2. Θα παρουσιάσουμε πολύ γρήγορα τη ροή ενός αιτήματος στον cached
3. Έρχεται request, το κάνουμε index, μπαίνουμε στη workq και πειράζουμε τα δεδομένα του και ανάλογα το cache policy το γράφουμε πίσω στον blocker αλλιώς τελειώσαμε
4. Optional σεναρια:
   - Αν γίνει ένα eviction, πρέπει να γράψουμε τα δεδομένα του πίσω με ασφάλεια. Επειδή το αντικείμενο μπορει να καταχωρηθεί, να μπει και να ξαναβγεί, πρέπει να είμαστε προσεκτικοί
   - Αν ξεμείνουμε από πόρους (χώρο στο hash table, buckets κτλ, πρέπει να συνεχίσουμε μονο όταν μπορούμε



## Cached flow

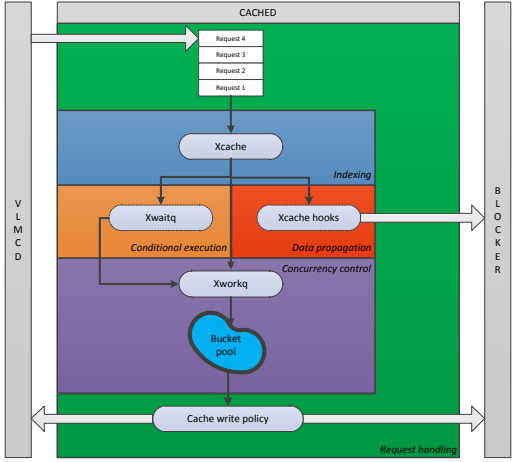


Table of Contents
Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



## Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



Οι μετρήσεις μας είναι εκτενείς και χωρίζονται σε τρεις κατηγορίες:

- Σύγκριση performance του cached και rados για workloads μικρότερα και μεγαλύτερα του cache size
- Αξιολόγηση εσωτερικών κομματιών του cached (multithreading, overhead του indexing μηχανισμου
- Μετρήσεις του Αρχιπελαγο για ένα πραγματικό VM

Benchmark methodology

We have conducted exhaustive benchmarks.
They are separated in three categories:
- Comparison between cached and RADOS
  - Peak behavior
  - Sustained behavior
- Internal comparison of cached
  - Multithreading overhead
  - Indexing mechanism overhead
- Evaluation of VM/Archipelago

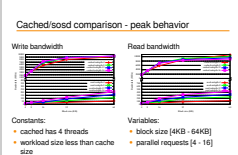


## Benchmark methodology

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

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



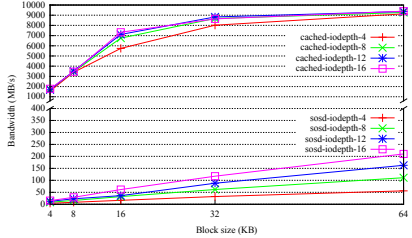



1. Σημεία προσοχής:Για μικρά writes είμαστε έως 100x γρηγορότεροι ενώ για μεγάλα έως 200x.
2. Ο cached μετά τα 16KB δεν κάνει scale - χτυπάμε το bandwidth της RAM
3. Έχουμε lock contention, δε θα έπρεπε να αυξάνεται η ταχύτητα για μεγάλα blocks και δεν αυξάνεται η ταχύτητα με parallel requests

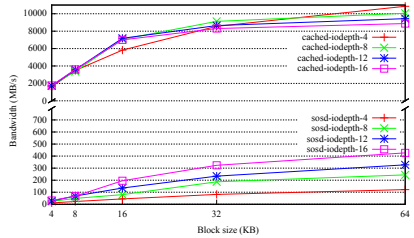


## Cached/sosd comparison - peak behavior

Write bandwidth




Read bandwidth



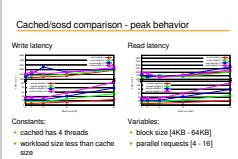

Constants:

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

Variables:

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



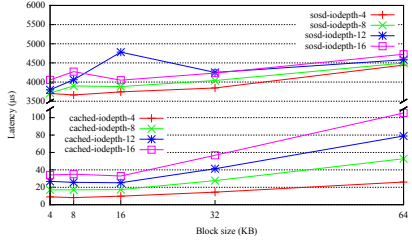



1. Αντίστοιχα, για μικρά reads είμαστε 50x γρηγορότεροι ενώ για μεγάλα έως 75x

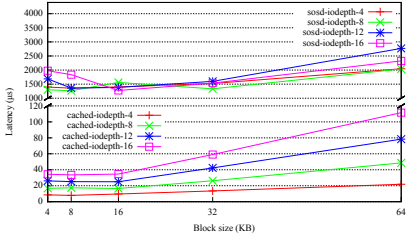


## Cached/sosd comparison - peak behavior

### Write latency




### Read latency



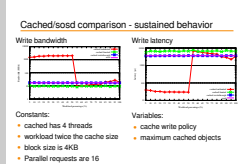

Constants:

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

Variables:

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



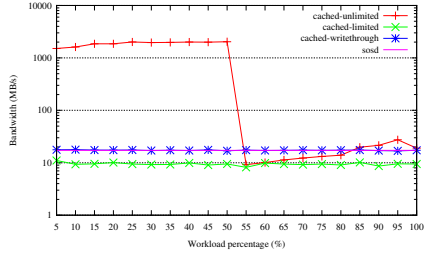



1. unlimited: έχουμε περισσότερα buckets απ' ότι objects
2. Σημεία προσοχής: Writethrough όσο και το Rados ενώ στα reads έχουμε παρατηρήσει καλύτερη ταχύτητα
3. Το performance πέφτει λόγω έλλειψης buckets, μεγαλώνει λόγω coalesces

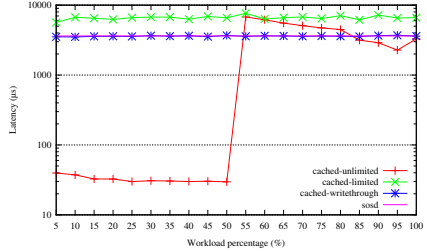


## Cached/sosd comparison - sustained behavior

### Write bandwidth




### Write latency

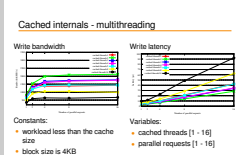



Constants:

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

Variables:

- cache write policy
- maximum cached objects

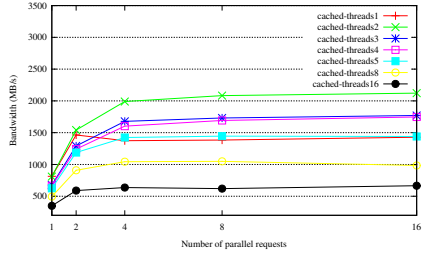



1. Μέχρι 2 είμαστε καλά γενικά. Αν έχουμε πολλά parallel requests, τότε φτάνουμε μέχρι και 5
2. Σαφή ένδειξη lock contention

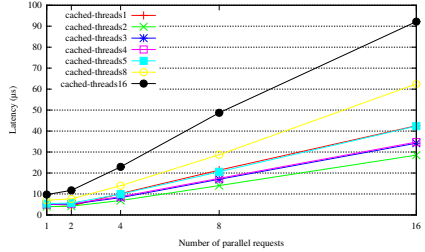


## Cached internals - multithreading

### Write bandwidth




### Write latency




### Constants:

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

### Variables:

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



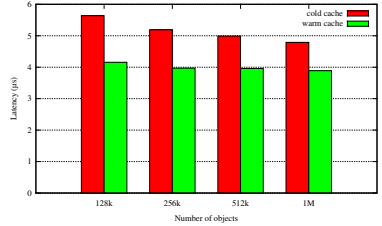
Cached internals - indexing
Latency of cold cache vs. warm cache
Constants:
• workload less than the cache size
• block size is 4KB
• no threads or parallel requests
Variables:
• number of objects [128k - 1G]

1. Σταθερό indexing overhead. Αν πέσει ερώτηση πες 2M hash table, το λειτουργικό δε δίνει αμέσως μνήμη



## Cached internals - indexing

Latency of cold cache vs. warm cache



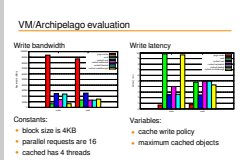

Constants:

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

Variables:

- number of objects [128k - 1G]





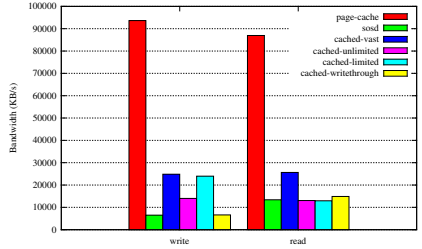

Σημεία προσοχής:

- page-cache: πολύ γρήγορη. Το 1ms latency λογικά μπαίνει λόγω του paravirtualized storage, filesystem, elevators
- sosd: είναι σίγουρα άσχημο αλλά σε αυτά τα test έχει συν 7ms latency για τα writes και 3ms latency για τα reads. Αυτό είναι πολύ μεγαλύτερο του 1ms του VM άρα κάτι παίζει με Αρχιπέλαγο
- cached-vast: 4x γρηγορότερη από sosd αλλά έχει 3ms latency που δεν είχε πριν, δηλαδή το archipelago βάζει 2ms
- cached-unlimited: 2.5x γρηγορότερο και ξεπέρασε πάλι τον sosd στο τέλος
- cached-limited: 4x γρηγορότερο, λογικά τα flushes είναι πολλα και μικρά και κρύβονται πίσω από το latency του Archipelago
- writethrough δεν είδαμε κάποια διαφορά

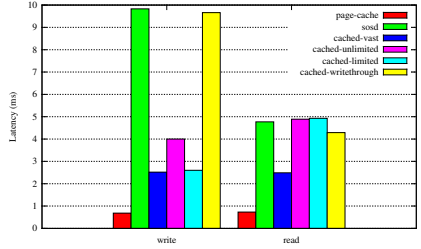


## VM/Archipelago evaluation

### 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
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



## Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



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?

1. Από τα προηγούμενα συμπεράσματα, μπορούμε να εξάγουμε ότι έχουμε τα εξής limitations ανεξαρτήτως latency Αρχιπελάγους: lock contention και έλλειψη από RAM
2. Αν ο cached τρέχει στον host όπου τρέχουν και τα VMs, θα έχουμε λιγότερη cpu -> περισσότερο contention και λιγότερη ram
3. Αν έτρεχε στους αποθηκευτικούς κόμβους;



## 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?



1. Αν έτρεχε εκεί θα TODO:

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

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.

1. Το synapsed σχεδιάστηκε για τα εξής:
   - Σύνδεση δυο Αρχιπέλαγο peers πάνω από network
   - Κατάλληλη προώθηση I/O αιτημάτων
   - Χρήση του tcp πρωτοκόλλου
   - Χρήση του signaling μηχανισμού του Αρχιπελάγους
   - Χρήση μεθόδων zero-copy
2. Η δημιουργία αντιγράφων μπορεί να προστεθεί εύκολα στα παραπάνω, αλλά εμείς δεν φτάσαμε ώς εκεί



## 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.



Table of Contents
Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



# Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



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.

1. Ο κύριως στόχος του synapsed είναι να προσφέρει τη δυνατότητα ή ελαστικότηατα αν το θέλετε, του να τρέχει ο cached ή κομματι του Archipelago σε άλλο κόμβο. Ας δούμε όμως την επίδοσή του



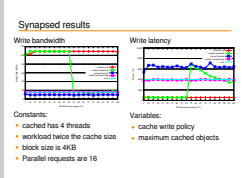
## Benchmark preamble

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.





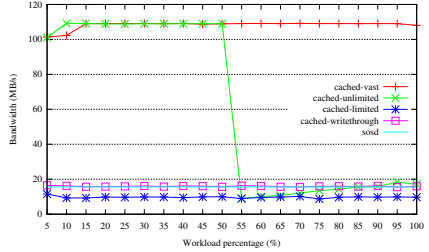

Σημεία προσοχής, πρακτικά υπάρχει πολύ μικρή διαφορά με το διάγραμμα της σελίδας 32. Απλά μπαίνει μικρότερο του 1ms latency που για το cached-vast φυσικά κάνει μεγάλη διαφορά.

Κατά τα άλλα, το latency αυτό είναι αμελητέο σε σχέση με το τρέχων latency, ενώ αν υπήρχε 10 ή 40Gbit δίκτυο, θα ήταν ακόμα καλύτερα τα πράγματα.

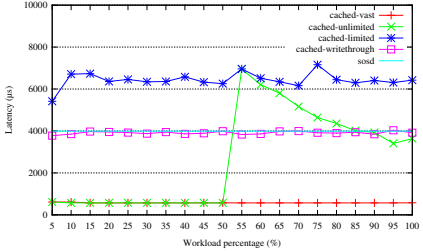


## Synapsed results

### 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
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
Conclusion



## Table of Contents

Introduction
VM Volume storage
Caching
Cached design
Cached evaluation
Synapsed design
Synapsed evaluation
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



# 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



That's all folks!

Questions?



## That's all folks!

Questions?