ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/TC-02/31
19 Dicembre 2002

LINMGR - UNO STRUMENTO PER LA GESTIONE CENTRALIZZATA DEL SOFTWARE SU SISTEMI LINUX REDHAT

Roberto Gomezel, Tullio Macorini, Claudio Strizzolo, Lucio Strizzolo, Alessandro Tirel

INFN, Sezione di Trieste

Sommario

In questo documento viene presentata una soluzione, realizzata presso la Sezione di Trieste dell'INFN, per la gestione centralizzata dei calcolatori con sistema operativo Linux RedHat, con l'obiettivo di semplificare l'installazione e l'aggiornamento dei pacchetti software, e la gestione dei sistemi.

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

INDICE

1 INTRODUZIONE

La gestione di un numero elevato di calcolatori con sistema operativo Linux RedHat comporta un notevole carico di lavoro. L'installazione, ma soprattutto il continuo aggiornamento dei pacchetti software, anche per risolvere eventuali problemi di sicurezza dei sistemi, sono onerosi in termini di tempo, specialmente se il numero dei nodi da gestire è considerevole.

Presso la Sezione di Trieste è stata messa in opera una soluzione per tentare di semplificare queste operazioni. Tale soluzione è facilmente configurabile e per questo motivo potrebbe essere portabile facilmente anche in altri siti.

2 REQUISITI

Tutto il software che compone il sistema di gestione è stato realizzato in Perl. Il primo requisito per l'utilizzo del sistema è la presenza di una versione recente dell'interprete Perl su tutte le macchine.

Il sistema fa un utilizzo intenso della condivisione di file tramite NFS. Viene richiesto un server su cui installare il software e tutti i file di configurazione, in una struttura esportabile (in lettura ed esecuzione) a tutti i client (le macchine Linux RedHat da gestire centralmente) via NFS. Questo per evitare di dover duplicare la struttura in diversi punti, con conseguente rischio di disallineamento dei dati.

Il server NFS viene utilizzato anche per la distribuzione delle release del sistema operativo.

Presso la Sezione di Trieste il server NFS esporta la directory `/dist/common/`, contenente il software di gestione centralizzata, e le directory `/dist/`*`OSRELEASE`*`/` (ad es. /dist/valhalla), contenenti le distribuzioni delle diverse release dei sistemi operativi, gli aggiornamenti, ed i file di configurazione necessari per l'installazione del sistema operativo via NFS, secondo la struttura utilizzata nelle distribuzioni del sistema operativo RedHat sui CD di installazione. I client NFS montano la directory `/dist/common/` e `/dist/`*`OSRELEASE`*`/` del server rispettivamente in `/misc/common/` e `/misc/`*`OSRELEASE`*`/`.

Il sistema permette anche di ottenere via Web la visualizzazione dello stato di aggiornamento delle macchine. Per attivare questa funzionalità, è necessario installare su un server http uno script per la comunicazione client-server ed uno script CGI per la generazione dei report visualizzabili via Web.

Riassumendo:

- Su tutti i nodi (client):
  - Interprete Perl
  - NFS client
- Sul server:
  - NFS server
  - Installazione di una o più distribuzioni di Linux RedHat

- Sul server Web:
  - Server HTTP
  - Creazione di un account di servizio non privilegiato

3 INSTALLAZIONE DEL SISTEMA

Il software necessario all'implementazione della soluzione può essere richiesto al Servizio Calcolo e Reti della Sezione di Trieste.

Il pacchetto fornito comprende 3 directory, che devono essere installate come segue:

- La directory `buildks` può essere installata su una qualsiasi macchina, anche se di solito si utilizza il server NFS, in una directory non esportata via NFS e leggibile solamente da root. Questa directory contiene il software utilizzato nella procedura di installazione delle macchine.
- La directory `linuxupdate` deve essere installata sulla macchina designata quale server NFS, in una directory esportata a tutti i client (presso la Sezione di Trieste /dist/common/). Questa directory contiene il software utilizzato per la manutenzione e l'aggiornamento delle macchine.
- La directory `websrv` deve essere installata sul server Web, in una directory accessibile in lettura, scrittura ed esecuzione ad un account di servizio non privilegiato. Questa directory contiene il software utilizzato per la centralizzazione delle informazioni sulla configurazione delle macchine e per la loro visualizzazione via Web. Il file ./cgi-bin/rpmcheck_status.pl contenuto in questa directory deve essere copiato nella directory contenente gli script cgi del server Web.

Dopo aver installato tutti i file, è necessario creare sul server NFS le strutture di directory contenenti i file di configurazione del sistema. Normalmente le configurazioni vengono mantenute in una directory `configs` sotto la directory contenente il software; all'interno della directory `configs` viene di solito creata una directory (*`OSRELEASE`*) per ogni release di Linux RedHat utilizzata (ad es. valhalla, enigma, …).

I riferimenti a queste strutture sono contenuti nei file `buildks.pl` e `linuxupdate.pl` e sono personalizzabili. Di default è prevista la seguente struttura:

```
./buildks/configs/OSRELEASE/templates/
./buildks/configs/OSRELEASE/constants/
./buildks/ks/
./linuxupdate/configs/OSRELEASE/
```

Nei seguenti capitoli verrà descritta più dettagliatamente tale struttura.

4 INSTALLAZIONE DI UNA NUOVA MACCHINA

L'installazione di una nuova macchina viene effettuata utilizzando Kickstart, che è un programma di utilità per l'installazione di RedHat Linux. Le risposte alle varie domande che Kickstart propone durante la normale installazione possono essere scritte in un file di configurazione. Tale file è letto da Kickstart e la nuova installazione può essere portata a termine senza ulteriori interventi.

Kickstart può essere installato su un dischetto di boot assieme al file di configurazione menzionato. Per installare una nuova macchina, dopo averla collegata alla LAN, è necessario eseguire il boot da questo dischetto. La distribuzione del sistema operativo verrà scaricata dal server NFS.

Maggiori informazioni su Kickstart sono disponibili sul sito RedHat.

Il file di configurazione di Kickstart è particolare per ogni macchina. In esso, infatti, sono scritti tutti i parametri variabili della macchina da installare, ad esempio l'indirizzo IP e l'hostname. Alcuni di questi parametri possono essere comuni per classi di macchine, ad esempio per macchine che dovranno essere collegate alla medesima LAN, oppure tutti i PC portatili, e così via.

Queste configurazioni dipendono da vari fattori; per questo motivo è stato creato un programma di utilità per la generazione dei files di configurazione di Kickstart. Tale programma è in grado di "personalizzare" in maniera semplice ed automatica il file di configurazione sulla base delle esigenze appena evidenziate.

4.1 Generazione dei file di Kickstart: buildks.pl

Il programma per la generazione dei file di Kickstart utilizza un template generico come struttura comune per tutte le configurazioni, all'interno del quale è possibile identificare alcune sezioni che verranno sostituite da file di dati esterni. I file di dati sono organizzati secondo una struttura ad albero (ved. fig. 1), in cui ogni nodo identifica una diversa configurazione delle macchine. Il nodo "radice" dell'albero identifica la sezione del template generico da sostituire con un file di dati esterno, ad esempio "network" per la sezione di configurazione della rete, "postinstall" per la sezione di comandi da eseguire nella fase di postinstallazione di Kickstart, ecc.

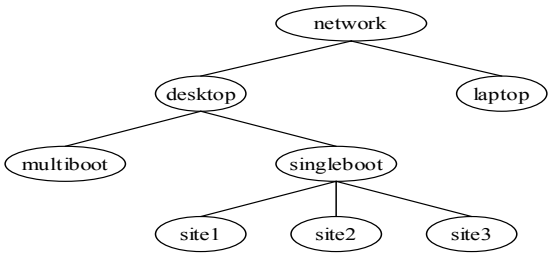

FIG. 1: Esempio di struttura ad albero dei file di dati.

La sintassi del programma per la generazione dei file di Kickstart è:

```
buildks.pl -c CLASSE -k HOSTNAME=HOSTNAME [-k OSRELEASE=OSRELEASE] [-k
                          ...] [-K ...] [-v]
```

Sulla riga di comando è necessario specificare la classe (con l'opzione `-c CLASSE`) ed il nome da assegnare alla macchina per la quale si sta creando il file di Kickstart (con l'opzione `-k HOSTNAME=HOSTNAME`). Il parametro classe specifica la configurazione della macchina, ed è utilizzato per determinare i file di dati da utilizzare all'interno della struttura ad albero precedentemente descritta. Nel caso della struttura di fig. 1, è possibile specificare ad esempio le seguenti classi: desktop.multiboot, desktop.singleboot.site2, laptop, ecc...

L'opzione `-v` permette di eseguire buildks.pl in modalità "verbose", ottenendo su standard output un numero maggiore di informazioni durante il processo di generazione del file di Kickstart.

Il programma prevede la seguente struttura di directory di configurazione, all'interno della directory in cui è installato il programma stesso:

`./configs/OSRELEASE/templates/`
È la directory contenente il template generico (`ks.default`) e i file di dati esterni.

`./configs/OSRELEASE/constants/`
È la directory contenente i file di costanti.

`./ks/`
È la directory in cui vengono creati i file di Kickstart (`ks.HOSTNAME`).

4.1.1 Definizione delle costanti

Buildks.pl permette di utilizzare delle costanti nella forma `%NOMECOSTANTE%` all'interno del template generico e dei file di dati. Quando viene generato il file di Kickstart, buildks.pl inserisce il valore assegnato alle costanti utilizzate. È possibile definire un numero illimitato di costanti.

Il valore delle costanti può essere assegnato in linea di comando con l'opzione

```
-k NOMECOSTANTE=valore
```

ad esempio:

```
-k NETMASK=255.255.255.0 -k NAMESERVER=10.11.12.13 -k SITE=site1
```

In alternativa, più costanti possono essere definite in un file esterno, che viene richiamato dalla linea di comando con l'opzione

`-K nomefile`

Nel file *nomefile* (che deve essere presente nella directory dei file di costanti, di solito `./configs/OSRELEASE/constants/`) su ogni riga viene definita una costante con la sintassi

NOMECOSTANTE=valore

Ad esempio:

```
NETMASK=255.255.255.0
NAMESERVER=10.11.12.13
SITE=site1
```

È importante ricordare che l'assegnazione dei valori alle costanti si effettua specificando *NOMECOSTANTE=valore*, mentre all'interno del template generico e dei file di dati le costanti vengono utilizzate specificando *%NOMECOSTANTE%*.

Le costanti indicate direttamente sulla linea di comando hanno priorità più alta rispetto alle costanti inserite in un file esterno, quindi in caso di una doppia definizione viene assegnato il valore indicato sulla linea di comando.

Buildks.pl prevede l'utilizzo di due costanti particolari:

- `HOSTNAME`: specifica il nome da assegnare alla macchina. Questa costante è utilizzata da buildks.pl per la composizione del nome del file di Kickstart generato (`ks.`*HOSTNAME*). Questa costante deve essere sempre definita.
- `OSRELEASE`: specifica il nome identificativo della release di Linux RedHat (ad es. valhalla, enigma, …). È utilizzata per la composizione dei nomi delle directory contenenti i file di configurazione (di solito `./configs/`*OSRELEASE*`/templates`) e i file di costanti (di solito `./configs/`*OSRELEASE*`/constants`).

4.1.2 Funzionamento di buildks.pl

Per prima cosa, è necessario avere a disposizione un template generico da utilizzare come struttura comune per tutte le configurazioni, all'interno del quale è necessario individuare le sezioni che possono essere personalizzate. Queste sezioni vengono quindi sostituite da righe di comando come:

`#INCLUDE nomesezione`

in cui *nomesezione* è un nome identificativo della sezione sostituita, ad esempio "network" per la configurazione della rete, "postinstall" per i comandi da eseguire nella fase di postinstallazione di Kickstart, ecc.

È quindi necessario preparare i file di dati esterni per buildks.pl, a cui fanno riferimento i comandi `#INCLUDE`. Questi file non sono altro che le sezioni appena sostituite; essi devono contenere righe di file come previsto per Kickstart, diversificate per le varie configurazioni. Anche all'interno di questi file possono essere inserite delle righe di comando `#INCLUDE` che si riferiscono a loro volta ad altri file esterni.

Sia nel template generico che nei file di dati è possibile inserire delle costanti nella forma *%NOMECOSTANTE%*. Durante la generazione del file di Kickstart, ad ogni costante verrà sostituito il valore ad essa assegnato (ved. Paragrafo 4.1.1).

Buildks.pl genera il file di Kickstart partendo dal template, inserendo al posto delle righe di comando `#INCLUDE` le linee contenute nei files inclusi, e al posto delle costanti *%NOMECOSTANTE%* il loro valore. La selezione del file di dati corretto e del valore delle costanti viene effettuata sulla base del parametro classe specificato sulla riga di comando di buildks con l'opzione `-c` *CLASSE*.

4.1.3 Struttura dei file "inclusi"

Ogni comando `#INCLUDE` fa riferimento ad un file esterno. Per consentire l'utilizzo di file di dati condivisi fra le varie configurazioni, buildks.pl determina il nome completo del file di dati combinando il *nomesezione* indicato nel comando `#INCLUDE` con la classe specificata sulla riga di comando. Buildks.pl prevede un'organizzazione dei file di dati con una struttura ad albero (ved. fig. 1), ed il parametro classe permette di raggiungere ogni elemento dell'albero.

Se ad esempio il comando specificato è:

```
#INCLUDE postinstall
```

e la classe è:

```
desktop.singleboot.site2
```

buildks.pl cercherà di includere uno dei seguenti file di dati, provando nell'ordine indicato e fermandosi al primo file disponibile:

```
postinstall.desktop.singleboot.site2
postinstall.desktop.singleboot
postinstall.desktop
postinstall
```

partendo dall'ultimo elemento dell'albero indicato in classe e risalendo di un livello ad ogni tentativo. Se ad esempio tutte le macchine desktop singleboot eseguono gli stessi comandi di postinstallazione, può essere sufficiente inserire le righe di Kickstart nel file `postinstall.desktop.singleboot`. Se invece una parte dei comandi è comune, mentre alcune righe differiscono per diversi gruppi di macchine (ad es. "site1", "site2", ecc...), in

`postinstall.desktop.singleboot` si possono inserire le righe comuni, mentre nei files `postinstall.desktop.singleboot.*` si devono inserire le righe che differiscono, più una riga `#INCLUDE postinstall.desktop.singleboot` in modo da includere le parti comuni. Ad esempio si potranno avere:

Nel file `postinstall.desktop.singleboot`:

```
cp -p /mnt/source/configs/pam.d/gdm /mnt/sysimage/etc/pam.d/gdm
cp -p /mnt/source/configs/pam.d/kde /mnt/sysimage/etc/pam.d/kde
cp -p /mnt/source/configs/pam.d/login /mnt/sysimage/etc/pam.d/login
cp -p /mnt/source/configs/pam.d/sshd /mnt/sysimage/etc/pam.d/sshd
cp -p /mnt/source/configs/pam.d/xdm /mnt/sysimage/etc/pam.d/xdm
```

Nel file `postinstall.desktop.singleboot.site1`:

```
#INCLUDE postinstall.desktop.singleboot
cp /mnt/source/configs/hosts /mnt/sysimage/etc/hosts
cp /mnt/source/configs/resolv.conf /mnt/sysimage/etc/resolv.conf
```

Nel file `postinstall.desktop.singleboot.site2`:

```
#INCLUDE postinstall.desktop.singleboot
rm -Rf /mnt/sysimage/var/spool/mail
cp /mnt/source/configs/syslog.conf /mnt/sysimage/etc/syslog.conf
```

4.2 Esecuzione guidata di buildks.pl: Wrapper.pl

Wrapper.pl è un'ulteriore script che è stata sviluppata per semplificare l'esecuzione di buildks.pl. Essa richiede solamente i pochi dati necessari per le diverse configurazioni, evitando così di specificare ogni volta un gran numero di costanti sulla riga di comando.

La script è localizzata tenendo presenti le esigenze specifiche della Sezione di Trieste, prevedendo l'utilizzo di specifiche classi e variabili, e richiedendo l'inserimento di dati diversi nelle diverse configurazioni; la script comunque è facilmente adattabile alle esigenze di altre realtà.

Per maggiore chiarezza, riportiamo di seguito alcuni esempi di funzionamento di wrapper.pl. Come si può notare i dati richiesti variano in base alla configurazione della macchina per la quale si sta creando il file di Kickstart.

- Creazione di Kickstart per una macchina desktop singleboot

```
[root]# ./wrapper.pl
System type [desktop|laptop]: desktop
Hostname: test
OS release: valhalla
Site [site1|site2|site3|site4]: site1
```

```
Boot [singleboot|multiboot]: singleboot
IP address: 10.11.12.13

That's all. I am going to execute the following script:
# ./buildks.pl -c desktop.singleboot.site1 -k HOSTNAME=test
-k IPADDR=10.11.12.13 -k TYPE=desktop -K CONSTANTS=site1
-k OSRELEASE=valhalla

Start [y|n]: y
```

- Creazione di Kickstart per una macchina laptop

```
[root]# ./wrapper.pl
System type [desktop|laptop]: laptop
Hostname: test
OS release: valhalla

That's all. I am going to execute the following script:
# ./buildks.pl -c laptop -k HOSTNAME=test -k TYPE=laptop
-k SITE="" -k OSRELEASE=valhalla

Start [y|n]: y
```

5 GESTIONE, MANUTENZIONE E AGGIORNAMENTO DELLE MACCHINE

Per quanto riguarda la gestione, la manutenzione e l'aggiornamento delle macchine, sono stati sviluppati degli script da eseguire su ogni client.

Le operazioni di gestione e controllo vengono normalmente eseguite al boot e tramite il servizio cron su ogni client. In particolare, presso la sezione di Trieste è stato deciso di operare come segue:

- Stazioni di lavoro desktop:
  - Verifica degli RPM installati: al boot e tramite cron.
    Inserire in `/etc/rc.local` la riga:

    ```
    /misc/common/linuxupdate/rpmcheck_client.pl > /dev/null 2>&1
    ```

    e nelle tabelle di cron di root (con il comando `crontab -e`) la riga:

    ```
    0,30 * * * * /misc/common/linuxupdate/rpmcheck_client.pl >
    /dev/null 2>&1
    ```

- Aggiornamento della configurazione: automatico tramite cron.
  Inserire nelle tabelle di cron di root (con il comando `crontab -e`) la riga:

  ```
  15 * * * 1,2,3,4,5 /misc/common/linuxupdate/linuxupdate.pl >
  /dev/null 2>&1
  ```

- Stazioni di lavoro laptop:
  - Verifica degli RPM installati: al boot.
    Inserire in `/etc/rc.local` la riga:

    ```
    /misc/common/linuxupdate/rpmcheck_client.pl > /dev/null 2>&1
    ```

  - Aggiornamento della configurazione: manuale.
    Quando necessario, da root viene eseguito il comando di aggiornamento:

    ```
    /misc/common/linuxupdate/linuxupdate.pl > /dev/null 2>&1
    ```

    È stato deciso di non effettuare automaticamente l'aggiornamento delle stazioni di lavoro laptop per evitare di eseguire le operazioni di aggiornamento nel caso in cui la stazione venga utilizzata all'esterno della rete locale della sezione.

Le informazioni sullo stato delle singole macchine vengono centralizzate utilizzando una comunicazione client-server verso il server Web; in questo modo è possibile ottenere la visualizzazione dello stato di aggiornamento delle macchine via Web mediante uno script CGI.

5.1 Controllo della configurazione degli RPM: rpmcheck.pl

La script rpmcheck.pl permette di ottenere su standard output un elenco dei pacchetti RPM mancanti e di quelli presenti ma non previsti nella configurazione di una macchina. La configurazione viene mantenuta sul server NFS, nella directory contenente la configurazione del software di gestione centralizzata (presso la Sezione di Trieste `./configs/OSRELEASE/`), in cui deve essere presente un file di dati per ogni client gestito (*`nomeclient`*`_list.txt`).

5.1.1 Sintassi dei file di dati

Per poter utilizzare rpmcheck.pl è necessario creare, sul server, un file di dati per ogni macchina, contenente la lista degli RPM che dovrebbero essere installati. All'interno di questo file è possibile inserire tre tipi di record:

- `+ nomerpm`: indica un RPM che dovrebbe essere installato.
- `- nomerpm`: indica un RPM che non dovrebbe essere installato.
- `@ commento`: riga di testo contenente un commento, ad esempio per una configurazione particolare del client o per la presenza di software non RPM.
- `#INCLUDE` *`nomefile`*: include le informazioni contenute nel file *`nomefile`*`_list.txt` presente nella stessa directory dei file di dati, che utilizza la medesima sintassi.

In questo modo è possibile utilizzare dei file di configurazione comuni, a cui ogni macchina può fare riferimento, ai quali è possibile aggiungere o rimuovere singoli RPM.

Ad esempio, è possibile creare un file di dati come il seguente:

```
#INCLUDE standard
#INCLUDE laptop
+ emacs-21.2-2
+ gcc-2.96-112
+ gcc-c++-2.96-112
- mpg321-0.2.9-3
- mtools-3.9.8-2
@ mozilla 1.1. in /home/
@ export di /usr/local+/
```

5.1.2 Esecuzione

Il programma interpreta il file di dati relativo alla macchina sulla quale è in esecuzione, creando una lista di RPM che dovrebbero essere installati sulla macchina. Confrontando questa lista con la lista di RPM effettivamente installati vengono determinati gli RPM mancanti e quelli presenti ma non previsti.

Il programma viene lanciato con la linea di comando:

```
# rpmcheck.pl
```

Sullo standard output si ottiene una lista di RPM; gli RPM mancanti vengono indicati nel formato `-nomerpm`, quelli presenti ma non previsti nel formato `+nomerpm`. Ad esempio:

```
+ghostscript-6.51-16
+fcpd-1.0.0-1
-ghostscript-6.51-16.2
```

indica la presenza di ghostscript-6.51-16 ed fcpd-1.0.0-1 (non previsti nella

configurazione) e la mancanza di ghostscript-6.51-16.2.

5.2 Centralizzazione dei dati di rpmcheck.pl e visualizzazione via Web

Il programma rpmcheck_client.pl esegue rpmcheck.pl ed invia l'output di questo comando al server Web mediante un socket TCP.

Sul server Web deve essere creato un account di servizio non privilegiato. Questo account viene utilizzato per controllare (operazioni di start e stop) un daemon in grado di ricevere i dati trasmessi dai client sul port prescelto, e di scriverli in un apposito spazio su disco. Normalmente il daemon viene fatto partire al boot della macchina o comunque non appena viene attivato il server Web.

Sul server Web deve essere stata installata la directory `websrv` del pacchetto fornito, che contiene la script per lo startup e lo shutdown del daemon (rpmcheck_server.pl), il file di configurazione del daemon (config.pl), il file contenente la lista dei client autorizzati a comunicare con il server Web (`authorized_clients.txt`) e la directory (`data/`) contenente i file (*`nomeclient`*`.txt`) in cui vengono scritti i dati ricevuti dai client. La directory `websrv` deve essere accessibile in lettura, scrittura ed esecuzione all'account di servizio non privilegiato.

Presso la Sezione di Trieste è stato deciso di disabilitare il login per l'account di servizio non privilegiato, e di eseguire rpmcheck_server.pl utilizzando il comando `su`.

La centralizzazione dei dati permette di ottenere la visualizzazione dello stato di aggiornamento delle macchine via Web mediante lo script CGI rpmcheck_status.pl installato nella directory contenente gli script CGI del server Web (ved. fig. 2).

Nella tabella vengono riportati i seguenti dati:

- Host: nome della macchina.
- OS rel.: stringa identificativa della release del sistema operativo; ad es: 7.2 (Enigma), 7.3 (Valhalla).
- Last Check: data ed ora dell'ultima esecuzione di rpmcheck.pl.
- + (plus): numero di RPM presenti sulla macchina ma non previsti nella configurazione.
- - (minus): numero di RPM non presenti sulla macchina ma previsti nella configurazione.
- Comments: numero di commenti sulla configurazione della macchina (corrisponde al numero di righe `@ commento` nel file di dati).
- Last Update: data ed ora dell'esecuzione dell'ultimo comando di aggiornamento (ved. par. 5.3).
- Last Update Command: stringa identificativa dell'ultimo aggiornamento eseguito (ved. par. 5.3).
- Last LinuxUpdate: data ed ora dell'ultima esecuzione di linuxupdate.pl (ved. par. 5.3).

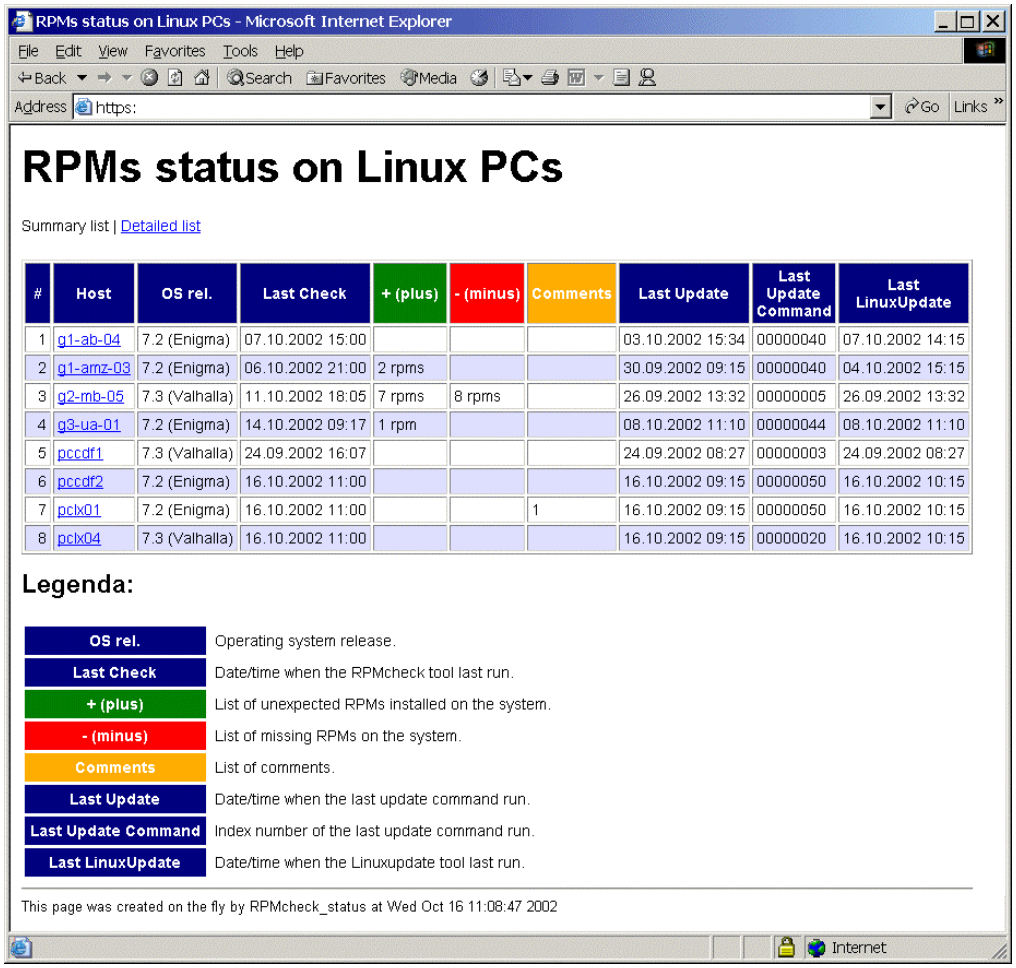

FIG. 2: Visualizzazione via Web dello stato di aggiornamento dei client.

È possibile ottenere una lista dettagliata (Detailed list) oppure la visualizzazione dettagliata della situazione di un singolo client facendo clic sul nome del client; in questi casi il contenuto di alcune colonne cambia come segue:

- + (plus): lista degli RPM presenti sulla macchina ma non previsti nella configurazione.
- - (minus): lista degli RPM non presenti sulla macchina ma previsti nella configurazione.
- Comments: commenti sulla configurazione della macchina.

5.2.1 Connessione client/server da parte di rpmcheck_client.pl

Il trasferimento dei dati dal client al server avviene per mezzo di una connessione TCP su un port non utilizzato del server, definito dall'amministratore.

Il server è in grado di gestire più connessioni di client in contemporanea, nel caso in cui più client trasmettano i dati nello stesso istante, senza eseguire forking di processi.

La connessione tra client e server avviene per mezzo di un protocollo molto semplice, che prevede un minimo di handshaking ed alcuni controlli da parte del server.

Quest'ultimo esegue delle verifiche minime sull'identità del client, e sui dati inviati, ed è in grado di chiudere o di ignorare la connessione qualora il client non comunichi le informazioni in modo corretto.

Il server accetta il trasferimento dei dati solamente da macchine che si trovano in una delle sottoreti "locali": questo viene fatto per evitare la comunicazione tra client e server per client esterni alla Sezione, siano questi ad esempio portatili utilizzati all'esterno della Sezione oppure macchine che cercano di accedere impropriamente alla comunicazione con il server.

5.3 Aggiornamento e sincronizzazione dei client: linuxupdate.pl

Questo programma esegue l'aggiornamento del software e la sincronizzazione della configurazione delle macchine Linux.

Sul server deve essere presente, nella directory contenente la configurazione del software di gestione centralizzata (presso la Sezione di Trieste `./configs/OSRELEASE/`), un file (`updates.upd`) contenente la sequenza dei comandi di aggiornamento da eseguire su tutte le macchine. Ad esempio:

```
00000001;/misc/common/linuxupdate/configs/valhalla/updates-scripts/f
lash-player-plugin-remove.sh
00000002;rpm -Fvh /misc/valhalla/updates/scrollkeeper-0.3.4-5.i386.r
pm 2>&1
00000003;rpm -Fvh /misc/valhalla/updates/gaim-0.59.1-0.7.3.i386.rpm
2>&1
00000004;rpm -Fvh /misc/valhalla/updates/tar-1.13.25-4.7.1.i386.rpm
2>&1
00000005;rpm -Fvh /misc/valhalla/updates/openafs-*.rpm 2>&1
00000006;rpm -Fvh /misc/valhalla/updates/tetex-*.i386.rpm 2>&1
00000007;rpm -Fvh /misc/valhalla/updates/xinetd-2.3.9-0.73.i386.rpm
2>&1
```

Ogni riga (record) è composta da due campi separati da un punto e virgola; il primo è una stringa ("indice") utilizzata per mantenere la sequenza dei comandi da eseguire correttamente ordinata, il secondo è il comando da eseguire. Per il corretto funzionamento del programma è necessario che il file sia ordinato in modo ascendente in base al primo campo di ogni record, in quanto questo campo viene utilizzato per tenere traccia dei comandi eseguiti e di quelli da eseguire.

Su ogni macchina, nella directory contenente i file di log di linuxupdate.pl (presso la Sezione di Trieste `/var/spool/linuxupdate/`), è presente un file (`linuxupdate.lst`)

contenente la stringa identificativa (il primo campo del record del file dei comandi), la data e l'ora dell'ultimo aggiornamento eseguito. Nella tabella ottenibile via Web questi ultimi dati vengono indicati come Last Update Command e Last Update. Se sulla macchina non è mai stato eseguito linuxupdate.pl, il file linuxupdate.lst non è presente. Quando viene lanciato linuxupdate.pl, vengono eseguiti in sequenza tutti i comandi indicati nel file presente sul server e non ancora eseguiti sulla macchina. Il confronto viene fatto sulla base della stringa presente nel file linuxudpate.lst. La struttura ad "indice" permette di tenere traccia degli aggiornamenti eseguiti e di quelli da eseguire; in questo modo è possibile aggiornare la configurazione di una macchina in qualsiasi momento.

Linuxupdate.pl aggiorna il file `linuxupdate.lst` dopo l'esecuzione di ogni comando presente nell'elenco esistente sul "server"; accoda l'output di ogni comando eseguito al file di log (`linuxupdate.log`) ed aggiorna il file (`linuxupdate.date`) contenente la data dell'ultimo aggiornamento eseguito (nella tabella ottenibile via Web questo dato viene riportato come Last LinuxUpdate).

5.4 Aggiornamento delle distribuzioni Linux: doupdate.pl

Ad ogni rilascio di nuove versioni di pacchetti RPM, è necessario inserire i nuovi file al posto di quelli più vecchi nelle directory di distribuzione del sistema operativo e degli aggiornamenti; presso la Sezione di Trieste queste directory sono rispettivamente `/dist/`*`OSRELEASE`*`/RedHat/RPMS` e `/dist/`*`OSRELEASE`*`/updates`.

A Trieste è stata creata una script per automatizzare, per quanto possibile, queste operazioni.

Gli RPM devono essere scaricati manualmente dal sito RedHat, ad esempio nella directory `/dist/`*`OSRELEASE`*`/updates-`*`AAAAMMGG`*`/`. Questa operazione non è stata automatizzata per permettere di scegliere di volta in volta quali RPM aggiornare.

Quindi si deve lanciare il programma con la linea di comando:

```
# doupdate.pl
```

ed inserire i dati richiesti:

```
OS release: OSRELEASE
Update dir: updates-AAAAMMGG
```

La script crea a sua volta una shell script `doupdate.`*`OSRELEASE`*`.updates-`*`AAAAMGG`*`.sh`; lanciando questa script vengono eseguite le seguenti operazioni:

- Rimozione di eventuali vecchie versioni dei file RPM dalle directory di distribuzione.
- Link alle nuove versioni dei file RPM dalle directory di distribuzione.
- Generazione dei file `/dist/`*`OSRELEASE`*`/RedHat/base/hdlist*` utilizzati in fase

di installazione.

- Aggiornamento dei file di dati per rpmcheck.pl (`/dist/common/linuxupdate/configs/`*`OSRELEASE`*`/*_list.txt`).

Al termine dell'esecuzione, è necessario inserire manualmente i comandi di aggiornamento nel file `updates.upd`.

La script è stata sviluppata per le esigenze specifiche della sezione di Trieste, ma è facilmente adattabile alle esigenze di altre realtà.