ISSN online: 2282-5665

Consiglio Nazionale delle Ricerche

CERIS ISTITUTO DI RICERCA SULL'IMPRESA E LO SVILUPPO

Giugno

2011

Rapporto tecnico N.37

STORAGE IN HA: CLUSTER ATTIVO/PASSIVO OPEN-SOURCE

Giancarlo Birello, Ivano Fucile, Valter Giovanetti, Anna Perin

RAPPPORTO TECNICO CNR-CERIS
Anno 6, N° 37; Giugno 2011

Direttore Responsabile
Secondo Rolfo

Direzione e Redazione
Cnr-Ceris
Istituto di Ricerca sull'Impresa e lo Sviluppo
Via Real Collegio, 30
10024 Moncalieri (Torino), Italy
Tel. +39 011 6824.911
Fax +39 011 6824.966
segreteria@ceris.cnr.it
http://www.ceris.cnr.it

Sede di Roma
Via dei Taurini, 19
00185 Roma, Italy
Tel. 06 49937810
Fax 06 49937884

Sede di Milano
Via Bassini, 15
20121 Milano, Italy
tel. 02 23699501
Fax 02 23699530

Segreteria di redazione
Maria Zittino
m.zittino@ceris.cnr.it

STORAGE IN HA:
CLUSTER ATTIVO/PASSIVO OPEN-SOURCE

Giancarlo Birello*, Ivano Fucile, Valter Giovanetti
(Cnr-Ceris, Ufficio IT)

Anna Perin
(Cnr-Ceris, Biblioteca)

Cnr-Ceris
Ufficio IT
Strada delle Cacce, 73
10135 Torino – Italy
Tel.: 011 3977533/534/535

Cnr-Ceris
Biblioteca Ceris
Via Real Collegio, 30
10024 Moncalieri (Torino), Italy
Tel.: 011 6824928

*Corresponding author: g.birello@ceris.cnr.it

ABSTRACT: Ceris-CNR IT Office manages the CNR Piedmont network infrastructure and provides network services for about 420 users.
The development of a large and reliable storage system is the answer to two needs: one from users, who must deal with ever more larger digital objects for personal and research use, and one from need to have a space system engineering to store virtual machines and backup copies.
To contain expenditure and without detracting from the reliability of the system, we choose to concentrate financial commitment on quality hardware and realize the whole project with open-source software, specifically the main components are: Linux, DRBD (Distributed Replicated Block Device), Corosync and Pacemaker.

KEY WORDS: cluster, open-source, linux, storage

Indice

Elenco figure

1 Introduzione

La rete telematica del CNR è distribuita su tutto il territorio nazionale per raggiungere capillarmente tutte le strutture presenti. La gestione territoriale è delegata ai Referenti che sono incaricati delle questioni inerenti la connettività ed i servizi di rete. Nel nostro caso, l'Ufficio IT del Ceris amministra l'Infrastruttura di rete CNR in Piemonte, il cui Referente Territoriale coincide col responsabile dell'Ufficio IT.
L'Ufficio IT del Ceris ha sede presso l'Area di Ricerca CNR di Torino e ospita la sala macchine che costituisce il centro-stella delle connessioni e dei servizi di rete per gli organi CNR afferenti, in totale circa 15 strutture CNR per un'utenza complessiva indicativa di 420 unità di personale.
La caratteristica dell'infrastruttura piemontese è stata da sempre la centralizzazione dei principali servizi di rete quali ad esempio la posta elettronica, i server WEB, il repository degli utenti, i sistemi di firewalling e sicurezza, le partizioni di backup, i server DNS. Questa impostazione è risultata nel tempo una scelta valida non solo economicamente ma anche per quanto riguarda l'aggiornamento e lo sviluppo di nuove applicazioni, delle quali tutta l'utenza ha potuto trarre vantaggio, in modo immediato e senza alcuno sforzo da parte delle singole strutture. Questo progetto è un chiaro esempio di come, con questa gestione centralizzata a livello territoriale, contenendo la spesa e con un minimo apporto da parte di alcune strutture CNR afferenti all'infrastruttura, si sia potuto sviluppare un servizio evoluto di memorizzazione e disponibile immediatamente per tutti gli utenti registrati sui nostri server.

La necessità di sviluppare un sistema di memorizzazione capiente ed affidabile è scaturita da esigenze specifiche sia degli utenti che di origine sistemistica.
Nel primo caso, il sempre maggior utilizzo di strumenti informatici nello sviluppo delle ricerche hanno fatto si che si debba ricorrere a informazioni sempre più voluminose che vanno conservate, in modo affidabile e per tempi non sempre ben definiti. È questo il caso di grossi file di immagini nel campo geologico oppure in sequenze di DNA in campo biologico.
A livello sistemistico, la migrazione degli ultimi anni della maggior parte dei server e dei servizi di rete su infrastrutture virtualizzate, rende lo storage uno dei punti vitali dell'intera struttura. Infatti, nel caso di macchine virtuali sono file i dischi virtuali dei server e sono file i dischi virtuali sui quali sono memorizzate le informazioni quali email, backup utenti, cartelle di gruppo.

Per il contenimento della spesa, vista la ridotta disponibilità economica, senza per questo togliere nulla all'affidabilità del sistema, dopo un'analisi iniziale di prodotti commerciali, si è scelto di implementare la soluzione con software open-source, lasciando l'intera somma disponibile per l'acquisto di hardware di qualità.
Questa scelta ha permesso di contenere i costi, richiedendo però un lavoro di studio di alcuni mesi per individuare ed approfondire il software più adatto a soddisfare le esigenze richieste dal progetto. Questo significa che il software open-source disponibile garantiva comunque le esigenze di affidabilità anche se non commerciale.

2 Scelte software e hardware

Come già accennato precedentemente, si sono inizialmente prese in considerazioni soluzioni chiavi in mano di alcuni produttori. Le soluzioni individuate prevedevano la fornitura dell'hardware e del software necessario a realizzare il sistema di memorizzazione in HA (High Availability), ma per restare su un livello di costi accettabile occorreva acquistare un prodotto che non garantiva una certa flessibilità ed espandibilità della quale invece avevamo bisogno.

Lo stesso discorso vale per soluzioni software commerciali, che se da un lato riducono i tempi di gestione, dall'altro prevedono un impegno economico che al momento non eravamo in grado di garantire.

Ci si è quindi orientati per ridurre al minimo i costi del software destinando gli importi disponibili all'acquisto dell'hardware.
Per il software si è andati perciò nella direzione dell'open-source, avendo rilevato che in tale campo esistono soluzioni affidabili, suffragati dall'esperienza della comunità su tali prodotti.
In prima battuta si è presa in considerazione una soluzione Solaris, Open HA Cluster, ma vista l'inesperienza sui sistemi Solaris e data la disponibilità di un prodotto equivalente su piattaforma Linux, che presenta sicuramente più flessibilità del primo, si è andati in quest'ultima direzione.

Il framework incaricato della gestione del cluster e delle risorse è stato individuato nella combinazione dei due prodotti Corosync e Pacemaker.
Il primo è un'implementazione dello standard OpenAIS e lavora in alternativa a Heartbeat, che sembra abbandonato dalle principali distribuzioni Linux a favore di OpenAIS. Questo strato si incarica della gestione dei servizi base del cluster, in particolare della comunicazione tra i nodi e dell'appartenenza degli stessi al cluster.
Pacemaker invece coordina la gestione delle risorse attive sul cluster, appoggiandosi in alternativa a Corosync o Heartbeat come strato inferiore di comunicazione tra i nodi. Nato alcuni anni fa quando Heartbeat ha iniziato a prevedere uno strato superiore a cui delegare la gestione delle risorse (CRM – Cluster Resource Manager) è diventato uno dei componenti principali nei cluster in ambiente open-source.
Oltretutto questi due componenti sono disponibili come pacchetti sulla distribuzione Ubuntu server che è quella solita utilizzata presso l'Infrastruttura di rete e sulla quale siamo più esperti. In tal modo è stata prescelta quindi come base del sistema operativo Ubuntu Server 10.04 LTS, la release a lungo termine che garantisce una certa stabilità nel tempo dell'installazione e dei relativi pacchetti.

Altrettanto disponibile come pacchetto di Ubuntu Server è il componente DRBD (Distributed Replicated Block Device), alla base dello stack del cluster, che effettua quello che si può considerare come un mirror di rete tra due partizioni su due nodi distinti. E' un componente diffuso e largamente impiegato proprio in questo tipo di soluzioni, pertanto è stata la scelta ovvia per la nostra realizzazione.

Infine andava scelto con che formato e protocollo rendere disponibile lo spazio di memorizzazione del cluster. Tra gli standard di condivisione di storage in rete disponibili in ambito open-source ci si è orientati su due possibili soluzioni: iSCSI (*Internet Small Computer System Interface)* e NFS (*Network File System*). In base agli impieghi a cui sarebbe stato destinato il cluster entrambi i protocolli offrivano i requisiti teorici richiesti, ma da alcune prove pratiche il primo si è rivelato di più semplice impiego ed altamente più affidabile proprio in situazioni di interruzione di un nodo del cluster e rispristino sul

secondo nodo della condivisione. In questi casi gli effetti sulle macchine virtuali attive sull'hypervisor e sulle macchine agganciate direttamente allo storage sono risultati nulli e senza ritardi nel ripristino, mentre altrettanto non si può dire con l'impiego di NFS. Non abbiamo approfondito le ragioni di questi problemi nel funzionamento di NFS, contenti delle ottime prestazioni di iSCSI prescelto come ultimo strato da implementare sul cluster.

Per poter meglio definire le richieste hardware abbiamo proceduto ad una prima realizzazione su un paio di macchine inutilizzate, coscienti delle prestazioni limitate delle stesse, ma che ci avrebbero permesso sperimentare ed approfondire la conoscenza dei componenti software prescelti e con le dovute cautele, valutare se la scelta fosse realmente quella più opportuna per soddisfare le nostre esigenze.
A parte il tempo impiegato (alcune decine di secondi) dal cluster in caso di errore di un nodo per commutare sul secondo, dovuto principalmente alla CPU debole delle macchine, e la ridotta velocità di trasferimento in lettura/scrittura, causato dalla lentezza del raid dei dischi fisici, in un caso addirittura raid software, per il resto il comportamento del cluster attivo/passivo si è rivelato esattamente come richiesto dalle nostre esigenze.
In particolare si è sperimentato che una macchina virtuale configurata con tre dischi virtuali, il primo sullo storage locale dell'hypervisor mentre gli altri due su due partizioni distinte del cluster, agganciate tramite iSCSI, se durante un'operazione di lettura/scrittura da una partizione all'altra, di quelle residenti sul cluster, si verificava il guasto di un nodo e la commutazione sul secondo, una volta riattivato l'operazione di lettura/scrittura continuava fino al termine senza introdurre alcun errore a livello del singolo bit.

Da questa prima installazione di test abbiamo quindi dedotto alcuni requisiti importanti per la realizzazione finale del cluster:

- il raid su cui appoggiare la partizione DRBD doveva essere assolutamente hardware e per ridurre le possibilità di errore con batterie tampone della cache di scrittura sui dischi;
- visti i vari livelli di ridondanza e le relative esigenze di velocità di trasferimento, i dischi avrebbero potuto essere dei SATA senza dover ricorrere ai più costosi SCSI;
- i nodi del cluster avrebbero dovuto avere CPU con buone prestazioni ma senza dover ricorrere a quelle di ultimissima generazione;
- la memoria RAM richiesta dal sistema non era particolarmente eccessiva e quindi un 4GB per macchina sarebbero stati sufficienti;
- vista l'importanza della connettività ethernet dei nodi, sia in termini di affidabilità del cluster che di prestazioni nella replica dei dati, si è rilevata l'opportunità di avere una certa ridondanza di interfacce di rete, scegliendo un modulo addizionale da 4 porte in aggiunta alle 2 disponibili di serie, tutte ovviamente al Gigabit.

Infine altre scelte ovvie, ma non legate strettamente a questa applicazione, sono state l'alimentazione ridondata e la possibilità di estensione, soprattutto in termini di dischi, che avrebbero dovuto offrire gli apparati.

3 Configurazioni

L'esperienza sulla prima installazione di test e le varie prove fatte, hanno permesso definire con chiarezza la configurazione necessaria per l'installazione definitiva del cluster.
Per sfruttare ulteriormente la fase di configurazione fintanto che il sistema non fosse messo in produzione, anche sulle macchine definitive sono stati installati gradualmente i vari dischi per provare sul nuovo hardware i passi necessari per l'ampliamento dello spazio di memorizzazione una volta che le macchine fossero in produzione.

Avendo già a disposizione il sistema di virtualizzazione e gli hypervisor in funzione, l'installazione è stata fatta tenendo anche presente le connessioni da e verso le macchine virtuali e potendo verificare così il funzionamento del cluster all'interno dell'infrastruttura esistente.
In figura 1 sono rappresentati in forma schematica il cluster ed i vari blocchi che lo compongono all'interno di una struttura virtualizzata.

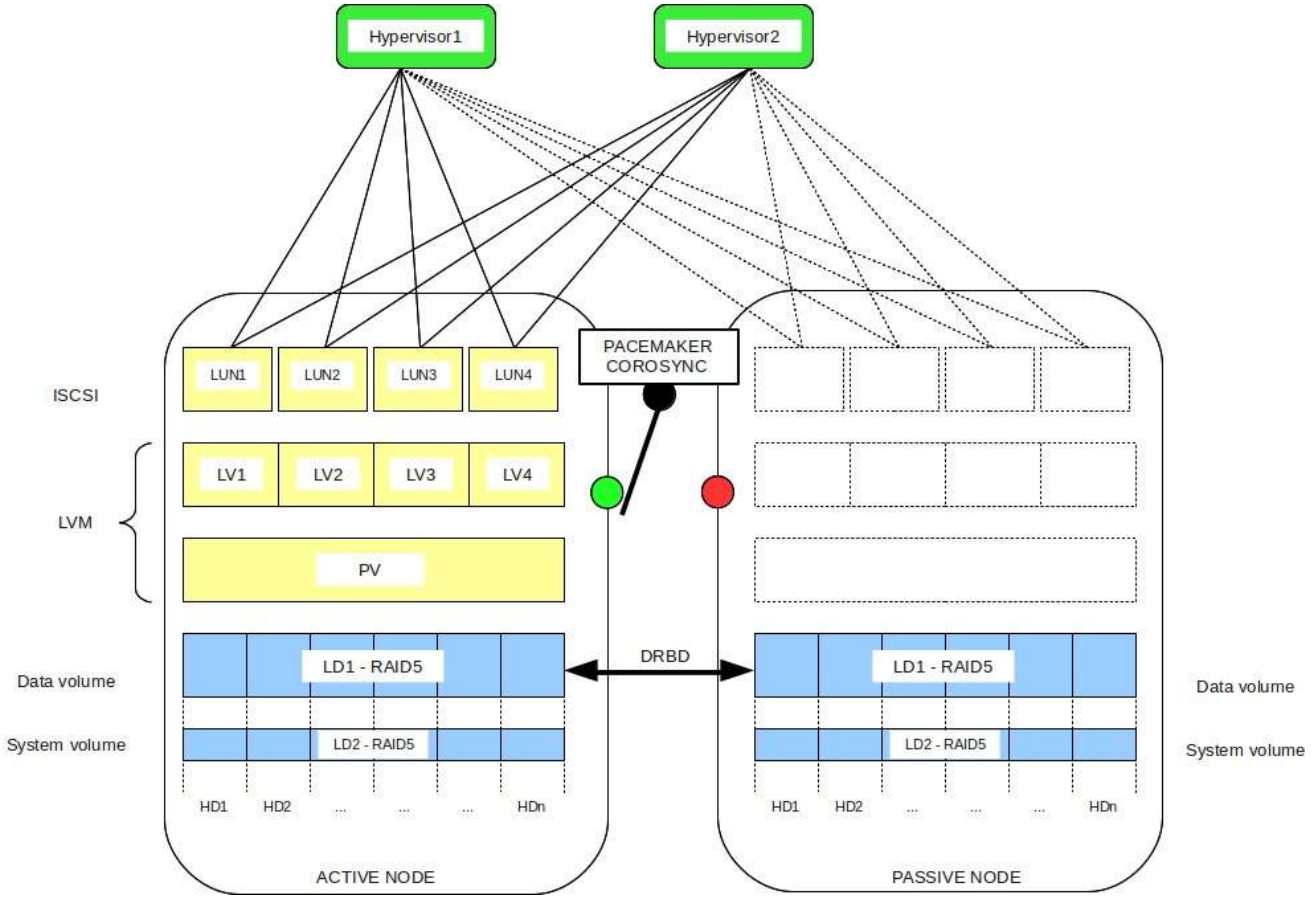

Figura 1: Schema a blocchi del cluster

3.1 Sistema operativo e interfacce di rete

3.1.1 Configurazione raid hardware

Come anticipato al punto 2 per lo strato a livello più basso si è scelto per l'affidabilità e le prestazioni di configurare i dischi in raid, in particolare scegliendo la tipologia Raid5. Inizialmente utilizzando solo 3 dei 4 dischi disponibili per ogni nodo in modo da lasciare aperta la possibilità di simulare in un secondo tempo un aumento di spazio on-line.
Sono stati definiti su ognuno dei due nodi i due seguenti Logical Drive:

- LD1: da 16GB destinato ad ospitare la partizione di sistema, quella di swap e quella per la memorizzazione dei metadati necessari a DRBD;
- LD2: lo spazio restante, che sarà la partizione messa in mirror tra i due nodi tramite DRBD.

3.1.2 Installazione base server

Definiti i dischi logici si è proceduto con l'installazione del sistema operativo, sempre su entrambi i nodi. Si è installato Ubuntu Server 10.04 LTS sulla prima delle partizioni del LD1 di circa 5GB e utilizzata una partizione di swap di 6GB. Non si è installato il sistema con LVM per evitare conflitti successivi avendo previsto l'utilizzo di LVM sopra lo strato DRBD prima dell'iSCSI.
All'installazione classica è stato solo aggiunto al termine il servizio SSH per accedere in remoto al nodo, ad ogni modo tutti i componenti necessari al funzionamento del cluster sono stati tutti installati dal repository della distribuzione, assicurandosi così l'aggiornamento dei pacchetti in modo automatico senza dover effettuare compilazioni e configurazioni particolari.

3.1.3 Interfacce di rete

Per tutti i componenti del cluster sono fondamentali le interfacce di rete, sia per quel che riguarda le prestazioni che per l'affidabilità dei dati.
Non essendo ancora disponibili interfacce a 10Gb di costo contenuto o quantomeno a costi che rientrassero nelle disponibilità del progetto, si è optato per interfacce a 1Gb in numero abbondante. Alle due interfacce disponibili di serie integrate sulla piastra madre, sono state aggiunte altre 4 interfacce sempre ethernet a 1Gb.
Nella configurazione delle interfacce abbiamo tenuto presente di un vincolo dato dalla possibilità di controllo dell'hardware del nodo tramite protocollo IPMI (Intelligent Platform Management Interface) solo disponibile sulle due interfacce integrate sulla piastra madre.
Si è inoltre valutato il traffico sviluppato dai vari blocchi, in particolare tenendo presente le prestazioni del protocollo di replica in rete DRBD, anche in vista di futuri sviluppi, considerato prioritario in termini di banda rispetto agli altri.
Infine si è scelto, almeno per questa prima realizzazione, di mantenere i due nodi del cluster vicini fisicamente, in questo modo connessioni dirette tra i nodi sono state realizzate senza altri apparati intermedi, che avrebbero introdotto ulteriori SPOF (Single Point Of Failure), tramite cavi crossed tra le interfacce di rete dei due nodi.

Ricapitolando la configurazione adottata, riassunta in figura 1, è la seguente:

- eth0: interfaccia integrata, connessione crossed, dedicata al controllo del cluster (Corosync) ed alla comunicazione IPMI;
- eth1: interfaccia integrata, connessione alla VLAN dedicata tramite switch, per comunicazione ridondata del cluster e connessione al nodo attivo del cluster da parte dei client;
- eth4 e eth5: interfacce aggiuntive, connessione crossed, messe in bond tra di loro e destinate alla replica DRBD;
- eth2 e eth3: interfacce aggiuntive, disponibili.

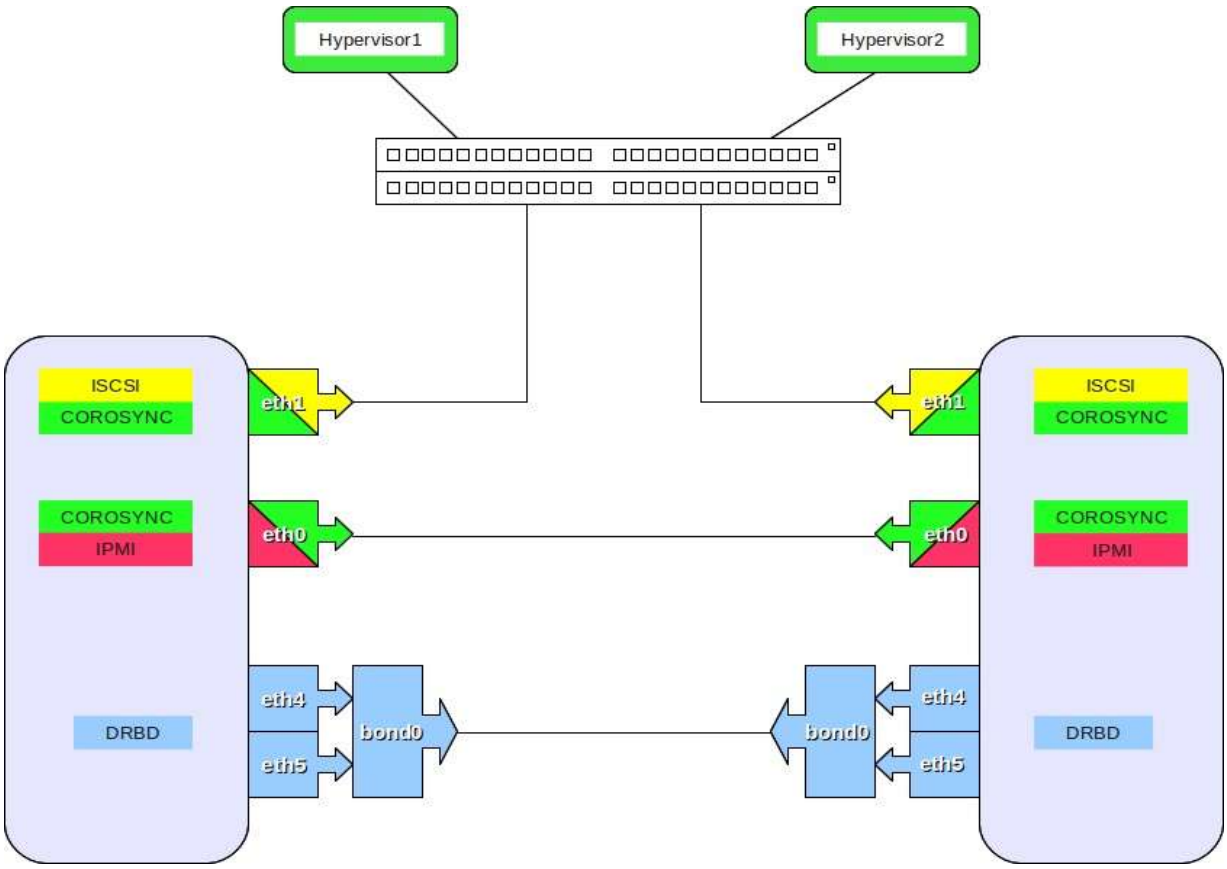

Figura 2: Interfacce di rete

Vale la pena evidenziare la configurazione in bonding delle interfacce, che permette allo stesso tempo un raddoppio delle prestazioni nel caso di due interfacce e l'incremento dell'affidabilità della connessione. Nel nostro caso la configurazione del bonding avviene nel seguente modo:

```
/etc/network/interfaces
[...]
auto bond0
iface bond0 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        bond-slaves none
        bond-mode 0
        bond-miimon 100
```

```
auto eth4
iface eth4 inet manual
        bond-master bond0

auto eth5
iface eth5 inet manual
        bond-master bond0
[...]
```

3.1.4 Configurazione IPMI per stonith

Una delle situazioni di stallo più critiche per un cluster di questo tipo è la condizione di “split-brain”, cioè quando per una serie di condizioni i due nodi perdono la comunicazione tra di loro ma entrambi sono ancora funzionanti. In questo caso entrambi i nodi crederanno di essere l'unico nodo attivo ed entrambi si crederanno il master della replica DRBD. Si verranno ad avere quindi due copie master dei dati con l'inevitabile invalidazione dei dati stessi.
Uno dei metodi da noi adottati per ridurre le possibilità che si verifichi uno split-brain è lo Stonith (Shoot The Other Node In The Head) dei nodi, cioè il primo nodo che “sente” di essere rimasto l'unico attivo del cluster si assicura che l'altro nodo sia realmente inattivo eseguendo un reset hardware dell'altra macchina. Questa operazione viene portata a termine tramite il protocollo IPMI sull'interfaccia dedicata allo scopo.
I server da noi acquistati sono dotati di questa possibilità e permettono definire un indirizzo IP specifico su cui la macchina è in ascolto per i comandi IPMI. L'interfaccia prescelta è la eth0, una di quelle integrate sulla piastra madre, e l'indirizzo assegnato al sistema di controllo a livello di chassis deve rientrare nella subnet degli indirizzi assegnati alla stessa interfaccia a livello di sistema operativo. Inoltre è necessario definire anche uno user ed una password, senza le quali i comandi IPMI non verrebbero eseguiti.
Infine la garanzia della comunicazione tra i due nodi così come quella della comunicazione per la procedura di stonith è rafforzata dal fatto che le due interfacce dei due nodi sono connesse tramite un semplice cavo di rete crossed senza aver introdotto ulteriori SPOF intermedi.

3.2 DRBD

Nella configurazione del mirror di rete DRBD si è tenuto conto di alcune condizioni quali:
- dimensionamento opportuno dei metadati per permettere l'espansione eventuale della partizione;
- possibilità di aggiungere una seconda partizione a quella esistente;
  introdurre anche a livello di DRBD accorgimenti per evitare la possibilità di split-brain.

Una volta installato dal repository della distribuzione e disattivato l'avvio automatico al boot della macchina, si è proceduto a configurare su entrambi i nodi il blocco DRBD, la cui configurazione è inclusa in un file generale ed in un file specifico per la partizione, non obbligatoria la suddivisione, ma è buona norma mantenere questa impostazione.

3.2.1 Parametri generali

Nel file dei parametri generali non sono state apportate modifiche ai valori di default e nello specifico sono semplicemente definiti alcuni gestori disponibili ad essere invocati qualora definite le condizioni nelle configurazioni successive. La sezione common presente nel file può essere ridefinita nella sezione specifica di una partizione. Il file si presenta così:

```
/etc/drbd.d/global_common.conf
global {
        usage-count yes;
}
common {
        protocol C;

        handlers {
                pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-degr.sh; /usr/lib/drbd/notify-
emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot -f";
#This handler is called if the node is primary, degraded and if the local copy of the data is inconsistent.

                pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-after-sb.sh; /usr/lib/drbd/notify-
emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot -f";
#The node is currently primary, but lost the after-split-brain auto recovery procedure. As as consequence, it should be
abandoned.

                local-io-error "/usr/lib/drbd/notify-io-error.sh; /usr/lib/drbd/notify-emergency-
shutdown.sh; echo o > /proc/sysrq-trigger ; halt -f";
#DRBD got an IO error from the local IO subsystem.

        }
        startup {
        }
        disk {
        }
        net {
        }
        syncer {
        }
}
```

Il modulo DRBD può interagire col cluster Pacemaker ed è prevista la possibilità di introdurre in DRBD alcune configurazioni specifiche quando usato in abbinamento con il cluster. In particolare anche a livello di DRBD si può introdurre un accorgimento per ridurre la possibilità che si verifichi la condizione di split-brain.
Le direttive da inserire possono essere introdotte a livello common o a livello specifico della partizione e fanno sì che, qualora si presenti un'interruzione sulla comunicazione di replica delle partizioni, il cluster non intervenga con la promozione del nodo secondario, evitando così la situazione di split-brain. Le normali operazioni del cluster vengono ristabilite una volta che la comunicazione di replica sia tornata attiva e le partizioni siano sincronizzate. Ovviamente, affinché questo intervento sia fattibile, è necessario che la comunicazione tra i nodi sia ridondata e differenziata da quella di replica delle partizioni.

In particolare le direttive da introdurre nella configurazione sono:

```
/etc/drbd.d/disk0.res
resource disk0 {
                ...
                disk {
                        ...
                        fencing resource-only;
#resource-only: If a node becomes a disconnected primary, it tries to fence the peer's disk. This is done by calling the
fence-peer handler. The handler is supposed to reach the other node over alternative communication paths and call
'drbdadm outdate res' there.
                }
                handlers {
                fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
#The handler is part of the fencing mechanism. This handler is called in case the node needs to fence the peer's disk. It
should use other communication paths than DRBD's network link.

                after-resync-target "/usr/lib/drbd/crm-unfence-peer.sh";
#DRBD calls this handler just after a resync operation finished on the node whose disk just became consistent after
being inconsistent for the duration of the resync. It might be used to remove a snapshot of the backing device that was
created by the before-resync-target handler.

                        ...
```

3.2.2 Parametri partizioni

Innanzitutto, prevedendo future espansioni della dimensione iniziale della partizione e visti i benefici che introducono nella gestione, è stato scelto di memorizzare i metadati di sincronizzazione su una partizione dedicata separata dalla partizione in replica. Opportunamente dimensionata, secondo la relazione 36kB + Backing-Storage-size / 32k che in prima approssimazione si riduce a 32kByte per 1GByte di storage arrotondato al primo MB superiore, la partizione per i metadati è stata individuata sul primo LD come riportato al punto 3.1.1.
Come protocollo di replica si è prescelto il tipo C che garantisce la massima sicurezza sull'integrità dei dati, cioè un'operazione di scrittura è considerata di esito positivo se è stata eseguita con successo su entrambi i nodi del cluster.
Nella configurazione dell'indirizzo IP e relativa porta si fa notare che il protocollo DRBD usa in realtà due porte, anche se è definita solo una nella direttiva, nel momento in cui si definiscano due partizioni in replica è necessario assegnare due porte distinte distanti almeno due unità (es. A.B.C.D:7788 e A.B.C.D:7790).
La configurazione completa della partizione di replica è la seguente:

```
/etc/drbd.d/disk0.res
resource disk0 {
        protocol C;
        disk {
                on-io-error detach;
                fencing resource-only;
        }
        handlers {
               fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
               after-resync-target "/usr/lib/drbd/crm-unfence-peer.sh";
        }
       net {
               cram-hmac-alg sha1;
               shared-secret "******************";
       }
       syncer {
       rate 200M;
```

```
        verify-alg sha1;
        al-extents 257;
        }

        on uclu1 {
                device /dev/drbd0;
                disk /dev/cciss/c0d1p1;
                address 192.168.1.1:7788;
                flexible-meta-disk /dev/cciss/c0d0p3;
        }
        on uclu2 {
                device /dev/drbd0;
                disk /dev/cciss/c0d1p1;
                address 192.168.1.2:7788;
                flexible-meta-disk /dev/cciss/c0d0p3;
        }
}
```

3.3 Corosync

Come anticipato, il componente Corosync è il blocco incaricato della gestione della comunicazione tra i due nodi del cluster e utilizza lo standard OpenAIS. Anch'esso è disponibile nel repository della distribuzione e facilmente installabile.
Dopo aver generato la chiave per le comunicazioni cifrate tra i nodi e ricopiata da un nodo all'altro, si può procedere con la configurazione.

Occorre tener presente innanzitutto che avremo bisogno di almeno due canali di comunicazione per ridondanza e per permettere al modulo DRBD di compiere le operazioni atte a ridurre la possibilità di split-brain. Inoltre, oltre a configurare i due canali, occorre anche definire come vengono utilizzati, cioè definire il valore del parametro rrp_mode (Redundant Ring Protocol) che può avere tre valori: none, active o passive. Nel nostro caso abbiamo prescelto il valore active, che aumenta leggermente il carico sulla CPU, ma riduce i tempi di risposta in caso di errori, mantenendo sempre attivi entrambi i canali di comunicazione.
Per quanto riguarda i parametri generali abbiamo mantenuto i valori di default considerandoli adatti alla nostra applicazione.

La configurazione, tralasciando alcune parti meno significative, risulta quindi la seguente:

/etc/corosync/corosync.conf

```
totem {
        version: 2
        [...]
        rrp_mode: active

        interface {
                ringnumber: 0
                bindnetaddr: 192.168.168.0
                mcastaddr: 226.94.1.10
                mcastport: 5405
        }
```

```
        interface {
                ringnumber: 1
                bindnetaddr:10.10.10.0
                mcastaddr: 226.94.1.15
                mcastport: 5407
        }

}

[...]
```

Configurato su entrambi i nodi ed avviato il servizio, che sarà comunque avviato in automatico al boot della macchina, si può facilmente verificare che i due canali siano attivi con il seguente comando:

```
root@nodo1:~# corosync-cfgtool -s
Printing ring status.
Local node ID 123456789
RING ID 0
id      = 192.168.168.1
status  = ring 0 active with no faults
RING ID 1
id      = 10.10.10.1
status  = ring 1 active with no faults
```

3.4 Pacemaker

Il modulo Pacemaker è installato dai repository della distribuzione è viene installato insieme a Corosync. Questo componente è incaricato della gestione dei vari servizi disponibili sul cluster. In sintesi si fa carico di attivare i servizi predefiniti su uno o sull'altro nodo in base alla situazione del cluster: se un nodo va in errore sposta i servizi sull'altro nodo, dove per errore si intende una condizione problematica hardware del nodo o anche una situazione di mancanza di connettività di rete.

Le informazioni sul cluster, sia le configurazioni che la situazione corrente, sono contenute nel Cluster Information Base (CIB) che viene gestito in modo interattivo tramite l'interfaccia a riga di comandi crm fornita con Pacemaker.

Prima di procedere con la configurazione abbiamo raccolto tutte le condizioni ed i servizi e fatto le opportune considerazioni, le ipotesi sono quindi state:

- cluster attivo/passivo a due nodi senza quorum;
- nessun nodo preferenziale, quando le risorse sono su un nodo ci restano salvo che il nodo stesso non vada in errore;
- abilitazione dello stonith;
- controllo connettività di rete;
- LVM e iSCSI sullo stesso nodo e avviati nell'ordine;
- LVM e iSCSI sullo stesso nodo dov'è il DRBD Master, che va promosso prima dell'avvio dei servizi;
- indirizzo IP virtuale sul nodo attivo, con un send ARP per rendere più rapido l'apprendimento e con un port block sulla porta 3260 utilizzata da iSCSI, per rendere più affidabile la commutazione.

La configurazione finale è quindi risultata la seguente, che commentiamo tra le righe:

```
root@nodo2:~# crm configure show
node uclu1 ¥
        attributes standby="off"
node uclu2 ¥
        attributes standby="off"
primitive IpA ocf:heartbeat:IPaddr2 ¥
        params ip="10.10.10.5" cidr_netmask="24" nic="eth1" ¥
        op monitor interval="5s"
```

definizione della risorsa IP virtuale, subnet e interfaccia sulla quale esporlo;

```
primitive IpArp ocf:heartbeat:SendArp ¥
        params ip="10.10.10.5" nic="eth1"
```

definizione della risorsa send ARP;

```
primitive Lvm ocf:heartbeat:LVM ¥
        params volgrpname="replica0"
```

definizione della risorsa LVM, con l'indicazione obbligatoria del nome del volume group;

```
primitive drbd-disk0 ocf:linbit:drbd ¥
        params drbd_resource="disk0" ¥
        op monitor interval="15s"
```

definizione della risorsa DRBD, indicando la partizione in replica;

```
primitive iscsi lsb:iscsitarget ¥
        op monitor interval="15s"
```

definizione della risorsa iSCSI, tutti i parametri sono nel file di configurazione specifico;

```
primitive ping ocf:pacemaker:ping ¥
        params host_list="10.10.10.254" multiplier="200" dampen="5s" ¥
        op monitor interval="10"
```

definizione della risorsa ping, include l'indirizzo rispetto cui fare la verifica di connettività;

```
primitive portblock0 ocf:heartbeat:portblock ¥
        params protocol="tcp" ip="10.10.10.5" portno="3260" action="block" ¥
        op monitor interval="10" timeout="10" depth="0"
```

definizione della risorsa blocco porta, si indica protocollo, indirizzo e numero porta;

```
primitive portunblock0 ocf:heartbeat:portblock ¥
        params protocol="tcp" ip="10.10.10.5" portno="3260" action="unblock" ¥
        op monitor interval="10" timeout="10" depth="0"
```

definizione della risorsa sblocco porta, si indica protocollo, indirizzo e numero porta;

```
primitive st-uclu1 stonith:external/ipmi ¥
        params hostname="uclu1" ipaddr="192.168.168.1" userid="userid" passwd="secret" ¥
        interface="lan" meta target-role="Started"
primitive st-uclu2 stonith:external/ipmi ¥
        params hostname="uclu2" ipaddr="192.168.168.2" userid="userid" passwd="secret" ¥
        interface="lan" meta target-role="Started"
```

definizione delle risorse per lo stonith, sono distinte a seconda del nodo sul quale sono attivate, si indica l'indirizzo IP, lo user e la password per il protocollo IPMI;

```
group HAServices portblock0 Lvm IpA IpArp iscsi portunblock0 ¥
        meta target-role="Started"
```

definizione di un gruppo che include le risorse che dovranno essere attivate sullo stesso nodo e definisce anche l'ordine di avvio/arresto delle stesse;

```
ms ms-drbd-disk0 drbd-disk0 ¥
        meta master-max="1" master-node-max="1" clone-max="2" clone-node-max="1" ¥      notify="true"
```

definizione della replica master/slave DRBD, indica che dev'esserci un solo master;

```
clone pingc ping ¥
        meta globally-unique="false"
```

definizione di un clone della risorsa ping, in modo tale che ce ne sia un'istanza su ognuno dei nodi;

```
location HA_on_connected HAServices ¥
        rule $id="HA_on_connected-rule" -inf: not_defined pingd or pingd lte 0
```

definizione del vincolo di non avere il gruppo HAServices su un nodo senza connettività verso il default gateway;

```
location l-st1-uclu1 st-uclu1 -inf: uclu1
location l-st2-uclu2 st-uclu2 -inf: uclu2
```

definizione dei vincoli dove attivare le due risorse distinte di stonith specifiche per ogni nodo;

```
colocation ms-drbd0-with-haservices inf: ms-drbd-disk0:Master HAServices
```

definizione del vincolo di attivare il gruppo dei servizi sul nodo master della replica DRBD;

```
order Drbd0BLvm inf: ms-drbd-disk0:promote HAServices:start
```

definizione dell'ordine di attivazione del gruppo servizi solo dopo la promozione a master della replica DRBD;

```
        property $id="cib-bootstrap-options" ¥
                dc-version="1.0.8-042548a451fce8400660f6031f4da6f0223dd5dd" ¥
                cluster-infrastructure="openais" ¥
                expected-quorum-votes="2" ¥
                stonith-enabled="true" ¥
```

abilitazione della funzione di stonith;

```
                no-quorum-policy="ignore" ¥
```

disattivazione della regola del quorum;

```
                last-lrm-refresh="1289819733"
        rsc_defaults $id="rsc-options" ¥
                resource-stickiness="100"
```

definizione per mantenere i servizi sul nodo attivo fino ad un suo eventuale errore.

A configurazione terminata, da uno dei due nodi, si può tramite il comando crm_mon controllare il funzionamento del cluster. In figura avete un esempio delle informazioni prodotte:

```
============
Last updated: Wed May 25 15:57:53 2011
Stack: openais
Current DC: uclu2 - partition with quorum
Version: 1.0.8-042548a451fce8400660f6031f4da6f0223dd5dd
2 Nodes configured, 2 expected votes
5 Resources configured.
============

Online: [ uclu1 uclu2 ]

 Resource Group: HAServices
     portblock0 (ocf::heartbeat:portblock):     Started uclu1
     Lvm        (ocf::heartbeat:LVM):   Started uclu1
     IpA        (ocf::heartbeat:IPaddr2):       Started uclu1
     IpArp      (ocf::heartbeat:SendArp):       Started uclu1
     iscsi      (lsb:iscsitarget):      Started uclu1
     portunblock0       (ocf::heartbeat:portblock):     Started uclu1
 Master/Slave Set: ms-drbd-disk0
     Masters: [ uclu1 ]
     Slaves: [ uclu2 ]
st-uclu1        (stonith:external/ipmi):        Started uclu2
st-uclu2        (stonith:external/ipmi):        Started uclu1
 Clone Set: pingc
     Started: [ uclu1 uclu2 ]
```

Figura 3: Comando crm_mon

3.5 LVM e iSCSI

Le due componenti LVM ed iSCSI costituiscono lo strato più esterno del cluster e vengono attivate alternativamente solo sul nodo attivo. Entrambe sono stati installate dal repository della distribuzione, dev'essere disattivato il loro avvio automatico al boot della macchina e devono essere configurate in modo particolare rispetto la configurazione del cluster.

Nella configurazione globale di LVM, su entrambi i nodi, per evitare problemi di identificazione delle partizioni coinvolte, occorre introdurre innanzitutto il seguente filtro:

```
filter = [ "a|drbd.*|", "r|.*|" ]
```

che permetterà di rilevare solo i volumi fisici ospitati sulle partizioni DRBD (es. /dev/drbd0).
Per definire i vari livelli di LVM, i volumi fisici, i gruppi ed i volumi logici, occorre agire da console sul nodo sul quale è attiva la replica master di DRBD. Una volta definiti i vari componenti, la configurazione è memorizzata all'interno delle partizioni stesse, quindi non è necessario ripetere l'operazione sull'altro nodo poiché alla prima commutazione, sul secondo nodo si ritroverà la stessa configurazione definita inizialmente.
Avendo alcuni vincoli sulle dimensioni massime della singola partizione iSCSI in grado di essere gestita dall'hypervisor, abbiamo configurato LVM in modo tale da presentare partizioni di un massimo di 2TB ciascuna.

Prima di attivare iSCSI occorre invece, su entrambi i nodi, effettuare un paio di configurazioni.
La prima relativa allo script di start, non fondamentale, ma che migliora sicuramente le prestazioni del modulo e la sicurezza. Consiste nell'andar a scrivere direttamente nello script l'indirizzo IP sul quale l'iSCSI Target (lato server del protocollo) sarà in ascolto per le connessioni da parte dell'iSCSI Initiator (lato client), vista la presenza di più interfacce di rete per evitare di averlo in ascolto su tutti gli indirizzi disponibili del nodo. In particolare la modifica è la seguente:

```
root@nodox:~# cat /etc/init.d/iscsitarget
#!/bin/sh
#
[…]
        start-stop-daemon --start --exec $DAEMON --quiet --oknodo -- -a 10.10.10.5
[…]
```

La seconda configurazione, anch'essa da ripetere su entrambi i nodi, è quella relativa alle partizioni che verranno rese disponibili tramite iSCSI. Nel nostro caso specifico di distribuzione e pacchetto iSCSI, la configurazione è inserita nel file `/etc/ietd.conf` e riportiamo qui un esempio per una partizione generica:

```
root@nodox:~# cat /etc/ietd.conf
Target iqn.AAAA-MM.cnr.to:storage.disco.U1.2tera
        IncomingUser utente password
Lun 0 Path=/dev/VG/LV0,Type=blockio,ScsiId=CNRT0123456,ScsiSN=12345678,IOMode=wt
        MaxConnections          1
        InitialR2T              Yes
        ImmediateData           No
        MaxRecvDataSegmentLength 131072
        MaxXmitDataSegmentLength 131072
        MaxBurstLength          262144
        FirstBurstLength        262144
```

```
DefaultTime2Wait        2
DefaultTime2Retain      20
MaxOutstandingR2T       8
DataPDUInOrder          Yes
DataSequenceInOrder     Yes
ErrorRecoveryLevel      0
HeaderDigest            None
DataDigest              None
```

Rispetto la configurazione di default non sono state fatte modifiche ai parametri specifici, sono solo state modificate le prime tre righe, in particolare si segnala:

- il nome del Target dev'essere univoco e si consiglia rispettare un certo standard nella sua stesura;
- le credenziali dell'IncomingUser è bene inserirle per l'autenticazione CHAP da parte dell'Initiator, per la sicurezza dell'accesso alla risorsa;
- il Path è quello che risulta per il volume logico fornito da LVM;
- lo ScsiId e lo ScsiSN devono essere inseriti nel nostro caso per permettere all'Initiator dell'hypervisor di riconoscere la partizione e riconnettersi in caso di commutazione tra i due nodi del cluster.

Nel caso di più partizioni offerte dall'iSCSI andranno inserite nel file di configurazione tanti gruppi di direttive come la precedente, con la ovvia accortezza che sia univoco il Target, lo ScsiId e lo ScsiSN.

4 Applicazioni

L'esigenza di avere ampio spazio di memorizzazione in alta affidabilità è nata principalmente per soddisfare le richieste di storage da parte dei sistemi di virtualizzazione e per le richieste, da gran parte dell'utenza, di avere a disposizione maggiore spazio non solo per le proprie esigenze di backup ma anche per le necessità di memorizzare dati da parte di gruppi di ricerca. Assimilabile a quest'ultima esigenza è anche la necessità come Ufficio IT di dotarsi di spazio consistente per mantenere le opere digitali all'interno del progetto Bess Digitale.

Per il sistema di virtualizzazione, lo storage è visto a blocchi di 2TB, formattati col file system dell'hypervisor, sul quale vengono memorizzati sia i dischi virtuali che le immagini di backup dei server virtuali installati.
Avere le immagini di backup sul storage permette il rapido ripristino dei server virtuali nel caso l'hypervisor sul quale sono in esecuzione dovesse presentare dei problemi. Abbiamo verificato che la macchina virtuale può essere eseguita senza problemi direttamente dallo storage del cluster senza doverla spostare necessariamente sullo storage locale dell'hypervisor con prestazioni più che accettabili.
E' inoltre anche molto utile in fase di test o semplicemente in fase di aggiornamento dei server virtuali, avere spazio sufficiente sul quale memorizzare una copia della macchina prima di procedere all'aggiornamento, di modo che, in caso di problemi, si possa rapidamente tornare alla situazione precedente l'intervento.

Un'alternativa all'uso tramite hypervisor è connettere direttamente il server, sia esso fisico o virtuale, alla partizione tramite iSCSI. In questo modo si elimina uno strato di virtualizzazione tra lo storage ed il server utilizzatore finale, quest'ultimo si connetterà col proprio iSCSI Initiator al disco presente sul cluster. Occorre però tener presente che, mentre nel caso tramite hypervisor il file system utilizzato è di tipo ad accesso multiplo, in questo caso, se si utilizza un file system tradizionale, potremmo avere un solo server alla volta connesso alla partizione iSCSI del cluster.

5 Conclusioni

L'esperienza maturata con questo progetto è stata molto positiva.
Da un lato l'approfondimento dei sistemi open-source ha rivelato come in caso di risorse economiche limitate, si possa sopperire a tale carenza sfruttando strumenti avanzati ed affidabili messi a disposizione dalla comunità, ovviamente con un certo impegno e studio, ma che trova conforto nei risultati ottenuti.
Dal punto di vista tecnico la soluzione è stata molto soddisfacente alle richieste iniziali sia per quanto riguarda l'aspetto sistemistico e di virtualizzazione, sia per gli utenti che da subito hanno apprezzato il nuovo servizio ed iniziato ad utilizzarlo in tutte le sue peculiarità.

Ovviamente, anche se ormai in produzione, la fase di studio e sviluppo non si può considerare terminata, in quanto sono da approfondire alcune questioni legate al mantenimento del sistema, ad esempio l'aggiornamento dei sistemi operativi sui due nodi, oltre allo studio delle possibilità di espansione legate ai limiti delle partizioni DRBD e quindi l'evoluzione del controllo del cluster per la gestione di più partizioni DRBD contemporanee.

Il sistema e la sua espandibilità hanno da subito lasciato intravvedere molteplici impieghi che possono trarre vantaggio da questa installazione. In particolare all'interno del progetto Bess Digitale, è fondamentale un sistema di memorizzazione dove poter conservare in modo affidabile e in grande quantità gli oggetti digitali che verranno prodotti dagli scanner, si tratta di circa 6000 volumi che saranno digitalizzati e di cui il nostro Ufficio IT dovrà curare la parte di storage, repository e presentazione. Non solo interessante l'aspetto di storage dei dati, ma anche il poter gestire i servizi richiesti dal progetto su sistemi virtuali che potranno fruire direttamente dello spazio di memorizzazione messo a disposizione dal cluster.

6 Bibliografia

[1] The DRBD User's Guide, http://www.drbd.org/users-guide/users-guide.html, acceduto maggio 2011.

[2] Pacemaker 1.1 - Clusters from Scratch, http://www.clusterlabs.org/doc/en-US/Pacemaker/1.1/html-single/Clusters_from_Scratch/index.html, acceduto maggio 2011.

[3] Test cases for cluster components in Ubuntu 10.04, https://wiki.ubuntu.com/ClusterStack/LucidTesting, acceduto maggio 2011.

[4] Logical Volume Manager, http://www.markus-gattol.name/ws/lvm.html, acceduto maggio 2011.

Working Paper Cnr-Ceris

ISSN (*print*): 1591-0709 ISSN (*on line*): 2036-8216

Download

↓

http://www.ceris.cnr.it/index.php?option=com_content&task=section&id=4&Itemid=64

Hard copies are available on request,
please, write to:

Cnr-Ceris
Via Real Collegio, n. 30
10024 Moncalieri (Torino), Italy
Tel. +39 011 6824.911 Fax +39 011 6824.966
segreteria@ceris.cnr.it http://www.ceris.cnr.it