ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Επισκόπηση και σύγκριση των τεχνολογιών κατασκευής διακομιστών διαδικτύου υψηλής διαθεσιμότητας (high availability (HA) web server) στα λειτουργικά συστήματα Microsoft Windows και Unix-based (Linux/BSD) και κατασκευή ενός τέτοιου συστήματος"

Του φοιτητή

Σπύρου Ανδρέου

Αρ. Μητρώου: 032159

Επιβλέπων καθηγητής

Δρ. Μηνάς Δασυγένης

Θεσσαλονίκη 2011

ΠΡΟΛΟΓΟΣ

Στην παρούσα εργασία θα αναφερθούμε στους διακομιστές υψηλής διαθεσιμότητας. Θα δούμε τις πιο δημοφιλείς τεχνολογίες που υπάρχουν στην αγορά σήμερα. Θα υλοποιήσουμε τρεις διαφορετικές τεχνολογίες και θα δείξουμε τις κρίσιμες διαφορές που υπάρχουν μεταξύ τους. Στόχος δεν είναι να απορρίψουμε κάποια τεχνολογία αλλά να παραθέσουμε τα αποτελέσματα των τεστ που θα τρέξουμε, ώστε να δούμε την απόκριση των συστημάτων σε συνδυασμό με το τι παρέχει κάθε τεχνολογία, δηλαδή ποια εργαλεία την συνοδεύουν, σε ποια συστήματα μπορούν να γίνουν εγκατάσταση, ποιες είναι οι απαιτήσεις τους αλλά και το εάν παρέχουν καλή τεκμηρίωση.

ΠΕΡΙΛΗΨΗ

Στην εισαγωγή αναφερόμαστε γενικά στο τι είναι το cluster computing στους ηλεκτρονικούς υπολογιστές αλλά και στο ποια ήταν η ανάγκη για να δημιουργηθεί κάτι τέτοιο. Ακόμη αναφερόμαστε στις κατηγορίες που υπάρχουν σήμερα. Αναλύουμε το εργαλείο όπου θα μας βοηθήσει να δημιουργήσουμε τα εικονικά συστήματα δηλαδή το Vbox. Τέλος αναφερόμαστε στο ποια είναι τα συστήματα RAID και SAN.

Στο πρώτο κεφάλαιο βλέπουμε πως υλοποιείται το High Availability στο LINUX, αναλύοντας το DRBD, δηλαδή το πως δουλεύει εσωτερικά αλλά και τα εργαλεία τα οποία εμπεριέχει. Ακόμη τι είναι το HeartBeat αλλά και που χρησιμεύει.

Τέλος, αναλύουμε πως υλοποιείτε ένα διαδραστικό site σε συνδυασμό με το DRBD και HeartBeat αλλά και την απόδοση τον συστημάτων κάτω από πίεση.

Στο δεύτερο κεφάλαιο καταγράφουμε πως υλοποιείται το High Availability στο FreeBSD, αναλύοντας το HAST και το UCARP, πώς δουλεύουν αλλά και που χρησιμοποιούνται. Έπειτα πως δημιουργούμε ένα διαδραστικό site σε συνδυασμό με το HAST και το UCARP αλλά και την απόδοση των συστημάτων.

Στο τρίτο κεφάλαιο περιγράφουμε πως υλοποιείται το High Availability στα Windows αναλύοντας το FailOver cluster και το FreeNAS. Έπειτα θα δούμε πως δημιουργούμε ένα διαδραστικό site σε συνδυασμό με το FailOver και το FreeNAS αλλά και την απόδοση των συστημάτων.

Τέλος θα βγάλουμε κάποια συμπεράσματα συγκρίνοντας τα αποτελέσματα των τεστ μεταξύ τους.

ABSTRACT

In the introduction, we refer generally to what is cluster computing and which was the need to create something like this. We also refer to the categories that exist today. We analyze the tool which will help us to create virtual systems, which is called Vbox. Finally we refer to what is RAID and SAN systems.

In the first chapter we see how we can implement the High Availability in LINUX analyzing DRBD, how it works internally and the tools which provide. We will also see what is HeartBeat and when we should use it.

Finally, we describe how we can implement an interactive site in conjunction with DRBD and HeartBeat and the system performance under pressure.

In the second chapter we describe how can be FreeBSD High Availability implemented by analyzing the HAST and UCARP as a reference to how they work and used. Then we will see how to create an interactive site in conjunction with HAST and UCARP and performance of the system.

In the third chapter we analyze the implementation of the High Availability in Windows analyzing FailOver cluster and FreeNAS. Then we will see how to create an interactive site in conjunction with FailOver and FreeNAS and the system performance.

Finally, we draw some conclusions from the results of the test.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Ευρετήριο πινάκων

Ευρετήριο εικόνων

ΕΙΣΑΓΩΓΗ

Καθώς όλο και περισσότερο οι ηλεκτρονικοί υπολογιστές είναι αναγκαίοι στην καθημερινότητά μας, οδηγούμαστε σε συστήματα τα οποία θα πρέπει να είναι προσπελάσιμα καθημερινά όλο το εικοσιτετράωρο. Ακόμη τα δεδομένα τα οποία αποθηκεύουμε σε αυτά είναι αρκετές φορές πολύ σημαντικά.

Αυτές οι δύο ανάγκες δημιούργησαν γύρω στο 1960 την υπολογιστική συστοιχία (cluster computing) την οποία χρησιμοποίησε η IBM για τις ανάγκες τον πελατών της για αντίγραφα ασφαλείας (backup) και για λόγους αποθηκευτικού χώρου. Η επίσημη όμως δημοσίευση ήρθε το 1967 από τον Gene Amdahl [1] ο οποίος εξέτασε διεξοδικά αυτή την τεχνολογία.

Σήμερα το cluster computing χωρίζεται σε τρεις κατηγορίες.

- Load-balancing clusters

Είναι η τεχνολογία με την οποία πολλοί υπολογιστές διασυνδέονται μεταξύ τους ώστε να διαμοιράσουν τον φόρτο εργασίας. Όλοι οι υπολογιστές δρουν και φαίνονται σαν μια οντότητα.

- Compute clusters

Εδώ έχουμε να κάνουμε με συστοιχία υπολογιστών οι οποίοι εκτελούν ένα πρόγραμμα ή εργασία παράλληλα.

- High Availability (HA) clusters

Η τεχνολογία αυτή προσφέρει αδιάλειπτη προσπέλαση σε υπηρεσίες. Με αυτήν την τεχνολογία θα ασχοληθούμε στην παρούσα εργασία. Η τεχνολογία αυτή χρησιμοποιείται κυρίως για Web εφαρμογές όπου η διαθεσιμότητα είναι κύριο στοιχείο σε συνδυασμό με βάση δεδομένων.

Σκοπός της εργασίας αυτής είναι να παραθέσει τις πιο δημοφιλείς υλοποιήσεις καθώς και την σύγκριση αυτών σε σχέση απόδοσης, παραμετροποίησης, υποστήριξης και την εξαγωγή συμπερασμάτων.

Για το σκοπό αυτό θα χρησιμοποιήσουμε το Virtual Box (Vbox) [2] της Sun το οποίο προσφέρει την δυνατότητα δημιουργίας εικονικών συστημάτων. Ο λόγος που επιλέξαμε το Vbox έγκειται στο ότι δεν έχουμε πολλαπλούς πραγματικούς υπολογιστές, έτσι η μόνη λύση είναι να δημιουργήσουμε εικονικά συστήματα. Όλα τα συστήματα θα έχουν 512MB μνήμη , 8MB κάρτα γραφικών , 1Mbit κάρτα δικτύου server edition της Intel , σαν CPU θα χρησιμοποιούν το ένα από τα δύο core που διαθέτει το σύστημα όπου τα φιλοξενεί με χρονισμό 3,2MHz. Το μέγεθος καθώς και ο αριθμός τον δίσκων αναλύονται στο κάθε κεφάλαιο ξεχωριστά.

Το Vbox τρέχει τόσο σε περιβάλλον 32 bit όσο και σε 64 bit, για χρήση διακομιστή ή πελάτη. Μπορεί να εγκατασταθεί σε Windows, Linux, Macintosh, BSD Solaris.

Τα εικονικά λειτουργικά συστήματα τα οποία μπορούμε να δημιουργήσουμε είναι αρκετά μερικά εκ' των οποίων είναι τα ακόλουθα, Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7), DOS/Windows 3.x, Linux (Arch, Debian, Fedora, Gentoo,Madriva, openSuse, Redhat, Turbolinux, Ubuntu, Xandros), Solaris,OpenSolaris, BSD (FreeBSD, NetBSD OpenBSD) IBM OS/2.

Σε κάθε σύστημα παρέχει υποστήριξη για τέσσερις κάρτες δικτύου και τέσσερις διαφορετικούς τύπους ελεγκτών (bus controllers) IDE,SATA,SCSI και Floppy.

Τα πιο δημοφιλή λειτουργικά συστήματα για διακομιστή είναι: Linux, FreeBSD,Windows.

Οι πιο δημοφιλείς τρόποι για HA είναι HAST (FreeBSD), DRDB (Linux), FailOver μέσω RAID (Redundant Array of Independent Disks) και NAS (Network-attached storage) (Windows Server).

Λόγω του ότι στα Windows χρειαζόμαστε RAID NAS και κάτι τέτοιο δεν διαθέτουμε, το Vbox σε συνδυασμό με το FreeNAS θα μας λύσουν το πρόβλημα.

Το NAS [3] είναι ένας υπολογιστής συνδεδεμένος σε ένα δίκτυο όπου παρέχει υπηρεσίες αποθήκευσης δεδομένων σε άλλες συσκευές του δικτύου. Αν και μπορεί τεχνικά να τρέχει και άλλες υπηρεσίες, δεν έχει σχεδιαστεί να τρέχει ως διακομιστής γενικής χρήσης. Οι μονάδες NAS συνήθως δεν διαθέτουν πληκτρολόγιο ή οθόνη, και ελέγχονται - ρυθμίζονται μέσω δικτύου, χρησιμοποιώντας ένα πρόγραμμα περιήγησης.

Το RAID [4] σημαίνει Πλεονάζουσα Παράταξη Ανεξάρτητων Δίσκων και αναφέρεται στον τρόπο οργάνωσης και σύνδεσης μιας ομάδας δίσκων με σκοπό την αύξηση της ταχύτητας προσπέλασης και εγγραφής αλλά και την προστασία των δεδομένων ενός συστήματος. Υπάρχουν δύο τρόποι κατασκευής συστοιχιών RAID. Με την χρήση υλικού ή λογισμικού. Η χρήση υλικού είναι πολύ πιο ακριβή, αλλά έχει το πλεονέκτημα της σταθερότητας και της προσθαφαίρεσης των δίσκων την ώρα λειτουργίας του συστήματος. Όπως γίνεται φανερό η hardware λύση δεν απευθύνεται στον απλό χρήστη. Αντίθετα η χρήση λογισμικού δίνει στον χρήστη το δικαίωμα να κατασκευάσει μια συστοιχία με IDE & SCSI δίσκους, κάτι που δεν γίνεται μέσω hardware.

Το FreeNAS δημιουργεί εικονικά RAID και NAS συστήματα οπότε θα το χρησιμοποιήσουμε για τις ανάγκες μας.

Οι παραπάνω τεχνολογίες υλοποιούν την ανταλλαγή δεδομένον σε δύο διακομιστές ώστε να διατηρούν τις ίδιες πληροφορίες. Για τον έλεγχο αυτών καθώς και την υλοποίηση του Failover θα χρησιμοποιήσουμε τα πιο δημοφιλή εργαλεία ανάλογα για την κάθε υλοποίηση, όπου αυτά είναι για το μεν DRBD το Heartbeat, HAST το UCARP και για το RAID-NAS το Cluster Failover.

Όλες οι τεχνολογίες βασίζονται στο μοντέλο πρωτεύων (master) / δευτερεύων (slave).

Δηλαδή, όταν ένας διακομιστής επιτελεί το ρόλο του master, εκτελεί όλες τις απαραίτητες εργασίες προκειμένου να ενημερώνει συνεχώς ένα slave αλλά και να παρέχει όλες τις εργασίες στους πελάτες (clients) με τις οποίες έχει επιφορτισθεί.

Το HA όπως αναφέραμε συντελεί στην υψηλή διαθεσιμότητα διακομιστών. Το συναντάμε κυρίως σε δυναμικά πληροφοριακά συστήματα. Τα συστήματα αυτά χρησιμοποιούν βάσεις δεδομένων όπως mysql [5], SQL, oracle, DB2, PostgreSQL κτλ και δυναμικές γλώσσες προγραμματισμού php [6],asp,jsp κτλ.

Οι δύο πρώτοι διακομιστές βάσεων είναι οι πιο δημοφιλείς. Για τον λόγο του ότι ο SQL διακομιστή είναι της Microsoft και δεν τρέχει σε Unix συστήματα θα χρησιμοποιήσουμε τον Mysql. Ακόμη σαν γλώσσα προγραμματισμού θα χρησιμοποιήσουμε php όπου είναι η πιο διαδεδομένη για δυναμικά site.

Τέλος ως εξυπηρετητή παγκόσμιου Ιστού θα χρησιμοποιήσουμε τον Apache [7] καθώς όπως φαίνεται και παρακάτω από τα στοιχεία είναι ο πιο διαδεδομένος.

Πίνακας 1: «Στοιχεία Χρήσης WEB Διακομιστών 2010» ([8])

Δημιουργός	**Προϊόν**	**Αριθμός sites**	**Ποσοστό**
Apache	Apache	179,720,332	60.31%
Microsoft	IIS	57,644,692	19.34%
Igor	Sysoev	22,806,060	7.65%
Google	GWS	15,161,530	5.09%
lighttpd	lighttpd	1,796,471	0.60%

Τα συστήματα που θα χρησιμοποιήσουμε είναι τα Ubuntu 8.04, freeBSD 8.2 και Windows Server 2008 και FreeNAS 7.3 όπου οι λεπτομέρειες κάθε συστήματος θα αναλυθούν στα κεφάλαια παρακάτω.

ΚΕΦΑΛΑΙΟ 1 - DRBD και HeartBeat

Στο κεφάλαιο αυτό θα αναλύσουμε μία από τις πιο διαδεδομένες τεχνολογίες για διακομιστές υψηλής διαθεσιμότητας, το Heartbeat το οποίο προσφέρει σε περίπτωση αποτυχίας διεκπεραίωσης κάποιων εργασιών από τον πρωτεύον διακομιστή, λόγο βλάβης υλικού ή προβλήματος στο λογισμικό, την δυνατότητα διεκπεραίωσης αυτών από τον δευτερεύων διακομιστή χωρίς ο χρήστης να καταλάβει την διάφορα (high availability).

Ακόμη θα αναλύσουμε το DRBD το οποίο αναλαμβάνει να ενημερώνει για τις αλλαγές που γίνονται στον δίσκο που είναι ενεργός επάνω στον πρωτεύον διακομιστή, στον δίσκο του δευτερεύων διακομιστή (data replication).Ακόμη θα δούμε πώς υλοποιείται πρακτικά ένας High Availability Web server με Ubuntu 8.04 με mysql server 5.0.67,php5 και apache2 υλοποιημένα σε Vbox. Τέλος θα δούμε την απόδοση του συστήματος καθώς και τους χρόνους προσπέλασης στα δεδομένα όταν και τα δύο συστήματα είναι ενεργά, όταν είναι μόνο ο δευτερεύων ενεργός και γενικά πώς συμπεριφέρονται κάτω από μη κανονικές συνθήκες.

1.1 DRBD

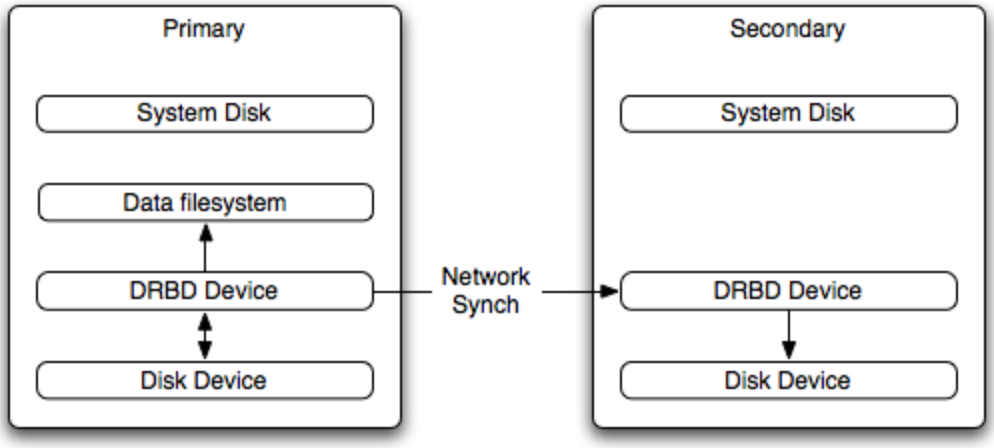
Το DRBD [9] (Distributed Replicated Block Device) (ως Block Device στο LINUX αναφέρεται σε τύπους συσκευής που σχετίζονται με αποθηκευτικό μέσο) δημιουργήθηκε από την Linbit και συνοδεύεται με GPL License και είναι διαθέσιμο μόνο σε Linux λειτουργικά καθώς είναι υλοποιημένο ως module στον πυρήνα του linux. Το DRBD δημιουργεί ένα εικονικό block συσκευής (DRBD Device) το οποίο

είναι συσχετισμένο με ένα φυσικό αλλά μη άμεσα εμφανές Block Device, το οποίο μπορεί να αναπαραχθεί από τον πρωτεύον διακομιστή στον δευτερεύων. Έτσι όταν δημιουργήσουμε ένα σύστημα αρχείων στην εικονική συσκευή τότε αυτή η πληροφορία αναπαράγεται στο Block Level του δευτερεύων διακομιστή.

Λόγω του ότι αναπαράγεται το Block Device και όχι μόνο τα δεδομένα, οι πληροφορίες που εμπεριέχονται είναι πιο αξιόπιστες, παρά με τις υλοποιήσεις που θέλουν να αναπαράγουν μόνο τα δεδομένα. Ακόμη το DRBD μπορεί να εξασφαλίσει την ακεραιότητα των δεδομένων μόνο όταν μια εγγραφή γίνει επιτυχώς στην φυσική συσκευή του πρωτεύων και του δευτερεύων διακομιστή.

Κάθε υπηρεσία του DRBD γράφει τις πληροφορίες από τον εικονικό δίσκο του DRBD στο φυσικό. Για παράδειγμα στον πρωτεύων διακομιστή όταν γίνεται μια εγγραφή εκτελείται στο φυσικό δίσκο, άλλα ταυτόχρονα μεταφέρεται στον εικονικό του DRBD στο δευτερεύων διακομιστή, στην συνέχεια στο δευτερεύων εκτελείται μια εγγραφή από τον εικονικό στον φυσικό. Στον πρωτεύων και δευτερεύων οι αναγνώσεις γίνονται απευθείας από τον φυσικό δίσκο. Οι πληροφορίες είναι κοινές και στους δύο DRBD διακομιστές καθώς είναι συγχρονισμένοι σε Block Level. Έτσι λοιπόν μπορούμε να το χρησιμοποιήσουμε σε συνδυασμό με την Mysql ώστε να παρέχουμε αδιάλειπτη προσπέλασή στα δεδομένα μια βάσης.

Στο σχήμα βλέπουμε σχηματικά την ροή τις πληροφορίας στον πρωτεύων και δευτερεύων διακομιστή. Όπως μπορούμε να διακρίνουμε οι έγγραφες περνάνε μέσω του DRBD από το δίκτυο στον δευτερεύων διακομιστή.



Εικόνα 1: «Ροή δεδομένων DRBD»

Το DRBD γενικά χρησιμοποιείται σε Single-Primary λειτουργία (με αυτήν θα ασχοληθούμε) κατά την οποία υπάρχει ένας και μόνο πρωτεύων διακομιστής και ένας τουλάχιστον οι περισσότεροι δευτερεύοντες.Σε αυτή την λειτουργία ο δίσκος που βλέπουμε είναι αυτός του διακομιστή που είναι πρωτεύων. Σαν αρχείο συστήματος μπορεί να χρησιμοποιηθεί οποιοδήποτε υποστηρίζεται από τον πυρήνα του LINUX πχ. Ext3,Xfs κτλ.

Ακόμη μπορεί να τρέχει σε λειτουργία Dual-Primary Mode το οποίο σημαίνει πως έχουμε δύο διακομιστές σε πρωτεύων κατάσταση. Σαν αποτέλεσμα έχουμε να βλέπουμε τον ίδιο δίσκο μέσα από δύο διαδρομές δηλαδή μέσα και από τους δύο κόμβους (Nodes). Σαν αρχείο συστήματος θα πρέπει να χρησιμοποιηθεί κάποιο από τα οποία παρέχει διαιτησία πχ Gfs, OCFS2. Δεν θα ασχοληθούμε με αυτή την λειτουργία καθώς δεν θέλουμε τα δεδομένα να είναι προσπελάσιμα ταυτόχρονα από δύο πηγές αλλά από μία.

Λειτουργίες αναπαραγωγής (Replication modes)

Πρωτόκολλο Α. Είναι το πρωτόκολλο για ασύγχρονη αναπαραγωγή. Εδώ κάθε εγγραφή στον πρωτεύων διακομιστή χαρακτηρίζετε ως επιτυχής εφόσον έχει ολοκληρωθεί η εγγραφή στον τοπικό δίσκο και προετοιμαστεί η πληροφορία για αποστολή μέσω TCP πακέτου. Σε περίπτωση αποτυχίας υπάρχει μεγάλος κίνδυνος κάποιο μέρος πληροφορίας να χαθεί.

Πρωτόκολλο Β. (Συγχρονισμού μνήμης). Εδώ επιτυχημένη ορίζετε μια εγγραφή όταν εκτελεστεί στον πρωτεύων διακομιστή και ολοκληρωθεί στον τοπικό δίσκο και ταυτόχρονα ληφθεί το πακέτο από τον δευτερεύων. Μικρότερη πιθανότητα να χαθεί πληροφορία.

Πρωτόκολλο Γ. Πρωτόκολλο Σύγχρονης αναπαραγωγής. Εδώ και στον πρωτεύων και δευτερεύων διακομιστή θα πρέπει να γίνει επιτυχής εγγραφή στους φυσικούς δίσκους, ώστε να μαρκαριστεί η καταχώριση ως έγκυρη. Δεν υπάρχει περίπτωση να χαθεί πληροφορία.

Γενικά χρησιμοποιείτε το πρωτόκολλο Γ για λόγους διασφάλισης της πληροφορίας,έτσι και εμείς θα ασχοληθούμε με αυτό.

Τα εργαλεία τα οποία προσφέρει είναι:

- drbdadm

Το εργαλείο αυτό υψηλού επιπέδου χειρίζεται το DRBD περνώντας όλες τις παραμέτρους που έχουμε ορίσει στο αρχείο /etc/drbd.conf ώστε να δουλέψει.

- drbdsetup

Το εργαλείο αυτό είναι χαμηλού επιπέδου και κάθε παράμετρος πρέπει να περαστεί με το χέρι σαν όρισμα. Το εργαλείο αυτό δίνει περισσότερη ευελιξία.

- drbdmeta

Όπως μας προϊδεάζει και το όνομα του, χρησιμοποιείται για να χειριζόμαστε τα “δεδομένα για τα δεδομένα” (meta data) του DRBD.

Θα δούμε παρακάτω, κατά την υλοποίηση, την χρήση των δύο πρώτων εργαλείων.

1.2 Heartbeat

Το Heartbeat [10] είναι δημιούργημα του Linux-HA project το οποίο προσφέρει υπηρεσίες υψηλής διαθεσιμότητας. Το Heartbeat έχει υιοθετηθεί από αρκετές Linux διανομές και συνοδεύεται με GPL License.Φυσικά μπορεί να χρησιμοποιηθεί για βάσεις δεδομένων αλλά χρήση βρίσκει και σε Mail Server, Web Servers, File Servers και DNS Servers.

Το Heartbeat υλοποιεί το heartbeat-protocol και είναι υλοποιημένο σαν module στον πυρήνα. Το heartbeat-protocol σημαίνει πώς τα μηνύματα στέλνονται σε τακτά χρονικά διαστήματα μεταξύ των διακομιστών εξού και το όνομά του heartbeat “χτύπος καρδιάς”. Έτσι λοιπόν εάν το μήνυμα δεν παραληφθεί από κάποιον διακομιστή σε ένα ορισμένο χρονικό διάστημα τότε αυτός υποθέτει πως υπάρχει κάποια αστοχία και έτσι θα εκτελέσει κάποια ενέργεια προκειμένου να επιτευχθεί η αδιάλειπτη προσπέλαση των υπηρεσιών που έχουμε ορίσει να τρέχουν στον πρωτεύων.

Το Heartbeat Υποστηρίζει τους ακόλουθους τύπους δικτύου, unicast UDP over Ipv4, broadcast UDP over Ipv4, multicast UDP over Ipv4 και serial link communications.

Γενικά το Heartbeat αναλαμβάνει να “καταλάβει” την αποτυχία που μπορεί να συντελεστεί στον πρωτεύων διακομιστή και να εκτελέσει όλες τις απαραίτητες

ενέργειες ώστε η απώλεια του πρωτεύων να μην δημιουργήσει κανένα πρόβλημα στην εκτέλεση οποιουδήποτε ρόλου έχει ανατεθεί σε αυτά τα δύο μηχανήματα.

1.3 Υλοποίηση

Στο δικό μας σενάριο έχουμε ένα http Server (apache) όπου είναι ενεργός στον πρωτεύων διακομιστή ο οποίος υλοποιεί ένα διαδραστικό site. Άρα λοιπόν έχουμε ένα sql server (mysql) ο οποίος είναι ενεργός και αυτός στον πρωτεύων διακομιστή. Όταν λοιπόν για τον οποιοδήποτε λόγο ο πρωτεύων δεν είναι προσπελάσιμος ο δευτερεύων θα αναλάβει να παρέχει τις παραπάνω υπηρεσίες.

Ας ξεκινήσουμε να δημιουργούμε τα δύο πειραματικά συστήματα.

Δημιουργούμε δύο εικονικούς δίσκους μέσω του Vbox με χωρητικότητα 5GB ο καθένας όπου θα εγκαταστήσουμε τα δυο ubuntu λειτουργικά. Ακόμη θα χρειαστούμε δύο δίσκους για το DRBD, εδώ μπορούμε να βάλουμε οποιοδήποτε μέγεθος μεγαλύτερο του 256 καθώς θα χωριστεί σε δύο διαμερίσματα (partitions) έκ' των οποίων το ένα χρησιμοποιείται για τα metadata από το DRBD και το μικρότερο μέγεθος πρέπει να είναι 128MB, εμείς θα ορίσουμε 2GB καθώς είναι αρκετό για τα δεδομένα που θα αποθηκεύσουμε για το διαδραστικό site.

Εφόσον έχουμε δημιουργήσει τους δίσκους δημιουργούμε δύο καινούρια συστήματα μέσα από το Vbox και ορίζουμε στο ένα σύστημα τον ένα δίσκο 5GB δίσκο και τον άλλο με τα 2GB.Προσαρτούμε στο εικονικό cd-rom την εικόνα (image) της εγκατάστασης (setup) του λειτουργικού και ξεκινάμε το σύστημα. Εκτελούμε την ίδια διαδικασία αντίστοιχα για το δεύτερο σύστημα. Από δω και πέρα θα αναφερόμαστε στο πρώτο σύστημα όπου θα είναι και το πρωτεύων ως drbda και στο δευτερεύων ως drbdb.

Κατά την εγκατάσταση του συστήματος ο μόνος δίσκος που θα πρέπει να μορφοποιηθεί είναι αυτός με τα 5GB όπου θα τον χωρίσουμε σε δύο partitions, ένα μέρος του θα διαμορφωθεί με Ext3 για τα δεδομένα του συστήματος και ένα άλλο μέρος ως swap για την εικονική μνήμη. Όταν ρωτηθούμε για τις παραμέτρους του δικτύου, αναλόγως των ρυθμίσεων στο μηχάνημα που φιλοξενεί τα δυο εικονικά, δίνουμε αντίστοιχα ip, μάσκα υποδικτύου (subnet mask), προεπιλεγμένη πύλη (default gateway). Στην προκειμένη περίπτωση σαν ip έχουμε 192.168.1.60 στο drbda και 192.168.1.61 drbdb αντίστοιχα και από κοινού subnet 24 και gateway 192.168.1.1

Πριν την πρώτη εκκίνηση των συστημάτων, μετά από την εγκατάσταση, θα πρέπει να αφαιρέσουμε το εικονικό cd-rom και να αλλάξουμε τις κάρτες δικτύου ως 1Gbit eth intel, για καλύτερη απόδοση του δικτύου καθώς και το mode σε bridged adapter, να ορίσουμε ένα από τα διαθέσιμα φυσικά interface από το λειτουργικό σύστημα το οποίο θα φιλοξενήσει τα δύο εικονικά, το οποίο να βγαίνει στο διαδίκτυο. Ακόμη θα πρέπει να ορίσουμε το μέγεθος τις μνήμης που θα έχει το κάθε σύστημα, στην προκειμένη περίπτωση ορίζουμε 512MB καθώς πρέπει να εξασφαλίσουμε την σωστή λειτουργία του συστήματος που φιλοξενεί τα άλλα δύο.Το ποσοστό που μπορούμε να δώσουμε στα εικονικά συστήματα έχει να κάνει με το πόσο διαθέτει το σύστημα που τα φιλοξενεί.

Εφόσον μας καλωσορίσουν τα λειτουργικά ανοίγουμε ένα τερματικό και εκτελούμε ifconfig ώστε να δούμε πως όλα είναι περασμένα σωστά στις κάρτες αλλά και για να σημειώσουμε το όνομα του interface όπου θα το χρειαστούμε αργότερα. Ακόμη καλό θα είναι να εκτελέσουμε ένα ping σε google.com ώστε να δούμε εάν βγαίνουν στον έξω κόσμο. Θα πρέπει να ορίσουμε στον κάθε διακομιστή το όνομα του ενός και του άλλου με την αντίστοιχη ip ώστε να μην έχουν προβλήματα στην ανάκτηση ονομάτων – διευθύνσεων. Για να το πετύχουμε αυτό επεξεργαζόμαστε (edit) με δικαιώματα υπερχρήστη (root) το αρχείο /etc/hosts και στα δύο μηχανήματα όπου προσθέτουμε τις γραμμές 192.168.1.60/24 drbda και 192.168.1.61/24 drbdb.

Ένα βήμα ακόμη πριν ξεκινήσουμε στην ρύθμιση του DRBD και HeartBeat είναι να χωρίσουμε τον 2GB δίσκο σε δύο partitions όπως αναφέραμε στην αρχή. Ανοίγουμε xterm και εκτελούμε την εντολή sudo (do as Superuser) gparted.Μόλις ανοίξει το πρόγραμμα διαχείρισης δίσκων του gnome επιλέγουμε τον δίσκο και απλά δημιουργούμε δύο διαμερίσματα. Φυσικά δεν προχωράμε σε διαμόρφωση και απλά προσέχουμε το ένα διαμέρισμα να είναι μεγαλύτερο του 128MB.

Εφόσον όλα πάνε καλά μπορούμε να ξεκινήσουμε εγκαθιστώντας τα απαραίτητα εργαλεία. Ανοίγουμε ένα τερματικό και εκτελούμε και στα δύο μηχανήματα sudo apt-get update, ώστε να λάβουμε τις τελευταίες ενημερώσεις. Έπειτα εκτελούμε sudo apt-get install ntp ntpdate ώστε να έχουμε σωστά συγχρονισμένα τα ρολόγια των διακομιστών μας.

Στην συνέχεια για να εγκαταστήσουμε το DRBD και HeartBeat εκτελούμε και στα δύο μηχανήματα sudo apt-get install linux-headers-`uname -r` drbd8-utils drbd8-source heartbeat.

Εφόσον ολοκληρωθεί η εγκατάσταση εκτελούμε και στα δύο μηχανήματα sudo m-a a-i drbd8-source για να δημιουργήσουμε το module για τον πυρήνα.Εφόσον ολοκληρωθεί η μεταγλώττιση (compile) εκτελούμε sudo modprobe drbd για να φορτώσουμε το module στον πυρήνα.

Έχοντας εγκαταστήσει με επιτυχία τα δύο modules μπορούμε να ξεκινήσουμε με την παραμετροποίηση των αρχείων ρυθμίσεων του DRBD και HeartBeat. Αρχικά θα πρέπει να επεξεργαστούμε το αρχείο /etc/drbd.conf

Στον drbda και drbdb το αρχείο αυτό θα είναι όπως φαίνεται παρακάτω.

```
Global {        # Εδώ δηλώνουμε πως ότι εμπεριέχεται μέσα στις
#αγκύλες είναι για global παραμέτρους. Μπορούμε να
#χρησιμοποιήσουμε μια από τις ακόλουθες minor-count, dialog-
#refresh, disable-ip-verification και usage-count

usage-count yes;    # Εδώ απλά ορίζουμε πώς θα δηλώνουμε την
#χρήση του DRBD στο DRBD's online usage counter το οποίο
#συλλέγει πληροφορίες για την χρήση του DRBD.

}

common {        # Εδώ μέσα δηλώνουμε παραμέτρους οι οποίοι
#κληρονομούνται σε όλους τους διακομιστές. Οι αποδεκτές
#παράμετροι είναι οι startup,  syncer,  handlers, net και
#disk.

  Syncer {

     rate 10M; }    # Εδώ ορίζουμε πώς ο ρυθμός συγχρονισμού
#θα είναι στα 10MB/s. Αυτό μπορούμε να το αυξήσουμε
#αναλόγως με το δίκτυό μας, π.χ σε Gigabit  μπορούμε να το
#πάμε άφοβα σε τιμές γύρω στο 120MB/s.Εμείς θα βάλουμε 10
#για τον λόγο του ότι το σύστημά μας είναι εικονικό.

}

resource r0 {  # Εδώ ορίζουμε το όνομα του resource

  protocol C;  # Ορίζουμε το πρωτόκολλο όπως  περιγράψαμε
         #παραπάνω. Θα χρησιμοποιήσουμε το Πρωτόκολλο Γ

  handlers {   # Ορίζουμε τους χειριστές γεγονότων (event
               # handlers)

     pri-on-incon-degr "echo o > /proc/sysrq-trigger ; halt
-f";
```

```
     pri-lost-after-sb "echo o > /proc/sysrq-trigger ; halt
-f";

    local-io-error "echo o > /proc/sysrq-trigger ; halt -f";

  }

  startup {         # Εδώ ορίζουμε το τι θέλουμε να περνάμε
#σαν παράμετρο κατά την εκκίνηση

    degr-wfc-timeout 120;    # ορίζουμε τον χρόνο, τον
#οποίο δεν θα εκτελέσει σύνδεση μετά από ένα αιφνίδιο θάνατο
#καθώς ένα μηχάνημα όταν έχει κάνει αιφνίδια επανεκκίνηση
#υπάρχει μεγάλη πιθανότητα να ξανασυμβεί κάτι παρόμοιο. Άρα
#βάζουμε ένα εύλογο χρόνο σε  δευτερόλεπτα.

  }

  disk {

    on-io-error   detach;    #εδώ ορίζουμε το τι θα συμβεί
#όταν υπάρξει πρόβλημα στον δίσκο. Με το detach ορίζουμε πως
#τα δεδομένα δεν θα γραφτούν στο lower level.

  }

  net {             # Περνάμε παραμέτρους που έχουν να κάνουν
                    # με το δίκτυο

          # Μπορούμε να βάλουμε τους ακόλουθους sndbuf-size,
#rcvbuf-size,timeout, connect-int, ping-int, ping-timeout,
#max-buffers,max-epoch-size, ko-count, allow-two-primaries,
#cram-hmac-alg, shared-secret, after-sb-0pri, after-sb-1pri,
#after-sb-2pri,    data-integrity-alg,    no-tcp-cork,    on-
#congestion,congestion-fill, congestion-extents

  }

  syncer {          # Ορίζουμε παραμέτρους για τον δαίμονα
                    # συγχρονισμού

    rate 10M;       # Ρυθμός συγχρονισμού.
```

```
    Al-extents 257; # Μετά από μία αποτυχία ο διακομιστής θα
#πρέπει να ανακτήσει τα χαμένα δεδομένα που τυχόν έχουν
#περαστεί όσο αυτός ήταν πεσμένος. Γενικά όσο μεγαλύτερη
#τιμή δώσουμε εδώ λιγότερος

#χρόνος χρειάζεται για να συγχρονιστούν οι διακομιστές αλλά
#περισσότερες εγγραφές στα Meta Data.Επιτρεπτές τιμές είναι
#από

# 7 έως 3843. 257 για το σύστημά μας είναι αποδεκτά καθώς
#δεν έχουμε μεγάλο χώρο για τα Meta Data.

  }

on drbda {          # Ορίζουμε παραμέτρους για τον πρώτο
                    # διακομιστή

    device /dev/drbd0;   # διαδρομή εικονικής συσκευής

    disk /dev/sdb1;           # διαδρομή φυσικού δίσκου

    address 192.168.1.60:7788;     # Διεύθυνση ip και πόρτα

    meta-disk /dev/sdb2[0];        # διαδρομή φυσικού δίσκου
                                   # για τα Meta Data

  }

on drbdb {                  #Ορίζουμε παραμέτρους για τον
                                #δεύτερο διακομιστή

    device /dev/drbd0;            #    διαδρομή    Εικονικής
συσκευή

    disk /dev/sdb1;               # διαδρομή φυσικού δίσκου

    address 192.168.1.61:7788;     # Διεύθυνση ip και πόρτα

    meta-disk /dev/sdb2[0];        # διαδρομή φυσικού δίσκου
για τα Meta Data

  }

}
```

Ολοκληρώνοντας το αρχείο παραπάνω προχωράμε στο να επεξεργαστούμε το /etc/ha.d/ha.cf.

Το αρχείο αυτό ορίζει παραμέτρους για το HeartBeat και είναι το ίδιο και στους δύο διακομιστές.

Θα μοιάζει όπως φαίνεται παρακάτω.

```
logfacility local0 # Ορίζουμε ποιόν δαίμονα θα #χρησιμοποιήσουμε (logging) για διαχείριση των logs να #χρησιμοποιεί. Εμείς ορίσαμε του συστήματος τον syslog

keepalive 1 # Πόσο συχνά σε δευτερόλεπτα θα #αποστέλλετε το heartbeat σήμα

deadtime 10 # Μετά από πόσα δευτερόλεπτα θεωρούν πως #ένας διακομιστείς δεν μπορεί να επικοινωνήσει.

bcast eth0 # Ο τρόπος που θα στέλνονται τα #heartbeat signals και από ποιο interface

auto_failback on # Όταν αρχικός πρωτεύων διακομιστής #επανέλθει εάν θα αναλαμβάνει τον ίδιο ρόλο, αποδεκτές #τιμές on/off

node drbda drbdb # Ορίζουμε τους διακομιστές
```

Έπειτα θα πρέπει να παραμετροποιήσουμε το αρχείο /etc/heartbeat/authkeys στο οποίο θα επιτρέψει στους διακομιστές να “εμπιστεύονται” ο ένας τον άλλο. Το αρχείο θα είναι όπως φαίνεται παρακάτω και στους δύο διακομιστές

```
auth 3 # τύπος ταυτοποίησης

3 md5 drbda # υπογραφή και συνθηματικό
```

Προσέχουμε το αρχείο να έχει δικαιώματα “ -rw------- “ καθώς μόνο ο root πρέπει να έχει πρόσβαση σε αυτό.

Για να το πετύχουμε τρέχουμε sudo chmod 600 /etc/heartbeat/authkeys.

Ήρθε η ώρα να παραμετροποιήσουμε το αρχείο /etc/ha.d/haresources το οποίο διαχειρίζεται όλες τις υπηρεσίες και τις παραμέτρους που θα εκτελεί ο κάθε διακομιστής όταν βρίσκετε σε primary mode και σε secondary mode, έτσι το αρχείο θα είναι ίδιο και στους δύο διακομιστές. Το αρχείο θα εμπεριέχει τα παρακάτω.

drbda IPaddr::192.168.1.100/24/eth0 drbddisk::r0 Filesystem::/dev/drbd0::/data::ext3

apache2 mysql

Οι παραπάνω γραμμές ορίζουν ότι η κοινή ip θα είναι η 192.168.1.100 με subnet 24 και θα δημιουργείται στο interface eth0.Το drbddisk::r0 ορίζει ότι το resource με όνομα r0 θα ξεκινήσει προκειμένου να γυρίσει ένα node από secondary σε primary mode.

Filesystem::/dev/drbd0::/data::ext3 σε πιο σημείο θα γίνει η προσάρτηση του εικονικού δίσκου και πιο σύστημα αρχείων χρησιμοποιεί. Ακολουθούν οι διεργασίες που θέλουμε να ανοίγουν όταν τρέχει σε Primary mode. Προς το παρόν τον apache και mysql δεν θέλουμε να προσπαθήσει να τους σηκώσει καθώς δεν θα μπορέσει και δεν θα προχωρήσει, άρα τα αφήνουμε προς το παρόν σαν σχόλια.Ακόμη εάν βάλουμε όπως είναι το αρχείο στον drbda και στον drbdb το αποτέλεσμα θα είναι πάντα να είναι πρωτεύον ο drbda εφόσον είναι προσπελάσιμος. Εάν θέλουμε να παραμένει πρωτεύων αυτός που ήταν τελευταίος προσπελάσιμος τότε στον drbdb θα πρέπει απλά να αλλάξουμε το drbda σε drbdb.

Στο σημείο αυτό έχουμε τελειώσει με τις ρυθμίσεις και δεν μένει παρά να δώσουμε την πρώτη εκκίνηση στις υπηρεσίες.

Εκτελούμε διαδοχικά στον drbda και στον drbdb τις επόμενες εντολές προκειμένου να δημιουργήσουμε τον δίσκο για τα μετά δεδομένα, sudo drbdadm create-md r0 και για να δημιουργήσει το interface με όνομα r0. Έπειτα drbdadm up all για να σηκώσει όλα τα interface.

Έπειτα λόγω του ότι ο drbda έχουμε αποφασίσει πως θα είναι ο πρωτεύων δίνουμε τις εντολές σε αυτόν

drbda$ sudo drbdsetup /dev/drbd0 primary -o ορίζει ως πρωτεύων το σύστημα

drbda$ sudo mkfs.ext3 /dev/drbd0 δημιουργεί το σύστημα αρχείων

drbda$ sudo mkdir /data δημιουργεί τον φάκελο που θα προσαρτήσουμε τον δίσκο

drbda$ sudo mount -t ext3 /dev/drbd0 /data προσαρτούμε τον δίσκο

drbda$ umount /data

στον drbdb$ sudo mkdir /data

Έπειτα τρέχουμε και στα δύο node για να ξεκινήσουμε τις υπηρεσίες

sudo /etc/init.d/drbd start

sudo /etc/init.d/heartbeat start

Μπορούμε να δούμε την κατάσταση του DRBD τρέχοντας

drbda# cat /proc/drbd

Και θα δούμε κάτι όπως το παρακάτω εφόσον είναι συγχρονισμένοι οι διακομιστές

```
cat /proc/drbd
version: 8.0.4 (api:86/proto:86)
SVN Revision: 2947 build by root@drbda, 2011-03-20 17:43:05
 0: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
    ns:2175704 nr:0 dw:99192 dr:2076641 al:33 bm:128 lo:0 pe:0 ua:0 ap:0
        resync: used:0/31 hits:134841 misses:135 starving:0 dirty:0 changed:135
        act_log: used:0/257 hits:24765 misses:33 starving:0 dirty:0 changed:33
```

Σε αντίθετη περίπτωση θα δούμε

```
cat /proc/drbd
version: 8.0.4 (api:86/proto:86)
SVN Revision: 2947 build by root@drbda, 2011-03-20 16:43:05
 0: cs:SyncSource st:Primary/Secondary ds:UpToDate/Inconsistent C r---
    ns:252284 nr:0 dw:0 dr:257280 al:0 bm:15 lo:0 pe:7 ua:157 ap:0
```

[==>.....] sync'ed: 12.3% (1845088/2097152)K

finish: 0:06:06 speed: 4,972 (4,580) K/sec

resync: used:1/31 hits:15901 misses:16 starving:0 dirty:0 changed:16

act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0

Ακόμη μπορούμε να δούμε τα μηνύματα του HeartBeat εκτελώντας tail -f /var/log/messages

Μία μη επιθυμητή κατάσταση είναι η λεγόμενη “split brain”. Το DRBD μας ενημερώνει με το ακόλουθο μήνυμα “Split-Brain detected, dropping connection”

Αυτό μπορεί να συμβεί όταν και τα δύο nodes είναι σε κατάσταση primary – primary. Τότε το DRBD κλείνει την σύνδεση και περιμένει να αποφασίσουμε απο πιο node θα πετάξουμε στα σκουπίδια τα δεδομένα που έχει και να το συγχρονίσουμε σύμφωνα με το άλλο.

Τα βήματα που πρέπει να ακολουθήσουμε είναι τα παρακάτω.

Sudo drbdadm secondary resource

Sudo drbdadm -- --discard-my-data connect resource

Sudo drbdadm connect resource

Όπου resource δίνουμε το όνομα που έχουμε δηλώσει στο drbd.conf

Πλέον είμαστε έτοιμοι να εγκαταστήσουμε και να ρυθμίσουμε τον apache, mysql,php, ώστε να έχουμε ένα διαδραστικό site το οποίο θα έχει ανοχή σε τυχών προβλήματα.

Εκτελούμε διαδοχικά τις παρακάτω εντολές σε ένα τερματικό και στα δύο nodes

Για να εγκαταστήσουμε τον apache

sudo apt-get install apache2

Για να εγκαταστήσουμε την php καθώς και υποστήριξη για να δουλέψει μαζί με τον apache

sudo apt-get install php5 libapache2-mod-php5

Επανεκκίνηση του apache

sudo /etc/init.d/apache2 restart

Για να εγκαταστήσουμε την mysql

sudo apt-get install mysql-server

Αλλαγή του συνθηματικού (password) για τον διαχειριστή της βάσης

mysql -u root

mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('νέο password');

Εγκατάσταση κατάλληλων βιβλιοθηκών για να δουλέψει ο apache με την mysql καθώς και εγκατάσταση του phpmyadmin ο οποίος βοηθάει να διαχειριζόμαστε την mysql μέσω γραφικού περιβάλλοντος.

sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin

επανεκκίνηση του apache.

sudo /etc/init.d/apache2 restart

ο apache διαβάζει από την παρακάτω διαδρομή. Άρα εκεί θα βρούμε και το index.html

/var/www/

Για τις δικές μας ανάγκες παραμετροποιούμε το index.html

```
<html>

<body>


<form action="testphp.php" method="post">
```

```
Customer ID: <input type="text" name="Cid" />

<br>

Customer name: <input type="text" name="Cname" />

<br>

Telephne: <input type="text" name="Ctelephone" />

<br>

<input type="submit" />

<br>

</form>

<form action="get_data.php" >

<input type="submit" name="read" value="Show Data" />

</form>

</body>

</html>
```

Ακόμη δημιουργούμε τα αρχεία get_data.php το οποίο μας επιστρέφει τις εγγραφές και testphp.php το οποίο καταχωρεί δεδομένα στην βάση

get_data.php

```
---------

<body>

<?php

$con = mysql_connect("localhost","root","omg");

if (!$con)

     {
```

```
            die('Could not connect: '. mysql_error());
        }
        mysql_select_db("hast", $con);
        echo "connection done!!!";
        $result = mysql_query("SELECT * FROM customer");
        echo "<table border='1'>
        <tr>
        <th>customer ID</th>
        <th>customer name</th>
        <th>customer telephone</th>
        </tr>";

while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>". $row['c_id']. "</td>";
  echo "<td>". $row['c_name']. "</td>";
  echo "<td>". $row['c_telephone']. "</td>";
  echo "</tr>";
  }
echo "</table>";
        mysql_close($con);
?>
```

```
<br>
<form action="index.html">
<input type="submit" name="Back" value="Back" />


</body>
</html>
```

testphp.php

```
<?php
$con = mysql_connect("localhost","root","omg");
if (!$con)
     {
          die('Could not connect: '. mysql_error());
     }
     mysql_select_db("hast", $con);
     echo "connection done!!!";
 //Isert DATA
$sql="INSERT INTO customer (c_id, c_name, c_telephone)
VALUES
('$_POST[Cid]','$_POST[Cname]','$_POST[Ctelephone]')";
if (!mysql_query($sql,$con))
     {
          die('Error: '. mysql_error());
```

```
     }
     echo "1 record added";
//END insert data
/* Post the data
     $result = mysql_query("SELECT * FROM customer");
     echo "<table border='1'>
     <tr>
     <th>customer ID</th>
     <th>customer name</th>
     <th>customer telephone</th>
     </tr>";
while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>". $row['c_id']. "</td>";
  echo "<td>". $row['c_name']. "</td>";
  echo "<td>". $row['c_telephone']. "</td>";
  echo "</tr>";
  }
echo "</table>";
//END Data
*/
     mysql_close($con);
```

```
?>

<br>

<form action="index.html">

<input type="submit" name="Back" value="Back" />
```

Αλλαγή διαδρομής για το που θα αποθηκεύονται οι βάσεις.

Σταματάμε τον mysql server στον drbda και στον drbdb

sudo /etc/init.d/mysql stop

Αντιγραφή των υπάρχων βάσεων στην νέα τοποθεσία όπου θα είναι στον δίσκο όπου έχουμε για το failover. Προσοχή καθώς ο δίσκος είναι ορατός μόνο στον primary, άρα θα το εκτελέσουμε μόνο στον drbda. Έτσι αυτομάτως θα δημιουργηθεί και στον drbdb.

Αντιγραφή του mysql φακέλου στον δίσκο

sudo cp -R -p /var/lib/mysql /data

διαγραφή όλων των αχρήστων αρχείων

sudo rm /data/mysql/*

Αλλάζουμε τα δικαιώματα ώστε να μπορεί να γράψει και ο drbdb όταν αναλάβει σε περίπτωση όπου ο drbda είναι μη προσπελάσιμος

sudo chown -R mysql:mysql /data/mysql

sudo chown -R mysql:mysql /data/mysql/mysql

Για να αλλάξουμε την διαδρομή όπου διαβάζει και γράφει τα δεδομένα η mysql επεξεργαστούμε το αρχείο my.cnf όπως φαίνεται παρακάτω στον drbda και drbdb

sudo gedit /etc/mysql/my.cnf

Βρίσκουμε την γραμμή όπου γράφει datadir =/var/lib/mysql και αλλάζουμε την διαδρομή ώστε να δείχνει στην νέα τοποθεσία άρα η παραπάνω γραμμή γίνετε datadir =/data/mysql

Για να έχει πρόσβαση ο server της mysql στον φάκελο θα πρέπει να το ορίσουμε στον apparmor ο οποίος είναι υπεύθυνος να δίνει πρόσβαση σε προγράμματα να γράφουν σε συγκεκριμένες τοποθεσίες. Άρα θα επεξεργαστούμε το παρακάτω αρχείο στον drbda και στον drbdb.

sudo gedit /etc/apparmor.d/usr.sbin.mysqld

και αντικαθιστούμε τις παρακάτω γραμμές

/mnt/mysql/data/ r,

/mnt/mysql/data/** rwk,

με

/data/mysql/ r,

/data/mysql/** rwk,

Έπειτα εξαναγκάζουμε τον apparmor να διαβάσει τις αλλαγές

sudo /etc/init.d/apparmor reload

Τέλος επανεκκινούμε την mysql μόνο στον drbda

sudo /etc/init.d/mysql start

Για τις ανάγκες μας θα δημιουργήσουμε μια βάση όπου θα την ονομάσουμε hast με ένα πίνακα με όνομα customer ο οποίος θα έχει τρία πεδία το c_id, c_name και c_telephone.Για να το πετύχουμε αυτό εκτελούμε στον drbda

mysql -u root -p omg

mysql>CREATE DATABASE hast;

mysql>USE hast;

mysql>CREATE TABLE customer (c_id INT, C_name VARCHAR(30), C_telephone INT);

Έχοντας ολοκληρώσει της ρυθμίσεις ήρθε η ώρα να ελέγξουμε εάν όλα είναι σωστά ανοίγοντας έναν περιηγητή και δίνοντας την εικονική διεύθυνση. Άρα http://192.168.1.100

Θα πρέπει τώρα να μην εκκινούν κατά την εκκίνηση ο mysql και apache αφήνοντας το HeartBeat να αποφασίσει για το πότε θα εκτελεστούν. εκτελούμε τις παρακάτω εντολές και στους δύο διακομιστές.

sudo update-rc.d mysql remove

sudo update-rc.d apache2 remove

Τέλος μπορούμε πλέον να αλλάξουμε το αρχείο /etc/ha.d/haresources έτσι ώστε να μπορεί το HeartBeat να σηκώσει τον Apache και Mysql βγάζοντας το σύμβολο των σχολίων όπως φαίνεται παρακάτω.

drbda IPaddr::192.168.1.100/24/eth0 drbddisk::r0 Filesystem::/dev/drbd0::/data::ext3 apache2 mysql

Πλέον μπορούμε να ελέγξουμε εάν δουλεύει απλά κλίνοντας τον drbda.

1.4 Stress Test – Αποτελέσματα

Στο σημείο αυτό θα πρέπει να αναφερθούμε στο εργαλείο το οποίο θα μας βοηθήσει να εκτελέσουμε τους ελέγχους για την απόδοση των συστημάτων.

Το εργαλείο αυτό είναι το siege [11] το οποίο είναι διαθέσιμο για unix συστήματα

Ως σκοπό έχει να δώσει στους προγραμματιστές Web εφαρμογών ένα δείγμα για την απόδοση του κώδικά τους κάτω από πίεση ορίζοντας το χρόνο που θα εκτελεστεί η συγκεκριμένη δοκιμή αλλά και το αριθμό τον ταυτόχρονων προσπελάσεων.

Περιλαμβάνει τις επιδόσεις όσο αφορά το χρόνο που πέρασε, το σύνολο των δεδομένων που μεταφέρθηκαν στον server, τον χρόνο απόκρισης, τον αριθμό των συναλλαγών και τον αριθμό των φορών όπου μια εγγραφή ήταν θετική.

Παρακάτω αναλύουμε τα στοιχεία που μας δίνει το πρόγραμμα μετά την ολοκλήρωσή του.

Όπου -c ο αριθμός τον clients -t ο χρόνος εκτέλεσης και ακολουθεί ο στόχος.

Lifting the server siege...	Μας δείχνει εάν ολοκληρώθηκε όλη η δοκιμή (test)
Transactions:	Αριθμός συναλλαγών
Availability:	Σε ποσοστό η διαθεσιμότητα
Elapsed time:	Χρόνος που πέρασε μετά την εκκίνηση
Data transferred:	Πόση πληροφορία μεταδόθηκε
Response time:	Μέσος χρόνος απόκρισης
Transaction rate:	Μέσος ρυθμός συναλλαγών ανά δευτερόλεπτο
Throughput:	Μέσος ρυθμός μεταδιδόμενης πληροφορίας ανά δευτερόλεπτο
Concurrency:	Ο μέσος αριθμός των ταυτόχρονων συνδέσεων, ο αριθμός όπου όσο αυξάνεται δηλώνει πως η επίδοση του διακομιστή μειώνεται.
Successful transactions:	Επιτυχείς μεταδόσεις
Failed transactions:	Αποτυχημένες μεταδόσεις
Longest transaction:	Μεγαλύτερος χρόνος που διήρκεσε μία μετάδοση
Shortest transaction:	Ο μικρότερος χρόνος

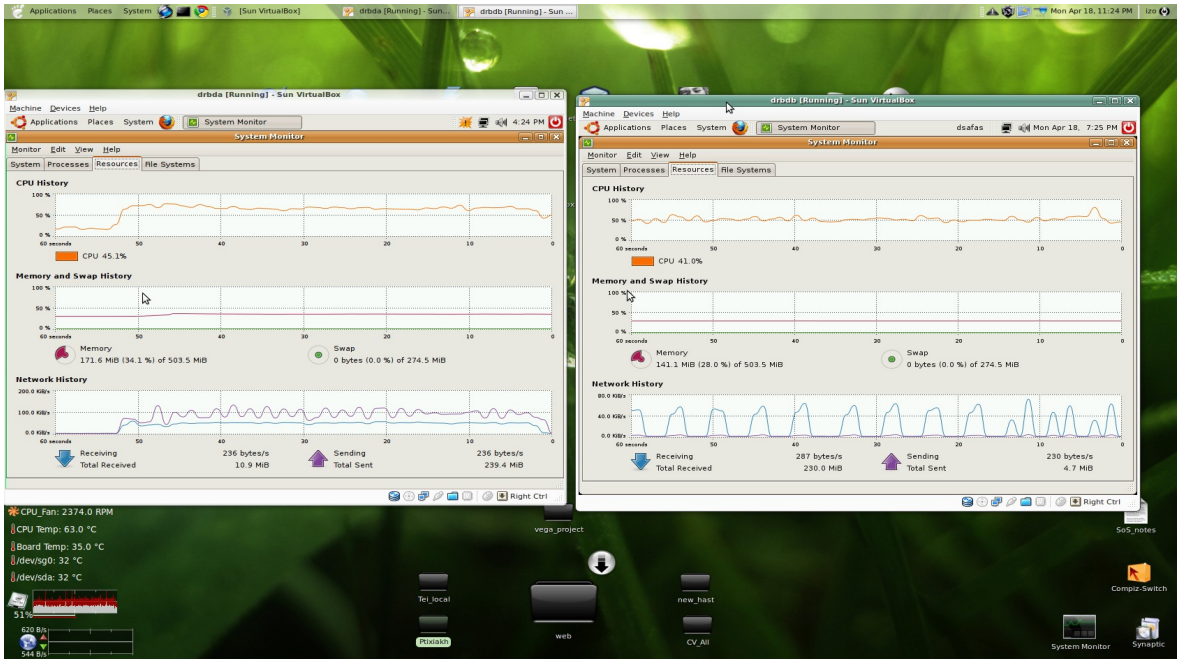
<u>Και τα δύο nodes ενεργά</u>

izo@i-book:~$ siege -c100 -t2m http://192.168.1.100/testphp.php

Lifting the server siege…	done.
Transactions:	11701 hits
Availability:	100.00 %
Elapsed time:	120.15 secs
Data transferred:	2.17 MB
Response time:	0.53 secs
Transaction rate:	97.39 trans/sec

Throughput:	0.02 MB/sec
Concurrency:	51.27
Successful transactions:	11701
Failed transactions:	0
Longest transaction:	1.60
Shortest transaction:	0.00

Εδώ βλέπουμε την χρήση του συστήματος cpu load, network load, memory, κάτω από πίεση 100 clients.Αριστερά είναι το drbda δεξιά το drbdb. Μπορούμε να παρατηρήσουμε απο το network load την κυματομορφή όπου είναι σαν χτύπος καρδιάς.



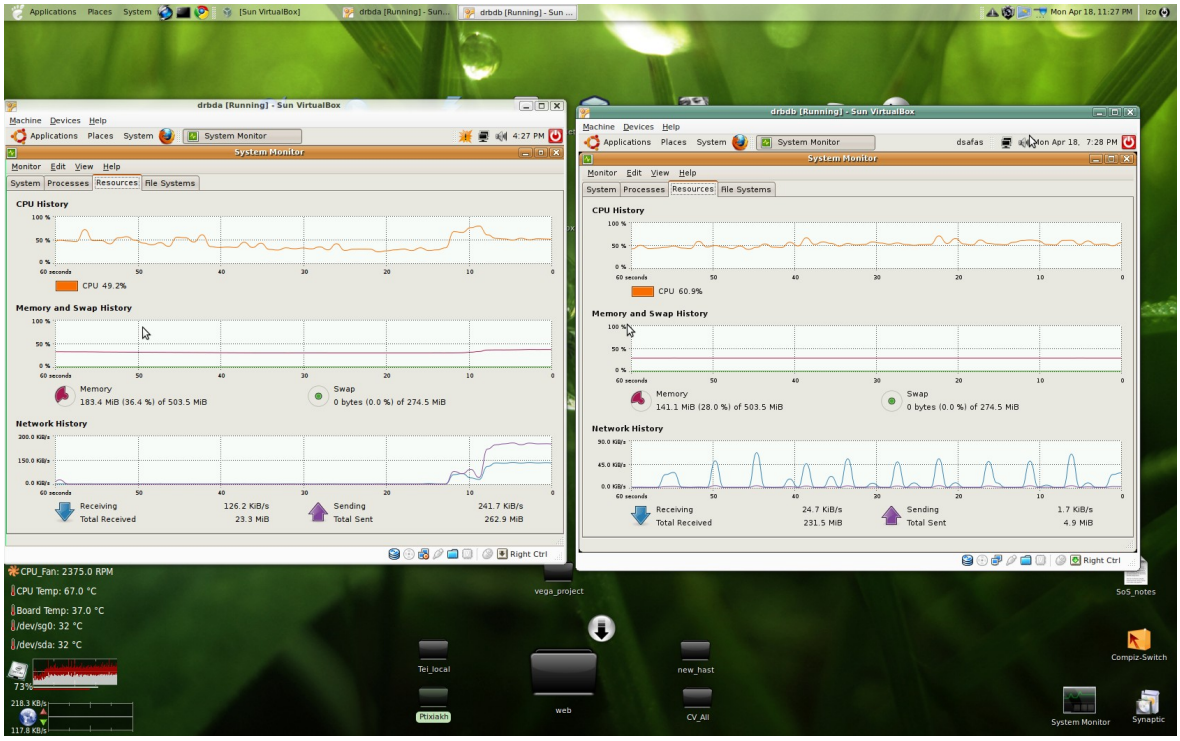
Εικόνα 2: «Χρήση πόρων DRBD 100 clients»

izo@i-book:~$ siege -c200 -t2m http://192.168.1.100/testphp.php

Lifting the server siege... done.

Transactions:	29437 hits
Availability:	100.00 %
Elapsed time:	120.19 secs
Data transferred:	9.17 MB
Response time:	0.30 secs
Transaction rate:	244.92 trans/sec
Throughput:	0.08 MB/sec
Concurrency:	74.46
Successful transactions:	29438
Failed transactions:	0
Longest transaction:	9.10
Shortest transaction:	0.00

Εδώ βλέπουμε την χρήση του συστήματος cpu load, network load, memory, κάτω από πίεση 200 clients.Η διαφορά που παρατηρούμε με την προηγούμενη εικόνα είναι στην ταχύτητα αποστολής δεδομένων απο τον drbda. Δεν παρατηρούμε μεγάλη αλλαγή στην χρήση μνήμης αλλά και cpu.

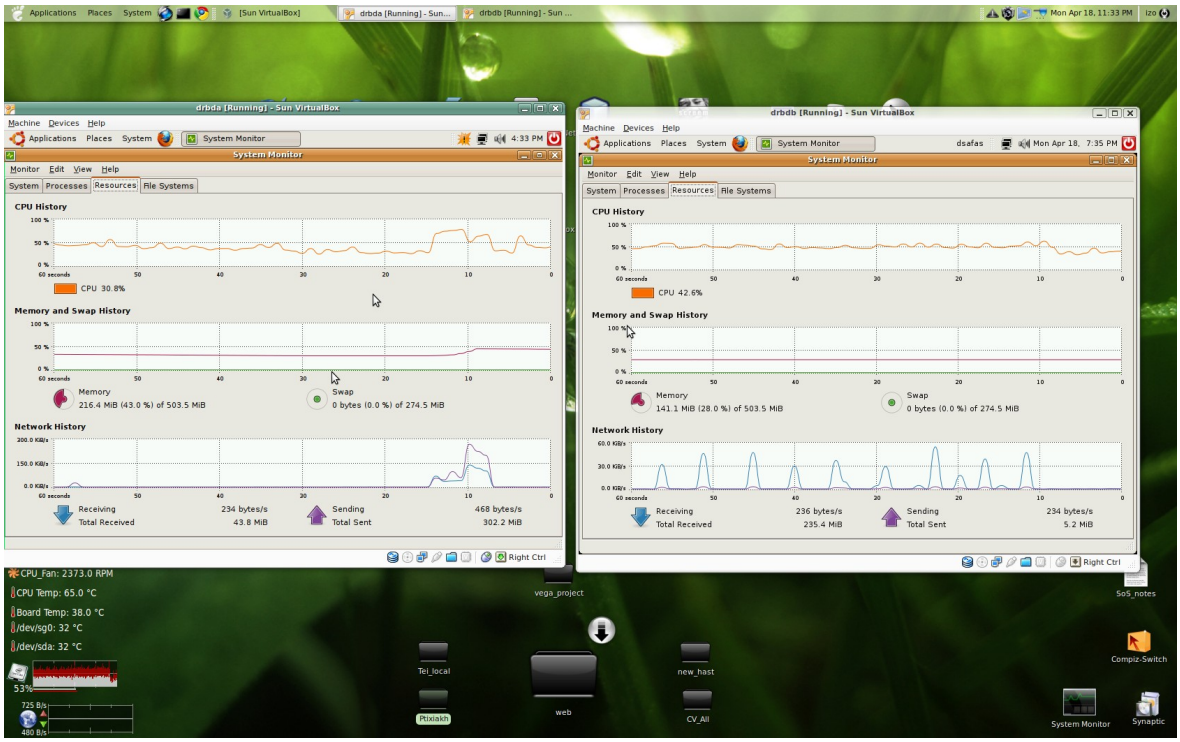


Εικόνα 3: «Χρήση πόρων DRBD 200 clients»

izo@i-book:~$ siege -c300 -t2m http://192.168.1.100/testphp.php

Lifting the server siege...	done.
Transactions:	27284 hits
Availability:	100.00 %
Elapsed time:	120.29 secs
Data transferred:	8.79 MB
Response time:	0.80 secs
Transaction rate:	226.82 trans/sec
Throughput:	0.07 MB/sec
Concurrency:	181.29
Successful transactions:	27284
Failed transactions:	0
Longest transaction:	16.02
Shortest transaction:	0.01

Εδώ βλέπουμε την χρήση του συστήματος cpu load, network load, memory, κάτω από πίεση 300 clients. Εδώ βλέπουμε μια στιγμιαία αύξηση στην χρήση τις cpu όπου φτάνει κοντά στο 80%.



Εικόνα 4: «Χρήση πόρων DRBD 300 clients»

Το σύστημα μπορεί να αντεπεξέλθει σε μεγαλύτερο αριθμό clients αλλά το μηχάνημα που τρέχει το παραπάνω τεστ δεν μπορεί να προσομοιώσει περισσότερα.

Για να συγχρονιστούν τα δύο nodes μετά από ένα αιφνίδιο θάνατο του ενός ~7 sec

Σε περίπτωση που πέσει ο πρωτεύων διακομιστής ο δευτερεύων χρειάζεται ~19 sec για να αναλάβει πλήρως.

<u>Το ένα node μή ενεργό.</u>

```
siege -c100 -t2m http://192.168.1.100/testphp.php
```

Lifting the server siege...	done.
Transactions:	15273 hits
Availability:	100.00 %
Elapsed time:	120.49 secs
Data transferred:	0.78 MB

Response time:	0.29 secs
Transaction rate:	126.76 trans/sec
Throughput:	0.01 MB/sec
Concurrency:	36.41
Successful transactions:	15273
Failed transactions:	0
Longest transaction:	0.83
Shortest transaction:	0.00

siege -c200 -t2m http://192.168.1.100/testphp.php

Lifting the server siege...	done.
Transactions:	31590 hits
Availability:	100.00 %
Elapsed time:	120.27 secs
Data transferred:	8.24 MB
Response time:	0.26 secs
Transaction rate:	262.66 trans/sec
Throughput:	0.07 MB/sec
Concurrency:	67.89
Successful transactions:	31590
Failed transactions:	0
Longest transaction:	3.90
Shortest transaction:	0.00

siege -c300 -t2m http://192.168.1.100/testphp.php

Lifting the server siege...	done.

Transactions:	40774 hits
Availability:	100.00 %
Elapsed time:	119.64 secs
Data transferred:	12.94 MB
Response time:	0.36 secs
Transaction rate:	340.81 trans/sec
Throughput:	0.11 MB/sec
Concurrency:	124.00
Successful transactions:	40775
Failed transactions:	0
Longest transaction:	14.99
Shortest transaction:	0.00

Ως αποτέλεσμα μπορούμε εύκολα να βγάλουμε πως όσο αυξάνει ο αριθμός τον αιτημάτων ανά δευτερόλεπτο, που πρέπει να διεκπεραιώσει ο διακομιστής πέφτει ο χρόνος απόκρισης. Ακόμη βλέπουμε την μείωση της διεκπεραιωτικής ικανότητας. Συγκρίνοντας τις τιμές που παίρνουμε όταν ο ένας διακομιστής είναι πεσμένος βλέπουμε πως όταν και οι δύο είναι ανεβασμένοι η απόδοση μειώνετε. Αυτό μπορεί να εξηγηθεί καθώς το μηχάνημα δεν αναπαράγει άμεσα τις εγγραφές και σε δεύτερο. Ακόμη θα πρέπει να λάβουμε υπόψη πως τα μηχανήματα μας είναι εικονικά άρα αφαιρώντας ένα μηχάνημα, σίγουρα το λειτουργικό όπου φιλοξενεί τα εικονικά συστήματα αφιερώνει περισσότερο χρόνο σε αυτό (περισσότερους πόρους).

ΚΕΦΑΛΑΙΟ 2 - HAST και UCARP

Το FreeBSD παρέχει High Availability υπηρεσίες μέσω του HAST σε συνδυασμό με το UCARP. Το HAST [12] δημιουργήθηκε από τον Pawel Jakub Dawidek και το UCARP [13] από τον Frank DENIS.

Σε αυτό το κεφάλαιο θα δούμε πώς λειτουργεί το HAST και το UCARP. Ακόμη θα δούμε πως στην πράξη μπορούμε να δημιουργήσουμε ένα διαδραστικό site στο FreeBSD και να ρυθμίσουμε το HAST και το UCARP ώστε το site μας να είναι πάντα διαθέσιμο. Τέλος θα δούμε την απόδοση των συστημάτων κάτω από πίεση.

2.1 HAST

Το HAST επιτρέπει την αποθήκευση δεδομένων σε δύο διαφορετικούς υπολογιστές, οι οποίοι είναι συνδεδεμένοι μέσω δικτύου TCP / IP. Αυτοί οι δύο υπολογιστές μαζί ονομάζεται cluster, ενώ ο καθένας ξεχωριστά αποτελεί ένα cluster node. Το HAST λειτουργεί με τη δομή Πρωτεύων - Δευτερεύων (Master-Backup, Master-Slave), το οποίο σημαίνει ότι μόνο ένα από τα cluster node μπορεί να είναι σε κατάσταση πρωτεύων μια συγκεκριμένη χρονική στιγμή. Το ενεργό node ονομάζεται πρωτεύων και είναι αυτό το οποίο μπορεί να χειρίζεται αιτήματα I / O στις συσκευές όπου διοικούνται από το HAST. Σήμερα, το HAST μπορεί να αποτελείται μόνο από δύο cluster nodes.

Το HAST λειτουργεί σαν σύμπλεγμα – παρέχει όλες τις συσκευές αποθήκευσης στο /dev/hast/ directory για χρήση από συστήματα αρχείων ή / και εφαρμογών. Όπως και το DRBD δουλεύει στο Block Level οπότε ενδείκνυται για αποθήκευση

δεδομένων. Δεν υπάρχει καμία διαφορά μεταξύ της χρήσης μιας συσκευής που παρέχει το HAST και άλλων σκληρών δίσκων ή partition.

Το HAST μπορεί να συγκριθεί με το RAID1 (mirror) όπου το ένα μηχάνημα είναι ο τοπικός δίσκος (στο πρωτεύων node) ενώ το δεύτερο είναι ο δίσκος στο απομακρυσμένο μηχάνημα (δευτερεύων node). Κάθε εργασία write, delete ή flush (BIO_WRITE, BIO_DELETE,BIO_FLUSH) στέλνεται και στο τοπικό και στον απομακρυσμένο δίσκο μέσω σύνδεσης TCP (εάν είναι διαθέσιμο το δευτερεύων node). Κάθε εργασία read εκτελείται από τον τοπικό δίσκο. Στην περίπτωση που δεν είναι ενημερωμένος (up-to-date) ή συμβεί σφάλμα I/O τότε η εργασία εκτελείται από το δευτερεύων node.

Η μείωση του χρόνου συγχρονισμού μετά την διακοπή λειτουργίας του κόμβου αποτελεί σημαντικό παράγοντα. Ο συγχρονισμός όλων των δεδομένων όταν η σύνδεση διακόπτεται για λίγα λεπτά δεν είναι τόσο αποδοτικός. Προκειμένου το HAST να πετύχει γρήγορο συγχρονισμό χαρακτηρίζει τις αλλαγές που συντελούνται στα δεδομένα (extents) ως “βρόμικα” (dirty) εάν αυτό χρειάζεται χρησιμοποιώντας την μέθοδο on-disk Bitmap. Κάθε εγγραφή extent αντιπροσωπεύεται από ένα bit στο bitmap, έτσι αποτελεί το μικρότερο block που μπορεί το HAST να αναγνωρίσει σαν dirty.

Το μέγεθος του extent είναι προεπιλεγμένο σε 2MB. Ακόμα και όταν μια εργασία write επιβεβαιωθεί από το αρχείο συστήματος, δηλαδή όταν επιβεβαιωθεί και από τα δύο nodes, είναι πιθανό να διακοπεί ο συγχρονισμός τους. Για το λόγο αυτό το extent πρέπει να χαρακτηρίζεται ως dirty πριν την εγγραφή των δεδομένων.

Φυσικά η διαδικασία αυτή θα ήταν πολύ αργή εάν το HAST για κάθε εγγραφή εκτελούσε συνέχεια της παρακάτω λειτουργίες:

1. Χαρακτήριζε το extent ως dirty και έγγραφε metadata.

2. Έγγραφε τα δεδομένα.

3. Χαρακτήριζε το extent ως καθαρό (clean) και έγγραφε metadata.

Αυτό θα σήμαινε ότι κάθε εργασία write θα μετατρεπόταν σε 3 εργασίες write. Για να το αποφύγουμε αυτό, το HAST κρατάει σταθερό τον αριθμό των extents που χαρακτηρίζει ως dirty συνεχώς. Εξ ορισμού αυτές είναι οι 64 πιο πρόσφατα extents. Αυτό βέβαια σημαίνει ότι όταν τα nodes συνδέονται πρέπει να συγχρονίσουν αυτομάτως 128MB δεδομένα, για τα οποία δεν υπάρχει στην πραγματικότητα ανάγκη να συγχρονιστούν. Σε επίπεδο όμως τοπικού δικτύου αυτή είναι μια γρήγορη εργασία και αξίζει να γίνεται. Στο σημείο αυτό σημειώνουμε

πως το μέγεθος του extent πρέπει να επιλέγεται προσεκτικά. Εάν το μέγεθος είναι πολύ μικρό, θα υπάρχουν πολλές αναβαθμίσεις των metadata, και θα μειωθεί η συνολική απόδοση του συστήματος. Εάν το μέγεθος του extent είναι πολύ μεγάλο, θα αυξηθεί ο χρόνος του συγχρονισμού το οποίο και πάλι θα μειώσει την απόδοση του συστήματος πριν ολοκληρωθεί ο συγχρονισμός.

Το HAST από μόνο του δεν μπορεί να καθορίσει τους ρόλους μεταξύ των δυο nodes (πρωτεύων και δευτερεύων). Υπεύθυνος για να καθορίσει το ρόλο του κάθε node θα πρέπει να οριστεί άλλο πρόγραμμα όπως το HeartBeat που παρουσιάσαμε στο προηγούμενο κεφάλαιο ή το ucarp.

Replication modes

Στη συνέχεια αναφέρουμε διαφορετικά Replication mode. Από την παρακάτω λίστα, μόνο το πρώτο replication mode υποστηρίζεται, τα υπόλοιπα απλά αναφέρονται για να δούμε τις διαφορές μεταξύ τους.

- memsync – Η εργασία write ολοκληρώνεται όταν το τοπικό node ολοκληρώσει την εντολή και όταν το απομακρυσμένο node αναγνωρίσει την άφιξη των δεδομένων αλλά πριν πραγματικά τα αποθηκεύσει. Τα δεδομένα στο απομακρυσμένο node θα αποθηκευτούν αμέσως μόλις σταλεί η απάντηση. Αυτός ο τύπος έχει σκοπό να μειώσει την καθυστέρηση και παρόλα αυτά να προσφέρει πολύ καλή αξιοπιστία. Η μόνη περίπτωση όπου κάποια μικρή ποσότητα δεδομένων θα μπορούσε να χαθεί είναι όταν τα δεδομένα αποθηκεύονται πρώτα στο πρωτεύων node και μετά στο δευτερεύων. Τότε το δευτερεύων node αναγνωρίζει τα δεδομένα και το πρωτεύων node κάνει αναφορά ολοκλήρωσης στην εφαρμογή. Πριν τα δεδομένα αποθηκευτούν πραγματικά στο δευτερεύων node, πέσει για κάποιο χρονικό διάστημα. Πριν το δευτερεύων node επιστρέφει, το κύριο node πεθαίνει εντελώς. Τότε το δευτερεύων node επανέρχεται και γίνεται το νέο πρωτεύων node. Δυστυχώς μια μικρή ποσότητα δεδομένων η οποία είχε επιβεβαιωθεί ότι είχε αποθηκευτεί χάνεται. Αλλά ο κίνδυνος για να συμβεί κάτι τέτοιο είναι πολύ μικρός. Το memsync mode δεν υποστηρίζεται.

- fullsync - Η εργασία write ολοκληρώνεται όταν και το τοπικό και το απομακρυσμένο node ολοκληρώσει την εγγραφή.Αυτός είναι ο ασφαλέστερος και ο πιο αργός τρόπος replication mode και είναι προεπιλεγμένος.

- async - Η εργασία write ολοκληρώνεται όταν το τοπικό node ολοκληρώσει την εγγραφή. Αυτό είναι το πιο γρήγορο αλλά και το πιο επικίνδυνο mode. Αυτός ο τύπος πρέπει να χρησιμοποιείται όταν το replication γίνεται σε απομακρυσμένο node όπου η καθυστέρηση είναι πολύ υψηλή. Το async mode σήμερα δεν εφαρμόζεται.

Πως λειτουργεί ο συγχρονισμός;

Όταν δουλεύουμε ως πρωτεύων και δεν υπάρχει δευτερεύων node, αυξάνουμε τον αριθμό συγχρονισμού (cnt) (ο οποίος υποδηλώνει σε ποια κατάσταση βρίσκετε το σύστημα) για τα metadata. Όταν συνδεθεί και συγχρονιστεί και το δευτερεύων node τότε ορίζουμε το τοπικό cnt ίσο με το απομακρυσμένο cnt, που θα πει πως λίγο πολύ τα 2 nodes είναι συγχρονισμένα.

Υπάρχουν πολλοί πιθανοί συνδυασμοί για τα διάφορα cnt που παρουσιάζονται όλοι στη συνέχεια. Σημασία δεν έχουν τόσο τα νούμερα όσο η σύγκριση μεταξύ τους. Παρακάτω συγκρίνουμε το τοπικό cnt του δευτερεύων με το απομακρυσμένο cnt του πρωτεύων και το απομακρυσμένο cnt του δευτερεύων με το τοπικό cnt το πρωτεύων. Σημειώστε ότι σε κάθε περίπτωση που το τοπικό cnt του πρωτεύων είναι μικρότερο από το απομακρυσμένο cnt του δευτερεύων και όπου το τοπικό cnt του δευτερεύων είναι μικρότερο από το απομακρυσμένο cnt του πρωτεύων δεν μπορεί να εφαρμοστεί.

Οπότε σε αυτή την περίπτωση εκτελούμε πλήρη συγχρονισμό. Αυτές οι περιπτώσεις σημειώνονται με αστερίσκο.

Με τον τυπικό συγχρονισμό εννοούμε ότι συγχρονίζονται μόνο οι προεκτάσεις που χαρακτηρίζονται ως dirty.

Πίνακας 2: «Αντιστοιχίες cnt – Τύπος συγχρονισμού»

Secondary metadata	Primary metadata	Synchronization type
local=3, remote=3	local=2, remote=2*	?! Full sync from secondary.
local=3, remote=3	local=2, remote=3*	?! Full sync from primary.
local=3, remote=3	local=2, remote=4*	?! Full sync from primary.
local=3, remote=3	local=3, remote=2	Primary is out-of-date, regular sync from secondary.
local=3, remote=3	local=3, remote=3	Regular sync just in case.
local=3, remote=3	local=3, remote=4*	?! Full sync from primary.
local=3, remote=3	local=4, remote=2	Split-brain condition.

local=3, remote=3	local=4, remote=3	Secondary out-of-date, regular sync from primary.
local=3, remote=3	local=4, remote=4*	?! Full sync from primary.

2.2 UCARP

Το UCARP επιτρέπει σε ένα ζεύγος διακομιστών να μοιράζονται κοινή εικονική διεύθυνση IP προκειμένου να παρέχει αυτόματο failover δηλαδή στο να σηκώνει αυτόματα υπηρεσίες. Το UCARP βασίζεται στο πρωτοκόλλου Common Address Redundancy (CARP). Χρησιμοποιούμε το UCARP καθώς το CARP είναι μέρος του πυρήνα BSD και επίσης παρέχει αρκετές επιλογές.

Τα σημαντικότερα χαρακτηριστικά του CARP πρωτοκόλλου είναι τα ακόλουθα:

1. πολύ μικρό overhead

2. κρυπτογραφημένα μηνύματα

3. συμβατότητα μεταξύ διαφορετικών λειτουργικών συστημάτων

4. δεν υπάρχει ανάγκη για επιπλέον σύνδεση μέσω δικτύου μεταξύ των επιπρόσθετων διακομιστών.

CARP

Το carp υλοποιείται σαν εικονική συσκευή η οποία υλοποιεί και ελέγχει το πρωτόκολλο CARP. Το carp επιτρέπει σε πολλαπλά συστήματα του ίδιου τοπικού δικτύου να μοιράζονται ένα σύνολο διευθύνσεων IP. Πρωταρχικός σκοπός του είναι να εξασφαλίσει ότι αυτές οι διευθύνσεις είναι πάντα διαθέσιμες, αλλά με συγκεκριμένες ρυθμίσεις το carp μπορεί επίσης να εξισορροπεί το φόρτο εργασίας.

2.3 Υλοποίηση

Δημιουργούμε δύο νέους εικονικούς δίσκους μεγέθους 5GB για την εγκατάσταση των συστημάτων και άλλους δύο δίσκους από 1GB για το HAST. Προσαρτούμε στο εικονικό cd-rom το αρχείο για την εγκατάσταση και ξεκινάμε τα συστήματα. Κάνουμε την εγκατάσταση στον πρώτο δίσκο αλλά χωρίς να πειράξουμε τον

δεύτερο, δηλαδή τον αφήνουμε αφορμάριστο και ορίζουμε της παραμέτρους του δικτύου καθώς και τα ονόματα τον μηχανημάτων παρόμοια με το ubuntu. Εμείς θα ορίσουμε σαν ονόματα για τον πρωτεύων διακομιστή nodeA με διεύθυνση ip 192.168.1.100 και για τον δευτερεύων nodeB ip 192.168.1.101 αντίστοιχα από κοινού subnet 24 και gateway 192.168.1.1

Όταν κάνουμε την πρώτη επανεκκίνηση θα πρέπει να αφαιρέσουμε το εικονικό cd-rom και να ορίσουμε την κάρτα δικτύου σε 1Gbit eth intel αλλά και το mode σε bridged adapter.

Εκκινούμε τα συστήματα και κάνουμε είσοδο ως root. Θα πρέπει να λάβουμε τις τελευταίες ενημερώσεις, οπότε εκτελούμε διαδοχικά και στα δύο μηχανήματα

```
portsnap fetch
```

Εφόσον δεν έχουμε περάσει ήδη τα ports κατά την εγκατάσταση, χρειάζεται η παρακάτω εντολή.

```
portsnap extract
```

```
portsnap update
```

Το HAST είναι μέρος του συστήματος στο FreeBSD άρα δεν χρειάζεται κάποια εγκατάσταση

Για να παραμετροποιήσουμε το HAST το μόνο που έχουμε να κάνουμε είναι να επεξεργαστούμε το αρχείο /etc/hast.conf.

Το αρχείο θα έχει την παρακάτω μορφή για τον nodeA και για τον nodeB

```
resource test {                         # Όνομα
        on nodeA {                  # Για ποιον διακομιστή
                local /dev/ad1          # Ο δίσκος με το
                                        #1GB για το HAST
                remote 192.168.1.101    # Πια είναι η ip
                                        #του άλλου διακομιστή
        }
        on nodeB {                  # Για ποιον διακομιστή
```

```
                local /dev/ad1      # Ο δίσκος με το 1GB για
                                    # το HAST

                remote 192.168.1.100    # Ποια είναι η
                                    #ip του άλλου διακομιστή

        }

}
```

Ας κάνουμε εγκατάσταση του UCARP

Εκτελούμε και στους δύο διακομιστές `cd /usr/ports/net/ucarp && make install clean`

Ακόμη για να εγκαταστήσουμε τον Apache, Mysql, PhP εκτελούμε και στους δύο διακομιστές

Εγκατάσταση του Apache

```
cd /usr/ports/www/apache22/ && make config install clean
```

Εγκατάσταση της PhP. Εδώ όταν ρωτηθούμε για ποιους διακομιστές θέλουμε να προσθέσουμε υποστήριξη θα δηλώσουμε τον Apache.

```
cd /usr/ports/lang/php5 && make config install clean
```

Εγκατάσταση της Mysql

```
cd /usr/ports/databases/mysql51-server && make install clean
```

```
mysql -u root
```

```
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('Νέος κωδικός');
```

Εγκατάσταση του PhPmyAdmin

```
cd /usr/ports/databases/phpmyadmin && make config install clean
```

Μόλις τελειώσει η εγκατάσταση μπορούμε να ξεκινήσουμε με την δημιουργία των απαραίτητων αρχείων ώστε να δουλέψει το UCARP.

Θα δημιουργήσουμε το αρχείο ucarp.sh το οποίο θα αρχικοποιεί το HAST καθώς και το UCARP, το vip-up.sh το οποίο θα εκτελείται μόλις ο διακομιστής γίνει πρωτεύων και vip-down.sh μόλις ο διακομιστής γυρίσει σε δευτερεύων. Όλα τα αρχεία θα οριστούν ως εκτελέσιμα, άρα μόλις τα δημιουργήσουμε θα εκτελέσουμε chmod 766 ucarp.sh vip-up.sh vip-down.sh.

Το αρχείο ucarp.sh στον node A όπως φαίνεται παρακάτω

```
#!/bin/sh

/sbin/hastd          # Αρχικοποίηση του HAST

# Αποθήκευση σε μεταβλητή το αποτέλεσμα από το ping με
#παραμέτρους αναμονής στο 1 δευτερόλεπτο (αυτό βέβαια πρέπει

#να αλλάξει αναλόγως του φόρτου του δικτύου. Μετά από

#έλεγχο και της απόδοσης του συγκεκριμένου δικτύου 1

#δευτερόλεπτο είναι αρκετό προκειμένου να αποφασίσουμε στο

#εάν η ip αυτή υπάρχει) και αποστολή ενός μόνο πακέτου,

#στην shared ip την οποία θα χρησιμοποιήσουμε για το

#failover και μετά από ξεκαθάρισμα των άχρηστων πληροφοριών

#με χρήση της grep.

var1="$(ping -c 1 -W 1 192.168.1.102 | grep '100')"

# Έλεγχος για το εάν το παραπάνω αποτέλεσμα είναι διάφορο

#του κενού. Τότε σημαίνει πως δεν υπάρχει άλλος ενεργός

#διακομιστής ως πρωτεύων, οπότε ο συγκεκριμένος διακομιστής

#θα οριστεί ως πρωτεύων.

if [ -n "$var1" ]; then

# Αρχικοποίηση του UCARP με παραμέτρους -B εκτέλεση σαν
#δαίμονας, interface το όνομα του interface στο οποίο θα
#δουλεύει το UCARP,  vhid μοναδικός αριθμός για τον
```

```
#εικονικό διακομιστή, pass συνθηματικό ώστε να μπορεί να

#εμπιστευτεί ο ένας τον άλλο,  preempt όταν είναι δυνατόν

#να γυρίσει σε πρωτεύων αμέσως, addr η εικονική ip που θα
#χρησιμοποιηθεί, upscript δήλωση της διαδρομής του αρχείου

#όπου θα εκτελεστεί όταν ο διακομιστής γίνει πρωτεύων,

#downscript δήλωση της διαδρομής του αρχείου όπου θα

#εκτελεστεί όταν ο διακομιστής γίνει δευτερεύων

     /usr/local/sbin/ucarp         -B         --interface=em0
--srcip=192.168.1.100    --vhid=102    --pass=omg    --preempt
--addr=192.168.1.102        --upscript=/etc/vip-up.sh       –
downscript=/etc/vip-down.sh

# Διαφορετικά γίνε δευτερεύων.Η μόνη διαφορά είναι ότι σε

#αυτή την περίπτωση δεν διαβάζετε από το UCARP το  vip-down

#κατά την εκκίνηση,θα πρέπει να δηλώσουμε και στο HAST πως

#αυτός είναι δευτερεύων σε αυτό το script. Οι υπόλοιπες

#ρυθμίσεις είναι ίδιες.

else

     /sbin/hastctl role secondary test

     /usr/local/sbin/ucarp         -B         --interface=em0
--srcip=192.168.1.100    --vhid=102    --pass=omg    --preempt
--addr=192.168.1.102                --upscript=/etc/vip-up.sh
--downscript=/etc/vip-down.sh

fi

exit 0 # δήλωση στο σύστημα πως το αρχείο εκτελέστηκε
#επιτυχώς
```

Το αρχείο στο nodeB είναι ίδιο εκτός από την διεύθυνση ip η οποία είναι του συγκεκριμένου διακομιστή δηλαδή η παράμετρος --srcip=192.168.1.101

```
#!/bin/sh

/sbin/hastd

var1="$(ping -c 1 -W 1 192.168.1.102 | grep '100')"

if [ -n "$var1" ]; then

     /usr/local/sbin/ucarp         -B         --interface=em0
--srcip=192.168.1.101    --vhid=102    --pass=omg    --preempt
--addr=192.168.1.102                --upscript=/etc/vip-up.sh
--downscript=/etc/vip-down.sh

else

     /sbin/hastctl role secondary test

     /usr/local/sbin/ucarp         -B         --interface=em0
--srcip=192.168.1.101    --vhid=102    --pass=omg    --preempt
--addr=192.168.1.102                --upscript=/etc/vip-up.sh
--downscript=/etc/vip-down.sh

fi



exit 0
```

Το αρχείο vip-up.sh θα είναι όπως παρακάτω και για τους δύο διακομιστές.

```
#!/bin/sh

/sbin/ifconfig em0 192.168.1.102 alias #Δήλωση της εικονικής
#ip

/sbin/hastctl role primary test        #Δήλωση στο HAST οτι
#ο παρόν διακομιστής είναι πρωτεύων

#Εδώ περιμένουμε κάποια δευτερόλεπτα προκειμένου το HAST να
#δημιουργήσει την εικονική συσκευή. Μετά από ελέγχους το

#σύστημα μας χρειάζεται περίπου 7 δευτερόλεπτα. Εδώ φυσικά
```

```
#αναλόγως με το σύστημα μπορεί να βελτιωθεί αυτό.

Sleep 7

#προσάρτηση του εικονικού δίσκου στο σημείο που θέλουμε

/sbin/mount -t ufs /dev/hast/test /mnt/test

#Αρχικοποίηση του mysql server

/usr/local/etc/rc.d/mysql-server onestart

#Αρχικοποίηση του apache

/usr/local/sbin/apachectl start
```

Το αρχείο vip-down.sh θα είναι ίδιο και στους δύο διακομιστές όπως φαίνεται παρακάτω

```
#!/bin/sh

#Σταμάτημα του apache

/usr/local/sbin/apachectl stop

#Αφαίρεση της κοινής ip

/sbin/ifconfig em0 192.168.1.102 -alias

# Σταμάτημα της  mysql

/usr/local/etc/rc.d/mysql-server onestop

#Αποπροσάρτηση του εικονικού δίσκου

/sbin/umount /mnt/test

#Δήλωση στο HAST πώς ο διακομιστής θα γίνει εφεδρικός

/sbin/hastctl role secondary test
```

Καθώς όταν κλίνει κανονικά ο διακομιστής θα πρέπει να κλείσουμε τις υπηρεσίες όπου έχουμε αρχίσει μέσα από το UCARP γιατί δεν γίνεται κλήση του /etc/vip-down.sh, θα πρέπει να τις δηλώσουμε μέσα στο αρχείο /etc/rc.shutdown το οποίο καλείται κατά τον τερματισμό του συστήματος. Άρα στην αρχή του αρχείου προσθέτουμε της παρακάτω γραμμές και στους δύο διακομιστές.

```
#Σταμάτημα της  mysql

/usr/local/etc/rc.d/mysql-server onestop

# Αποπροσάρτηση του εικονικού δίσκου

/sbin/umount /mnt/test

#Αφαίρεση της κοινής ip

/sbin/ifconfig em0 192.168.1.102 -alias
```

Ακόμη λόγου του ότι υπάρχει μεγάλη πιθανότητα μερικές υπηρεσίες να μην κλείσουν λόγω καθυστέρησης στο τέλος του αρχείου εκτελούμε εξαναγκασμένο σταμάτημα με την εντολή η οποία φαίνεται παρακάτω:

```
/sbin/halt
```

Έτσι λοιπόν είμαστε έτοιμοι να αρχικοποιήσουμε το HAST και να τοποθετήσουμε τα αρχικά meta data στους δίσκους, δίνοντας τις παρακάτω εντολές στον node Aκαι στον nodeB αναλόγως

```
nodeA# hastctl create test

nodeA# hastd

nodeB# hastctl create test

nodeB# hastd
```

Έπειτα θα πρέπει να ορίσουμε χειρωνακτικά τον ρόλο του κάθε διακομιστή στο HAST καθώς δεν θα τρέξουμε το αρχείο αρχικοποίησης πριν την πρώτη επανεκκίνηση. Άρα δίνουμε της παρακάτω δυο εντολές για στους διακομιστές αναλόγως, όπου η πρώτη εντολή ορίζει ως πρωτεύων το nodeA και η δεύτερη ως δευτερεύων το nodeB.

```
nodeA# hastctl role primary test

nodeB# hastctl role secondary test
```

Λόγω του ότι ο nodeA είναι ο πρωτεύων μόνο σε αυτόν θα είναι ορατός ο εικονικός δίσκος άρα λοιπόν σε αυτόν θα ορίσουμε των τύπο του αρχείου συστήματος

```
nodeA# newfs -U /dev/hast/test
```

Τέλος μετά της παραπάνω εντολές τα συστήματα θα ξεκινήσουν να συγχρονίζονται. Για να δούμε την πρόοδο εκτελούμε στον πρωτεύων διακομιστή την εντολή.

```
nodeA# hastctl status test
```

Για να διαπιστώσουμε εάν έχουν συγχρονιστεί θα παρατηρήσουμε το πεδίο που γράφει dirty το οποίο δηλώνει πόσα bit δεν έχουν συγχρονιστεί.

Μόλις ολοκληρωθεί ο συγχρονισμός μπορούμε να προσαρτήσουμε τον δίσκο στον πρωτεύων ώστε να περάσουμε τα δεδομένα της mysql. Άρα εκτελούμε μόνο στον nodeA

```
nodeA#/sbin/mount -t ufs /dev/hast/test /mnt/test
```

Έπειτα αντιγράφουμε τα δεδομένα με την εντολή

```
nodeA#cp -R -p /var/db/mysql /mnt/test
```

Ορίζουμε τα δικαιώματα όπως και στο κεφάλαιο 1.3, με τις παρακάτω εντολές

```
nodeA#chown -R mysql:mysql /mnt/test/mysql
```

```
nodeA#chown -R mysql:mysql /mnt/test/mysql/mysql
```

Τέλος και στους δύο διακομιστές αντιγράφουμε και τα αρχεία για την δοκιμαστική σελίδα μας στον προεπιλεγμένο φάκελο που διαβάζει ο Apache δηλαδή την /usr/local/www/data

Οι ρυθμίσεις έχουν ολοκληρωθεί. Άρα σκοτώνουμε τις τρέχουσες διεργασίες προκειμένου να τις σηκώσουμε αυτόματα μέσα από το ucarp.sh, οπότε εκτελούμε και στους δύο διακομιστές.

Προσέχουμε την σειρά.

```
nodeB#killall ucarp
```

```
nodeB#killall hastd
```

```
nodeA#killall ucarp
```

```
nodeA#umount /mnt/test
```

```
nodeA#killall hastd
```

Τώρα μπορούμε να εκτελέσουμε το ucarp.sh πρώτα στο nodeA και έπειτα στο nodeB

Μια απροσδόκητη κατάσταση είναι όπως και στο DRBD το split-brain (το αναφέραμε και στην αρχή του κεφαλαίου. Για να ξεπεράσουμε αυτό το πρόβλημα θα πρέπει να αδειάσουμε τα δεδομένα από τον ένα από τους δύο διακομιστές και να τον συγχρονίσουμε σύμφωνα με τον άλλο.

```
hastctl role init test

hastctl create test

hastctl role secondary test
```

2.4 Stress Test – Αποτελέσματα

Θα χρησιμοποιήσουμε το ίδιο εργαλείο δηλαδή το siege

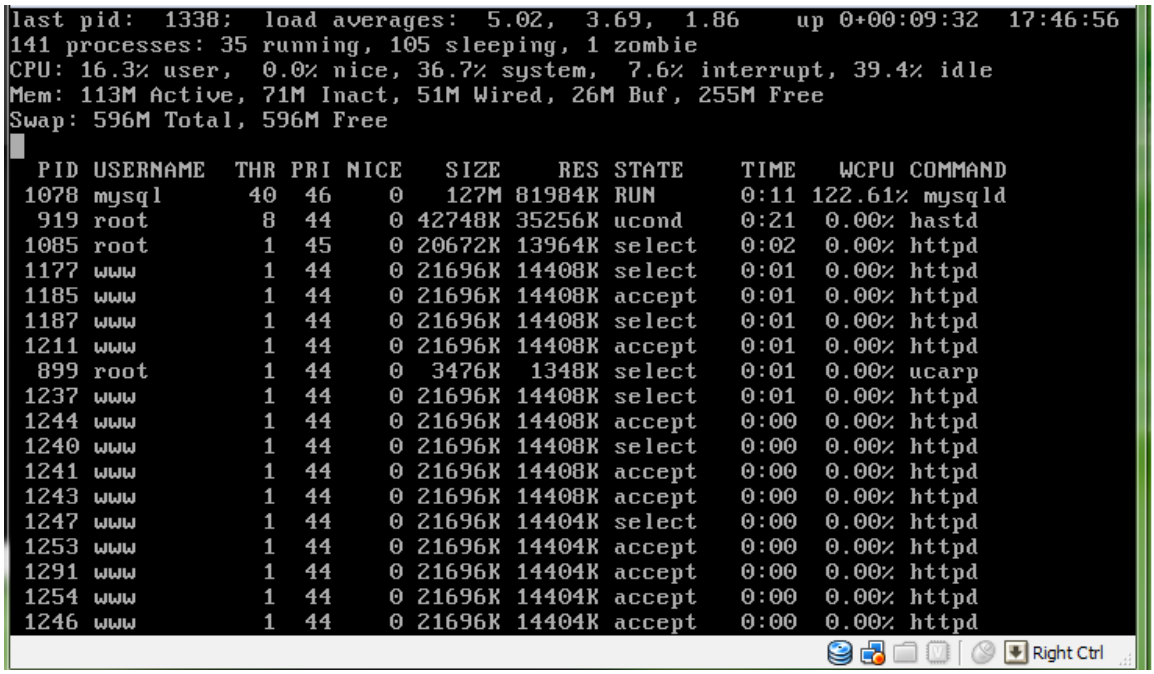
Και οι δύο διακομιστές ενεργοί

```
siege -c100 -t2m http://192.168.1.102/testphp.php
```

Lifting the server siege...	done.
Transactions:	6335 hits
Availability:	100.00 %
Elapsed time:	120.52 secs
Data transferred:	2.91 MB
Response time:	1.39 secs
Transaction rate:	52.56 trans/sec
Throughput:	0.02 MB/sec
Concurrency:	73.08
Successful transactions:	6335
Failed transactions:	0

Longest transaction:	13.94
Shortest transaction:	0.02

Εδώ βλέπουμε την χρήση του συστήματος cpu load, memory, κάτω από πίεση 100 clients. Παρατηρούμε πώς η χρήση της CPU 5%, ram 50% και το interrupt 7%, έτσι δεν έχει κάποιο πρόβλημα να διευθετήσει τους 100 clients.



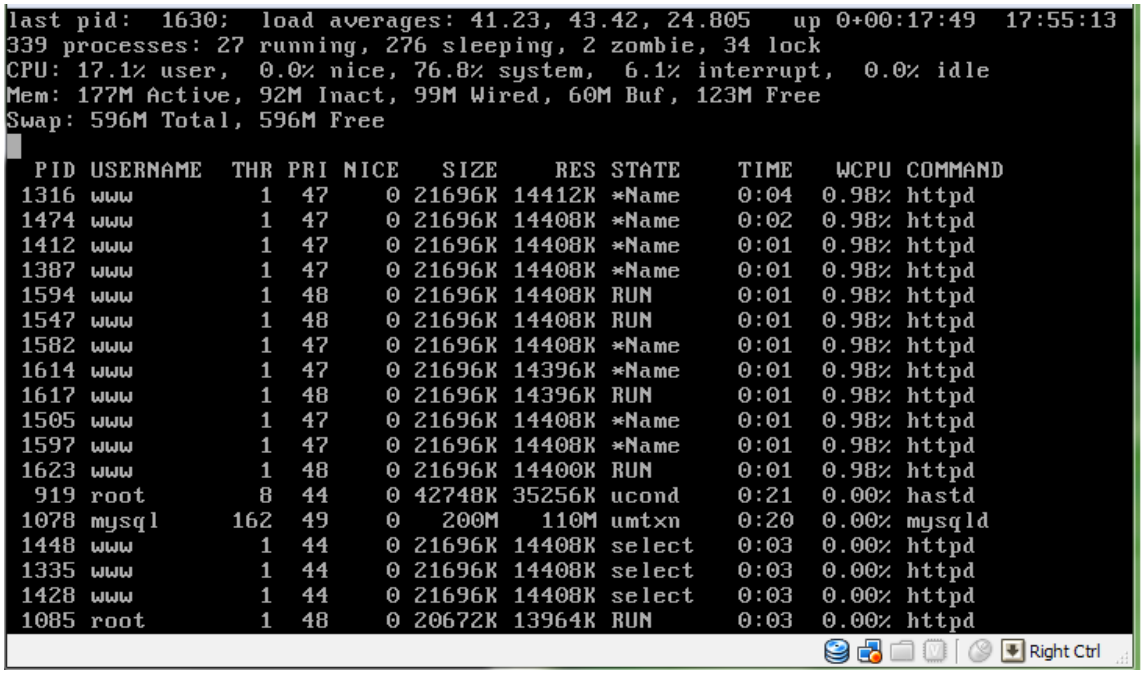
Εικόνα 5: «Χρήση πόρων HAST 100 clients»

```
siege -c200 -t2m http://192.168.1.102/testphp.php
```

Lifting the server siege...	done.
Transactions:	16961 hits
Availability:	99.35 %
Elapsed time:	120.38 secs
Data transferred:	7.63 MB
Response time:	0.69 secs
Transaction rate:	140.90 trans/sec

Throughput:	0.06 MB/sec
Concurrency:	97.05
Successful transactions:	16961
Failed transactions:	111
Longest transaction:	24.63
Shortest transaction:	0.00

Εδώ βλέπουμε την χρήση του συστήματος cpu load, memory, κάτω από πίεση 200 clients. Ο μέσος όρος χρήσης της cpu φαίνεται στο 41% , αλλά και η μνήμη στο 70%, και το interrupt σε πολύ χαμηλά επίπεδα 6,1%, κάτι το οποίο δεν δικαιολογεί την μη διεκπεραίωση 111 αιτημάτων .



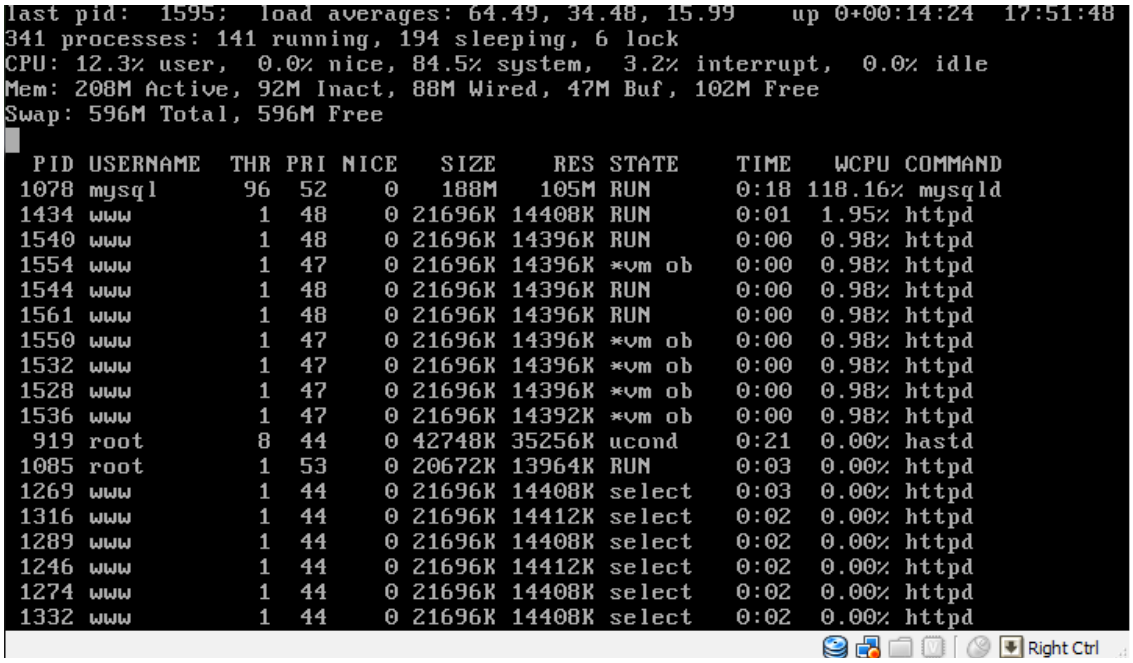
Εικόνα 6: «Χρήση πόρων HAST 200 clients»

```
siege -c300 -t2m http://192.168.1.102/testphp.php
```

Lifting the server siege...	done.
Transactions:	16095 hits

Availability:	95.73 %
Elapsed time:	119.59 secs
Data transferred:	7.25 MB
Response time:	1.30 secs
Transaction rate:	134.58 trans/sec
Throughput:	0.06 MB/sec
Concurrency:	174.68
Successful transactions:	16095
Failed transactions:	718
Longest transaction:	29.95
Shortest transaction:	0.00

Εδώ βλέπουμε την χρήση του συστήματος cpu load, memory, κάτω από πίεση 300 clients. Το σύστημά μας έχει φτάσει στα όριά του όχι μόνο από το ότι η χρήση της φυσικής μνήμης είναι στο 80% αλλά κυρίως από το πλήθος των αιτημάτων που έχει απορρίψει (718).



Εικόνα 7: «Χρήση πόρων HAST 300 clients»

Για να συγχρονιστούν τα δύο nodes μετά από ένα αιφνίδιο θάνατο του ενός χρειάζονται ~7 sec.

Σε περίπτωση που πέσει ο πρωτεύων διακομιστής ο δευτερεύων χρειάζεται ~20 sec για να αναλάβει πλήρως.

Το ένα node μή ενεργό.

```
siege -c100 -t2m http://192.168.1.102/testphp.php
```

Lifting the server siege...	done.
Transactions:	8287 hits
Availability:	100.00 %
Elapsed time:	119.87 secs
Data transferred:	3.81 MB
Response time:	0.93 secs
Transaction rate:	69.13 trans/sec
Throughput:	0.03 MB/sec
Concurrency:	64.42
Successful transactions:	8287
Failed transactions:	0
Longest transaction:	2.90
Shortest transaction:	0.04

```
siege -c200 -t2m http://192.168.1.102/testphp.php
```

Transactions:	11674 hits
Availability:	98.42 %
Elapsed time:	119.84 secs
Data transferred:	3.98 MB
Response time:	1.02 secs
Transaction rate:	97.41 trans/sec

Throughput:	0.03 MB/sec
Concurrency:	99.52
Successful transactions:	11674
Failed transactions:	188
Longest transaction:	28.26
Shortest transaction:	0.00

```
siege -c300 -t2m http://192.168.1.102/testphp.php
```

Lifting the server siege...	done.
Transactions:	18580 hits
Availability:	99.02 %
Elapsed time:	120.56 secs
Data transferred:	8.39 MB
Response time:	1.12 secs
Transaction rate:	154.11 trans/sec
Throughput:	0.07 MB/sec
Concurrency:	172.79
Successful transactions:	18580
Failed transactions:	184
Longest transaction:	23.90
Shortest transaction:	0.00

Τα αποτέλεσμα έχουν διαφορετικές τιμές αλλά σαν συμπέρασμα στην προκειμένη φάση, συγκρίνοντας μόνο τα δύο συστήματα, βγάζουμε τα ίδια συμπεράσματα με το DRBD δηλαδή όσο αυξάνονται τα αιτήματα που πρέπει να διεκπεραιώσει ο διακομιστής τόσο περισσότερο αυξάνεται η χρήση της μνήμης αλλά και η χρήση της cpu.

ΚΕΦΑΛΑΙΟ 3 - FailOver cluster και FreeNAS

Τα Windows Server 2008 παρέχουν αποθηκευτικές λύσεις για μικρές, μεσαίες και μεγάλες επιχειρήσεις. Υποστηρίζουν High available services μέσο του failover cluster σε συνδυασμό με Microsoft iSCSI Software. Σε αυτό το κεφάλαιο θα δούμε πως δουλεύει το High Availability στα Windows αλλά και το πώς μπορούμε να δημιουργήσουμε ένα web server και συγκεκριμένα ένα διαδραστικό site. Ακόμη θα δούμε την απόδοση των συστημάτων κάτω από πίεση.

3.1 FailOver cluster

Ένα failover cluster [14] είναι μια ομάδα ανεξάρτητων υπολογιστών οι οποίοι εργάζονται από κοινού για να αυξήσουν τη διαθεσιμότητα των εφαρμογών και υπηρεσιών. Οι διακομιστές, ονομάζονται nodes, συνδέονται με καλώδια και λογισμικό. Αν ένα από τα node του cluster αποτύχει, ένα άλλο node αρχίζει να παρέχει τις υπηρεσίες (μια διαδικασία γνωστή ως failover). Έτσι οι χρήστες έχουν ελάχιστες διακοπές στις υπηρεσίες που παρέχονται.

Η Microsoft υποστηρίζει μια λύση failover cluster μόνον αν όλα τα μηχανήματα (hardware) είναι πιστοποιημένα για Windows Server 2008. Επιπλέον, η διαμόρφωση (server, δίκτυο, και αποθηκευτικά μέσα), πρέπει να περάσει όλες τις

δοκιμές από τον οδηγό “Validate a Configuration”, ο οποίος περιλαμβάνεται στο Failover Cluster Manager.

Ποιες νέες λειτουργίες παρέχει το failover clustering;

Νέα λειτουργία επικύρωσης. Με αυτήν τη λειτουργία, μπορούμε να ελέγξουμε εάν το σύστημα, η αποθήκευση και η διαμόρφωση του δικτύου είναι κατάλληλα για cluster.

Support for GUID partition table (GPT). Οι δίσκοι GPT μπορούν να χωρίζονται σε τμήματα (particions) μεγαλύτερα από δύο terabytes και έχουν built-in redundancy στον τρόπο που αποθηκεύονται οι πληροφορίες στα particion σε αντίθεση με τους δίσκους master boot record (MBR).

Ένα failover cluster όπως ήδη αναφέραμε είναι μια ομάδα από ανεξάρτητους υπολογιστές. Τα nodes, είναι φυσικά συνδεδεμένοι σε ένα τοπικό δίκτυο (LAN) ή σε ένα ευρείας περιοχής δίκτυο (WAN).

Η ομάδα των nodes λογίζεται ως ενιαίο σύστημα και μοιράζεται κοινό namespace. Η ομάδα συνήθως περιλαμβάνει πολλαπλές συνδέσεις δικτύου και συσκευές αποθήκευσης δεδομένων που συνδέονται με τα nodes μέσω των δικτύων αποθήκευσης χώρου (SANs). Αυτή είναι και η μεγάλη διαφορά με τα προηγούμενα συστήματα.

Το failover cluster λειτουργεί μετακινώντας τους πόρους ανάμεσα στα nodes σε περίπτωση που κάποιος επιμέρους υπολογιστής διακόψει τη λειτουργία έτσι ώστε να παρέχονται συνεχώς οι υπηρεσίες του συστήματος.

Συνήθως όταν ένας server που εκτελεί μια συγκεκριμένη εφαρμογή σταματήσει λόγω σφάλματος, τότε η εφαρμογή δεν είναι διαθέσιμη μέχρι να επιδιορθωθεί ο server. Το failover clustering μπορεί να αντιμετωπίζει αυτό το πρόβλημα καθώς εντοπίζει τα σφάλματα του λογισμικού ή των μηχανημάτων και αμέσως ξαναρχίζει την εφαρμογή σε διαφορετικό node χωρίς να χρειάζεται παρέμβαση από τον διαχειριστή του συστήματος. Η διαδικασία αυτή είναι γνωστή ως failed-over. Έτσι οι χρήστες συνεχίζουν να έχουν πρόσβαση στις υπηρεσίες χωρίς να γνωρίζουν ότι αυτές παρέχονται από διαφορετικό server.

Ορολογία του Failover Clustering

Οι ακόλουθοι όροι και οι έννοιες χρησιμοποιούνται στο failover clustering:

- Resource (Πόροι). Ένα τμήμα του εξοπλισμού ή του λογισμικού σε ένα failover cluster (όπως ένας δίσκος, μια διεύθυνση IP ή ένα όνομα δικτύου)

- Resource group (ομάδα πόρων). Ο συνδυασμός των πόρων που διαχειρίζονται ως μια οντότητα στο failover.

- Dependency (αλληλεξαρτηση). Μια συνεργασία μεταξύ δύο ή περισσότερων πόρων στην αρχιτεκτονική ενός cluster.

- Quorum (απαρτία). Μια κοινή εικόνα των μελών (nodes και resources) σε ένα cluster. Προκειμένου να διασφαλιστεί ότι μόνο ένα τμήμα των μελών του cluster λειτουργεί σε μια χρονική στιγμή, η πλειοψηφία των μελών πρέπει να είναι ενεργή και σε επικοινωνία μεταξύ τους. Έτσι αποφεύγεται η περίπτωση του να υπάρχουν δύο διαφορετικές ομάδες μελών που προσπαθούν να εκτελέσουν μια εντολή γράφοντας στον ίδιο δίσκο, η οποία θα προκαλούσε πρόβλημα στην συστοιχία. Κάθε node αντιπροσωπεύει μία ψήφο στο σύστημα. Ένας φυσικός δίσκος ή ένα κοινόχρηστο αρχείο μπορεί επίσης να χρησιμεύσει ως quorum resource και να συνεισφέρει μια μοναδική ψήφο στο σύστημα.

- Heartbeat. Είναι ο μηχανισμός που παρακολουθεί τη σωστή λειτουργία του cluster ανάμεσα στα nodes. Έχουμε μιλήσει στο πρώτο κεφάλαιο για αυτό.

- Active / Active failover cluster model. Όλα τα nodes του failover cluster λειτουργούν και εξυπηρετούν πελάτες. Εάν ένα node αποτύχει, ο πόρος θα μετακινηθεί σε άλλο node και θα συνεχίσει να λειτουργεί κανονικά, με την προϋπόθεση ότι ο νέος server έχει αρκετή χωρητικότητα για να εκτελέσει τον πρόσθετο φόρτο εργασίας.

- Active / Passive failover cluster model. Ένα node στο failover cluster παραμένει συνήθως σε αδράνεια μέχρι να συμβεί ένα failover. Μόλις συμβεί, το node που ήταν σε αδράνεια γίνεται ενεργό και παρέχει υπηρεσίες στους πελάτες. Επιπλέον επειδή ήταν σε αδράνεια, προφανώς έχει αρκετή χωρητικότητα για να εξυπηρετήσει την εφαρμογή χωρίς να μειωθούν οι επιδόσεις του συστήματος.

- Shared storage. Όλα τα nodes στο failover cluster πρέπει να μπορούν να έχουν πρόσβαση στα δεδομένα σε ένα κοινό χώρο αποθήκευσης. γράφουν τα δεδομένα τους σε κοινό χώρο αποθήκευσης. Έτσι, εάν ένα node αποτύχει, όταν η

διαδικασία ξαναρχίζει σε διαφορετικό node, το νέο node μπορεί να διαβάσει τα ίδια στοιχεία από τον κοινό χώρο αποθήκευσης. Ο κοινός χώρος αποθήκευσης μπορεί να δημιουργηθεί με iSCSI.

3.2 FreeNAS

Το FreeNAS [15] είναι ένα είναι ελεύθερο λειτουργικό και έρχεται με την άδεια BSD. Το FreeNAS μετατρέπει έναν υπολογιστή σε NAS. Υποστηρίζει την σύνδεση με Microsoft Windows, Apple OS X, Linux και FreeBSD. Επίσης, υποστηρίζει RAID, έχει ένα απλό web GUI και μικρές απαιτήσεις συστήματος.

Το FreeNAS είναι συμπαγές, αποδοτικό και αφιερωμένο σε ένα μόνο έργο, στην περίπτωσή μας στο NAS. Μόλις το FreeNAS εγκατασταθεί σε έναν υπολογιστή, ο υπολογιστής μετατρέπεται σε dedicated NAS και δεν μπορεί να εκτελέσει ταυτόχρονα άλλα tasks.

Το FreeNAS υποστηρίζει τα ακόλουθα πρωτόκολλα πρόσβασης στο δίκτυο:

- CIFS (via Samba)
- FTP
- NFS
- AFP
- RSYNC
- iSCSI

Επίσης διαθέτει:

- Υποστήριξη για S.M.A.R.T
- Local and Active Directory
- Λογισμικό RAID (0,1,5)

3.3 Υλοποίηση

Δημιουργούμε 5 εικονικούς δίσκους δύο με χωρητικότητα 20GB για τα windows server 2008 δύο για το failover με 512 για Quorum disk και ένα με 5GB για τα δεδομένα και τέλος ένα με 5GB για την εγκατάσταση του freenas.

Εγκαθιστούμε το Freenas έχοντας προσαρτήσει από το Vbox μόνο τον δίσκο που προορίζετε για την εγκατάσταση του. Κατά την πρώτη επανεκκίνηση δημιουργούμε δύο εικονικούς scsi ελεγκτές όπου εκεί θα προσαρτήσουμε τους δύο δίσκους για το failover.

Ακόμη όπως και σε όλα τα συστήματα αλλάζουμε την κάρτα δικτύου σε 1GB adapter intel MT server σε bridged mode. Εκκινούμε το σύστημα. Μόλις το σύστημα μας έχει ολοκληρώσει την φόρτωση παρατηρούμε την γραμμή όπου γράφει Lan ipv4 address και βλέπουμε την διεύθυνση που έχει το σύστημα.

Έτσι ανοίγουμε ένα περιηγητή και δίνουμε την παραπάνω διεύθυνση προκειμένου να παραμετροποιήσουμε το σύστημα μέσα από το δικτυακό περιβάλλον που παρέχει. Όταν μας ζητήσει όνομα χρήστη και κωδικό βάζουμε τα προεπιλεγμένα τα οποία είναι admin και freenas αντίστοιχα.

Για να ορίσουμε τον Quorum δίσκο πηγαίνουμε στο menu και επιλέγουμε το Disks και έπειτα στο management. Στην νέα σελίδα όπου θα εμφανιστεί κάνουμε κλικ στο + και κατόπιν επιλέγουμε τον δίσκο με τα 512MB δίνουμε και μία περιγραφή αλλάζοντας το πεδίο του Description έπειτα αλλάζουμε την επιλογή στο Preformatted file system σε ZFS storage pool device και έπειτα κάνουμε κλικ στο Add κουμπί. Σε κάθε αλλαγή που κάνουμε θα πρέπει να πάτατε και εφαρμογή προκειμένου το Freenas να ενεργοποιήσει τις αλλαγές αυτές, άρα πατάμε το κουμπί Apply changes. Παρατηρούμε πως ο δίσκος πλέον είναι σε κατάσταση ONLINE στο Status. Έπειτα πηγαίνουμε στο μενού Disks και επιλέγουμε την κατηγορία ZFS κατόπιν πηγαίνουμε στο Virtual devices και κάνουμε κλικ στο + και δίνουμε ένα όνομα πχ quorum αφήνοντας τις υπόλοιπες επιλογές όπως έχουν επιλέγοντας μόνο το δίσκο από το πεδίο που γράφει devices και επιλέγοντας Add και Apply. Πηγαίνουμε στην καρτέλα που γράφει Management δίνουμε ένα όνομα πχ quorum_disk και επιλέγουμε την εικονική συσκευή που δημιουργήσαμε προηγουμένως δηλαδή το quorum κατόπιν κάνουμε add και Apply. Έπειτα πάμε στο μενού Services και κάνουμε κλικ στο iSCSI Target κατόπιν επιλέγουμε το Enable και πατάμε στο save και έπειτα restart ώστε να ενεργοποιηθεί η υπηρεσία. Στην ίδια κατηγορία κάνουμε κλικ στο portals προκειμένου να εφαρμόσουμε της ρυθμίσεις για το δίκτυο δηλαδή ip και την πόρτα. Απλά πατάμε add και Apply. Στη συνέχεια πηγαίνουμε στο Initators και επιλέγουμε add και apply.

Μετά από τις παραπάνω ρυθμίσεις πηγαίνουμε πάλι στο Targets και πατάμε το + στο Extend αλλάζοντας εδώ την διαδρομή στο path πεδίο πχ /mnt/quorum_disk/quorum. Στο πεδίο του File size δίνουμε το μέγεθος ΠΡΟΣΟΧΗ εδώ θα πρέπει να δώσουμε λιγότερα MB από την πραγματική χωρητικότητα του δίσκου πχ 450MB κατόπιν επιλέγουμε add και apply.

Από το ίδιο μενού δηλαδή στο Targets πατάμε το + στο target και δίνουμε ένα όνομα πχ quorum κατόπιν προσέχουμε το Flags να είναι στο Read/Write και το Storage στο extent0 και έπειτα πατάμε στο add και apply.

Έχουμε τελειώσει με τον quorum το επόμενο βήμα είναι να ορίσουμε τον δίσκο με τα 5GB τον οποίο θα χρησιμοποιήσουμε για τα δεδομένα εκτελώντας τα βήματα από την αρχή παρακάμπτοντας αυτά που αφορούν το δίκτυο και αυτό της ενεργοποίησης της υπηρεσίας δηλαδή τα Settings, Portals και Initiators. Στα ονόματα φυσικά θα χρησιμοποιήσουμε διαφορετικά πχ share

Επόμενο βήμα είναι να κάνουμε εγκατάσταση τους δύο διακομιστές τον πρώτο με όνομα node1 με ip 192.168.1.200 και τον δεύτερο με όνομα node2 και ip 192.168.1.201 και φυσικά αλλάζοντας όπως και σε όλους τους διακομιστές την κάρτα δικτύου από το Vbox. Για να ξεκλειδώσει το Failover θα πρέπει να δημιουργήσουμε ένα domain. Έτσι μετά την εγκατάσταση τρέχουμε στον node1 το dcpromo. Στο παράθυρο που μας εμφανίζετε επιλέγουμε το Use advanced mode installation. Έπειτα επιλέγουμε το Create a new domain in a new forest στο επόμενο παράθυρο ορίζουμε το όνομα του domain πχ hawin.net στο επόμενο παράθυρο αφήνουμε το ίδιο όνομα για το netbios και πατάμε επόμενο. Στο επόμενο παράθυρο ορίζουμε το 2008 και αφήνουμε τις προεπιλογές για τα επόμενα τρία παράθυρα. Στο τελευταίο παράθυρο ορίζουμε τον κωδικό πρόσβασης για τον διαχειριστή του Domain. Μόλις ολοκληρωθεί η εγκατάσταση κάνουμε επανεκκίνηση και κατόπιν εισερχόμαστε στο σύστημα σαν διαχειριστές του Domain.

Έχοντας ολοκληρώσει με επιτυχία τον πρώτο ελεγκτή του Domain θα ορίσουμε σαν δεύτερο ελεγκτή τον node2.Άρα εκτελούμε και σε αυτόν το dcpromo και εδώ επιλέγουμε το Use advanced mode installation στο επόμενο παράθυρο επιλέγουμε το Add a domain controller to an existing domain έπειτα δίνουμε το όνομα του υπάρχοντος domain δηλαδή το hawin.net έπειτα επιλέγουμε το Alternate credentials και δίνουμε το όνομα του διαχειριστή του Domain και το συνθηματικό που είχαμε ορίσει στον node1 και κατόπιν πατάμε επόμενο στα επόμενα παράθυρα. Μετά την επανεκκίνηση κάνουμε είσοδο στο σύστημα σαν διαχειριστές του Domain.

Και στους δύο διακομιστές μέσα από την κονσόλα ελέγχου του διακομιστή κάνουμε εγκατάσταση στο failover. Έπειτα στον πρώτο διακομιστή πηγαίνουμε στο administrative tools και έπειτα στο iSCSI initiator. Εδώ θα πρέπει να δώσουμε την διεύθυνση ip του freenas στο target ώστε να προσαρτήσουμε τους δίσκους στα windows. Βλέπουμε πως και οι δύο δίσκοι είναι σε κατάσταση inactive άρα τους επιλέγουμε και πατάμε το connect ακόμη επιλέγουμε να γίνονται αυτόματα reconnect κατά την εκκίνηση του συστήματος. Έπειτα κάνουμε κλικ στην καρτέλα που Volumes and Devices tab και πατάμε στο Auto Configure. Τέλος πηγαίνουμε στην διαχείριση τον δίσκων και βάζουμε online τους δίσκους. Έπειτα τους διαμορφώνουμε ως NTFS και μόλις τελειώσει η διαμόρφωση τους ξαναγυρνάμε σε offline. Έχοντας τελειώσει με τον πρώτο διακομιστή εκτελούμε τα ίδια βήματα εκτός από την διαμόρφωση των δίσκων.

Είμαστε έτοιμοι να ρυθμίσουμε το failover άρα ανοίγουμε την διαχείριση των clusters και κάνουμε κλικ στο Validate a configuration ώστε να δούμε πως έχουμε όλα τα απαραίτητα προκειμένου να στήσουμε το failover. Εφόσον ολοκληρωθεί με επιτυχία πατάμε στο create cluster ώστε να δημιουργήσουμε το failover. Επιλέγουμε τα node που θα περιλαμβάνει το failover δηλαδή το node 1 και node2 καθώς και το όνομα και την διεύθυνση που θα χρησιμοποιούν σαν κοινή. Έχοντας ολοκληρώσει το failover κατεβάζουμε το WAMP το οποίο εγκαταστήσει όλα μαζί το apache, mysql, php,phpmyadmin. Κάνουμε αντιγραφή του φακέλου όπου εμπεριέχει τα δεδομένα μέσα από φάκελος που έγινε εγκατάσταση το wamp/bin/mysql στον δίσκο που έχουμε για τα δεδομένα του failover. Από το πρόγραμμα διαχείρισης του wamp πηγαίνουμε στο mysql my.cnf και αλλάζουμε την διαδρομή του datadir με την διαδρομή του δίσκου που έχουμε κάνει αντιγραφή προηγουμένως. Τέλος κάνουμε restart all services στο πρόγραμμα διαχείρισης και έπειτα put online, εκτελούμε τις ίδιες ενέργειες και στον δεύτερο διακομιστή εκτός από την αντιγραφή του φακέλου καθώς ο δίσκος δεν είναι ορατός καθώς ο διακομιστής είναι ορισμένος ως δευτερεύων. Πηγαίνουμε στο πρόγραμμα διαχείρισης του failover και επιλέγουμε να εισάγουμε μια καινούργια διεργασία η οποία θέλουμε να ξεκινάει αυτόματα και γενικά να διαχειρίζεται από το failover. Άρα επιλέγουμε το wamp.

3.4 Stress Test – Αποτελέσματα

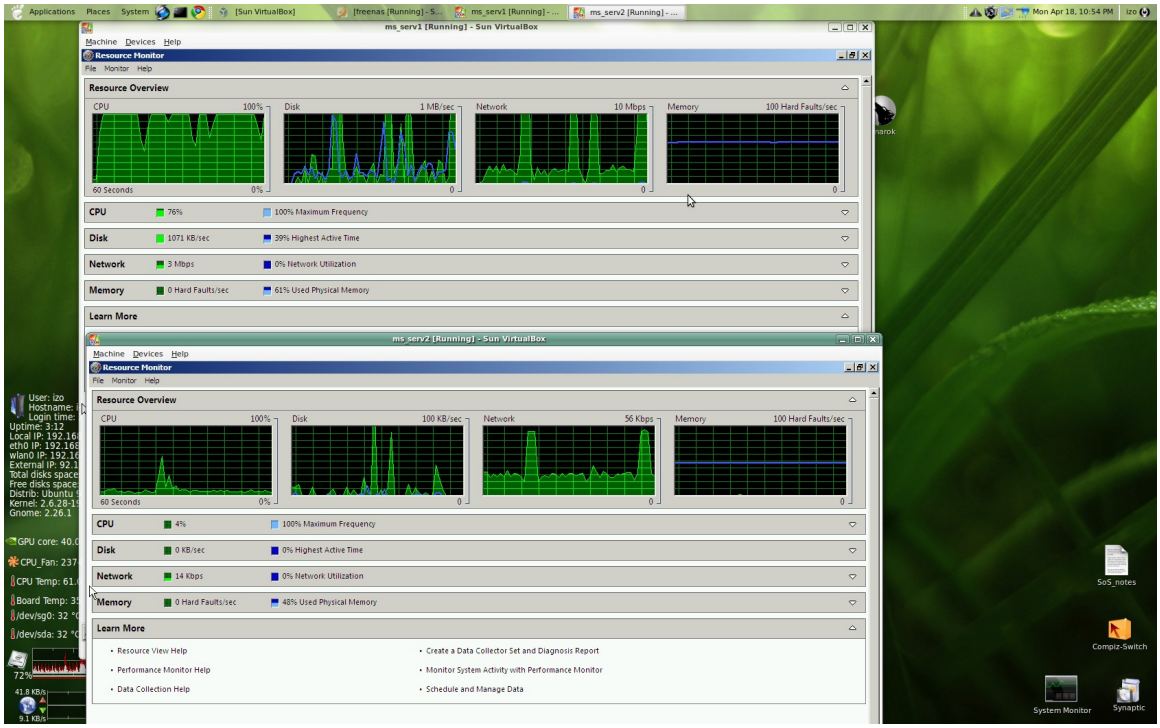
<u>Και οι δύο διακομιστές ενεργοί.</u>

```
izo@i-book:~$siege-c100-t2m http://192.168.1.202/testphp.php
```

Lifting the server siege... done.

Transactions:	3366 hits
Availability:	100.00 %
Elapsed time:	119.84 secs
Data transferred:	9.31 MB
Response time:	2.98 secs
Transaction rate:	28.09 trans/sec
Throughput:	0.08 MB/sec
Concurrency:	83.81
Successful transactions:	3366
Failed transactions:	0
Longest transaction:	7.86
Shortest transaction:	0.68

Εδώ βλέπουμε την χρήση του συστήματος cpu load, network load, memory, κάτω από πίεση 100 clients από πάνω είναι ο node1 και κάτω ο node2. Παρατηρούμε την χρήση τις CPU στο node1 όπου είναι στο μεγαλύτερο μέρος στο 100%.



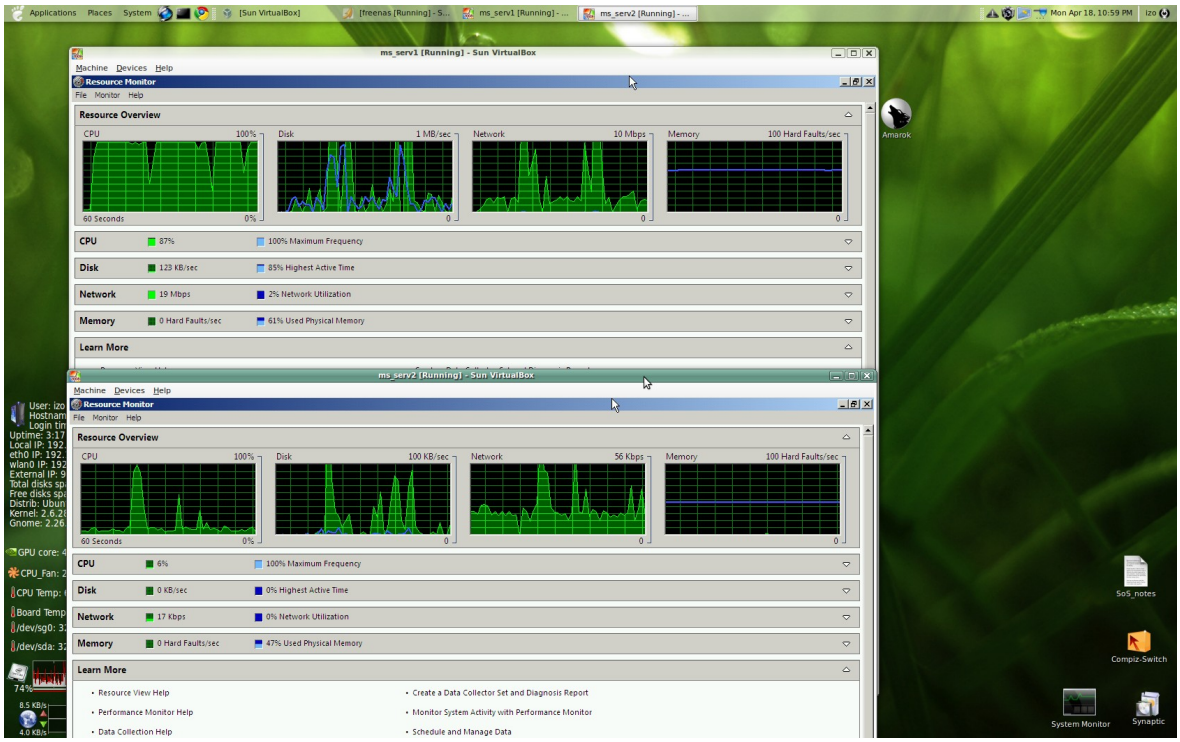
Εικόνα 8: «Χρήση πόρων FailOver 100 clients»

Παρατηρούμε ότι ήδη η CPU είναι στο 100% αρκετό χρόνο.

izo@i-book:~$ siege -c200 -t2m http://192.168.1.202/testphp.php

Lifting the server siege...	done.
Transactions:	3322 hits
Availability:	100.00 %
Elapsed time:	120.52 secs
Data transferred:	9.18 MB
Response time:	6.55 secs
Transaction rate:	27.56 trans/sec
Throughput:	0.08 MB/sec
Concurrency:	180.55
Successful transactions:	3322
Failed transactions:	0
Longest transaction:	12.72
Shortest transaction:	1.31

Εδώ βλέπουμε την χρήση του συστήματος cpu load, network load, memory, κάτω από πίεση 200 clients node1 και κάτω ο node2. Εδώ μπορούμε να παρατηρήσουμε την αύξηση χρήσης του δίσκου αλλά και του δικτύου κάτι που είναι φυσικό. Και εδώ έχουμε 100% χρήσης της CPU.



Εικόνα 9: «Χρήση πόρων FailOver 200 clients»

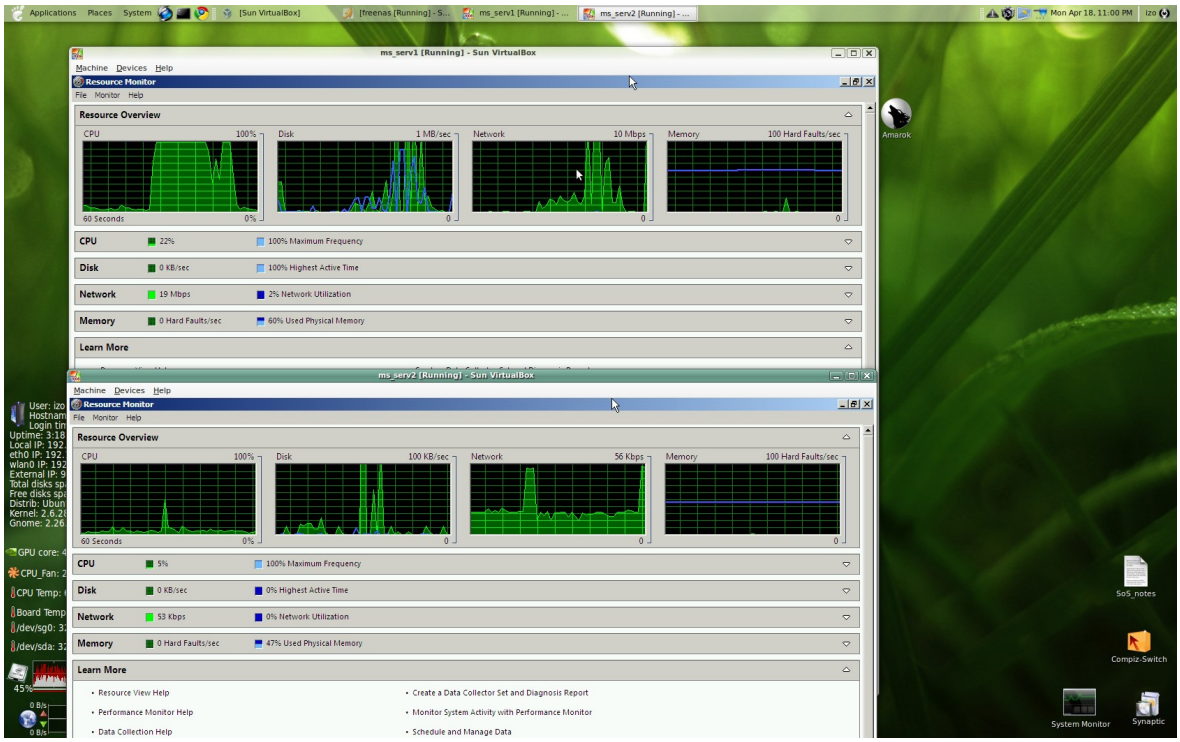
izo@i-book:~$ siege -c300 -t2m http://192.168.1.202/testphp.php

siege aborted due to excessive socket failure;

Transactions:	678 hits
Availability:	38.99 %
Elapsed time:	23.04 secs
Data transferred:	1.87 MB
Response time:	7.21 secs
Transaction rate:	29.43 trans/sec
Throughput:	0.08 MB/sec
Concurrency:	212.21
Successful transactions:	678
Failed transactions:	1061
Longest transaction:	10.19

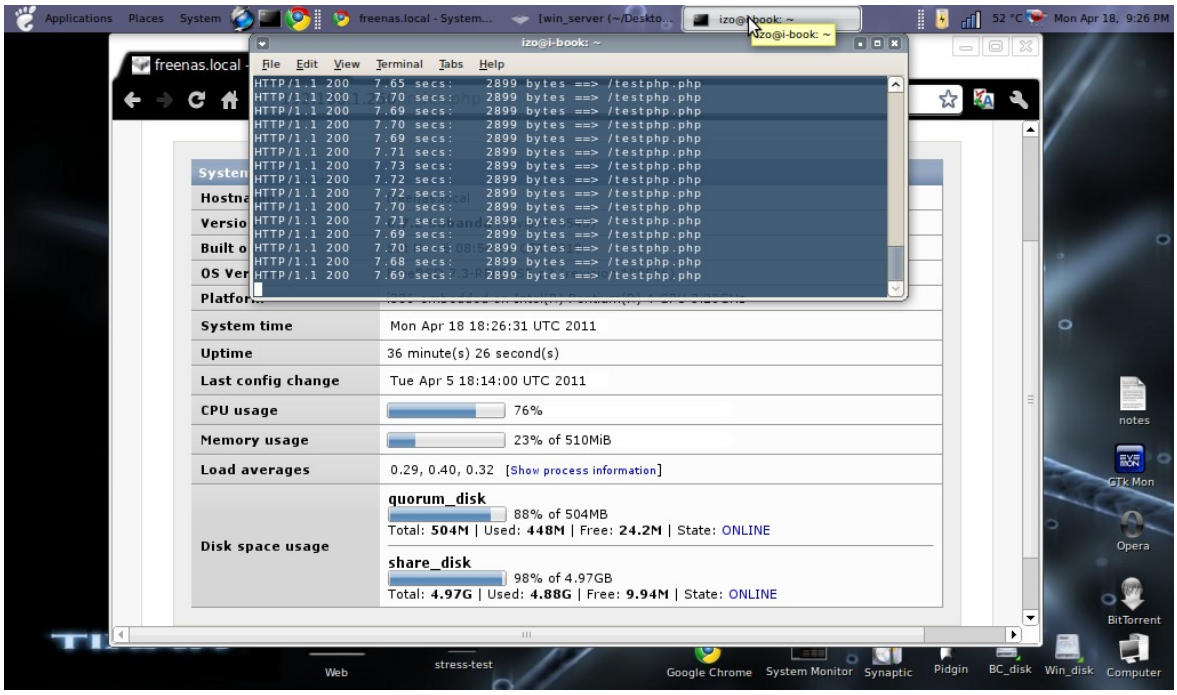
Shortest transaction: 0.88

Εδώ βλέπουμε την χρήση του συστήματος cpu load, network load, memory, κάτω από πίεση 300 clients node1 και κάτω ο node2. Εδώ βλέπουμε να έχουμε μια έκρηξη στην χρήση όλων των πόρων στο 100%. Αυτός είναι και ο λόγος για τον οποίο το έχουμε αρκετές μη πραγματοποιήσιμες συναλλαγές.



Εικόνα 10: «Χρήση πόρων FailOver 300 clients»

Σε αυτό το σημείο θα παραθέσουμε την χρήση των πόρων του FreeNAS ώστε να δείξουμε πώς τα αποτυχημένα αιτήματα δεν έχουν να κάνουν με αυτό. Το CPU usage είναι στο 76% και αυτό είναι στιγμιαία το μέγιστο. Γενικά δηλαδή δεν χρησιμοποιεί πάνω από 40-50%.



Εικόνα 11: «Χρήση πόρων FreeNAS 300 clients»

Σε περίπτωση που πέσει ο πρωτεύων διακομιστής ο δευτερεύων χρειάζεται ~50ms για να αναλάβει πλήρως

<u>Το ένα node μή ενεργό.</u>

izo@i-book:~$ siege -c100 -t2m http://192.168.1.202/testphp.php

Lifting the server siege..	done.
Transactions:	3588 hits
Availability:	100.00 %
Elapsed time:	120.42 secs
Data transferred:	9.92 MB
Response time:	2.75 secs
Transaction rate:	29.80 trans/sec
Throughput:	0.08 MB/sec
Concurrency:	81.92
Successful transactions:	3588

Failed transactions:	0
Longest transaction:	14.19
Shortest transaction:	0.11

izo@i-book:~$ siege -c200 -t2m http://192.168.1.202/testphp.php

Lifting the server siege...	done.
Transactions:	3454 hits
Availability:	100.00 %
Elapsed time:	119.92 secs
Data transferred:	9.55 MB
Response time:	6.27 secs
Transaction rate:	28.80 trans/sec
Throughput:	0.08 MB/sec
Concurrency:	180.62
Successful transactions:	3454
Failed transactions:	0
Longest transaction:	16.46
Shortest transaction:	0.75

izo@i-book:~$ siege -c300 -t2m http://192.168.1.202/testphp.php

siege aborted due to excessive socket failure;

Transactions:	629 hits
Availability:	37.07 %
Elapsed time:	20.44 secs
Data transferred:	1.74 MB
Response time:	6.76 secs
Transaction rate:	30.77 trans/sec
Throughput:	0.09 MB/sec

Concurrency:	208.06
Successful transactions:	629
Failed transactions:	1068
Longest transaction:	11.71
Shortest transaction:	0.75

Τα αποτέλεσμα έχουν διαφορετικές τιμές αλλά σαν συμπέρασμα στην προκειμένη φάση δηλαδή συγκρίνοντας μόνο τα δύο συστήματα βγάζουμε τα ίδια συμπεράσματα με το DRBD και HAST. Ένα στοιχείο όμως όπου δεν μπορούμε να παραβλέψουμε είναι ότι γενικά ο χρόνος απόκρισης είναι πολύ μεγάλος αλλά και τα αποτυχημένα αιτήματα είναι πάρα πολλά.

ΚΕΦΑΛΑΙΟ 4 - Σύγκριση Αποτελεσμάτων – Συμπεράσματα

Όπως βλέπουμε συγκρίνοντας τους πίνακες αλλά και τα γραφήματα θα παρατηρήσουμε πως τα αποτελέσματα δείχνουν αδιαμφισβήτητα υπεροχή του DRBD σε όλα τα σημεία, σαν δεύτερο σίγουρα είναι το HAST και τελευταίο σίγουρα το FailOver των Windows το οποίο έχει και την πιο κακή απόδοση, εάν δούμε προσεκτικά έχει πάρα πολλά αιτήματα τα οποία δεν διεκπεραίωσε. Φυσικά αυτό ίσως να οφείλεται σε μεγάλο ποσοστό στο ότι για τα Windows χρησιμοποιούμε τρία συστήματα. Αυτό φυσικά σε καμία περίπτωση δεν αλλάζει το συμπέρασμα πως είναι το πιο αργό κάτι το οποίο ήταν και εν μέρει αναμενόμενο λόγω του ότι οι δίσκοι είναι δικτυακοί προσθέτοντας έξτρα κίνηση στο δίκτυο. Αυτό ίσως μπορεί να βελτιωθεί χρησιμοποιώντας για το NAS άλλο ζεύγος καρτών δικτύου και άλλο ζεύγος για την επικοινωνία των δύο διακομιστών. Τα Windows βέβαια λόγο του ότι χρησιμοποιούν αρκετούς πόρους μόνο και μόνο για το βασικό σύστημα ίσως εάν τα υλοποιούσαμε σε πραγματικά μηχανήματα τα αποτελέσματα να ήταν διαφορετικά. Ακόμη είναι αρκετά δημοφιλές από τις εταιρίες λόγω της καλής υποστήριξης αλλά και της άριστης τεκμηρίωσης που παρέχει η Microsoft.

Το HAST φαίνεται να ολοκληρώνει με επιτυχία και τα τρία τεστ. Παρόλα αυτά το σύστημα ήταν σε οριακή κατάσταση καθώς 718 περιπτώσεις δεν μπόρεσε να τις διεκπεραιώσει. Ακόμη εάν λάβουμε υπόψη μας πως το load averages είναι στο 64%, αυτό μας οδηγεί στο να συμπεράνουμε πως η απόδοση δεν έχει να κάνει τόσο με τους πόρους αλλά με την απαίτηση για την μεταφορά πληροφοριών μέσα στο δίκτυο. Το HAST λόγο του ότι ανήκει σε ένα λειτουργικό σύστημα το οποίο

είναι αρκετά δημοφιλές αλλά περισσότερο λόγο της κοινότητάς του, στο μέλλον ίσως να δούμε κάτι καλύτερο. Επίσης δεν μετράει αρκετό χρόνο ζωής αλλά ήδη μετράει αρκετούς οπαδούς οπότε η περαιτέρω ανάπτυξή του είναι θέμα χρόνου.

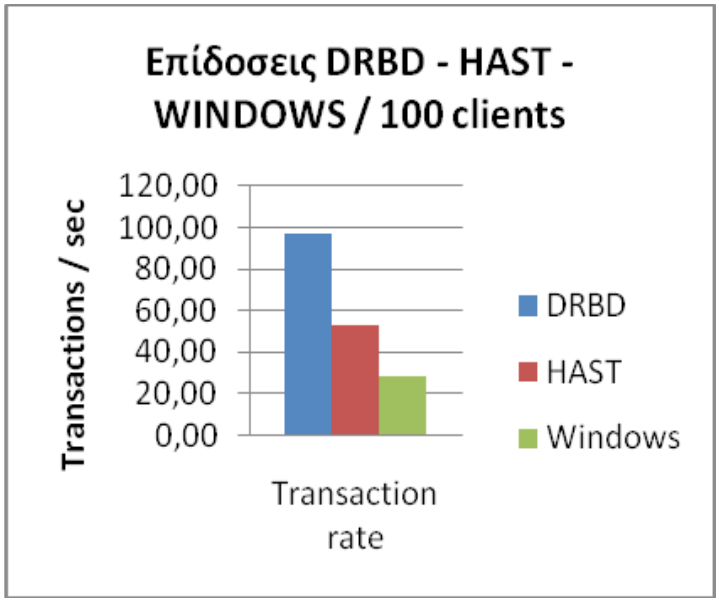
Το DRBD με τις καλύτερες επιδόσεις μπορεί άνετα να χειριστεί 226 συναλλαγές το δευτερόλεπτο ακόμη και σε ένα εικονικό σύστημα με ποσοστό 100% όσο αφορά τις επιτυχείς διεκπεραιώσεις. Το σύστημα μπορούσε να χειριστεί ακόμα περισσότερες αλλά αυτό ήταν αδύνατο να ελεγχθεί καθώς στο σύστημα όπου έτρεχε το siege δεν μπορούσε να προσομοιώσει άλλους clients. Τέλος το ότι μπορεί να στηθεί επάνω σε LINUX συστήματα το κάνει πάρα πολύ ελκυστικό άλλα και λόγου της άριστης τεκμηρίωσης που παρέχεται.

Πίνακας 3: «Επιδόσεις των συστημάτων με 100 clients»

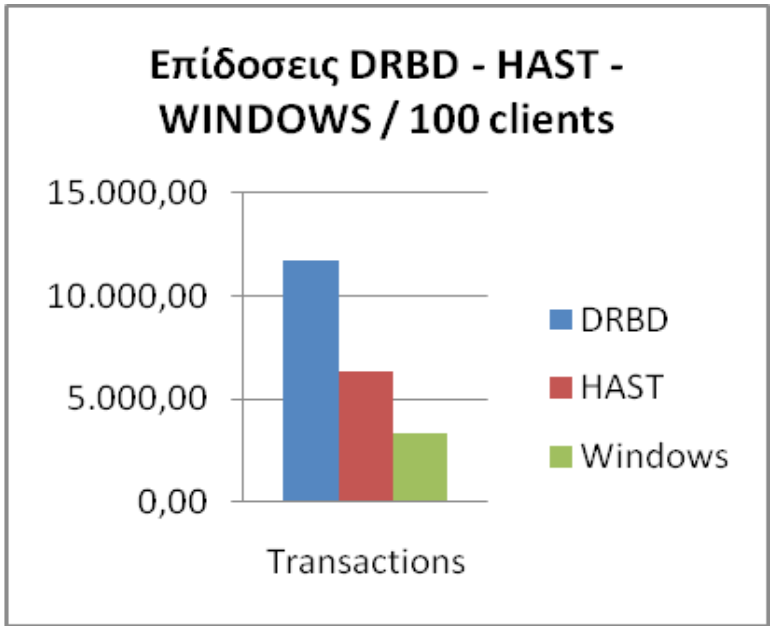
Αριθμός Clients	Παράμετρος	DRBD	HAST	Windows
100	Failed transactions	0,00	0,00	0,00
100	Longest transaction	1,60	13,94	7,86
100	Response time	0,53	0,69	2,98
100	Transaction rate	97,39	52,56	28,09
100	Transactions	11.701,00	6.335,00	3.366,00

Στο παρακάτω διάγραμμα βλέπουμε την απόδοση όσο αφορά τα αιτήματα όπου μπορούν να διεκπεραιώσουν ανά δευτερόλεπτο το DRBD, HAST, Windows κάτω από την πίεση 100 client.

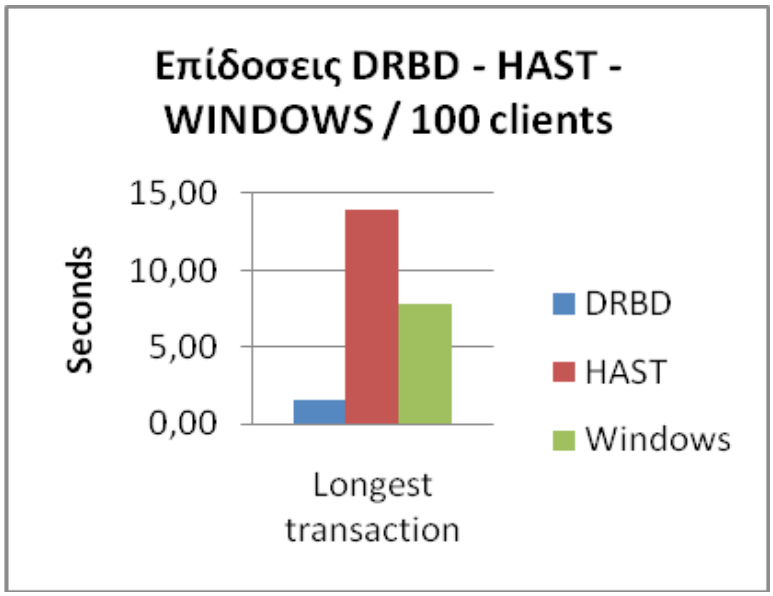
Εικόνα 12: «Διαγράμματα Επιδόσεις των συστημάτων με 100 clients»



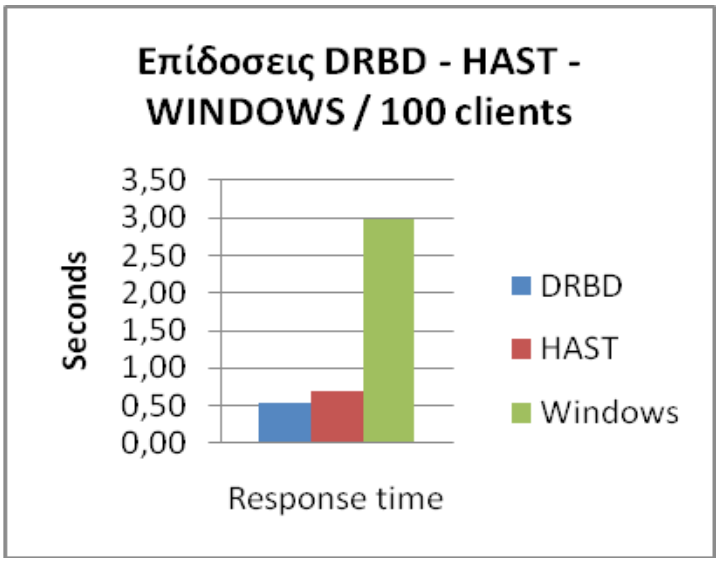
Στο παρακάτω διάγραμμα βλέπουμε τον συνολικό αριθμό αιτημάτων όπου μπορούν να διεκπεραιώσουν σε περίοδο 120” το DRBD, HAST, Windows κάτω από την πίεση 100 client.



Στο παρακάτω διάγραμμα βλέπουμε την μεγαλύτερη καθυστέρηση σε δευτερόλεπτα που υπήρξε σε συναλλαγή στο DRBD, HAST, Windows κάτω από την πίεση 100 client.



Στο παρακάτω διάγραμμα βλέπουμε τον μέσο χρόνο απόκρισης σε δευτερόλεπτα στα αιτήματα στο DRBD, HAST, Windows κάτω από την πίεση 100 client.

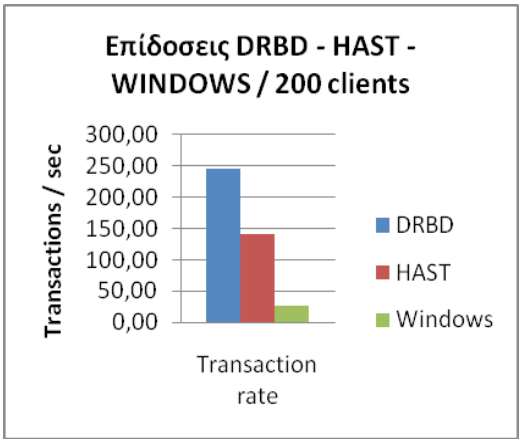


Πίνακας 4: «Επιδόσεις των συστημάτων με 200 clients»

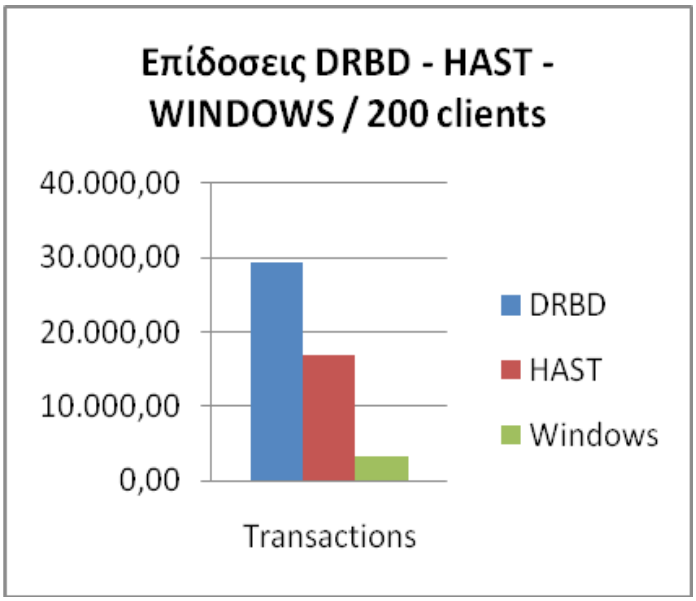
Αριθμός Clients	Παράμετρος	DRBD	HAST	Windows
200	Failed transactions	0,00	111,00	0,00
200	Longest transaction	9,10	24,63	12,72
200	Response time	0,30	0,69	6,55
200	Transaction rate	244,92	140,90	27,56
200	Transactions	29.437,00	16.961,00	3.322,00

Στο παρακάτω διάγραμμα βλέπουμε την απόδοση όσο αφορά τα αιτήματα όπου μπορούν να διεκπεραιώσουν ανά δευτερόλεπτο το DRBD, HAST, Windows κάτω από την πίεση 200 client.

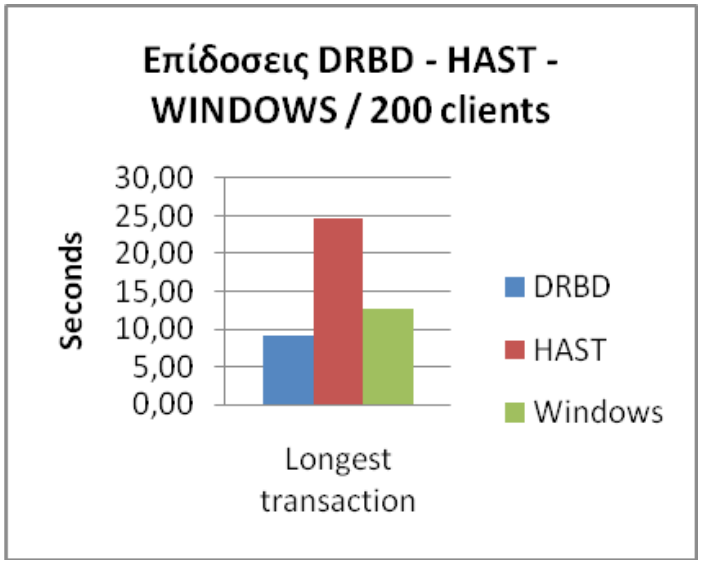
Εικόνα 13: «Διαγράμματα Επιδόσεις των συστημάτων με 200 clients »



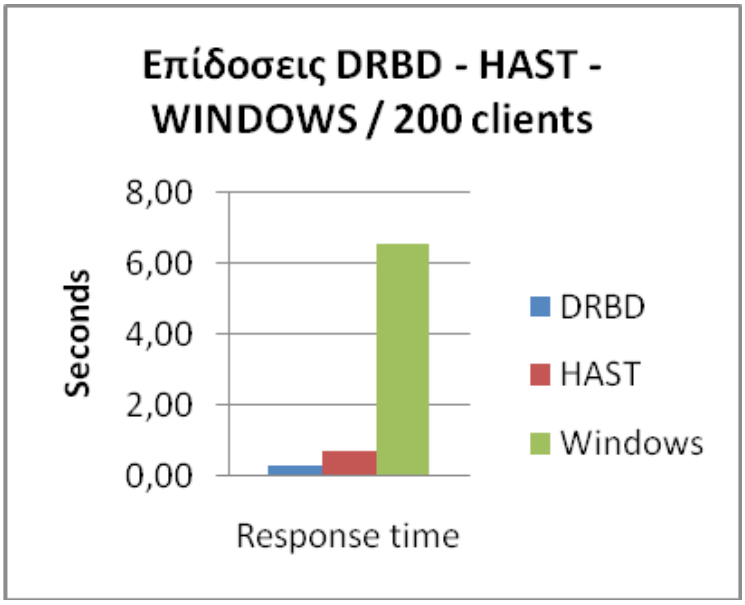
Στο παρακάτω διάγραμμα βλέπουμε τον συνολικό αριθμό αιτημάτων όπου μπορούν να διεκπεραιώσουν σε περίοδο 120” το DRBD, HAST, Windows κάτω από την πίεση 200 client.



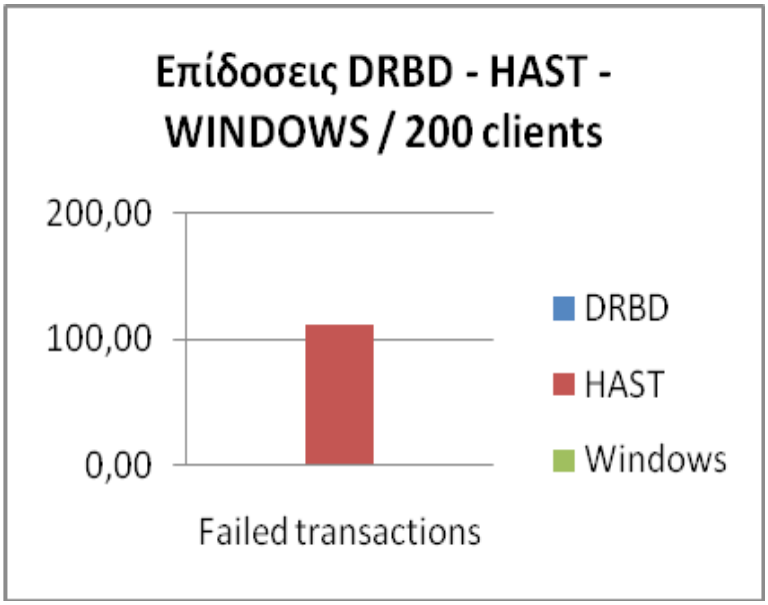
Στο παρακάτω διάγραμμα βλέπουμε την μεγαλύτερη καθυστέρηση σε δευτερόλεπτα που υπήρξε σε συναλλαγή στο DRBD, HAST, Windows κάτω από την πίεση 200 client.



Στο παρακάτω διάγραμμα βλέπουμε τον μέσο χρόνο απόκρισης σε δευτερόλεπτα στα αιτήματα στο DRBD, HAST, Windows κάτω από την πίεση 200 client.



Στο παρακάτω διάγραμμα βλέπουμε τον αριθμό αποτυχημένων αιτημάτων στο DRBD, HAST, Windows κάτω από την πίεση 200 client.

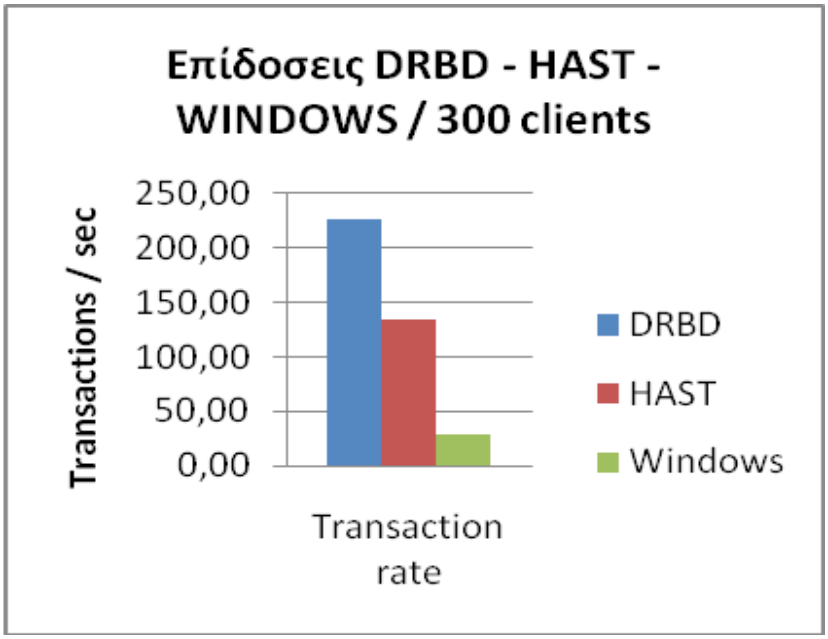


Πίνακας 5: «Επιδόσεις των συστημάτων με 300 clients»

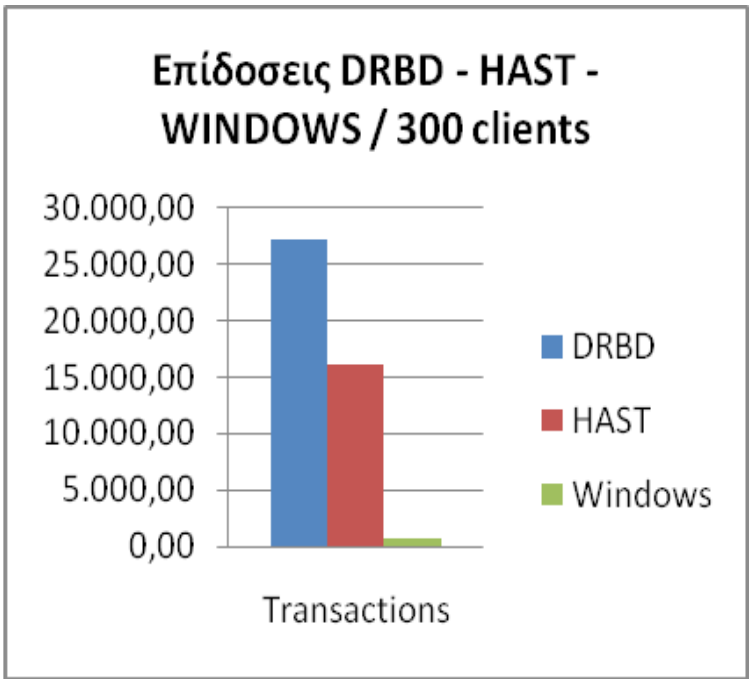
Αριθμός Clients	Παράμετρος	DRBD	HAST	Windows
300	Failed transactions	0,00	718,00	1.061,00
300	Longest transaction	16,02	29,95	10,19
300	Response time	0,80	1,30	7,21
300	Transaction rate	226,82	134,58	29,43
300	Transactions	27.284,00	16.095,00	678,00

Στο παρακάτω διάγραμμα βλέπουμε την απόδοση όσο αφορά τα αιτήματα όπου μπορούν να διεκπεραιώσουν ανά δευτερόλεπτο το DRBD, HAST, Windows κάτω από την πίεση 300 client.

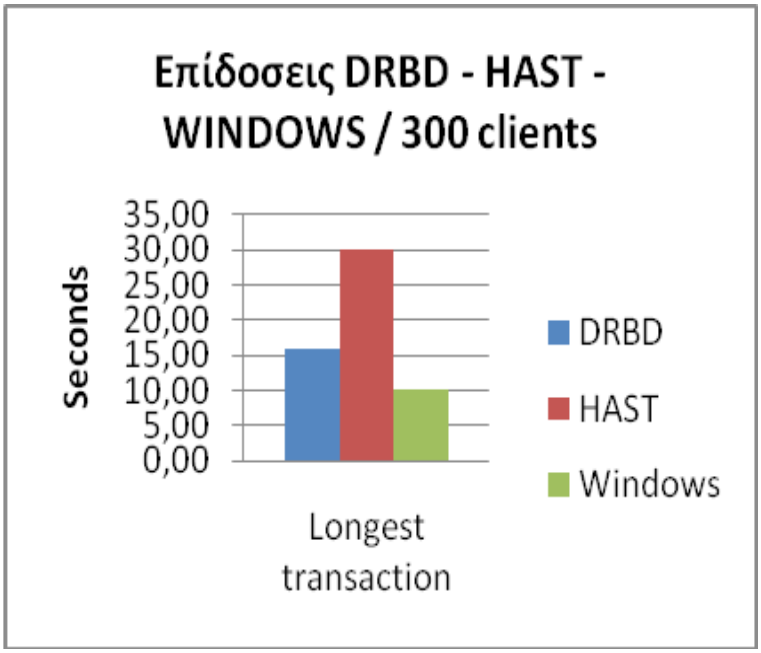
Εικόνα 14: «Διαγράμματα Επιδόσεις των συστημάτων με 300 clients »



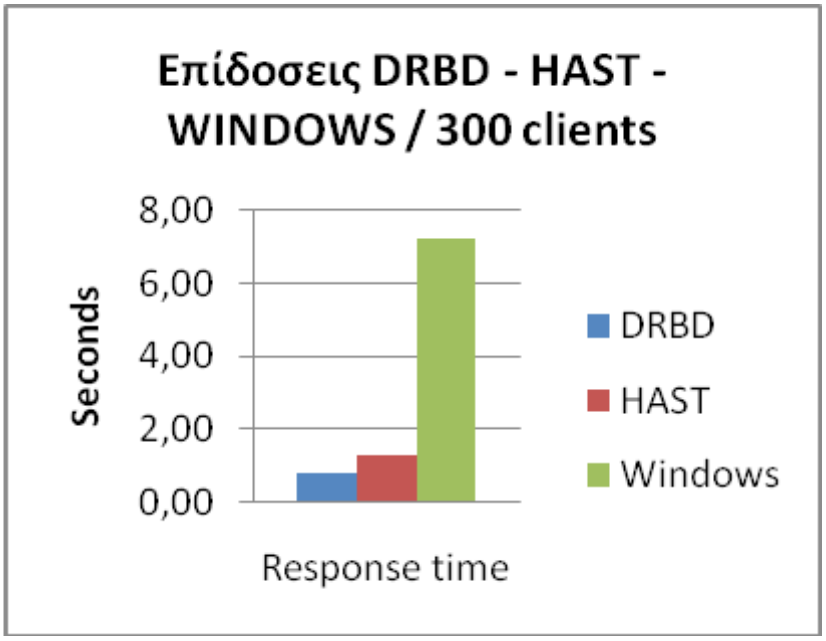
Στο παρακάτω διάγραμμα βλέπουμε τον συνολικό αριθμό αιτημάτων όπου μπορούν να διεκπεραιώσουν σε περίοδο 120” το DRBD, HAST, Windows κάτω από την πίεση 300 client.



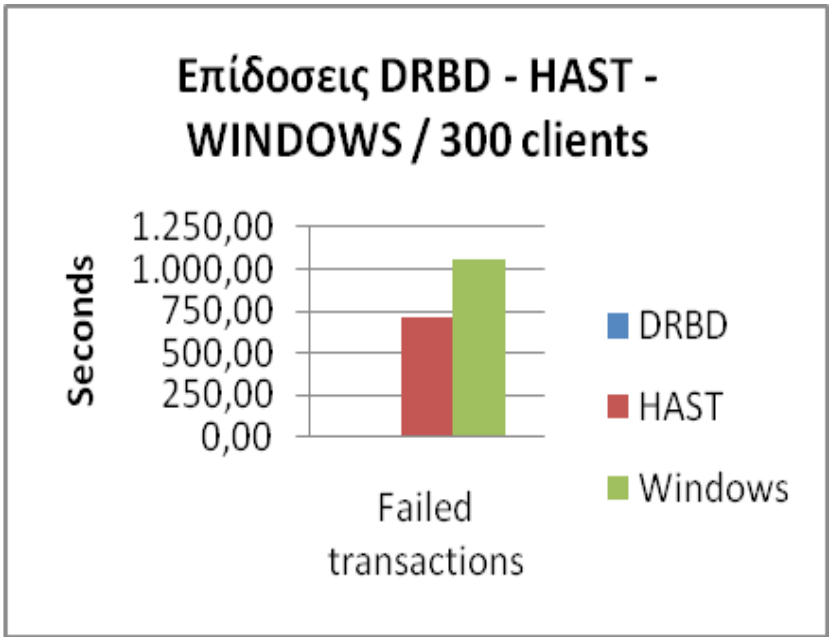
Στο παρακάτω διάγραμμα βλέπουμε την μεγαλύτερη καθυστέρηση σε δευτερόλεπτα που υπήρξε σε συναλλαγή στο DRBD, HAST, Windows κάτω από την πίεση 300 client.



Στο παρακάτω διάγραμμα βλέπουμε τον μέσο χρόνο απόκρισης σε δευτερόλεπτα στα αιτήματα στο DRBD, HAST, Windows κάτω από την πίεση 300 client.



Στο παρακάτω διάγραμμα βλέπουμε τον αριθμό αποτυχημένων αιτημάτων στο DRBD, HAST, Windows κάτω από την πίεση 300 client.



ΕΠΙΛΟΓΟΣ

Στην πτυχιακή αυτή εξετάσαμε εξονυχιστικά τις 3 πιο δημοφιλείς υλοποιήσεις για High Availability. Και οι τρεις μπορούν άνετα να χρησιμοποιηθούν (και ήδη χρησιμοποιούνται) σε συστήματα τα οποία έχουν πολύ μεγάλες απαιτήσεις στην διεκπεραίωση αιτημάτων, εάν λάβουμε υπόψιν μας δε πώς τα εικονικά συστήματα φιλοξενήθηκαν από ένα απλό προσωπικό υπολογιστή .

Σίγουρα το DRBD μαζί με το Heartbeat είναι οι νικητές. Το DRBD έχει χρησιμοποιηθεί από την Mysql παρέχοντας μία σουίτα για HA. Στην αγορά υπάρχουν και άλλες υλοποιήσεις από άλλες εταιρίες όπως στο Solaris της Sun, IBM, Hp η μελέτη των οποίων ξεφεύγει από τα όρια της παρούσας εργασίας. Σίγουρα το μέλλον είναι το HA καθώς όλο και περισσότερες επιχειρήσεις παντός τύπου χρειάζονται τέτοιου είδους υπηρεσίες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Computer Cluster - http://en.wikipedia.org/wiki/Computer_cluster (2010)

[2] VirtualBox - http://www.virtualbox.org (2010)

[3] NAS - http://en.wikipedia.org/wiki/Network-attached_storage (2011)

[4] RAID - http://en.wikipedia.org/wiki/RAID (2011)

[5] Mysql - http://www.mysql.com/ (2010)

[6] PhP - http://www.php.net/ (2010)

[7] Apache - http://www.apache.org (2010)

[8] Web Server - http://en.wikipedia.org/wiki/Web_server (2010)

[9] DRBD - http://www.drbd.org (2010)

[10] HeartBeat - http://www.linux-ha.org/wiki/Main_Page (2010)

[11] SIEGE - http://joedog.org (2011)

[12] HAST - http://wiki.freebsd.org/HAST (2010)

[13] UCARP - http://www.ucarp.org/project/ucarp (2010)

[14] FailOver - http://technet.microsoft.com/en-us/library/ff182326(WS.10).aspx (2011)

[15] freenas - http://freenas.org (2011)