



# 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 και αντικείμενο της  
διπλωματικής, 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. Παρ'όλα αυτά όμως, αν κάποια  
από αυτές τις έννοιες είναι άγνωστες ή για όποια απορία κατά τη  
διάρκεια της παρουσίασης, μπορείτε να με διακόψετε και να  
ρωτήσετε.



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 και τι είναι η υπηρεσία okeanos
- Έπειτα, μιλάμε για τον τρόπο που διαχειριζόμαστε το storage και κατ'επέκταση για το archipelago.
- Στη συνέχεια μιλάμε για το τι είναι caching και αναφέρουμε κάποιες σύγχρονες λύσεις για caching.
- Τα επόμενα δυο κεφάλαια έχουν να κάνουν με τον cached, και συγκεκριμένα με την παρουσίαση της σχεδιάσής του και της απόδοσής του
- Αντίστοιχα παρουσιάζουμε το synapsed, τη σχεδίαση και υλοποίησή του
- Τέλος, συνοψίζουμε όσα ειπώθηκαν παραπάνω και μιλάμε για μελλοντικές 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



- 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



## 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
- ...and of course powered by Synnefo.



## Table of Contents

Introduction

VM Volume storage

Caching

Cached design

Cached evaluation

Synapsed design

Synapsed evaluation

Conclusion



## 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

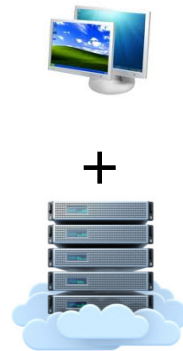
1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:
2. **CLICK!**
3. Το VM που τρέχει σε ένα host server
4. **CLICK!**
5. Και τους storage servers
6. **CLICK!**
7. Το ερώτημα είναι λοιπόν, πως το 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

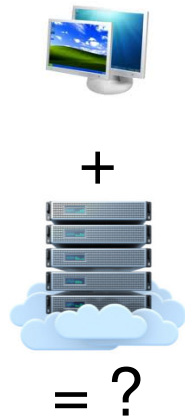
1. Σε μεγάλες εγκαταστάσεις, ξεφεύγουμε από το κλασσικό μοντέλο του PC μας (το μηχάνημα έχει το σκληρό του δίσκο). Συγκεκριμένα σε μια cloud υποδομή έχουμε:
2. **CLICK!**
3. Το VM που τρέχει σε ένα host server
4. **CLICK!**
5. Και τους storage servers
6. **CLICK!**
7. Το ερώτημα είναι λοιπόν, πως το 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?

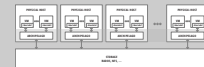


└ VM Volume storage

└ Our solution

Our solution

Archipelago



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

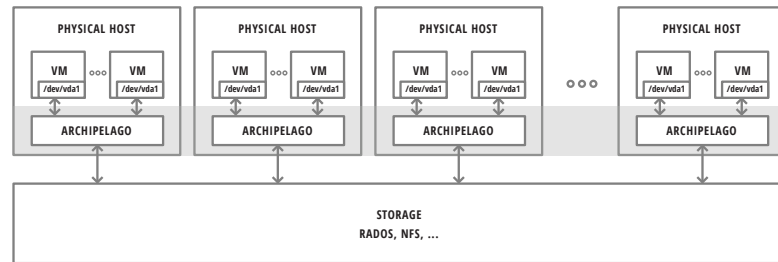
Η λύση που χρησιμοποιήσαμε είναι το Archipelago

- 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



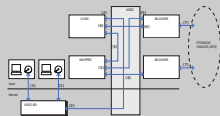
Key features: 1) Software-defined 2) Distributed 3) Modular 4) Copy-On-Write 5) Storage agnostic



└ VM Volume storage

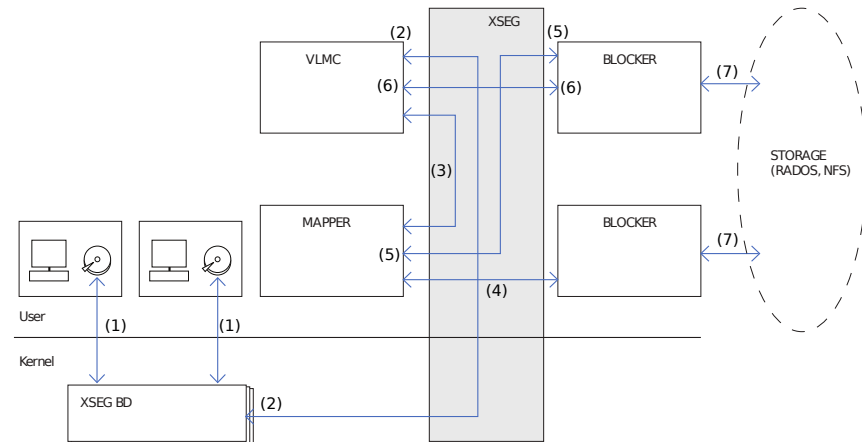
└ Archipelago Architecture

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



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.



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

## Table of Contents

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

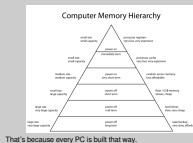
### Solution: Caching

#### 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

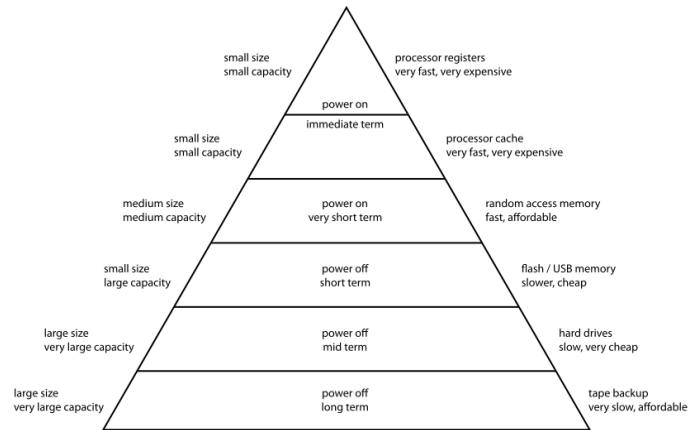
#### Sounds familiar?





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-based caching
- Object-based caching

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



- Bcache
- Flashcache
- EnhanceIO

## Block-based caching solutions

1. Δε θα επεκταθούμε γιατί έχουν κάποια βασικά κοινά:
  - Kernel modules
  - expose εικονικά block devices που δείχνουν σε γρήγορα μέσα
  - Καθαρά caching μηχανισμοί με policies
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)

Most notable examples:

- Bcache
- Flashcache
- EnhanceIO

Typically scale-up solutions.

Pros: Simple, scale-up

Cons: Unaware of CoW, kernel solutions



- Memcached
- Couchbase

## Object-based caching solutions

Τα πράγματα γίνονται πιο ενδιαφέροντα εδώ:

- Κατανεμημένα συστήμα χωρίς ενιαίο σημείο αποτυχίας (SPOF), μπορούν να τρέχουν εκτός host οπότε δεν το επιβαρύνουν
- Memcached δε διατηρεί τα δεδομένα, couchbase δεν μπορεί να μιλήσει με rados

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?

Αν χρησιμοποιούσαμε την page-cache Εύκολη λύση, σταθερή και γρήγορη. Όμως, δεν καταλαβαίνει από COW, είναι προσκολλημένη σε ένα block device

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
- 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

## Conclusions

### TODO:

- 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



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

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

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

- Δημιουργία ενός peer 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 τρέχει, 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



└ Cached design

└ Cached design

Cached design



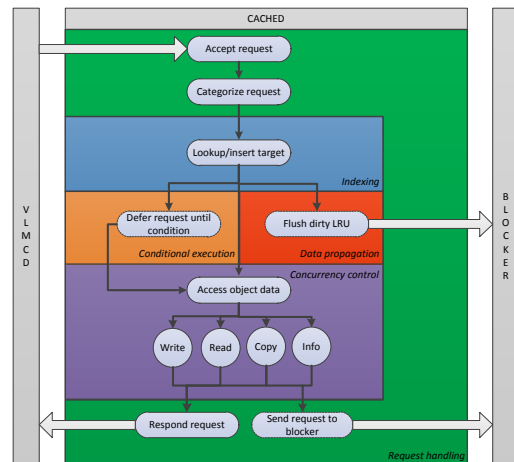
Και εδώ βλέπουμε τα  
διακριτά κομμάτια που υλοποιούν τα παραπάνω και τα οποία θα  
συζητήσουμε ευθύς αμέσως

Εδώ βλέπουμε το design του cached. Ο cached μπαίνει ανάμεσα  
στον vmc και στον blocker και cach-άρει ότι αίτημα για αντικείμενα  
πάει στο storage. Οι εργασίες του cached χωρίζονται σε 5  
κατηγορίες: item

1. Στην διαχείριση των αιτημάτων από και προς vmc, blocker
2. Στο indexing (εύρεση και καταχώρηση) των αντικειμένων
3. Στην ασύγχρονη ή κατά όρους εκτέλεση εργασιών
4. Στην ασφαλή μετάδοση των cachαρισμένων δεδομένων στο storage
5. Καθώς επίσης και στην ασφαλή επεξεργασία των cachαρισμένων  
δεδομένων
6. **CLICK!**

## Cached design

## Cached design



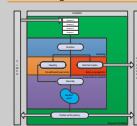
Και εδώ βλέπουμε τα  
διακριτά κομμάτια που υλοποιούν τα παραπάνω και τα οποία θα  
συζητήσουμε ευθύς αμέσως



└ Cached design

└ Cached design

Cached design



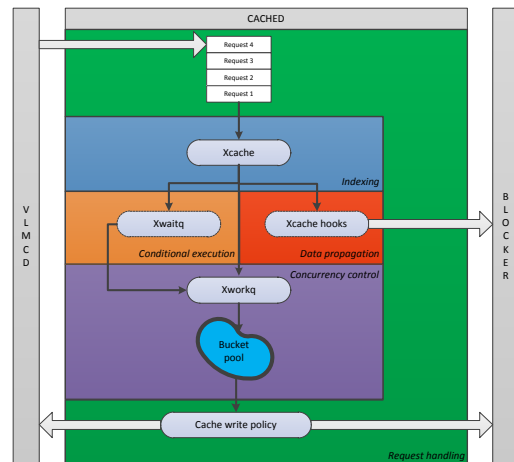
Και εδώ βλέπουμε τα  
διακριτά κομμάτια που υλοποιούν τα παραπάνω και τα οποία θα  
συζητήσουμε ευθύς αμέσως

Εδώ βλέπουμε το design του cached. Ο cached μπαίνει ανάμεσα στον vmlc και στον blocker και cach-άρει ότι αίτημα για αντικείμενα πάει στο storage. Οι εργασίες του cached χωρίζονται σε 5 κατηγορίες: item

1. Στην διαχείριση των αιτημάτων από και προς vmlc, blocker
2. Στο indexing (εύρεση και καταχώρηση) των αντικειμένων
3. Στην ασύγχρονη ή κατά όρους εκτέλεση εργασιών
4. Στην ασφαλή μετάδοση των cachαρισμένων δεδομένων στο storage
5. Καθώς επίσης και στην ασφαλή επεξεργασία των cachαρισμένων δεδομένων
6. **CLICK!**

## Cached design

## Cached design



Και εδώ βλέπουμε τα  
διακριτά κομμάτια που υλοποιούν τα παραπάνω και τα οποία θα  
συζητήσουμε ευθύς αμέσως



└─ Cached design

└─ Xcache design

Xcache design

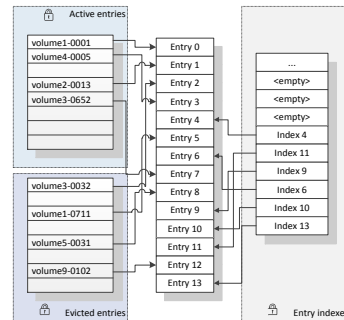


Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control

Cached design

## Xcache design

- Αυτό είναι το xcache, που είναι υπεύθυνο για την
  - Εύρεση και καταχώριση των αντικειμένων
  - Έξωση αντικειμένων από την cache με τη χρήση LRU
  - Χειρισμό πολλαπλών threads
- Εχουμε ένα hash table <αυτό> που κρατάει τα ονόματα των αντικειμένων. 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
- CLICK!**
- Κάθε item έχει ένα reference counter για να ξέρουμε πόσοι το χρησιμοποιούν, όνομα, lru



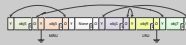
Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control



└─ Cached design

└─ Xcache design

Xcache design

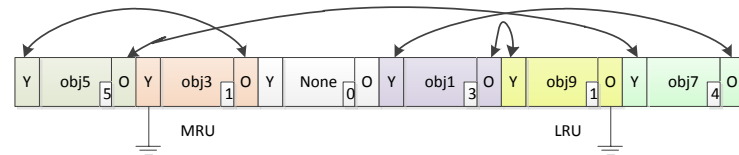


Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control

Cached design

## Xcache design

- Αυτό είναι το xcache, που είναι υπεύθυνο για την
  - Εύρεση και καταχώριση των αντικειμένων
  - Έξωση αντικειμένων από την cache με τη χρήση LRU
  - Χειρισμό πολλαπλών threads
- Εχουμε ένα hash table <αυτό> που κρατάει τα ονόματα των αντικειμένων. 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
- CLICK!**
- Κάθε item έχει ένα reference counter για να ξέρουμε πόσοι το χρησιμοποιούν, όνομα, lru



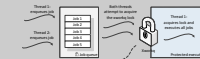
Xcache is responsible for: 1) entry indexing, 2) entry eviction, 3) concurrency control



└─ Cached design

└─ Xworkq design

Xworkq design

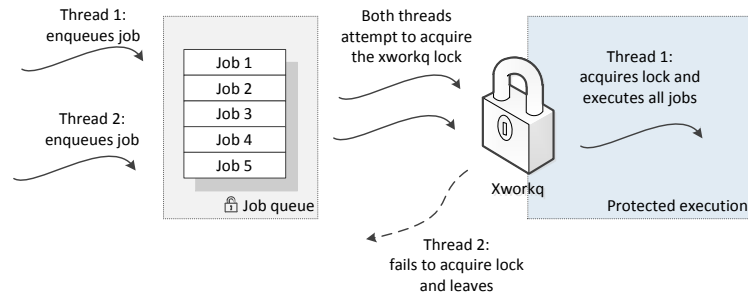


Xworkq is responsible for concurrency control

Cached design

## Xworkq design

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



Xworkq is responsible for concurrency control

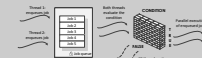




└─ Cached design

└─ Xwaitq design

Xwaitq design

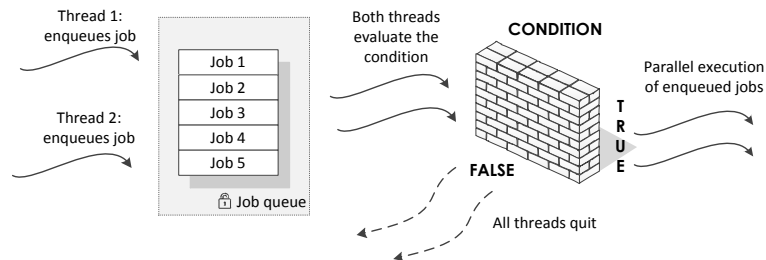


Xwaitq is responsible for deferred execution

Cached design

## Xwaitq design

1. xwaitq υπεύθυνο για την κατά συνθήκη εκτέλεση 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 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

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 cachάραμε μικρό αριθμό από objects
3. Ιδανικά θέλουμε να διαχωρίσουμε την καταχώρη/όνομα του αντικειμένου από τα δεδομένα του. Δυσνητικά θα μπορούμε να καταχωρούμε πάρα πολλά αντικείμενα αλλά θα έχουμε μικρότερο χώρο
4. Preallocated χώρος, όλοι παίρνουν indexes από αυτό (Θυμίζει xcache)



└─ 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

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
- Φροντίζει ώστε οι εργασίες να μπορούν να γίνουν ασύγχρονα
- Και φυσικά φροντίζει τα δεδομένα να γράφονται σωστά στο storage

Several other key-tasks are:

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



└─ Cached design

└─ Cached flow

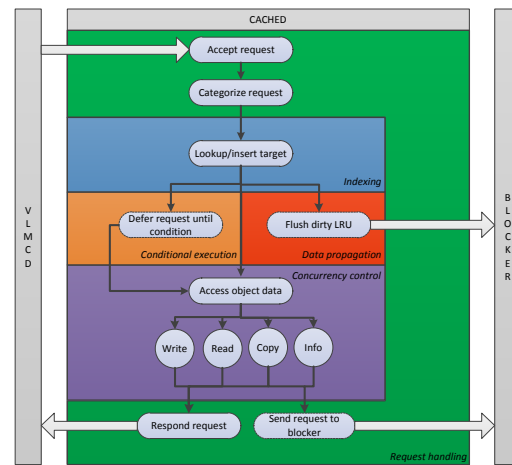
Cached flow



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 μπορεί να καταχωρηθεί, να μπει και να ξαναβγεί, πρέπει να είμαστε προσεκτικοί
  - Αν ξεμείνουμε από πόρους (χώρο στο hash table, buckets κτλ, πρέπει να συνεχίσουμε μόνο όταν μπορούμε

## Cached design

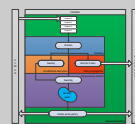
### Cached flow



└─ Cached design

└─ Cached flow

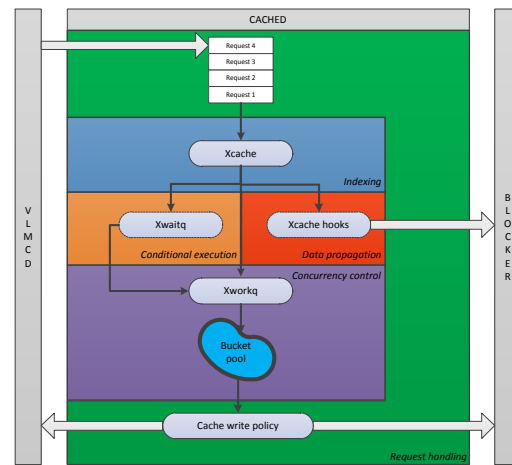
Cached flow



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 μπορεί να καταχωρηθεί, να μπει και να ξαναβγεί, πρέπει να είμαστε προσεκτικοί
  - Αν ξεμείνουμε από πόρους (χώρο στο hash table, buckets κτλ, πρέπει να συνεχίσουμε μόνο όταν μπορούμε

## Cached design

### Cached flow



## 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 μηχανισμού)
- Μετρήσεις του Αρχιπελάγου για ένα πραγματικό VM

## Benchmark methodology

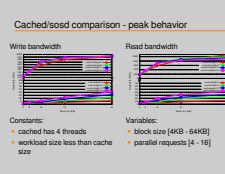
We have conducted exhaustive benchmarks.  
They are separated in three categories:

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



└─ Cached evaluation

└─ Cached/sosd comparison - peak behavior

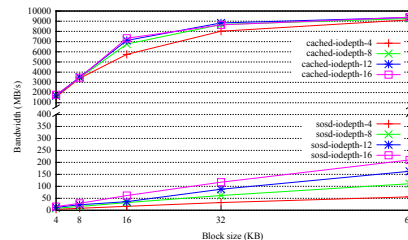


1. Σημεία προσοχής: Για μικρά writes είμαστε έως 100x γρηγορότεροι ενώ για μεγάλα έως 200x.
2. Ο cached μετά τα 16KB δεν κάνει scale - χτυπάμε το bandwidth της RAM
3. Έχουμε lock contention, δε θα έπρεπε να αυξάνεται η ταχύτητα για μεγάλα blocks και δεν αυξάνεται η ταχύτητα με parallel requests

## Cached evaluation

### Cached/sosd comparison - peak behavior

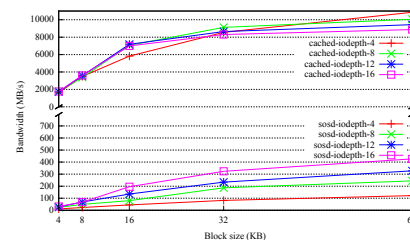
#### Write bandwidth



#### Constants:

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

#### Read bandwidth



#### Variables:

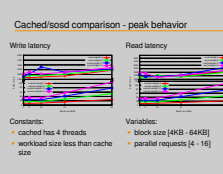
- block size [4KB - 64KB]
- parallel requests [4 - 16]





└─ Cached evaluation

└─ Cached/sosd comparison - peak behavior

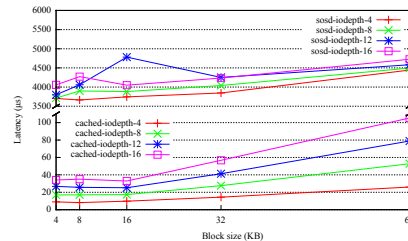


1. Αντίστοιχα, για μικρά reads είμαστε 50x γρηγορότεροι ενώ για μεγάλα έως 75x

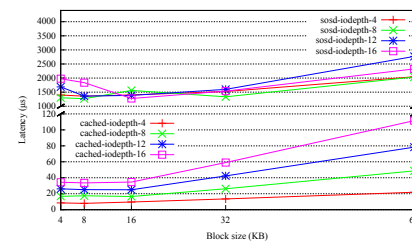
## Cached evaluation

### Cached/sosd comparison - peak behavior

#### Write latency



#### Read latency



#### Constants:

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

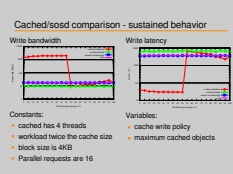
#### Variables:

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



└─ Cached evaluation

└─ Cached/sosd comparison - sustained behavior

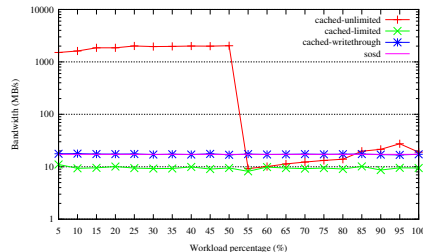


1. unlimited: έχουμε περισσότερα buckets απ' ότι objects
2. Σημεία προσοχής: Writethrough όσο και το Rados ενώ στα reads έχουμε παρατηρήσει καλύτερη ταχύτητα
3. Το performance πέφτει λόγω έλλειψης buckets, μεγαλώνει λόγω coalesces

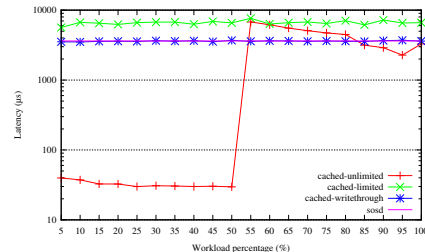
## Cached evaluation

### 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

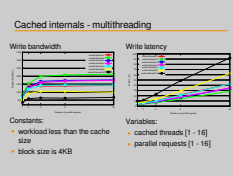
#### Variables:

- cache write policy
- maximum cached objects



└─ Cached evaluation

└─ Cached internals - multithreading

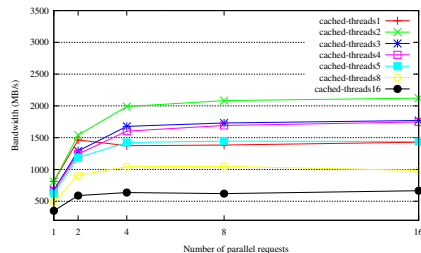


1. Μέχρι 2 είμαστε καλά γενικά. Αν έχουμε πολλά parallel requests, τότε φτάνουμε μέχρι και 5
2. Σαφή ένδειξη lock contention

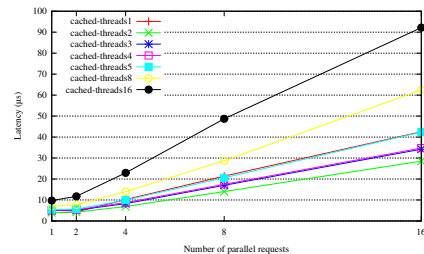
## Cached evaluation

### 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:

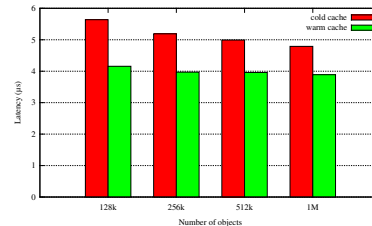
- number of objects [128k - 1G]

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

## 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

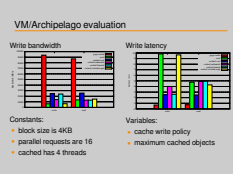
Variables:

- number of objects [128k - 1G]



└─ Cached evaluation

└─ VM/Archipelago evaluation



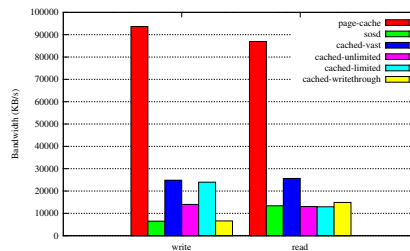
### Σημεία προσοχής:

- 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 πολλά και μικρά και κρύβονται πίσω από το latency του Archipelago
- writethrough δεν είδαμε κάποια διαφορά

## Cached evaluation

### VM/Archipelago evaluation

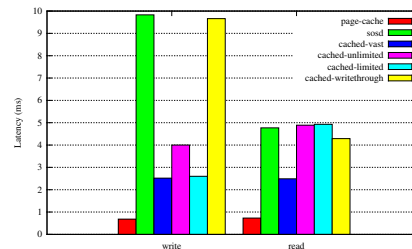
#### Write bandwidth



#### 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 design**

Synapsed evaluation

Conclusion



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

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

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?



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.

1. Αν έτρεχε εκεί θα **TODO**:





- 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

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 ως εκεί

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



## Benchmark preamble

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

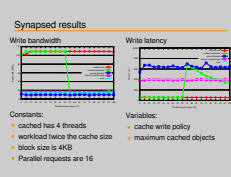
The most important part is that synapsed works. We are **now** able to run cached or part of Archipelago in the storage nodes.

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.



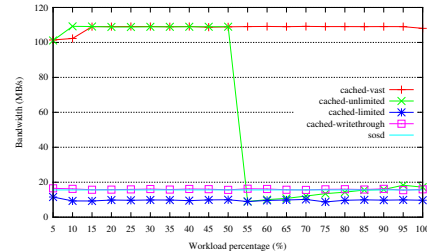


Σημεία προσοχής, πρακτικά υπάρχει πολύ μικρή διαφορά με το διάγραμμα της σελίδας 32. Απλά μπαίνει μικρότερο του 1ms latency που για το cached-vast φυσικά κάνει μεγάλη διαφορά.

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

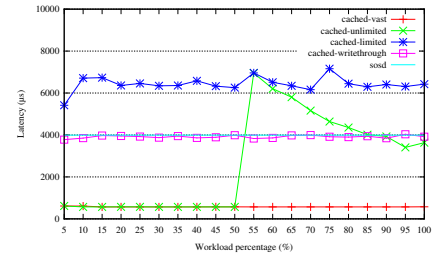
### Write bandwidth



#### 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 design](#)[Synapsed evaluation](#)[Conclusion](#)

Conclusion

Concluding remarks

#### 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

### 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



└ Conclusion

└ That's all folks!

That's all folks!

Questions?

## Conclusion

That's all folks!

Questions?

