Monitoring e accounting code LSF

Andrea Simonetto
andrea.simonetto@cnaf.infn.it

21-Jun-2013

1 Panoramica

Monitoring e accounting, pur presentando alcune similitudini superficiali, sono task molto diversi. Il monitoraggio tiene traccia dei job presenti nel batch system, ed è ottenuto tramite una serie di istantanee prese ad intervalli di pochi minuti, che permettono di conoscere lo stato del batch system momento per momento. Per contro l'accounting effettua una analisi su base giornaliera, per conoscere a posteriori le quantità di risorse consumate dai job.

La differenza principale può essere evidenziata con un esempio: in una giornata tipo l'intera farm lavora a pieno regime e ogni job è occupato per tutto il tempo nell'elaborazione dei dati. Il monitoraggio mostrerà un'elevata quantità di calcolo effettuato. Supponendo che tutti i job restino in esecuzione superando la mezzanotte (momento in cui viene calcolato l'accounting), non risulteranno job terminati agli occhi dell'accounting, che quindi registrerà zero ore di calcolo effettuate. Quando i job termineranno, l'accounting registrerà le ore complessive di calcolo di ogni job. Nella pratica i job entrano ed escono dall'esecuzione a tempi che possono essere considerati casuali, e pertanto la probabilità che nessun job termini entro le 24 ore è trascurabile. Tuttavia questo esempio limite mostra come monitoraggio e di accounting forniscano informazioni diverse sul lavoro della farm e del batch system.

Monitoraggio e accounting sono accomunati dal tipo dei dati che raccolgono: serie temporali. Pertanto possono essere collocati sotto un unico framework per la raccolta, l'elaborazione e la visualizzazione di serie temporali. È stato scelto Graphite, un progetto Open Source sotto licenza Apache 2.0. I dati, sia come grafici, sia in vari formati testuali, sono esportati grazie al server web httpd di Apache.

2 API

Le serie temporali raccolte da monitoring e accounting sono accessibili dal web server `http://farm.cr.cnaf.infn.it`, richiedendo una risorsa tra quelle generate dalla seguente grammatica:

Resource	::=	/Type/Submit/Queue/Period.Unit[-*W*x*H*].Format
Type	::=	`monitoring` | `accounting`
Submit	::=	`grid` | `local` | `all`
Queue	::=	*queue*[*] | `each` | `all`
Period	::=	*date*[-*date*] | *n*`h` | *n*`d` | *n*`w` | `day` | `week` | `month` | `year`
Unit	::=	`jobs` | `efficiency` | `sec` | `hs06` | `pledge`
Format	::=	`png` | `json` | `csv` | `raw`

Zucchero sintattico:

- /*Type*/*Queue*/... → /*Type*/all/*Queue*/...
- /*Type*/*Period*... → /*Type*/all/all/*Period*...

Ad ogni coda è associata una serie temporale. L'intervallo di tempo è espresso dalla categoria *Period*: i formati day, week, month e year corrispondono ai dati della giornata corrente a partire dalla mezzanotte, della settima a partire da lunedì, del mese a partire dal primo, dell'anno a partire dal primo Gennaio; i formati *n*`h`, *n*`d`, *n*`w` permettono di selezionare delle finestre temporali di dimensione costante, nella fattispecie, le ultime n ore, giorni e settimane rispettivamente. Infine il formato generico date permette di scegliere un momento preciso con granularità di 1 minuto; se il secondo date è omesso, s'intende il periodo tra la data scelta ed adesso.

Le grandezze selezionabili per ogni serie (categoria *Unit*) sono:

- `jobs`: per il monitoraggio, il numero di job sottomessi e il loro stato all'interno del batch system (running, pending, suspended, unknown), per l'accounting, il numero di job completati e il loro stato di uscita (normale o killato dal batch system);
- `efficiency`: efficienza delle code calcolata come "CPU Time / Wall Clock Time" (aka CPT/WCT);
- `sec`: secondi di calcolo (WCT e CPT);
- `hs06`: HepSpec06 utilizzati (WCT e CPT);
- `pledge`: quantità di HepSpec06 (WCT) correntemente assegnati ad ogni coda.

Per il monitoraggio, non sono disponibili tutte le grandezze: solo jobs e efficiency vengono raccolte.

I tipi di sottomissione sono grid e local, a seconda che i job provengano da un CE o da una UI. Sia per le code che per i tipi di sottomissione, è presente una keyword all che permette di ricevere la somma di tutte le serie corrispondenti a una grandezza.

Le code possono inoltre essere selezionate giustapponendo al loro nome un asterisco, che significa "somma di tutte le serie aventi questo nome come prefisso". Infine la keyword each permette di ricevere i dati di ogni singola coda. Quest'ultima possibilità è presente solo per le grandezze `sec`, `hs06` e `pledge`.

Ogni risorsa è disponibile in diversi formati:

- `png`: grafico dei dati come immagine PNG di dimensione WxH (800x600 di default);
- `json`: serie di dati come oggetto json;
- `csv`: formato CSV, utilizzabile, ad esempio, per l'importazione in un foglio di calcolo;
- `raw`: formato testuale personalizzato. Le serie sono ritornate per riga, ognuna avente il seguente formato:

  `target_name,start_timestamp,end_timestamp,series_step|[data]*`

Alcuni esempi:

- `http://farm.cr.cnaf.infn.it/monitoring/week.jobs.png`
- `http://farm.cr.cnaf.infn.it/monitoring/grid/all/week.efficiency.png`
- `http://farm.cr.cnaf.infn.it/monitoring/ams*/week.efficiency.png`
- `http://farm.cr.cnaf.infn.it/accounting/year.hs06.png`
- `http://farm.cr.cnaf.infn.it/accounting/20130501-20130531.efficiency.png`

Sono infine presenti alcuni report "preconfezionati" alle pagine:

- /*Type*/*Queue*/*Period*.html
- /*Type*/*Period*.html

ad esempio:

- `http://farm.cr.cnaf.infn.it/monitoring/day.html`
- `http://farm.cr.cnaf.infn.it/monitoring/week.html`
- `http://farm.cr.cnaf.infn.it/monitoring/ams*/week.html`
- `http://farm.cr.cnaf.infn.it/accounting/year.html`

3 Architettura

Tutto il codice è mantenuto nel repository Subversion:

`https://svn.forge.cnaf.infn.it/svn/farming/simtools/monacct`

ed è installato in `/opt/farm` su `farm.cr.cnaf.infn.it`. Gli script Python `update_monitoring` e `update_accounting` raccolgono i dati dal batch system e li inviano a Carbon, il demone del progetto Graphite responsabile dell'immagazzinamento delle serie temporali. Carbon e Graphite hanno le rispettive configurazioni nelle cartelle `/etc/carbon/` e `/etc/graphite-web/`. In particolare Carbon salva i dati delle serie temporali in `/var/lib/carbon/whisper/`.

La schedulazione dei questi task è controllata da `crond`, tramite il file `conf/farm.cron`:

```
LSF_SERVERDIR=/usr/share/lsf/7.0/linux2.6-glibc2.3-x86_64/etc
LSF_LOGDIR=/usr/share/lsf/work/tier1-lsf/logdir
LD_LIBRARY_PATH=/usr/share/lsf/7.0/linux2.6-glibc2.3-x86_64/lib

# Update monitoring stat every 3 minutes
*/3 * * * * root /opt/farm/monitoring_update 2>/dev/null

# Update accounting stat once a day
10 02 * * * lsfadmin /opt/farm/accounting_update $LSF_LOGDIR
```

Nella sottocartella `web/` si trova il codice PHP che si occupa di esporre le API attraverso il web server Apache (nella stessa cartella è presente il `.htaccess` che filtra e invia le richieste allo script principale, `api.php`. Gli altri script PHP `{accounting,monitoring}_{queue,all}.php` gestiscono la visualizzazione delle pagine "preconfezionate", assieme allo script `js/customgraph.js`.

Il virtual host `http://farm.cr.cnaf.infn.it/` è configurato in `/etc/httpd/conf.d/farm.conf` in questo modo:

```
<VirtualHost *:80>
   ServerName farm.cr.cnaf.infn.it
   DocumentRoot "/opt/farm/web"
   <Directory "/opt/farm/web">
      Options Indexes FollowSymLinks
      AllowOverride All
      Order allow,deny
      Allow from all
   </Directory>
</VirtualHost>
```

Graphite dispone di una propria interfaccia web, accessibile tramite il *CNAME* `http://graphite-farm.cr.cnaf.infn.it/`, e immagazzina i dati degli utenti, dei grafici salvati e delle dashboard in un database SQLite (in `/var/lib/graphite-web/graphite.db`).

Tutti i grafici prodotti (anche quelli esportati in formato testuale) vengono salvati in cache per 3 ore, grazie al demone memcached. Le cache sono indicizzate per URL, quindi per avere **dati freschi** bisogna specificare il periodo con un formato tra `day`, `week`, *`date`* o *`date-date`* esplicitando **ore e minuti** oppure uno tra *n*`h`, *n*`d`, *n*`w`.