

CRESCO - SPI.2

Strumenti di monitoraggio e aggiornamento

Deliverable 2° obiettivo realizzativo intermedio

Codice Commessa:	ENE06008
Autore:	Team di Progetto (Anna Jannace)
Versione:	x.yy
Data:	01 Dicembre 2007
Verificato da:	Verificatore Nome Cognome
Approvato da:	Approvatore Nome Cognome
Lista di distribuzione:	

Controllo della configurazione

Titolo: Strumenti di monitoraggio e aggiornamento

Riferimento: SPI.2. Strumenti di monitoraggio e aggiornamento.doc

Creato da: Anna Jannace

Storia del documento

Ver.	Stato	Autore	Data	Memorizzato in
0	Iniziale	Jannace	01/12/07	"C:\Documents and Settings\jannace\Documenti\Jannace\CRESCO\per capirci\relazioni"

Storia delle revisioni

Ver.	Data	Autore	Verif.	Approv.	Motivo Revisione

Modifiche previste

Al momento non sono ipotizzabili le modifiche necessarie.

Stampato il: 01 Dicembre 2007

CRESCO - SPI.2

Strumenti di monitoraggio e aggiornamento

Deliverable 2° obiettivo realizzativo intermedio

Abstract

La ricerca è stata effettuata nell'ambito delle attività dello stack SP.I.2 del Progetto PON Cresco (Centro computazionale di RicErca sui Sistemi COMplessi) ENEA.

Il progetto ha come obiettivo la realizzazione, presso il Centro Ricerche ENEA di Portici (NA), di un importante Polo di calcolo multidisciplinare per lo studio dei sistemi complessi di natura biologica e tecnologica, collegato con facilities di elaborazione, archiviazione e visualizzazione 3D di dati ubicate presso i Centri ENEA di Brindisi e Trisaia (MT).

Il sottoprogetto SPI consiste nella realizzazione del Polo di calcolo e nello sviluppo di nuove funzionalità di GRID Computing.

In questo contesto si inserisce anche la scelta di un adeguato strumento di monitoraggio delle risorse disponibili in ENEA-GRID con opportuni script di aggiornamento e misura idonei all'infrastruttura stessa. Il presente documento contiene una visione generale del sistema ipotizzato, descrivendone funzionalità e caratteristiche, al fine di effettuare una scelta mirata e consapevole delle tecnologie da utilizzare nella implementazione della relativa prospettiva all'interno del progetto.

Indice dei contenuti

Abstract	iv
Indice dei contenuti	v
Indice delle tabelle	vii
Indice delle figure	viii
0. Introduzione	9
1. Scopo e Campo di applicazione	10
2. Strumenti di monitoraggio e aggiornamento	11
2.1.1 <i>Ganglia</i>	11
2.1.2 <i>Nagios</i>	12
3. I motivi della scelta	14
4. Gmetad	17
4.1 Installazione	17
4.2 Configurazione	18
5. Gmond	21
5.1 Installazione	21
5.2 Configurazione	22
6. Web Front End	24
6.1 Installazione	25
6.2 Configurazione	25
7. RRD Tool	27
7.1 Installazione	27
8. Gmetric e Gstat	29

8.1 Gmetric.....	29
8.2 Gstat.....	30
9. Gexec.....	32
9.1 Authd.....	32
10. Studio delle metriche.....	33
11. Funzionalità del sistema (focus sul monitoraggio).....	35
12. Architettura del sistema	37
12.1 Server Centrale	37
12.2 Server di Sito.....	37
12.3 Nodo.....	38
13. Componenti del sistema (focus sul monitoraggio).....	39
13.1 Database delle Risorse	39
13.1.1 Mago.sql.....	39
13.1.2 Tabelle statiche in AFS	39
13.2 Strumenti di monitoraggio e aggiornamento.....	39
13.2.1 Ambiente locale MAGO.	40
13.2.2 magoupdate.sh	40
13.2.3 Scripts di misura delle metriche aggiuntive.....	41
14. Conclusioni finali.....	42
15. Eventuali sviluppo futuri	43
16. Riferimenti Bibliografici	44
17. Definizioni.....	45
Appendice.A Prima Appendice	46

Indice delle tabelle

Tabella n.1 -	<i>Esempio di file gmnetad.conf.....</i>	<i>20</i>
Tabella n.2 -	<i>Esempio di assegnazione al gruppo multicast di indirizzo</i>	
<i>239.2.11.71</i>	<i>23</i>	
Tabella n.3 -	<i>File conf.php configurazione di default.....</i>	<i>25</i>
Tabella n.4 -	<i>Visualizzazione sintassi di gmetric.....</i>	<i>29</i>
Tabella n.5 -	<i>Visualizzazione sintassi di gstat.....</i>	<i>31</i>
Tabella n.6 -	<i>Metriche di Ganglia.....</i>	<i>34</i>

Indice delle figure

Figura n.1 -	<i>Testbed CRIA</i>	14
Figura n.2 -	<i>I tre livelli dell'Architettura MAGO</i>	37
Figura n.3 -	<i>Visione generale del sistema</i>	38

0. Introduzione

In questo documento si descrivono le caratteristiche del sistema denominato genericamente **MAGO: Monitoring All GRID Objects**.

Sotto il nome di “**MAGO: Monitoring All GRID Objects**” si indica qui il sistema completo che comprende al suo interno varie funzionalità:

- a. La memorizzazione di tutti i dati riguardanti le risorse disponibili nella rete di calcolo ENEA-GRID
- b. Il monitoraggio di tutte le risorse della rete di calcolo ENEA-GRID**
- c. La comunicazione tra il database suddetto ed il sistema di ENEA-GRID

Il focus di questo documento è centrato sul punto b. per descrivere dettagliatamente gli strumenti di monitoraggio e aggiornamento.

Nel prossimo paragrafo si descrive lo scenario di riferimento.

1. Scopo e Campo di applicazione

Il progetto PON Cresco ha come obiettivo la realizzazione, presso il Centro Ricerche ENEA di Portici (NA), di un importante Polo di calcolo multidisciplinare per lo studio dei sistemi complessi di natura biologica e tecnologica, collegato con facilities di elaborazione, archiviazione e visualizzazione 3D di dati ubicate presso i Centri ENEA di Brindisi e Trisaia (MT).

Di seguito si descrive il capitolo I.2.4 della attività programmatica di CRESCO con titolo “Analisi e Realizzazione di tool innovativi a supporto delle funzionalità GRID” al fine di esporre il motivo dell'utilizzo di Ganglia e le potenzialità che se ne intendono sfruttare.

Il capitolo I.2.4 della attività programmatica di CRESCO con titolo “Analisi e Realizzazione di tool innovativi a supporto delle funzionalità GRID” si rivolge esplicitamente ai temi del “Monitoring”, “Resource Discovery”, “Accounting” e “Semantic & Knowledge GRID”.

L'obiettivo è estendere in modo innovativo le potenzialità della attuale infrastruttura ENEA-GRID, all'interno della quale viene collocata la realizzazione del progetto CRESCO. A tal fine si assumono come vincoli i componenti costitutivi di base di ENEA GRID, ovvero il sistema di gestione delle risorse basato sul prodotto LSF Multicluster e il file system distribuito OpenAFS.

Questa attività specifica dovrà portare ad arricchire in modo innovativo le funzionalità di ENEA-GRID in modo compatibile con l'infrastruttura esistente, estendendole in particolare in quelle aree che le componenti attualmente utilizzate non riescono a coprire in modo pienamente soddisfacente.

Il progetto si basa su GANGLIA e si propone di integrare le informazioni fornite di default da ganglia con informazioni su demoni, servizi e software che sono specifici delle infrastruttura di ENEA-GRID.

L'insieme di tutte le informazioni viene centralizzato su un DB, implementato in MySQL.

L'implementazione fa uso di AFS, componente di base di ENEA-GRID per la distribuzione delle informazioni statiche di configurazione ma identifica in modo esplicito tale approccio in modo che sia possibile implementare le stesse funzionalità attraverso altri metodi in ambienti che non fanno uso di AFS.

2. Strumenti di monitoraggio e aggiornamento

Si descrivono in seguito le caratteristiche salienti dei due sistemi di monitoraggio Open Source più conosciuti, dalle quali si evince che la scelta per ENEA-GRID è caduta su Ganglia come dettagliatamente specificato nel relativo paragrafo.

2.1.1 *Ganglia*

Ganglia è un sistema di monitoraggio scalabile e distribuito per sistemi di computazione ad alte prestazioni come i clusters ed i Grids.

È basato su un disegno gerarchico dedicato alle federazioni di clusters. Poggia su un protocollo richiesta/risposta basato sul multicast per monitorare lo stato tra i clusters ed usa un albero di connessioni punto-punto attraverso i nodi rappresentativi dei clusters ed aggregare i loro stati.

Fa leva su tecnologie ampiamente usate quali XML per la rappresentazione di dati, XDR per trasporto compatto e portatile di dati e RRDtool per salvataggio dei dati e visualizzazione.

Usa con attenzione strutture di dati ed algoritmi per realizzare overheads molto bassi per-nodo ed alta concorrenza. L'implementazione è robusta, portata ad un vasto insieme di sistemi operativi e di architetture di processori, ed è attualmente in uso su oltre 500 cluster nel mondo. È stato usato per collegare clusters attraverso campus universitari ed intorno al mondo e può regolare per maneggiare clusters con 2000 nodi.

Il sistema Ganglia è costituito da due demoni unici, un web frontend basato sul PHP ed alcuni altri piccoli programmi pratici.

Il sistema memorizza i dati in un database RRD (Robin Round Database) ed attraverso rrdtool visualizza i grafici richiesti dall'utente attraverso la web application.

Le sonde sono installabili in hosts con qualsiasi sistema operativo.

Il demone server si chiama “gmetad”, la sonda si chiama “gmond”.

Questo strumento è sicuramente apprezzato da sistemisti linux / unix che devono gestire ambienti eterogenei e complessi. Può inoltre essere parte di architetture GRID in cui Ganglia si pone come strumento di monitoraggio delle risorse distribuite o come strumento di supporto al bilanciamento di carico.

2.1.2 Nagios

Nagios® è una applicazione di monitoraggio dei sistemi e delle reti. Osserva gli hosts ed i servizi specificati, avvisando quando le cose vanno male e quando vanno meglio.

Nagios originalmente è stato disegnato per funzionare sotto Linux, anche se dovrebbe funzionare sotto la maggior parte degli altri unices pure.

Alcune delle molte caratteristiche di Nagios® includono:

- Verifica e Controllo dei servizi di rete (smtp, POP3, HTTP, NNTP, RUMORE METALLICO, ecc.)
- Verifica e Controllo delle risorse ospite (carico del processor, uso di memoria e del disc, processi correnti, lime di ceppo, ecc.)
- Verifica e Controllo dei fattori ambientali quale la temperatura
- Semplici disegni di plugin che consentono agli utenti di sviluppare facilmente i propri controlli di servizi
- Controlli di servizi paralleli
- Capacità di definire la gerarchia di rete tra gli host usando "parent" hosts, permettendo la rilevazione di una distinzione tra gli host che sono down e quelli che sono riavviabili
- Mette in contatto con le notifiche quando accadono problemi di host o di servizio e quando vengono risolti (via email, pager, o metodi definiti dall'utente)
- Escalation facoltativa delle notifiche di servizio ed ospite ai gruppi differenti del contatto
- Tempo morto previsto per le notifiche supressing di servizio ed host durante i periodi dei guasti previsti
- Interfaccia web opzionale per visualizzare lo stato corrente della rete, la storia delle notifiche e dei problemi, il file di log, etc
- Schema semplice di autorizzazione che concede limitate che utenti possono vedere e fare dall'interfaccia di fotoricettore
- Capacità di definire collegamenti tra eventi per essere eseguiti durante gli eventi host o servizi per la risoluzione proattiva del problema
- Rotazione automatica del file di log
- Supporto per l'implementazione ridondante del monitoraggio degli hosts

La sola richiesta per l'esecuzione di Nagios è una macchina con SO Linux (o variante UNIX) e un compilatore C.

Probabilmente bisognerà avere il protocollo TCP/IP configurato, come la maggior parte dei servizi di rete da controllare.

Scaricando l'ultima releasedi Nagios si avrà solamente il nucleo del programma che è in grado di eseguire i plugins, mandare le notifiche e generare l'interfaccia Web.

Le sonde sono prevalentemente plugin eseguibili mediante il client NRPE (Nagios Remote Plugin Executor). È il client che esegue in remoto i plugin e restituisce al server Nagios il codice di ritorno (1, 2, 0). Il suo file di configurazione è *nrpe.cfg* che contiene i path dei plugin.

I file di configurazione di Nagios sono:

- *cgi.cfg* e *nagios.cfg* che contengono i parametri generali per l'ambiente
- *checkcommands.cfg* che contiene i path e la sintassi dei plugin
- *contact.cfg* e *contactgroups.cfg* che contengono i contatti per le notifiche
- *host.cfg*, *hostgroups.cfg*, *hostextinfo.cfg* e *services.cfg* che contengono le configurazioni degli oggetti monitorati
- *Timeperiods* che contiene le fasce orarie per i check e le notifiche
- *dependencies.cfg* e *escalations.cfg* che contengono la funzioni avanzate.

3. I motivi della scelta

Dopo un'accurata fase di scouting è stato scelto il sistema di monitoraggio GANGLIA.

Il CRIAI ha messo a disposizione la propria rete come testbed del sistema di monitoraggio, costruito in modo da **emulare** le caratteristiche dell'infrastruttura dei nodi di calcolo ENEA nella quale il sistema andrà in produzione.

Il CRIAI sta realizzando la sperimentazione controllando sia le metriche di default di Ganglia che misure aggiuntive opportunamente implementate.

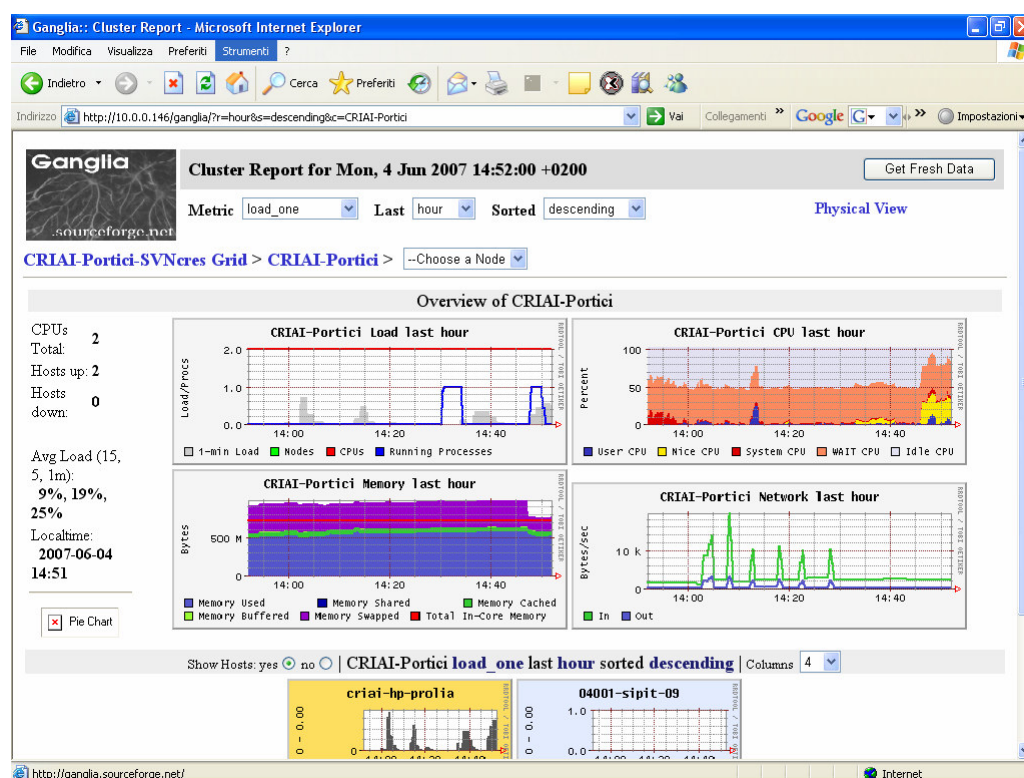


Figura n.1 - Testbed CRIAI

Ganglia è un sistema di monitoraggio distribuito che permette di controllare diversi parametri di un gruppo di nodi di calcolo, come:

- carico del processore

- uso della memoria
- I/O
- spazio disco
- ...

Ganglia è un sistema di monitoraggio scalabile e distribuito per sistemi di calcolo ad alte prestazioni come i **cluster** ed i **Grid**.

È basato su un disegno gerarchico destinato alle federazioni di cluster.

Poggia su un protocollo **request/response** per monitorare lo stato **tra i cluster** ed usa un albero di **connessioni punto-punto attraverso i nodi** rappresentativi dei clusters basato sul multicast per aggregare i loro stati.

Fa leva su tecnologie consolidate e diffuse quali :

- **XML** per il trasporto dei dati
- **XDR** per la portabilità dei dati
- **RRDtool** per il salvataggio dei dati e visualizzazione grafica

Usa strutture di dati ed algoritmi che realizzano un overhead molto basso su ciascun nodo e favoriscono l'alta concorrenza dell'insieme dei nodi.

L'implementazione è robusta, portata ad un vasto insieme di sistemi operativi e di architetture di processori, ed è attualmente in uso su moltissimi cluster nel mondo.

È stato usato per collegare cluster attraverso campus universitari ed intorno al mondo e può essere configurato per controllare cluster con una grande quantità di nodi.

È un sistema apprezzato in ambienti linux / unix per gestire siti eterogenei e complessi.

Può essere integrato in architetture **GRID**, come **strumento di monitoraggio delle risorse distribuite** (performance evaluation) o come strumento di **supporto al bilanciamento di carico** (capacity planning)

Queste osservazioni rilevate durante la fase di scouting e verificate nella fase di sperimentazione e soprattutto **un'attenta valutazione delle risorse presenti nell'infrastruttura ENEA-Grid** come il file system distribuito AFS e lo scheduler LSF, hanno determinato la scelta di Ganglia come sistema più idoneo a sfruttare le potenzialità intrinseche di ENEA-Grid.

4. Gmetad

Il collegamento in Ganglia è realizzato usando un albero di connessioni punto-punto tra i nodi rappresentativi del cluster per aggregare lo stato di clusters multipli.

Ad ogni nodo nell'albero, un demone Ganglia Meta Daemon (`gmetad`) periodicamente ottiene una collezione di dati dalle sorgenti discendenti, analizza l'XML collezionato, salva tutte le metriche numeriche volatili nel database round-robin ed esporta gli XML aggregati sulle socket TCP verso i clients.

Le sorgenti di dati potrebbero essere i demoni `gmond`, che rappresentano specifici clusters.

Le sorgenti dati usano l'indirizzo IP di provenienza per Data sources use source IP addresses per il controllo di accesso e possono essere specificati usando indirizzi IP multipli per il failover.

La seconda caratteristica è naturale per l'aggregazione di dati dai clusters dal momento in cui ciascun demone `gmond` contiene l'intero stato del suo cluster.

4.1 Installazione

Pur monitorando i nodi di un cluster indipendentemente dal sistema operativo dei nodi stessi, è necessario installare il demone server su una macchina con SO Linux.

Dalle analisi compiute l'installazione più efficiente risulta essere quella effettuata mediante pacchetto rpm.

Si descrivono di seguito i passi per una corretta installazione.

1. scaricare dal sito ufficiale il pacchetto rpm relativo alla distribuzione Linux ed all'architettura del processore della macchina (`ganglia-gmetad-3.0.3-1.fc4.i386.rpm`)

```
> cd /usr/local/tmp
```

```
> wget http://downloads.sourceforge.net/ganglia/ganglia-gmetad-3.0.3-1.fc4.i386.rpm?modtime=1145294052&big_mirror=0
```

2. installare il pacchetto mediante comando

```
> rpm -Uvh ganglia-gmetad-3.0.3-1.fc4.i386.rpm
```

3. una volta terminata l'installazione, il demone viene avviato automaticamente quindi per configurarlo è necessario fermarlo col comando

```
> service gmetad stop
```

4.2 Configurazione

Il comportamento del Ganglia Meta Daemon è completamente controllato da un singolo file di configurazione che per default è
`/etc/gmetad.conf`

Per gmetad per fare qualcosa di utile devi specificare almeno una sorgente di dati nella configurazione.

Il formato della riga “data_source” è il seguente

```
data_source "Cluster A" 127.0.0.1 1.2.3.4:8655  
data_source "Cluster B" 1.2.4.4:8655
```

In questo esempio ci sono due sole sorgenti di dati, il “Cluster A” ed il “Cluster B”.

Il data source “Cluster A” ha due sorgenti ridondanti. Se gmetad non può ottenere i dati dalla prima sorgente, continuerà a cercare le altre sorgenti in ordine.

È importante ricordare in questo punto che il sistema Ganglia funziona in multicast, questo vuol dire che per monitorare una intera sottorete di un cluster è sufficiente puntare ad uno qualsiasi degli host della sottorete (che sia necessariamente una macchina con SO Linux) perché vengano monitorati tutti gli host della stessa sottorete che dichiarano di appartenere al suo stesso gruppo multicast, nella modalità che verrà descritta nel prossimo paragrafo.

Se non si specifica un numero di porta, gmetad assume che la porta di default per ganglia è la 8649.

Ecco un esempio di configurazione del file gmetad.conf con i commenti.

```
# This is an example of a Ganglia Meta Daemon configuration file  
# http://ganglia.sourceforge.net/  
#  
#-----  
# Setting the debug_level to 1 will keep daemon in the foreground and  
# show only error messages. Setting this value higher than 1 will make  
# gmetad output debugging information and stay in the foreground.  
# default: 0  
# debug_level 10  
#  
#-----
```

```
# What to monitor. The most important section of this file.
#
# The data_source tag specifies either a cluster or a grid to
# monitor. If we detect the source is a cluster, we will maintain a complete
# set of RRD databases for it, which can be used to create historical
# graphs of the metrics. If the source is a grid (it comes from another gmetad),
# we will only maintain summary RRDs for it.
#
# Format:
# data_source "my cluster" [polling interval] address1:port addreses2:port ...
#
# The keyword 'data_source' must immediately be followed by a unique
# string which identifies the source, then an optional polling interval in
# seconds. The source will be polled at this interval on average.
# If the polling interval is omitted, 15sec is assumed.
#
# A list of machines which service the data source follows, in the
# format ip:port, or name:port. If a port is not specified then 8649
# (the default gmond port) is assumed.
# default: There is no default value
#
# data_source "my cluster" 10 localhost my.machine.edu:8649 1.2.3.5:8655
# data_source "my grid" 50 1.3.4.7:8655 grid.org:8651 grid-backup.org:8651
# data_source "another source" 1.3.4.7:8655 1.3.4.8

data_source "my cluster" localhost

#
#-----
# Scalability mode. If on, we summarize over downstream grids, and respect
# authority tags. If off, we take on 2.5.0-era behavior: we do not wrap our output
# in <GRID></GRID> tags, we ignore all <GRID> tags we see, and always assume
# we are the "authority" on data source feeds. This approach does not scale to
# large groups of clusters, but is provided for backwards compatibility.
# default: on
# scalable off
#
#-----
# The name of this Grid. All the data sources above will be wrapped in a GRID
# tag with this name.
# default: Unspecified
# gridname "MyGrid"
#
#-----
# The authority URL for this grid. Used by other gmetads to locate graphs
# for our data sources. Generally points to a ganglia/
# website on this machine.
# default: "http://hostname/ganglia/";,
# where hostname is the name of this machine, as defined by gethostname().
# authority "http://mycluster.org/newprefix/";
#
#-----
# List of machines this gmetad will share XML with. Localhost
# is always trusted.
```

```
# default: There is no default value
# trusted_hosts 127.0.0.1 169.229.50.165 my.gmetad.org
#
#-----
# If you want any host which connects to the gmetad XML to receive
# data, then set this value to "on"
# default: off
# all_trusted on
#
#-----
# If you don't want gmetad to setuid then set this to off
# default: on
# setuid off
#
#-----
# User gmetad will setuid to (defaults to "nobody")
# default: "nobody"
# setuid_username "nobody"
#
#-----
# The port gmetad will answer requests for XML
# default: 8651
# xml_port 8651
#
#-----
# The port gmetad will answer queries for XML. This facility allows
# simple subtree and summation views of the XML tree.
# default: 8652
# interactive_port 8652
#
#-----
# The number of threads answering XML requests
# default: 4
# server_threads 10
#
#-----
# Where gmetad stores its round-robin databases
# default: "/var/lib/ganglia/rrds"
# rrd_rootdir "/some/other/place"
```

Tabella n.1 - Esempio di file gmnetad.conf

Inoltre gmetad ha una opzione `--conf` per consentire di specificare i file di configurazione alternativa, ecco il comando

```
> gmetad -conf=/tmp/my_custom_config.conf
```

5. Gmond

Gmond è un demone multi-thread che è attivo su ciascun nodo del cluster che si vuole monitorare.

L'installazione è facile. Non è necessario avere un filesystem NFS comune o un database backend, installare account speciali, mantenere file di configurazione o altre fastidiose incombenze.

Gmond ha quattro responsabilità principali:

1. monitorare i cambiamenti di stato negli host,
2. annunciare i cambiamenti rilevanti,
3. ascoltare lo stato di tutti gli altri nodi ganglia attraverso un canale unicast o multicast
4. e rispondere alla richiesta con una descrizione XML dello stato del cluster.

Ciascun gmond trasmette informazioni in due differenti modi: stato degli host nel formato XDR (external data representation) usando messaggi UDP in unicast/multicast o inviando XML su una connessione TCP.

5.1 Installazione

Il demone sonda gmond deve essere installato su ciascun host da monitorare ed è installabile su qualsiasi sistema operativo.

Le prove effettuate e quindi le installazioni descritte sono relative alle seguenti piattaforme:

- SO Linux Fedora Core
- SO Windows XP

I rispettivi pacchetti scaricati dal sito Ganglia al link dei downloads, http://sourceforge.net/project/showfiles.php?group_id=43021&package_id=35280 sono :

- [ganglia-gmond-3.0.3-1.fc4.i386.rpm](#) per le macchine con SO Linux
- [ganglia-3.0.0-setup.exe](#) per le macchine con SO Windows

L'installazione del pacchetto rpm nel SO Linux viene effettuata col comando

```
> rpm -Uvh ganglia-gmond-3.0.3-1.fc4.i386.rpm
```

L'installazione in Windows viene effettuata mediante doppio click sull'eseguibile scaricato.

Una volta terminata l'installazione, il demone viene avviato automaticamente quindi per configurarlo è necessario fermarlo.

Nel SO Linux col comando

```
> service gmond stop
```

Nel SO Windows con

MenuAvvio -> Programmi -> ganglia -> Stop Ganglia
Monitoring Service

5.2 Configurazione

A seguito dell'installazione è possibile configurare il servizio mediante il file `gmond.conf` presente sull'host da monitorare.

Nel SO Linux esso si trova direttamente nella directory
`/etc.`

Nel SO Windows si trova nella directory di installazione di Ganglia, raggiungibile mediante

MenuAvvio -> Programmi -> ganglia -> Configure
Ganglia Monitoring Service

Questo file è importante per indicare i gruppi multicast di appartenenza, mediante i parametri `udp_send_channel` e `udp_recv_channel`, assegnando lo stesso multicast join per tutti gli hosts che intendono essere monitorati in quel gruppo multicast.

Attenzione è importante che il master host, quello che scambia i messaggi direttamente col server gmetad sia una macchina con SO Linux.

In realtà dopo l'installazione viene assegnato per default il seguente indirizzo multicast:

```
mcast_join = 239.2.11.71
```

e da questo possono essere poi stabiliti differenti gruppi multicast assegnando progressivamente gli indirizzi successivi (239.2.11.72, 239.2.11.73, etc.).

```
udp_send_channel {  
    mcast_join = 239.2.11.71
```

```
host = 127.0.0.1
port = 8649
}
udp_recv_channel {
  mcast_join = 239.2.11.71
  port = 8649
}
```

Tabella n.2 - *Esempio di assegnazione al gruppo multicast di indirizzo
239.2.11.71*

6. Web Front End

Il web frontend di Ganglia fornisce una visione delle informazioni raccolte attraverso pagine web dinamiche in tempo reale. Maggiormente importante, esso visualizza i dati Ganglia in un modo significativo per gli amministratori di sistemi e gli utenti di computer. Benchè il web frontend di ganglia parte come una semplice vista HTML di un albero XML, è evoluto in un sistema che mantiene una storia colorata di tutti i dati collezionati.

Il web frontend di Ganglia soddisfa gli amministratori di sistemi e gli utenti.

Per esempio, uno può visualizzare l'utilizzo della CPU durante l'ora passata, il giorno, la settimana, il mese o l'anno. Il web frontend mostra grafici simili per l'utilizzo della memoria, l'utilizzo del disco, le statistiche di rete, il numero di processi in esecuzione e tutte le altre metriche di Ganglia.

Il web frontend dipende dall'esistenza del gmetad che lo rifornisce con i dati dei molteplici sorgenti Ganglia.

In particolare, il web frontend aprirà la porta locale 8651 (per default) ed aspetta di ricevere un albero XML di Ganglia. Le pagine web in sé sono altamente dinamiche; ogni cambiamento ai dati Ganglia appare immediatamente sul sito. Questo comportamento conduce ad un sito molto sensibile, ma richiede l'intero albero XML sia analizzato ad ogni accesso alla pagina. Di conseguenza, il web frontend Ganglia dovrebbe essere eseguito su una macchina dedicata abbastanza potente se è presente una grande quantità di dati.

Il web frontend può dunque essere installato su una macchina differente dal server Ganglia (cioè quella dove viene eseguito il demone gmetad), per la configurazione si rimanda al relativo paragrafo.

Il web frontend Ganglia è scritto nel linguaggio di scripting PHP, e usa grafici generati dal gmetad per visualizzare la storia delle informazioni.

È stato testato su molte distribuzioni Unix (principalmente Linux) con il webserver Apache ed il modulo PHP 4.1.

6.1 Installazione

Anche il web frontend va installato preferibilmente su una macchina con SO Linux.

Il pacchetto [ganglia-web-3.0.3-1.noarch.rpm](http://sourceforge.net/project/showfiles.php?group_id=43021&package_id=35280) è scaricabile dal sito Ganglia al link dei downloads,
http://sourceforge.net/project/showfiles.php?group_id=43021&package_id=35280.

L'installazione del pacchetto rpm nel SO Linux viene effettuata col comando

```
> rpm -Uvh ganglia-web-3.0.3-1.noarch.rpm
```

Basta questa sola operazione per rendere eseguibile la web application direttamente dal browser.

6.2 Configurazione

A seguito dell'installazione è possibile visualizzare la pagina web iniziale digitando nella barra degli indirizzi del browser:

`http://IpAddress/ganglia`

dove `IpAddress` è l'indirizzo IP della macchina sulla quale è installato il web frontend di Ganglia.

Nella configurazione di default che viene avviata direttamente con l'installazione del web frontend, viene invocato il servizio gmetad che risiede in "localhost", cioè sulla stessa macchina dove è in esecuzione la web application stessa.

Nel caso in cui si scelga di installare il web frontend su una macchina diversa dal server, è necessario effettuare la seguente modifica.

Nel file `/var/www/html/ganglia/conf.php` si deve valorizzare il parametro `$ganglia_ip` con l'indirizzo IP del server dove risiede gmetad.

```
$ganglia_ip = "127.0.0.1";  
$ganglia_port = 8652;
```

Tabella n.3 - File `conf.php` configurazione di default



7. RRD Tool

RRD è l'acronimo per Round Robin Database. Esso è un sistema per memorizzare e visualizzare dati in serie temporale (ad esempio larghezza di banda delle reti, temperatura della sala macchina, media di caricamento del server,...).

Il termine round-robin è usato in molti contesti ed usualmente significa che un numero di cose sta aspettando il turno per qualcosa, per esempio in UK “round-robin” è frequentemente usato per indicare una lettera copiata ed inviata a molti destinatari. Nello sport un round-robin è spesso il termine usato per le fasi di gruppo di un torneo nel quale ciascun concorrente o team compete contro ciascun altro nel loro gruppo, la vittoria dei competitori passa attraverso un passaggio di eliminazione del torneo.

RRDtool è un round robin database creato da Tobias Oetiker. Esso è designato per maneggiare dati cronologici.

Il dato è memorizzato in un database round robin così che lo spazio occupato dal sistema di memorizzazione rimane costante nel tempo.

Inoltre include i tools per estrarre i dati RRD in forma grafica.

RRDtool ha la licenza GNU GPL.

Attenzione con l'installazione del pacchetto rpm del demone server gmetad, viene fornito anche il database rrd. Infatti nella cartella `/var/lib/ganglia/rrds` vengono memorizzati i dati relativi ai clusters monitorati ed ai singoli host.

La modalità di memorizzazione consiste in un file system, contenente come unità atomiche i singoli file di formato “rrd” relativi ai singoli grafici memorizzati.

Ma se sulla macchina non è installato anche il RRDtool vero e proprio tali dati non possono essere visualizzati sottoforma di grafici.

7.1 Installazione

Il tool RRD va installato sulla stessa macchina con SO Linux dove è stato installato precedentemente il server gmetad e dove si trova il relativo file system contenente il database.

Il pacchetto `rrdtool-1.0.50-3.fc3.rf.i386.rpm` è scaricabile dal sito

RRDtool al link dei downloads,

<http://oss.oetiker.ch/rrdtool/download.en.html>.

L'installazione del pacchetto rpm nel SO Linux viene effettuata col comando

```
> rpm -Uvh rrdtool-1.0.50-3.fc3.rf.i386.rpm
```

Basta questa sola operazione per visualizzare correttamente i grafici nella web application direttamente dal browser.



8. Gmetric e Gstat

Si tratta di due semplici programmi a linea di comando che vengono forniti automaticamente con gmetad.

Usare gmetric se si vogliono aggiungere nuove metriche e gstat se si vuole visualizzare lo stato del cluster.

8.1 Gmetric

Il **Ganglia Metric Tool (gmetric)** consente di monitorare facilmente ogni arbitraria metrica host che si desidera espandere sul nucleo di metriche che gmond misura per default.

Per avere aiuto sulla sintassi di gmetric basta usare l'opzione a linea di comando "help".

```
> gmetric --help
```

```
gmetric 3.0.0
Purpose:
  The Ganglia Metric Client (gmetric) announces a metric
  on the list of defined send channels defined in a
  configuration file
Usage: gmetric [OPTIONS]...
  -h, --help           Print help and exit
  -V, --version        Print version and exit
  -c, --conf=STRING    The configuration file to use for finding
                        send channels
                        (default='/etc/gmond.conf')
  -n, --name=STRING    Name of the metric
  -v, --value=STRING    Value of the metric
  -t, --type=STRING     Either string | int8 | uint8 | int16 |
                        uint16 | int32 | uint32 | float | double
  -u, --units=STRING   Unit of measure for the value e.g.
                        Kilobytes, Celcius (default='')
  -s, --slope=STRING   Either zero|positive|negative|both
                        (default='both')
  -x, --tmax=INT        The maximum time in seconds between
                        gmetric calls (default='60')
  -d, --dmax=INT        The lifetime in seconds of this metric
                        (default='0')
```

Tabella n.4 - Visualizzazione sintassi di gmetric

Gmetric invia la metrica specificata sulla linea di comando a tutti gli `udp_send_channels` specificati nel file di configurazione `/etc/gmond.conf` per default.

Se si vuole inviare metriche a `udp_send_channels` alternativi, si può specificare un differente file di configurazione come:

```
> gmetric --conf=./custom.conf -n "wow" -v "it
works" -t "string"
```

Tutte le metriche in ganglia hanno un nome, un valore, un tipo ed opzionalmente unità. Se per esempio si vuole misurare la temperatura della CPU (alcuni gmond non lo fanno per default) allora si può inviare questa metrica con name="temperature", value="63", type="int16" e units="Celcius".

Si assume di avere un programma chiamato cputemp il cui outputs è in testo la temperatura della CPU

```
> cputemp
63
```

Si potrebbe facilmente inviare questo dato a tutti i gmonds in ascolto durante l'esecuzione

```
> gmetric --name temperature --value `cputemp` --
type int16 --units Celcius
```

Esaminando il valore di uscita di gmetric per vedere se ha inviato il dato con successo (0) o con errore (-1).

Per esaminare costantemente questa metrica temperatura è necessario aggiungere anche questo comando alla tabella cron.

8.2 Gstat

Il **Ganglia Cluster Status Tool (gstat)** è una utility a linea di comando che consente di ottenere un report sullo stato del tuo cluster.

Per ottenere aiuto con le opzioni a linea di comando, semplicemente si esegue:

```
> gstat --help
```

```
gstat 3.0.0
Purpose:
  The Ganglia Status Client (gstat) connects with a
  Ganglia Monitoring Daemon (gmond) and output a load-balanced
  List of cluster hosts
Usage: gstat [OPTIONS]...
  -h          --help          Print help and exit
  -V          --version       Print version and exit
  -a          --all           List all hosts. Not just hosts
                             running gexec (default=off)
  -d          --dead          Print only the hosts which are
                             dead (default=off)
  -m          --mpifile       Print a load-balanced mpifile
                             (default=off)
  -l          --single_line   Print host and information all
                             on one line (default=off)
  -l          --list          Print ONLY the host list
```

		(default=off)
-iSTRING	--gmond_ip=STRING	Specify the ip address of the gmond to query (default='127.0.0.1')
-pINT	--gmond_port=INT	Specify the gmond port to query (default=8649)

Tabella n.5 - Visualizzazione sintassi di gstat

Nota: gstat senza opzioni mostrerà solo gli host abilitati a gexec. Per vedere tutti gli hosts che sono UP (indipendentemente dai loro stati gexec) si aggiunge il flag **--all**.

```
> gstat --all
```

9. Gexec

GEXEC è un sistema scalabile di esecuzione dei cluster remoti che fornisce velocemente, esecuzioni autenticate RSA di jobs paralleli e distribuiti.

Fornisce spedizioni trasparenti di standard input, standard output, standard di errori e segnali verso e da processi remoti, prevede la propagazione dell'ambiente locale ed è designato per essere robusto e scalabile su sistemi di oltre 1000 nodi.

Internamente, GEXEC funziona costruendo un n-albero di sockets e threads TCP tra demoni gexec e propagando le informazioni di controllo su e giù attraverso l'albero.

Usando il controllo gerarchico, GEXEC distribuisce entrambi il lavoro e l'utilizzo delle risorse associate con il massimo impiego di parallelismo attraverso i nodi multipli, quindi eliminando i problemi associati con i limiti dei singoli nodi (ad esempio limiti sul numero di file descrittori sui nodi front-end).

Una release iniziale del software consiste di un demone, un programma client, e una libreria che fornisce una interfaccia programmatica al sistema GEXEC.

9.1 Authd

Authd è un semplice meccanismo di autenticazione basato su SSL che riesce facilmente ad autenticare gli utenti sui clusters.

Authd è stato scritto da Brent N. Chun ed è parte del package Ganglia. Authd è usato da gexec per lanciare processi sui nodi dei clusters.

Sui sistemi con installati Ganglia/authd/gexec installed, uno può aggiungere ssh modificate ed usare authd per autenticazioni sul cluster prive di password ragionevolmente sicure senza alcun set-up richiesto dall'utente.

10. Studio delle metriche

Per vedere una lista completa delle metriche che un particolare gmond supporta, si esegue il comando:

```
> gmond -m
```

e gmond produrrà tutte le metriche che è capace di collezionare ed inviare.

Questa tabella descrive tutte le metriche che ganglia colleziona e mostra su quali piattaforme le metriche sono supportate. (La seguente tabella è solo parzialmente completa).

Metric Name	Description	Platforms
boottime	System boot timestamp	l, f
bread_sec		
bwrite_sec		
bytes_in	Number of bytes in per second	l, f
bytes_out	Number of bytes out per second	l, f
cpu_idle	Percent of time since boot idle CPU	l
cpu_arm		
cpu_avm		
cpu_idle	Percent CPU idle	l, f
cpu_intr		
cpu_nice	Percent CPU nice	l, f
cpu_num	Number of CPUs	l, f
cpu_rm		
cpu_speed	Speed in MHz of CPU	l, f
cpu_ssys		
cpu_system	Percent CPU system	l, f
cpu_user	Percent CPU user	l, f
cpu_vm		
cpu_wait		
cpu_wio		
disk_free	Total free disk space	l, f
disk_total	Total available disk space	l, f
load_fifteen	Fifteen minute load average	l, f
load_five	Five minute load average	l, f
load_one	One minute load average	l, f
location	GPS coordinates for host	e
lread_sec		
lwrite_sec		
machine_type		
mem_buffers	Amount of buffered memory	l, f
mem_cached	Amount of cached memory	l, f
mem_free	Amount of available memory	l, f
mem_shared	Amount of shared memory	l, f
mem_total	Amount of available memory	l, f

mtu	Network maximum transmission unit	l,f
os_name	Operating system name	l,f
os_release	Operating system release (version)	l,f
part_max_used	Maximum percent used for all partitions	l,f
phread_sec		
phwrite_sec		
pkts_in	Packets in per second	l,f
pkts_out	Packets out per second	l,f
proc_run	Total number of running processes	l,f
proc_total	Total number of processes	l,f
rcache		
swap_free	Amount of available swap memory	l,f
swap_total	Total amount of swap memory	l,f
sys_clock	Current time on host	l,f
wcache		
Platform key:		
l = Linux, f = FreeBSD, a = AIX, c = Cygwin		
m = MacOS, i = IRIX, h = HPUX, t = Tru64		
e = Every Platform		

Tabella n.6 - Metriche di Ganglia

Se si è interessati in quale modo le metriche sono collezionate, si guardi la directory `./src/lib/libmetrics` nella distribuzione dei sorgenti.

C'è un singolo file sorgente nella directory per ciascuna piattaforma supportata.

11. Funzionalità del sistema (focus sul monitoraggio)

In sintesi il sistema MAGO (**M**onitoring **A**ll **G**RID **O**bjects) implementa tre funzionalità principali: la memorizzazione dati di tutti gli oggetti (risorse), il monitoraggio di ogni singolo nodo; l'utilizzo e l'analisi del database siffatto.

Il **monitoraggio** delle risorse della rete di calcolo ENEA-GRID è costituito dalle seguenti funzionalità implementate su tutti i nodi, che ovviamente sono differenziate in base alle specificità di calcolo del nodo stesso che si evince dalle Tabelle Statiche inizializzate all'avvio dell'intero sistema:

- Funzionalità implementate su tutti i nodi da monitorare:
 - Sonda **gmond** del sistema Ganglia in grado di misurare i seguenti parametri:
 - `boottime` timestamp di avvio del sistema
 - `bread_sec` byte(?) letti al secondo
 - `bwrite_sec` byte(?) scritti al secondo
 - `bytes_in` numero bytes in al secondo
 - `bytes_out` numero bytes out al secondo
 - `cpu_idle` % tempo dall'avvio idle CPU
 - `cpu_idle` % di idle della CPU
 - `cpu_intr`
 - `cpu_nice` percentuale di CPU buona
 - `cpu_num` numero di CPU
 - `cpu_sintr`
 - `cpu_speed` velocità in MHz della CPU
 - `cpu_system` % della CPU di sistema
 - `cpu_user` percentuale della CPU utente
 - `cpu_wio` tempo attesa dell'I/O
 - `disk_free` spazio libero totale del disco
 - `disk_total` disco totale
 - `gexec` tool aggiuntivo
 - `heartbeat`
 - `load_fifteen` media load in 15 minuti
 - `load_five` media load in 5 minuti
 - `load_one` media load in 1 minuto
 - `location`
 - `lread_sec` linee(?) lette al secondo
 - `lwrite_sec` linee(?) scritte al secondo
 - `machine_type` tipo di macchina

- mem_arm
- mem_avm
- mem_buffers quantità memoria bufferizzata
- mem_cached quantità di memoria nascosta
- mem_free quantità di memoria libera
- mem_rm
- mem_shared quantità di memoria comune
- mem_total quantità memoria disponibile
- mem_vm
- mtu
- os_name nome del sistema operativo
- os_release release (versione) sist op
- part_max_used massima % utilizzo di tutte le partizioni
- pthead_sec pacchetti(?) letti al secondo
- phwrite_sec pacchetti(?) scritti secondo
- pkts_in pacchetti in al secondo
- pkts_out pacchetti out al secondo
- proc_run num tot processi in esecuzione
- proc_total numero totale dei processi
- rcache cache in lettura
- swap_free q.memoria scambio disponibile
- swap_total q.totale di memoria di scambio
- sys_clock ora di sistema
- wcache cache in scrittura

— Script di test per ciascuna **METRICA** aggiunta. Col termine **metrica** si intende la singola richiesta tra:

- test di ciascuno specifico **software** che restituisce lo **stato** dell'eseguibile sul nodo
- test di ciascuno specifico **software** che restituisce la **versione** installata sul nodo
- test di controllo della corretta esecuzione di un dato **demone** sul nodo, che restituisce uno **stato** (OK, Non OK)
- test di controllo erogazione di un servizio su nodi dedicati all'erogazione di servizi generici verso altri nodi

12. Architettura del sistema

Il sistema si compone di 3 livelli distinti: Server Centrale, Server di Sito e Nodo.

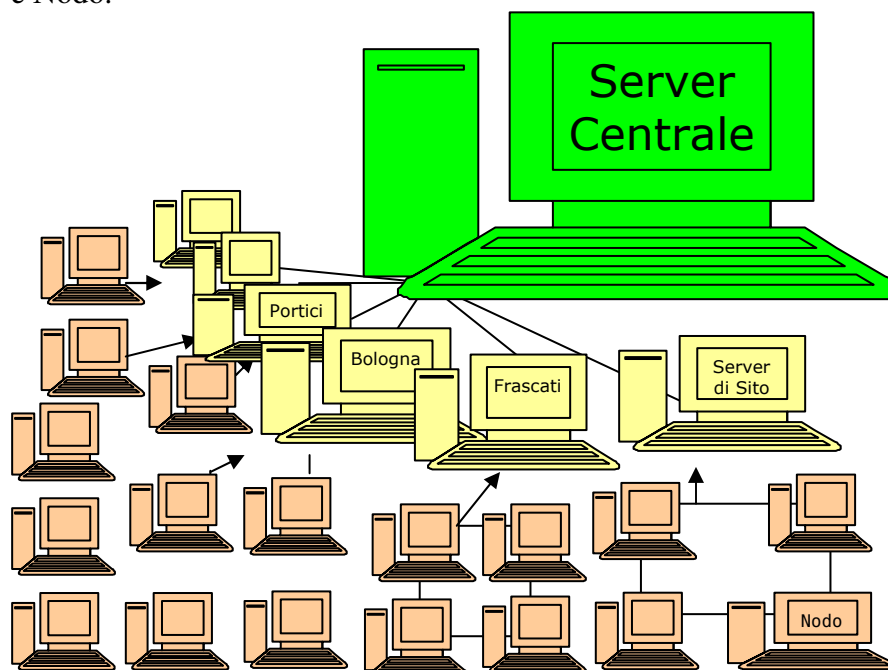


Figura n.2 - I tre livelli dell'Architettura MAGO

12.1 Server Centrale

Il Server centrale (ad es. svncres) ha i seguenti componenti:

1. Server di monitoraggio di Ganglia (gmetad);
2. Database delle Risorse: MySQL;
3. Componente di caricamento del database: **MMasterLoader**;
4. Componente di configurazione iniziale del sistema: **MSetup**;
5. Componente di visualizzazione del monitoraggio: **MMonitor**;
6. Utility per l'analisi del database: **MEnquiry**.

12.2 Server di Sito

Il Server di sito è una macchina per ciascuno dei siti ENEA, dedicata al sistema MAGO.

Ogni Server di sito ha i seguenti componenti:

1. Server di monitoraggio di Ganglia (gmetad), quindi fornisce comunque le informazioni grafiche tipiche di Ganglia sulle metriche

- del sito ed è inoltre client (gmond) perchè è esso stesso un nodo da monitorare;
2. Componente di caricamento del database locale di sito: **MSiteLoader**;
 3. Componente di fornitura dei dati verso il Server centrale: **MSiteProvider**.

12.3 Nodo

Ogni Nodo della rete ENEA-GRID ha i seguenti componenti:

1. Client di monitoraggio (gmond) sempre attivo come servizio ed esso fornisce le informazioni standard di Ganglia.
2. Metriche aggiuntive che vengono raccolte tramite l'esecuzione di script e fanno uso di gmetric per passare l'informazione a gmond. Le metriche aggiuntive vengono eseguite dal crontab di un utente locale dedicato ("mago"). La loro configurazione è definita dalla configurazione dell'ambiente locale MAGO.
3. Ambiente locale MAGO.
4. Script-sonda generalizzato per tutti i nodi **magoupdate.sh**

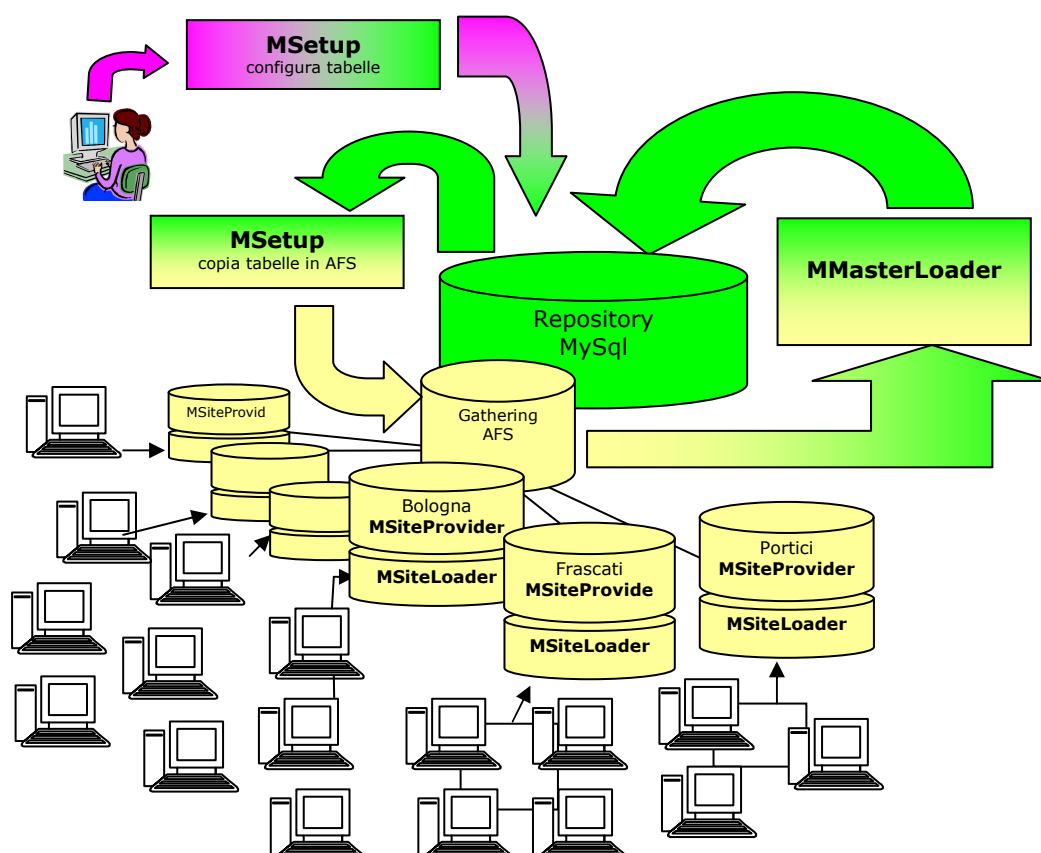


Figura n.3 - Visione generale del sistema

13. Componenti del sistema (focus sul monitoraggio)

In questo paragrafo si descrivono le funzionalità dei componenti che realizzano il sistema di monitoraggio sui singoli nodi e l'aggiornamento negli stessi dei sistemi di misurazione previsti.

13.1 Database delle Risorse

Qui si definiscono i componenti predisposti per la configurazione iniziale del sistema.

13.1.1 *Mago.sql*

Database delle risorse di ENEA-GRID implementato con MySQL, e presente fisicamente sul server centrale dedicato.

Tale database contiene sia le tabelle statiche di configurazione che vengono caricate da MSetup ed utilizzate dagli altri componenti del sistema, sia le tabelle dinamiche contenenti i dati aggiornati dell'intera infrastruttura di monitoraggio.

I dettagli della struttura del DB sono descritti nell'apposito capitolo di questo documento.

13.1.2 *Tabelle statiche in AFS*

Struttura delle tabelle statiche in AFS

- `/afs/enea.it/system/mago/` directory generale in AFS del sistema MAGO
- `.../magoupdate.sh` versione ultima aggiornata dello script generalizzato per tutti i nodi
- `.../hosts/hostname/mago.crontab` tabella dei tempi di esecuzione per ciascuno script di monitoraggio di ciascun nodo
- `.../hosts/hostname/metric.list` lista delle metriche e relativi scripts per ciascun nodo

Si sottolinea che la logica sulla quale si basa tale scelta è tale da rendere la configurazione dell'intero sistema quanto più centralizzata possibile, senza dover fare eccessivi interventi a livello di singolo nodo.

13.2 Strumenti di monitoraggio e aggiornamento

Qui si determinano i componenti che devono essere predisposti per l'esecuzione del monitoraggio e l'aggiornamento dei sistemi di misura.

13.2.1 *Ambiente locale MAGO.*

L'ambiente locale su ciascun nodo della rete ENEA-GRID dedicato al sistema MAGO è costituito da:

- `/usr/local/mago/magoupdate.sh` script generalizzato per tutti i nodi
- `/usr/local/mago/mago.crontab` tabella dei tempi per l'esecuzione automatica degli script
- `/usr/local/mago/metric/metric.list` elenchi delle metriche e relativi script
- `/usr/local/mago/metric/scripts/` tutti gli scripts di metrica

13.2.2 *magoupdate.sh*

Questo componente di ciascun nodo esplica le seguenti funzionalità:

- Verifica l'accessibilità ad AFS
- Se AFS non è accessibile termina (o provvede a ricevere le informazioni da canali alternativi), altrimenti:
- Copia da AFS la versione aggiornata di `magoupdate.sh`
- Copia da AFS le versioni specifiche per il nodo di:
 - `mago.crontab`
 - `metric.list`
 - gli script definiti da `metric.list` (che è un elenco del tipo
chiave=id_script, valore=path completo in AFS, cioè
`/afs/enea.it/system/mago/scripts/type`
`[=software,daemon,service]` /group
`[=AFS,matlab,gcc,...]` /nome_script_metrica
questo nome è un link alla versione corrente dello script)
- Aggiorna il crontab locale dell'utente "mago"

Come conseguenza di tali funzioni si ottengono i seguenti risultati:

- Esecuzione dei rispettivi script secondo la tempistica stabilita da crontab
- Accodamento del risultato ottenuto da ciascuno script a gmond mediante gmetric.
- Invio automatico, grazie all'esecuzione di gmond, del file XML di Ganglia sulla porta 8649 (default) contenente anche i valori delle metriche aggiunte ed eseguite

13.2.3 *Scripts di misura delle metriche aggiuntive*

Ciascuno di questi “n” componenti di ciascun nodo esplica la funzionalità di “misurare” lo specifico valore per cui è stato costruito.

La condizione generale che tali scripts devono rispettare è essere rigorosamente gmetric-compliant, cioè restituire un UNICO VALORE, che poi attraverso gmentric possa essere accodato alla sonda gmond di Ganglia.

14. Conclusioni finali

Queste attività prevedono le fasi di studio, installazione e configurazione dei rispettivi demoni sui server e su tutti gli host della rete ENEA-Grid.

In particolare:

- verificare il funzionamento multiplatforma
- test della piramide MMasterLoader, SiteLoader/ServiceProvider, nodi
- funzionamento di gmond su tutte le piattaforme: AIX, Irix, MacOSX, Linux i386, IA64, x86_64.

Un'altra attività consiste nella procedura di installazione di mago sul nodo:

- Installare ganglia-gmond
- Configurare /etc/gmond.conf eventualmente utilizzando le informazioni della tabella **Host**
- Definire l'ambiente per mago: la user dedicata

15. Eventuali sviluppo futuri

Se applicabile al caso in esame.

16. Riferimenti Bibliografici

[1] NA

17. Definizioni

CRESCO - Centro computazionale di RicErca sui Sistemi Complessi

MAGO - Monitoring All GRID Objects

Appendice.A *Prima Appendice*

Se applicabile al caso in esame.