Sistemes de computació d'altes prestacions (Clusters) i programari lliure

Gabriel Verdejo Álvarez - Iván Couto Vivas
Laboratori de Càlcul de LSI (LCLSI), Universitat Politècnica de Catalunya
gabriel@lsi.upc.edu - ivanc@lsi.upc.edu

Maig de 2008

Resum

Aquest document reflecteix l'experiència de més de quatre anys del Laboratori de Càlcul del Departament de Llenguatges i Sistemes Informàtics (LCLSI) dintre de l'àmbit de la computació massiva.

Conforme augmentaven les necessitats de còmput dels diferents grups de recerca del Departament de LSI, el LCLSI ha anat proporcionant diferents solucions per la creació d'aquests sistemes.

El compromís amb el programari lliure de la Universitat juntament amb l'aposta personal del Departament de LSI ha fet que volguem compartir la nostre experiència amb la comunitat dintre d'aquestes Jornades. Actualment i aprofitant la necessitat d'actualitzar els sistemes de computació existents, el LCLSI ha realitzat un gran esforç per tal de posar a disposició del departament les últimes tecnologies en aquest camp.

Aquestes document recull les nostres experiències i conclusions.

1 Introducció

Des de l'any 1965 quan el senyor Gordon Moore [1], cofundador d'Intel, va publicar la seva famosa llei[1], els ordinadors han continuat creixent en potència sense que actualment es vegi un sostre definitiu. Tot i això, el segment de problemes amb grans necessitats de potència de càlcul, criptografia o simulacions per exemple, continua creixent més ràpid del que la llei de Moore pot assolir.

[1] El número de transistors als xips es duplica cada dos anys.

L'augment de la quantitat de dades a processar, imaginem per exemple una gran companyia com Google, i la complexitat de les operacions a realitzar encara poden col·lapsar qualsevol ordinador actual.

Tanmateix la proliferació de les xarxes locals LAN [16] i l'augment del seu rendiment (ara mateix tenim xarxes a gigabit com a gamma domèstica) fa més viable que mai la creació d'agrupacions d'ordinadors (Clusters [6]) per assolir la computació d'aquests problemes.

2 Dels entorns tancats de computació cap a les solucions lliures

Fins fa pocs anys les solucions aportades per les grans empreses com IBM, SUN o HP s'apropaven a la computació massiva mitjançant solucions en entorns homogenis i tancats basats únicament en els seus productes de software i hardware.

L'aparició de sistemes operatius lliures (Linux, BSD...) i la gran quantitat de gent i organismes (generalment universitats i grups de recerca) que van començar a col·laborar en el seu desenvolupament, va propiciar la interoperativitat d'aquests sistemes funcionant en entorns de màquines heterogènies.

A mesura que es consolidà l'ús d'aquests sistemes operatius a la societat, el seu grau de maduresa va començar a fer-los atractius per entorns de producció tant d'universitats com d'empreses. Paral·lelament van anar sorgint les primeres plataformes de desenvolupament de projectes lliures, inicialment liderada per FSF-GNU [11], com Sourceforge [2] i Savannah [3].

D'aquesta manera, com generalment passa amb el programari lliure, una mica per portar les coses al límit i experimentar la distribució de tasques entre diferents ordinadors i una mica per crear una solució no propietària pels ordinadors amb sistemes operatius lliures, van començar els primers projectes de computació massiva. Beowulf [4] és probablement el primer projecte d'aquestes característiques que va aparèixer per Linux.

3 Problemàtica dels Clusters OpenMosix

Tot i que l'aproximació a la computació distribuïda és força complexa i actualment existeixen diferents possibilitats, nosaltres ens centrarem en el sistema existent al Laboratori de Càlcul del Departament de Llenguatges i

Sistemes Informàtics (LCLSI) de la Universitat Politècnica de Catalunya [8].

El LCLSI gestiona actualment 3 Clusters de computació per grups de recerca amb uns 45 ordinadors en total. Actualment aquests Clusters són independents entre sí i funcionen mitjançant OpenMosix [18].

Aquest model de computació es basa en la migració de processos entre els diferents nodes que conformen el Cluster. D'aquesta manera donat un procés iniciat a una màquina es divideix en dues parts:

Part d'execució local: Part de codi resident al node on s'inicia l'execució del procés.

Part d'execució remota: Tros de codi que es mou al node amb menor càrrega de CPU i que s'executa remotament. Durant la seva execució, si hi ha un altre node lliure o amb més potència de CPU disponible, el planificador el pot tornar a migrar.

Els problemes que ens hem trobat amb aquest tipus de sistema són que no funciona adequadament amb molts tipus de programes que fan servir memòria compartida o threads (com per exemple processos JAVA) ja que la migració no és pas trivial i la reubicació del procés al nou node no sempre funcionava bé. En el cas de l'ús de threads dintre dels processos és encara més complexa la migració i fer transparents les crides a sistema (system calls) dels diferents threads.

D'altra banda no hi ha mecanismes de control i seguretat sobre els processos dels usuaris. D'aquesta forma un usuari *despistat* pot crear un gran número de processos que saturin el sistema de Cluster. Cal destacar també que el projecte OpenMosix va deixar de ser mantingut oficialment l'1 de Març del 2008!

Respecte el sistema de fitxers, es va optar inicialment per NFS on una màquina servidora de disc s'encarregava de servir les zones de dades. Aquest esquema aviat va mostrar problemes de rendiment en moments d'alta concurrència de processos amb molta E/S, col·lapsant el node servidor de disc i allargant el temps de resposta per peticions d'E/S fins a fer-lo en determinades circumstàncies gairebé inaccessible.

Per tal de pal·liar els problemes de rendiment també es van fer proves amb l'automounter. D'aquesta forma diferents zones de dades es servien des de diferents nodes i només es "muntaven" sota demanda. Aquest model va millorar el rendiment del sistema original NFS, però tot i ser vàlid, en les nostres proves va demostrar que pateix força de rendiment i escalabilitat

en sistemes amb molts nodes (més de 20) comparat amb altres sistemes de fitxers en xarxa [23] [24] [22].

4 Nou sistema de computació massiva per LCLSI

Dissenyar un nou sistema de clustering pel Departament de LSI i els seus grups de recerca és un procés molt complex, ja que cada grup té requisits específics i variables. Per una altra banda el creixement de nodes de còmput que estem experimentant i que projectem pels propers anys, fa que necessitem una solució robusta i flexible.

D'aquesta manera, per tal d'aprofitar no només les CPUs dels diferents nodes, sinó també els discos locals de les màquines (40 nodes representen molt d'espai de disc tenint en compte que és difícil trobar actualment discos inferiors a 250Gbytes) vam optar per un model que permetés:

1. La creació d'un volum de disc global (compartit) a partir de tots els discos dels diferents nodes.
2. Un sistema de cues que controlés el número de treballs en execució a cada node (aprofitament de processadors multi-core) i establir restriccions de recursos pels usuaris.

Les nostres premisses per ordre d'importància a l'hora de triar projectes candidats tant pel sistema de distribució de tasques entre els nodes com pel sistema de fitxers en xarxa han estat:

- Projectes madurs/estables i amb continuïtat
- Projectes escalables i amb un bon suport dels desenvolupadors
- Projectes consolidats dintre de la comunitat de computació massiva
- Projectes amb bon rendiment

D'aquesta forma, després de dedicar moltes hores a llegir fòrums, examinar característiques i consultar molta bibliografia, vam destriar els següents projectes:

Sistemes de cues:	*SunGrid Engine [13]* *Condor [7]* *OpenPBS [19]*
Sistemes de fitxers:	*Linux Virtual File System [15]* *GlusterFS [12]* *Lustre [17]*

4.1 Selecció del sistema de cues: SunGrid Engine

Per triar el sistema de control de cues vam avaluar tant la part de gestió de la planificació com les eines que proporciona el sistema als usuaris. Tot i que els tres projectes son força coneguts i utilitzats al món dels clusters, SunGrid Engine per la seva facilitat d'ús, flexibilitat de configuració i eines gràfiques per usuaris va ser l'opció triada.

El sistema **Condor** és complex de configurar i fer servir pels usuaris ja que requereix d'una sintaxi pesada per a cada execució. D'altra banda, la gestió tant administrativa com d'usuari es troba menys treballada que al SunGrid Engine.

El projecte **OpenPBS**, tot i que actualment continua el seu desenvolupament com a TORQUE [21], ha deixat de ser un referent capdavanter en la distribució de tasques. A més el seu suport als usuaris no és tant bo com es podria esperar i per tant va ser descartat.

4.2 Selecció del sistema de fitxers: Lustre + DRDB

Per la selecció del sistema de fitxers en xarxa vam premiar l'estabilitat per sobre de tot. Un sistema amb més de quaranta nodes i un centenar d'usuaris ha de proporcionar una estabilitat i uns mecanismes de recuperació totalment eficients i consolidats.

No entrarem a detallar totes les proves realitzades ja que queda fora de l'abast d'aquest document i les proves de rendiment poden ser consultades més endavant. Sí que farem notar que el nostre joc intern de proves constava de 52Gbytes distribuït en fitxers des d'1Kbyte fins als 2Gbytes i més d'un miler de directoris amb mil fitxers cadascun.

El GlusterFS ha demostrat ser una opció interessant i flexible que ens ha proporcionat un rendiment molt raonable en les proves realitzades. També permet molta flexibilitat de configuració i proporciona un model de funcionament força àgil. Malauradament ha presentat dos inconvenients prou greus com per descartar-lo per un sistema en producció.

1. La utilització del mode AFR (*Automatic File replication*) per la sincronització de les parelles de nodes és inestable i de tant en tant en les proves ens hem trobat que el sistema deixava de funcionar.

2. El sistema GlusterFS fa servir la funcionalitat del FUSE [10] per interaccionar amb el sistema de fitxers local de la màquina, així tenim

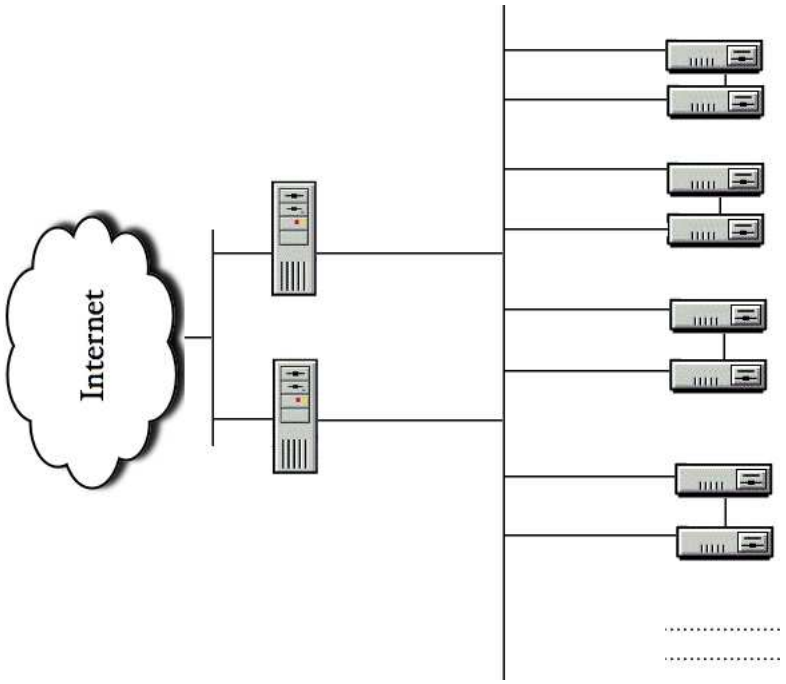

Figura 1: Arquitectura del sistema de Clustering.

que la funcionalitat de bloqueig (POSIX-locks) proporcionada no és total. A més ens hem trobat amb problemes d'estabilitat (el sistema es penja) al fer servir la versió de FUSE optimitzada per la gent de GlusterFS.

D'altra banda, el sistema proporcionat pel LVFS no ens garantia la flexibilitat (creixement, redundància, mecanismes de recuperació) que necessitàvem per el nostre projecte.

El sistema Lustre ha demostrat ser un sistema molt estable amb un projecte força consolidat a la comunitat. Tot i que es troba al "roadmap" per properes funcions, no proporciona cap mecanisme de replicació de la informació. D'aquesta forma vam haver de fer servir el projecte DRDB i el protocol Heart-beat per tal de completar la solució tolerant a fallades.

4.3 Arquitectura del nou sistema

Podem observar a la figura 1 l'esquema de la nova arquitectura que farem servir amb tots els nodes dels clusters de LSI. Cal destacar l'existència de dos nodes d'entrada al sistema i com a més de la connexió per xarxa entre tots els nodes del sistema també els tenim enllaçats per parelles.

1. **Nodes d'entrada:** La configuració de dos equips al sistema d'entrada al cluster permet que el sistema tingui redundància i en cas de problemes el sistema continuï funcionant. Tenint més de 40 nodes no té sentit dependre d'un únic ordinador!

2. **Nodes de còmput:** Aquests nodes tenen un doble paper ja que actuen alhora com nodes de còmput (tots els nodes reben treballs) i com a part del sistema de fitxers en xarxa (nodes agrupats per parelles).

   Per la part del sistema de fitxers en xarxa, també s'ha optat per una configuració redundant on si cau un node, els continguts (dades del usuaris) es mantinguin accessibles. El compromís entre fiabilitat i rendiment és una altre qüestió a discutir (3 nodes?, 4 nodes?... quin és el límit?).

   La sincronització entre els dos nodes de la parella es realitza automàticament per la segona targeta de xarxa dels ordinadors amb el programa DRDB [9]. El node primari (el que es troba actiu) rep les peticions d'escriptura i les propaga al node secundari de la parella. Les parelles de nodes es comuniquen entre sí fent servir el protocol `heart-beat` [14] de forma que si el node primari de la parella cau, el node secundari s'activa i comença a servir les dades.

   Cal notar que el sistema de fitxers en xarxa Lustre controla les peticions que realitzen els processos en execució per tal que en cas de que no estiguin disponibles les dades els processos no abortin. Una vegada activat com a primari el node secundari de la parella, els processos continuen la seva execució sense problemes.

5 Resultats

Seguint l'ànim divulgatiu d'aquest document no entrarem a detallar exhaustivament totes les proves realitzades. Aquest treball ha estat un procés de més de cinc mesos i deu fulles no poden recollir totes les experiències, alegries i decepcions que ens hem trobat. Per una altre banda són totalment accessibles les nostres adreços de correu així com la WWW on es penjarà el PFC de l'Iván Couto. Sempre podeu contactar amb nosaltres.

Tanmateix, sí que farem cinc cèntims de la part del rendiment obtingut dintre del nostre sistema de proves. Pel nostre entorn de treball hem fet servir un mini-cluster amb 9 servidors Dell Poweredge 860 Quad core amb 8 Gbytes de RAM, dos discos interns SATA de 250Gbytes configurats en

RAID 1 [20] per software i una xarxa Gigabit amb un switch dedicat 3Com.

Aquest equipament ha estat dividit en:

- **Nodes master o Nodes d'entrada al cluster:** Dos nodes configurats com *master-slave* encarregats de portar el control del SunGrid, Lustre i serveis auxiliars (dhcp, Rembo, ntp...). D'aquesta manera en cas de problemes amb un dels nodes, l'altre manté el funcionament del sistema.
- **Nodes de computació:** Set nodes encarregats de l'execució de tasques assignades pel SunGrid i servidores de disc agrupades en parelles. D'aquesta manera tenim 3 servidors de disc (node1-node2, node3-node4, node5-node6) i un client que no serveix disc (node 7).

Per realitzar les proves de rendiment de disc vam fer servir un joc de proves propi dissenyat específicament i diferents programes de test de disc que es poden trobar al Linux test Tools [25].

La part del joc de proves dissenyat consta d'un total de 52Gbytes distribuits en 4096 directoris amb 1024 directoris de profunditat. Cadascun d'aquests directoris inclou 1024 arxius d'1KByte o 1024 arxius d' 1MByte o 1024 arxius de 4Mbytes o 1024 arxius de 10Mbytes o 1024 arxius de 100Mbytes o 1024 arxius de 500MBytes o 4 arxius de 2GBytes.

Per aquest document i per tal que els resultats siguim comparables amb altres sistemes hem fet servir un dels test més reconeguts al món del rendiment de discos i arrays, el Bonnie++ [5].

La prova executada per obtenir les gràfiques de les figures 2 i 3 ha estat la creació d'un fitxer de 16Gbytes per les proves de rendiment seqüencial i la creació de 1024 directoris amb 1024 fitxers de 4Mbytes cadascun:

bonnie++ -s 16384 -n 1:4096:4096:1024 -d [directori] -u [usuari]

A les figures 2 i 3 podem observar els resultats gràfics de les proves realitzades. A continuació adjuntem les taules amb els resultats en Kbytes/segon obtinguts amb les proves seqüencials amb un fitxer de 16Gbytes (fig. 2):

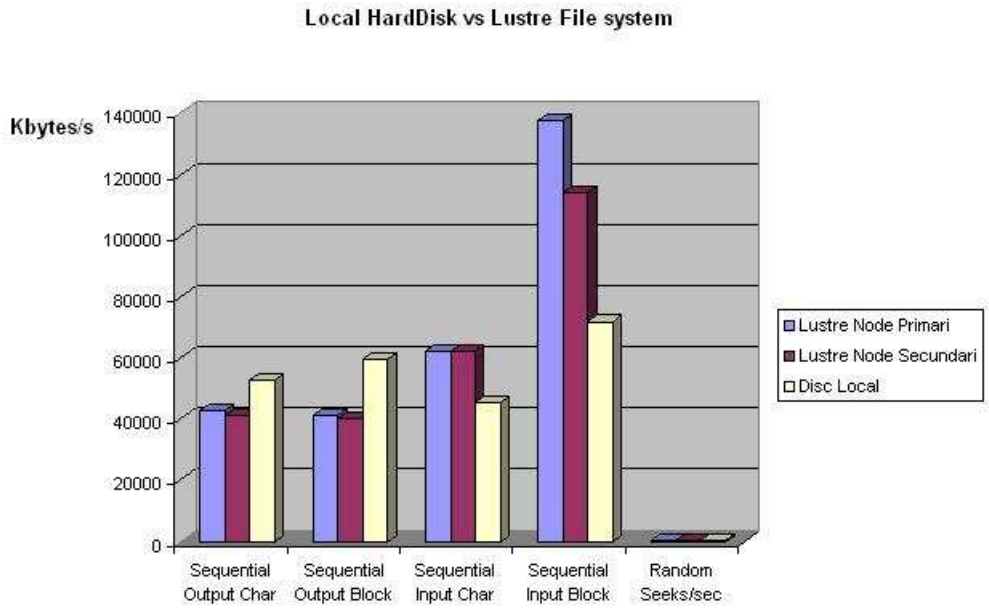

Figura 2: Rendiment de disc seqüencial.

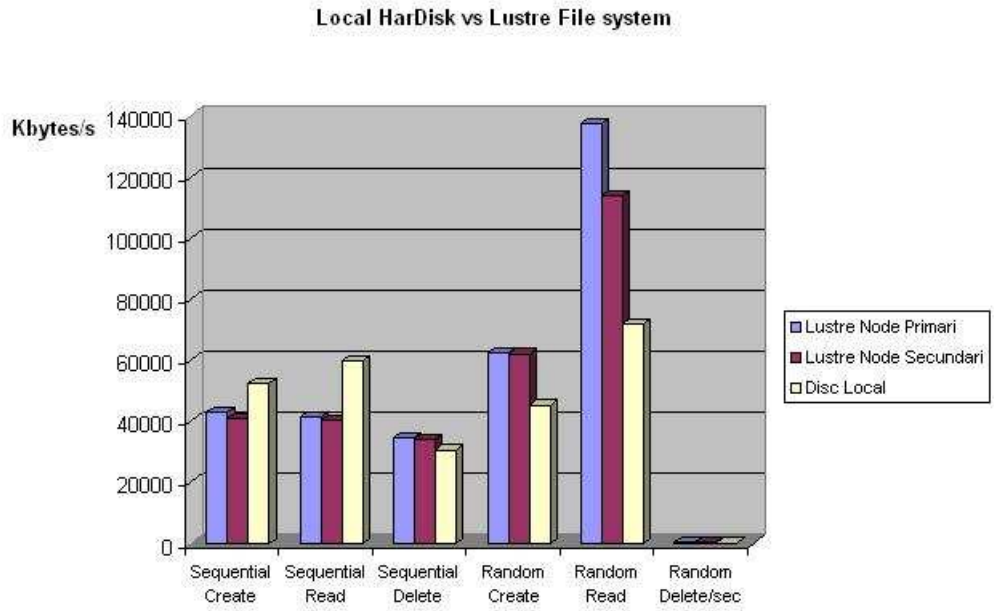

Figura 3: Rendiment de disc d'accés aleatori.

	Lustre Primari	Lustre Secundari	Disc Local
Sequential Output Char	42810	41026	52488
Sequential Output Block	41269	40328	59717
Sequential Input Char	62226	62165	45357
Sequential Input Block	137354	113770	71840
Random Seeks/sec	348	358	99

I la taula amb les proves realitzades amb fitxers i directoris (fig. 3):

	Lustre Primari	Lustre Secundari	Disc Local
Sequential Create	42810	41026	52488
Sequential Read	41269	40328	59717
Sequential Delete	34457	34189	30685
Random Create	62226	62165	45357
Random Read	137354	113770	71840
Random Delete/sec	348	358	99

Cal fer notar que "només"treballant amb 3 parelles de nodes ja observem una millora clara del rendiment del sistema, especialment en peticions aleatòries (random). Aquesta tendència es manté i escala perfectament conforme augmenta el número de parelles de nodes que serveixin més disc. D'aquesta manera observem que per sistemes multi-usuari amb diferents processos l'ús d'un sistema de fitxers en xarxa és sempre una millora de rendiment per tot el cluster.

Aquest treball és el Projecte Final de Carrera (PFC) de Enginyeria Superior en Informàtica de l'Iván Couto Vivas que es presentarà al Juliol de 2008. Tota aquesta documentació i el PFC resultant es pot trobar a:

http://gabriel.verdejo.alvarez.googlepages.com/cluster

6 Conclusions

La creació d'un sistema de computació massiva amb programari lliure és actualment una realitat. L'existència de diferents projectes i paradigmes (Condor, SunGrid, Beowulf) per distribuir la càrrega entre els diferents nodes fa viable l'ús de projectes lliures en entorns de producció.

Respecte a la part del sistema de fitxers en xarxa, els resultats han estat pitjors dels esperats. Els projectes que actualment aporten nous models distribuïts com el GlusterFS queden encara lluny de poder ser instal·lats en

sistemes reals en producció. Projectes com el Lustre que han passat a ser lliures fa relativament poc, poden començar a ser alternatives vàlides en lloc dels clàssics sistemes com NFS.

7 Llicència

Referències

[1] Moore's law. http://www.intel.com/technology/mooreslaw/index.htm, 1965.

[2] http://sourceforge.net/, 2008.

[3] http://savannah.nongnu.org/, 2008.

[4] Beowulf. http://www.beowulf.org, 2008.

[5] Bonnie++. http://www.coker.com.au/bonnie++/, 2008.

[6] Computer cluster. http://en.wikipedia.org/wiki/Computer_cluster, 2008.

[7] Condor. http://www.cs.wisc.edu/condor/, 2008.

[8] Departament de lsi. http://www.lsi.upc.edu, 2008.

[9] Drbd. http://www.drbd.org/, 2008.

[10] Filesystem in userspace. http://fuse.sourceforge.net/, 2008.

[11] Free software foundation. http://www.fsf.org/, 2008.

[12] Glusterfs. http://www.gluster.org/docs/index.php/GlusterFS, 2008.

[13] Gridengine. http://gridengine.sunsource.net/, 2008.

[14] Heartbeat: High availability. http://www.linux-ha.org/Heartbeat, 2008.

[15] Linux virtual file system. http://www.coda.cs.cmu.edu/doc/talks/linuxvfs/, 2008.

[16] Local area network. http://en.wikipedia.org/wiki/Local_area_network, 2008.

[17] Lustre. http://www.lustre.org/, 2008.

[18] The openmosix project. http://openmosix.sourceforge.net, 2008.

[19] Pbs gridworks. http://www.openpbs.org/, 2008.

[20] Redundant array of inexpensive drives. http://en.wikipedia.org/wiki/RAID, 2008.

[21] Torque resource manager. http://www.clusterresources.com/pages/products/torque-resource-manager.php, 2008.

[22] Christopher M. Boumenot. The performance of a linux nfs implementation. http://www.wpi.edu/Pubs/ETD/Available/etd-0523102-121726/unrestricted/boumenot.pdf, 2002.

[23] Michael Ewan. Exploring clustered parallel file systems and object storage. http://softwarecommunity.intel.com/articles/eng/2640.htm, 2008.

[24] Brice Goglin and Loïc Prylli. Performance analysis of remote file system access over high bandwidth local network. http://lara.inist.fr/bitstream/2332/838/1/RR2003-22.pdf, 2003.

[25] Jeff Martin. Linux test tools. http://ltp.sourceforge.net/tooltable.php, 2008.