ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/XXXXX/XX
26 settembre 2006

CCR-08/2006/P

TECNOLOGIA iSCSI – PRIME ESPERIENZE

Alessandro Tirel[1]

[1)]INFN-Sezione di Trieste c/o Area di Ricerca, loc. Padriciano, 99, I-34012 Trieste, Italy

Abstract

Questo documento descrive le prove sulla tecnologia iSCSI effettuate presso la Sezione di Trieste. Non tutti gli aspetti del protocollo sono stati testati, sono state valutate in particolare le performance di soluzioni basate su dispositivi hardware e software. Questa attività è stata promossa dal gruppo di lavoro Storage su mandato della Commissione Calcolo e Reti dell'INFN.

*Published by **SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

CONTENUTI

1 INTRODUZIONE

La Commissione Calcolo e Reti dell'INFN ha istituito un gruppo di lavoro, chiamato Gruppo Storage, le cui finalità principali sono:

- la sperimentazione di nuove soluzioni hardware/software
- la valutazione dei costi/benefici di particolari soluzioni storage
- l'implementazione di Cluster File System,
- studio degli aspetti derivanti dall'impiego in ambiente Grid di middleware dedicato allo storage

Alla fine del 2005 si è ritenuto interessante iniziare ad esplorare il mondo iSCSI, poiché in molte Sedi dell'INFN si sente l'esigenza di riorganizzare le risorse di storage attraverso delle Storage Area Network ma i costi elevati della tecnologia Fibre Channel rappresentano un ostacolo.

2 IL PROTOCOLLO iSCSI

Il protocollo Internet SCSI (iSCSI), divenuto standard IETF nel febbraio 2003, si propone come alternativa economica al protocollo Fibre Channel del quale eredita molte caratteristiche. Entrambi, infatti, sono utilizzati per incapsulare dati e comandi SCSI; ciò rende possibile il dialogo tra il sistema operativo (initiator) ed un dispositivo di storage, tipicamente disco o nastro (target), attraverso l'infrastruttura di rete, che nel caso del protollo iSCSI, è costituita dalla rete LAN. Poiché il trasporto è fornito dal protocollo TCP/IP, questo permette di veicolare i dati e comandi SCSI anche in ambito WAN, ovviamente, in questo caso bisognerà considerare i tempi di latenza.

Il protocollo prevede diverse metodologie per assicurare l'integrità dei dati ed impedire l'accesso non autorizzato alle informazioni, nel primo caso sono utilizzati degli algoritmi di checksum di tipo CRC-32 che possono essere applicati sia all'header del pacchetto iSCSI (PDU, Protocol Data Unit) sia alla parte dati, mentre nel secondo caso, i seguenti protocolli di autenticazione permettono di limitare l'accesso alle risorse: CHAP (Challenge Handshake Authentication Protocol), SRP (Secure Remote Password), Kerberos 5 e SPKM-1/SPKM-2 (Simple Public Key Mechanism). Nel caso in cui il traffico dati attraversi reti non sicure è possibile utilizzare le funzionalità IPSec per cifrare il flusso.

Il protocollo definisce due tipi di notazioni per definire i nomi da attribuire ai nodi iSCSI, che sono: eui (extended unique identifier) ed iqn (iSCSI qualified name). Entrambe le forme prevedono che i nomi attribuiti siano unici ed indipendenti dalla locazione fisica, ecco due esempi: eui.02004567A425678D ed iqn.2001-04.com.example:storage.se.disk1. Mentre il primo ricorda la forma del WWNN (World Wide Node Name) tipico del protocollo Fibre Channel il secondo è, come era logico aspettarsi, molto simile ad un nome a dominio, la cui notazione quindi può essere così schematizzata:

iqn.<date>.<reversed FQDN>[:<iscsi unique string>]

Dove reversed FQDN è la notazione inversa del nome a dominio dell'host, la data rappresenta l'anno ed il mese di assegnazione da parte della naming authority del root name all'organizzazione proprietaria dell'host, ad esempio nel caso dell"INFN si potrà avere un nome del tipo: iqn.1996-02.it.infn.ts.iscsi-i0. La parte opzionale dopo i due punti rappresenta un metodo per evitare duplicati dello stesso nome all'interno di un'organizzazione complessa.

Le funzionalità di failover rendono il protocollo adatto anche per installazioni clusterizzate e dove sia richiesta un'alta affidabilità.

3 TESTBED

L'ambiente di test è stato costruito utilizzando del materiale presente in Sezione, nella figura seguente è rappresentato schematicamente.

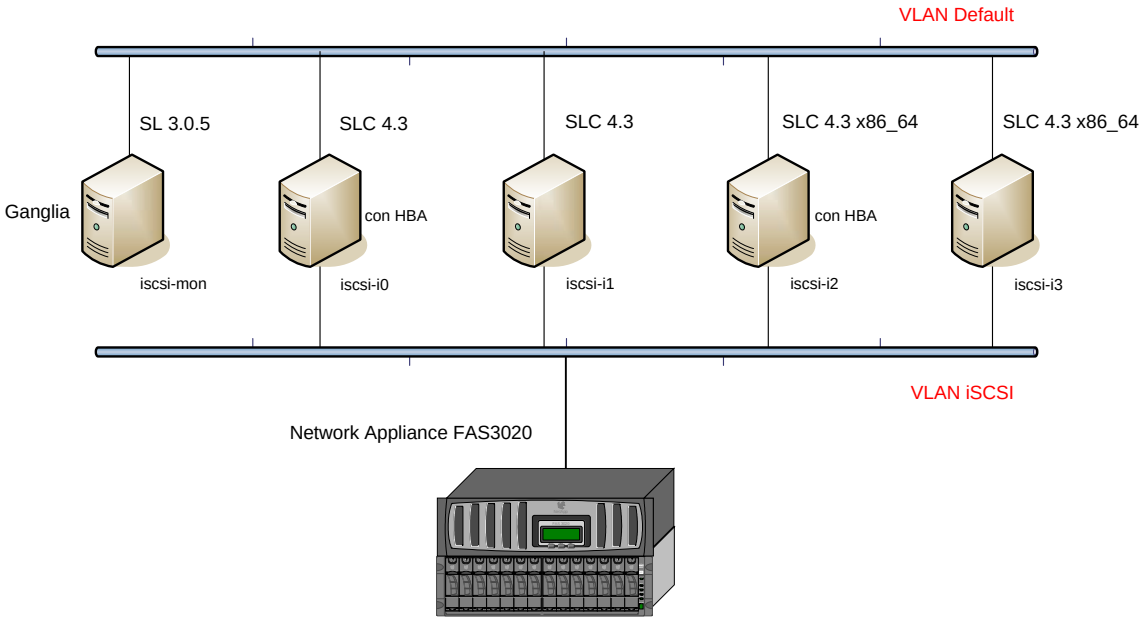

FIG. 1: Testbed

L'architettura prevede quattro nodi per i test che fungono da initiator ed uno che monitorizza il sistema e raccoglie dei dati statistici per verificare quelli prodotti dai test. Tutti i nodi sono dei server 1U Tyan configurati nel seguente modo:

- Scheda madre S2882
- 2 Dual Core AMD Opteron(tm) Processor 275 2.2 Ghz
- 4 GB DDR 400 Reg. ECC
- Hard Disk SATA Seagate 80 GB ST380817AS

- 2 Ethernet Broadcom Corporation NetXtreme BCM5704

Il sistema operativo utilizzato sugli initiator è Scientific Linux Cern 4.3 con kernel 2.6.9-34.0.2 sia a 32 che 64 bit. Due dei nodi initiator sono stati dotati di HBA iSCSI Qlogic e più precisamente la QLA4052C e la QLA4050C.

3.1 iSCSI Target

Il dispositivo iSCSI target è rappresentato dal FAS3020C, fornito in prova da Network Appliance S.r.l.. La macchina in questione era cosi configurata:

- Processore Xeon con Hyper Threading
- 2 GB RAM
- 512 NVRAM
- JBOD con 14 Hard Disk ATA Maxtor 250 GB
- 4 Gigabit Ethernet
- 4 Fibre Channel 2 Gb/s

Due delle quattro porte FC sono utilizzate per collegare i dischi, mentre le restanti due possono essere utilizzate per un'eventuale connessione ad una SAN o direttamente a dei server. Il sistema prevede come sistema di protezione dei dati il solo livello RAID-4 con singola o doppia parità. Il FAS offre molte funzionalità che sono attivate tramite codici di licenza forniti dal costruttore, il modello in prova offriva la possibilità di utilizzare il protocollo iSCSI, l'NFS, il CIFS ed il Fibre Channel per la connessione ad una SAN. Tra le peculiarità del sistema relative alla gestione dei volumi sono da segnalare gli snapshot periodici, l'espansione dinamica e le quote basate sullo userid. La gestione può essere effettuata tramite l'interfaccia web oppure con accesso ssh/telnet, nel primo caso sono previsti dei wizard che ne facilitano la configurazione.

3.2 iSCSI Area Network (iSAN)

La Storage Area Network in ambiente iSCSI viene definita iSAN, come già accennato essa non è altro che una normale LAN su cui vengono veicolati i dati e comandi SCSI. Il frame Ethernet che ne risulta può essere così schematizzato:

Ethernet Header	IP Header	TCP Header	iSCSI Header	iSCSI Data	*Ethernet Trailer*
14 bytes	*20 bytes*	*20 bytes*	*48 bytes*	*variabile*	*4 bytes*

In linea di principio l'impiego di switch ethernet gigabit con la possibilità di utilizzare MTU più grandi di 1500 bytes (jumbo frame) è da preferire, inoltre, la separazione del traffico IP 'normale' da quello iSCSI è fortemente consigliata, essa può avvenire tramite strutture fisicamente separate o più semplicemente mediante l'utilizzo

delle VLAN. Per il testbed è stato utilizzato l'Extreme Networks Summit400 uno switch di classe stackable con 48 porte gigabit ethernet e possibilità di uplink a 10GbE. Come si può vedere dalla fig. 1 è stata definita una VLAN denominata iSCSI che comprende le connessioni del target e degli initiator, siano essi dotati di HBA o di semplice interfaccia gigabit ethernet.

3.3 Monitoring

Una delle finalità del test era rappresentata dalla valutazione del carico sulla CPU quando si utilizza una normale interfaccia ethernet con l'initiator iscsi software. Questo test è importante per definire la fattibilità di una soluzione iSCSI+GPFS in installazione complesse come le farm di calcolo ove l'impiego massivo di HBA iSCSI potrebbe rivelarsi economicamente svantaggioso. Per ottenere dei dati statistici da mettere in relazione con quelli forniti dai benchmark è stato installato sull'host iscsi-mon il software Ganglia[1] molto diffuso nell'ambiente scientifico.

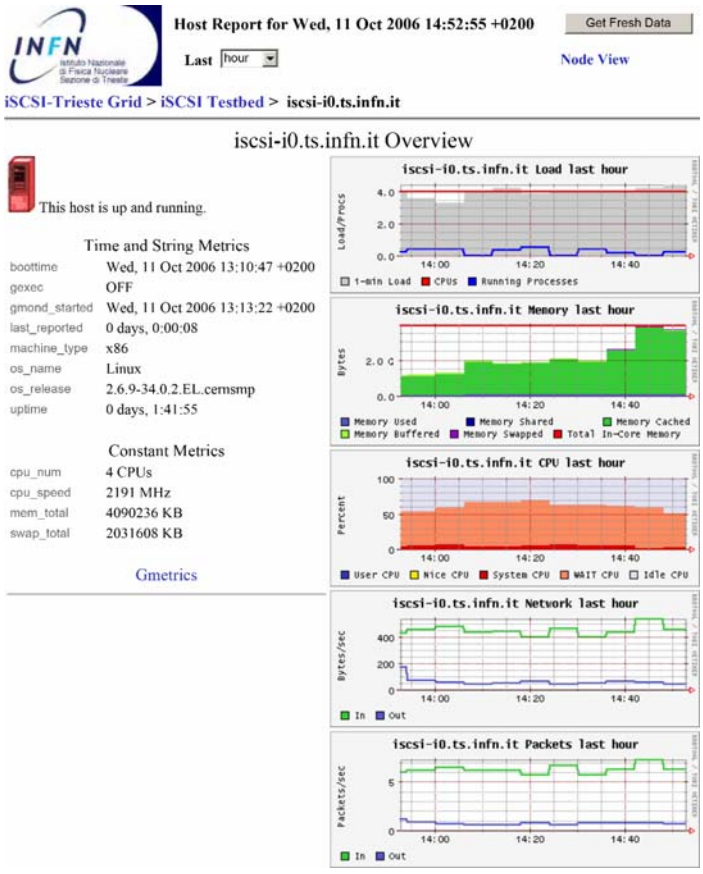

FIG. 2: Ganglia

[1] http://ganglia.info

Per ottenere un numero maggiore di parametri di sistema ed in particolare quelli relativi all'I/O verso i dispositivi di storage è stato installato sugli initiator il pacchetto Performance Co-Pilot[1]. Nell'appendice A è riportato lo script che utilizzando gmetric manda i dati al server Ganglia. Inoltre per facilitare la generazione dei grafici relativi ai test si è utilizzato drraw[2] che estrapola i dati dal database di Ganglia e può generare grafici singoli o multipli (dashboard).

3.4 Benchmark

La scelta di un benchmark con cui valutare le performance di un apparato di storage rappresenta un problema piuttosto serio, infatti, al contrario di quelli utilizzati per le CPU (es. SPECint, SPECfp, Linpak,..) che sono diventati degli standard, in ambito storage la scelta non è così ovvia. Ci sono molti programmi utilizzati per testare questo tipo di soluzioni, tuttavia la maggior parte di loro misura le performance dei filesystems, mentre in questo caso è necessario individuare un benchmark orientato al trasferimento in modalità a blocchi. Un primo candidato è stato iometer prodotto da Intel ora rilasciato su licenza Intel Open Source License, molto utilizzato dai vari vendor era esclusivamente utilizzabile in ambiente Microsoft Windows; ora, dopo il rilascio, è stato effettuato il porting per la piattaforma Linux, purtroppo dalle prime prove effettuate i risultati non si sono rivelati affidabili. Quindi la scelta è caduta su disktest che fa parte del pacchetto Linux Test Project il cui scopo è quello di creare una suite di test per certificare i sistemi Linux, la versione utilizzata è la 1.2.8.

4 CONFIGURAZIONE TESTBED

Per prima cosa si è provveduto a configurare sullo switch ethernet la VLAN iSCSI, relativa alla rete privata ip 192.168.1.0/24, alla quale sono state assegnate alcune porte gigabit, quindi su ogni initiator è stata configurato, sull'interfaccia ethernet o sull'HBA iSCSI, il protocollo ip e l'initiator name nella forma **iqn.1996-02.it.infn.ts.initiator*x*** (dove x=0…3). Quindi è stata la volta del FAS3020 sul quale è stato creato un unico aggregato (raid array) composto di dodici dischi (dieci dischi dati + due di parità), i rimanenti due sono stati definiti come spare; all'interno dell'aggregato sono stati creati quattro volumi di uguale dimensione (circa 300 GB). Affinché gli initiator possano accedere ai volumi è necessario definire un initiator group che ne contiene i nomi. Dal punto di vista della connessione in rete è possibile costruire delle interfacce virtuali sommando le interfacce fisiche, in questo caso, due interfacce fisiche gigabit ethernet concorrono a formare l'interfaccia virtuale iscsi, che utilizza un algoritmo di bilanciamento

[1] http://oss.sgi.com/projects/pcp/

[2] Draw Round Robin Archives on the Web - http://web.taranis.org/drraw/

del traffico basato sull'indirizzo ip sorgente. Quindi per rendere visibili agli initiator i volumi definiti, si è provveduto ad associare ad essi delle LUN.

Nella configurazione del protocollo iSCSI sugli initiator si è preferito mantenere i parametri di default limitandosi alla sola definizione del Discovery Address ovvero l'indirizzo ip del target a cui l'initiator richiede l'elenco delle risorse disponibili.

5 ESECUZIONE DEI TEST

Per eseguire i test è stato utilizzato il comando disktest nella seguente forma:

disktest -K*n* -IB -PT -h3 -T2h -w -pl *device* -B *blocksize*

dove:

-K	definisce il numero *n* di thread
-IB	definisce la modalità di Block I/O
-PT	mostra le statistiche di performance
-h3	mostra le statistiche ogni 3 secondi
-T2h	esegue il test per 2 ore
-w	esegue un test di scrittura (-r = lettura)
-pl	esegue le operazioni *sequenzialmente*
device	device utilizzato per il test (es: /dev/sdd)
-B *blocksize*	specifica la grandezza del blocco da trasferire (in bytes)

Il primo test è stato quello di verificare il carico sulla CPU (cpu system) degli initiator, in particolare la differenza tra quelli forniti di HBA e quelli senza. I seguenti grafici mettono in relazione i risultati ottenuti facendo delle operazioni di lettura e scrittura sequenziali con blocchi di 64 KB e MTU a 1500 bytes e 9000 bytes.

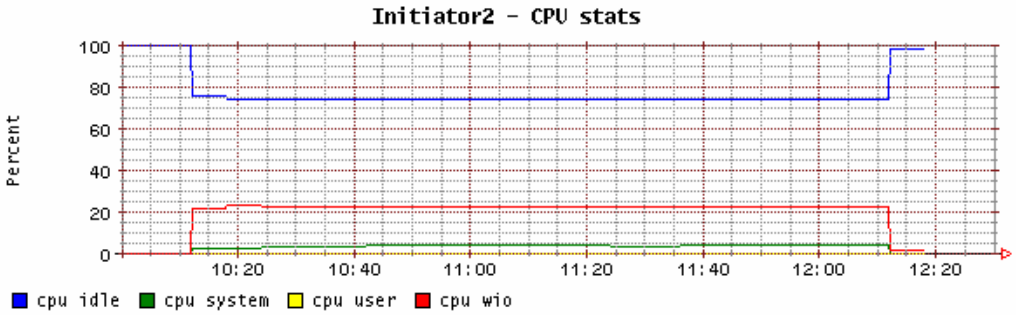

FIG. 3: HBA/Read/MTU 1500

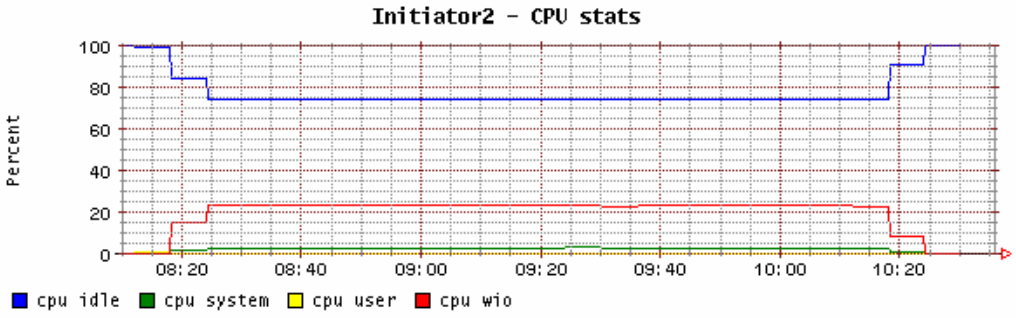

FIG. 4: HBA/Read/MTU 9000

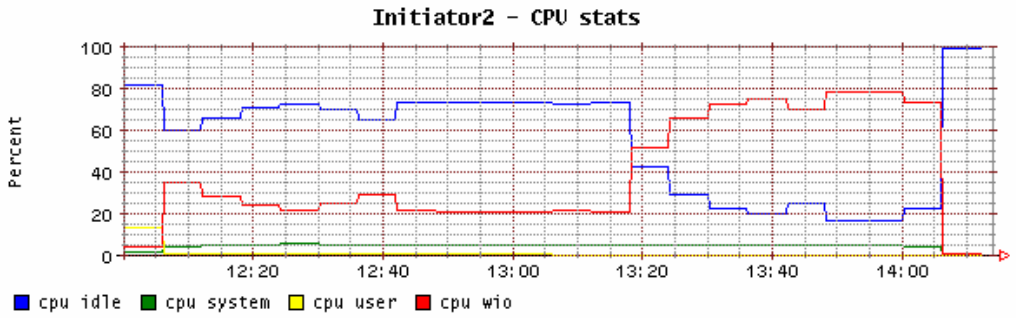

FIG. 5: HBA/Write/MTU 1500

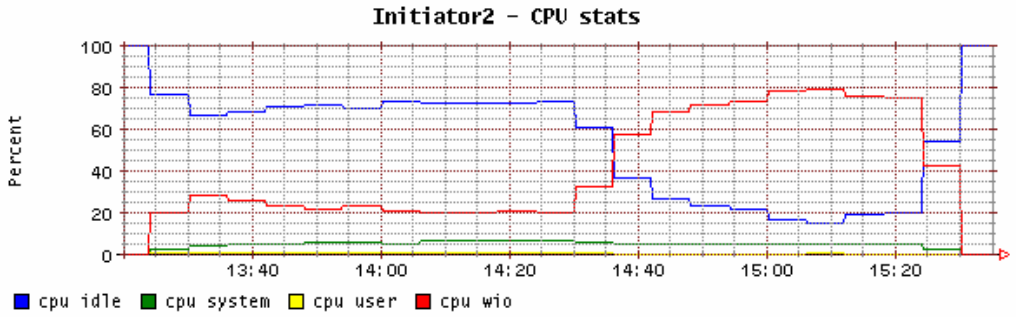

FIG. 6: HBA/Write/MTU 9000

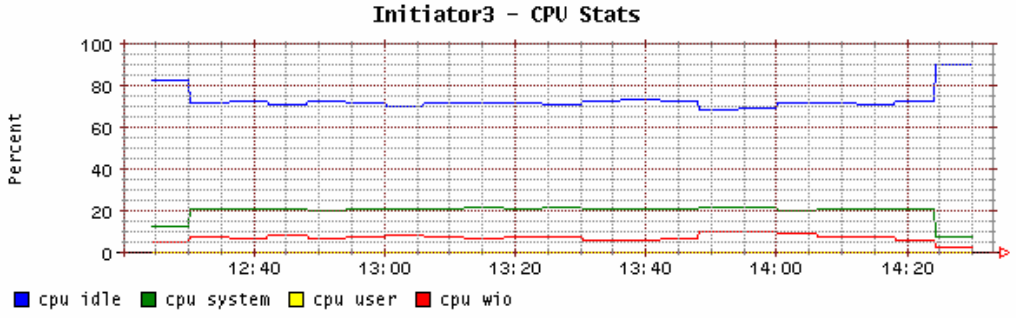

FIG. 7: no HBA/Read/MTU 1500

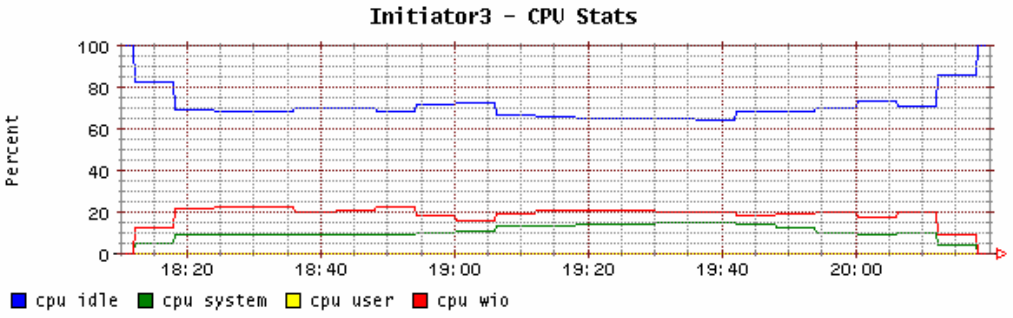

FIG. 8: no HBA/Read/MTU 9000

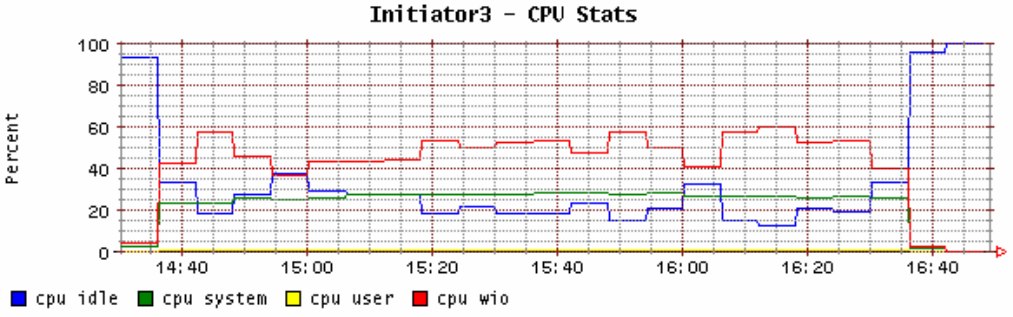

FIG. 9: no HBA/Write/MTU 1500

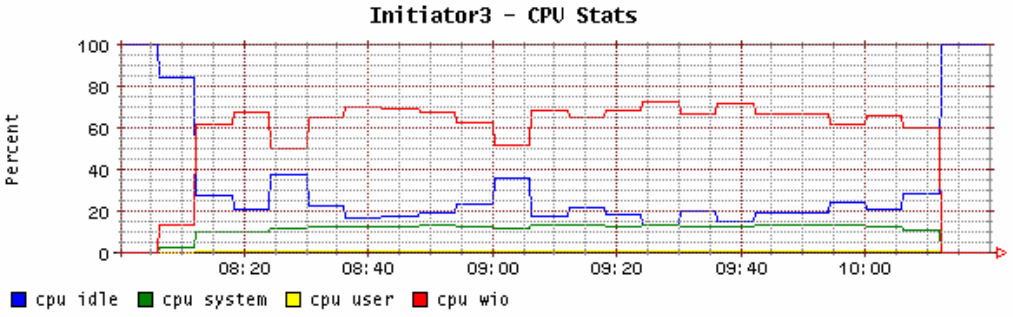

FIG. 10: no HBA/Write/MTU 9000

Si può notare come sia trascurabile il carico sulla CPU in caso di utilizzo dell'HBA intorno al 5%, mentre, sia piuttosto importante nell'altro caso, 20-25%, il che significa che uno dei core della CPU è dedicato al lavoro di I/O. Un altro interessante risultato si nota, quando è utilizzata la MTU a 9000 bytes, in particolare, se non è presente l'HBA, il carico sulla CPU quasi si dimezza.

Un altro aspetto importante da valutare sono le performance di trasferimento (espresse in MB/s). In questo caso i test sono stati effettuati con varie modalità, servendosi di uno o più initiator. La seguente tabella riporta i risultati derivati dalle prove citate nella valutazione del carico di CPU.

TAB. 1: Singolo initiator

Operazione	HBA	MTU	Trasferimento medio	Operazioni per secondo
Read	Si	1500	70.89	1134.2
Read	Si	9000	53.34	853.4
Write	Si	1500	67.41	1078.5
Write	Si	9000	71.78	1148.5
Read	No	1500	58.75	940.0
Read	No	9000	50.81	812.9
Write	No	1500	60.22	963.5
Write	No	9000	61.17	978.7

Nota

I seguenti test sono stati svolti utilizzando una grandezza del blocco di 64 KB in modalità sequenziale.

La seconda prova effettuata consiste nell'attivare quattro diversi thread di lettura o scrittura contemporanei da un singolo initiator.

TAB. 2: Multithread

Operazione	HBA	MTU	Trasferimento medio	Operazioni per secondo
Read	Si	1500	82.77	1324.3
Write	Si	1500	68.81	1101.0
Read	No	1500	61.68	986.9
Write	No	1500	65.73	1051.7

Invece di utilizzare un singolo disktest con più thread che potrebbe beneficiare della presenza di cache a vario livello si sono eseguiti quattro diversi processi disktest con un singolo thread in modo da simulare quattro batch distinti, come può accadere in un nodo di una farm di calcolo.

TAB. 3: Quattro processi

Operazione	HBA	MTU	Trasferimento medio aggregato	Operazioni per secondo aggregate
Read	Si	1500	64.60	1033.6
Write	Si	1500	64.80	1036.9
Read	No	1500	62.49	1000.0
Write	No	1500	66.62	1065.8

Nota

In questi test il carico sulla CPU degli initiator si è mantenuto costante.

Questi primi test condotti con un singolo initiator alla volta, non hanno prodotto un carico significativo sul FAS3020c, quindi per stressare l'iSCSI target si è provveduto ad utilizzare tutti e quattro i nodi a disposizione per controllare la risposta del sottosistema dischi e del bilanciamento del traffico sull'interfaccia virtuale iscsi. Il primo tentativo ha prodotto risultati insolitamente scarsi, il problema è stato in seguito individuato nella configurazione dello switch Ethernet, infatti, è stato necessario inserire nella configurazione la seguente linea "configure sharing address-based ip-source-dest" per "sincronizzare" gli algoritmi di bilanciamento del FAS e dello switch.

Per controllare l'efficacia del bilanciamento e le differenze in termini di througput, il test è stato effettuato prima assegnando un'unica interfaccia fisica a quella virtuale e poi aggiungendone una seconda.

TAB. 4: Singola interfaccia

Operazione	Initiator-0	Initiator-1	Initiator-2	Initiator-3	Aggregato
Read	21.21	14.64	21.28	13.27	70.40
Write	14.43	17.14	15.60	17.46	64.63

TAB. 5: Doppia interfaccia

Operazione	Initiator-0	Initiator-1	Initiator-2	Initiator-3	Aggregato
Read	21.21	14.04	21.23	16.94	73.42
Write	12.44	19.77	12.85	20.03	65.09

Come si può notare i risultati sono pressoché uguali questo probabilmente è da imputare al sottosistema dei dischi che non riesce a fornire un adeguato flusso dati tale da saturare l'interfaccia virtuale, quasi certamente una diversa tecnologia dei dischi (Fibre Channel, Serial Attached SCSI o SATA), peraltro disponibile tra le opzioni del FAS3020c, potrebbe raggiungere prestazioni migliori. Questa valutazione è stata confermata dal test effettuato utilizzando la connessione Fibre Channel, infatti, come controprova è stata installata un HBA QLogic QLA2340 nel nodo iscsi-i1, nella seguente tabella vengono riportati i risultati.

TAB. 6: Fibre Channel

Operazione	Block size	Trasferimento medio	Operazioni per secondo
Read	64	79.45	1271.2
Write	64	54.03	864.4
Read	256	82.97	331.9
Write	256	48.95	195.8

Nel valutare l'utilizzo della CPU si sono notati alti livelli di I/O wait, in questo stato il processore sta aspettando che delle operazioni di I/O siano completate e quindi le sue

capacità di calcolo vengono per così dire sprecate. Tuttavia i livelli sembravano troppo elevati, quindi si è utilizzato un benchmark basato sul calcolo matriciale, per verificare gli effetti sul trasferimento dati.

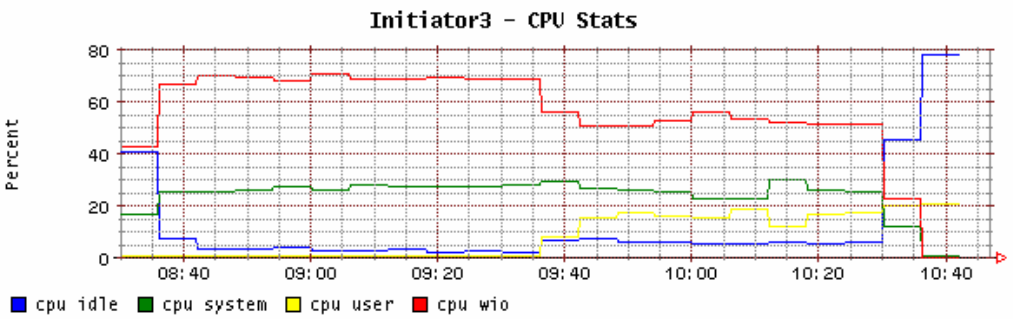

FIG. 11: Disktest + SparseBench

Il test si è limitato ad osservare la condizione meno favorevole, rappresentata dalla scrittura eseguita da un initiator senza HBA iSCSI, infatti come si nota dal grafico riportato in fig. 11 il livello iniziale di wio si avvicina al 70%, quando viene eseguito SparseBench il livello di user si alza e quello di wio si abbassa ma il trasferimento dati non ne risente, come viene riportato nella seguente tabella.

TAB. 7: SparseBench

Operazione	MTU	Block size	SpareBench	Trasferimento medio	Operazioni per secondo
Write	1500	64	No	60.22	963.5
Write	1500	64	Si	60.15	962.4

Questo comportamento può essere spiegato in questo modo: poiché le operazioni di I/O non appartengono a nessuno dei processori il sistema indica tutti i processori in I/O wait quindi le CPU non sono da considerarsi in attesa di I/O ma piuttosto liberi in attesa di un eventuale operazione di I/O.

Come ultimo test si è preso in considerazione l'aspetto filesystem, per effettuare questa prova uno dei volumi è stato formattato in ext3 e quindi con l'utilizzo di iozone[1] si sono ottenuti i seguenti risultati:

[1] http://www.iozone.org

TAB. 8: Iozone

Operazione	Block size 64	Block size 128	Block size 256	Block size 512	Block size 1024	Block size 2048	Block size 4096	Block size 8192	Block size 16384
Read Seq.	54755	54347	54623	53406	52382	52904	52451	52656	53291
Write Seq.	39406	41012	39093	42352	42484	42705	33519	45488	38800
Read Ran.	6884	10238	13006	16122	20332	24254	32674	44394	55136
Write Ran.	38672	41869	26089	28732	27983	32386	32962	38398	33573

La tabella riporta i dati relativi alle performance ottenute, quando l'effetto delle cache si è esaurito; la grandezza del file trasferito era di 8 GB, mentre l'MTU, in questo caso, era di 9000 bytes.

6 CONCLUSIONI

La tecnologia iSCSI si è rivelata promettente e degna di ulteriori investigazioni, data la sua flessibilità può essere impiegata in diversi ambiti, dal semplice utilizzo per fornire spazio disco agli utenti a configurazioni complesse come il clustering, la virtualizzazione o l'impiego con cluster filesystem. Questa prima esperienza ha permesso la costruzione di un modello di test utilizzando degli strumenti che si sono rivelati affidabili ma anche perfettibili.

Le schede gigabit ethernet integrate nei nodi si sono rivelate molto vicine in termini di throughput alle HBA iSCSI probabilmente perché implementano delle funzioni di Tcp Offload Engine, tuttavia, il carico sulla CPU risulta comunque significativo. Un'altra questione da chiarire rimane lo strano comportamento del sistema, quando si utilizzano i jumbo frames (MTU=9000), la soluzione forse potrà scaturire da un analisi più approfondita dei parametri di configurazione del protocollo iSCSI.

7 RINGRAZIAMENTI

Un particolare ringraziamento va ad Andrea Masiero che ha reso possibile per questo test l'utilizzo dell'hardware Network Appliance.

APPENDICE A

Di seguito è riportata lo script perl prodotto da Claudio Strizzolo per la comunicazione dei dati relativi alle operazioni di input/output al sistema di monitoring

```
#!/usr/bin/perl

$cmdPminfo="/usr/bin/pminfo -f ";
$cmdGmetric="/usr/bin/gmetric";

@pcpmetric=("disk.dev.read_bytes","disk.dev.write_bytes", \
 "disk.dev.blkread", "disk.dev.blkwrite");
@name=("diskio_readbytes", "diskio_writebytes", "diskio_readblks", \
 "diskio_writeblks");
@unit=("Kbytes/s", "Kbytes/s", "Blocks/s", "Blocks/s");
@type=("uint32", "uint32", "uint32", "uint32");

for ($i=0;$i<=$#pcpmetric;$i++) {
  $gm=$pcpmetric[$i];
  chomp($out1=qx { $cmdPminfo $gm });
  sleep (1);
  chomp($out2=qx { $cmdPminfo $gm });

  @lines1=split(/\n/,$out1,99999);
  @lines2=split(/\n/,$out2,99999);
  foreach $line (@lines1) {
    $line=~s/^[ \t]+//g;
    if (!($line =~ /^inst/)) { next; }
    ($garbage1,$disk,$garbage2)=split(/"/,$line,3);
    ($garbage1,$value1{$disk})=split(/ value /,$line,2);
  }
  foreach $line (@lines2) {
    $line=~s/^[ \t]+//g;
    if (!($line =~ /^inst/)) { next; }
    ($garbage1,$disk,$garbage2)=split(/"/,$line,3);
    ($garbage1,$value2)=split(/ value /,$line,2);
    $diff=$value2-$value1{$disk};
    $cmd=sprintf("%s -n %s_%s -v %s -t %s -u \"%s\"\n", $cmdGmetric, \
     $name[$i], $disk,$diff,$type[$i], $unit[$i]);
    qx{$cmd};
  }
}
```