Normalizzazione puntuale dei dati di accounting

Proposta per uno strumento automatizzato

Peter Solagna, INFN Padova
Felice Rosso, CNAF
Guido Guizzunti, CNAF
Andrea Guarise, INFN Torino

Sommario

- Sistema corrente
  - Limiti, problemi
- La nostra proposta
  - Struttura database
  - Client/Server
- Commenti/Critiche
  - FONDAMENTALI

Normalizzazione, sistema corrente

- Site Manager deve:
  - Calcolare il valore di benchmark medio/core
    - HEP_SPEC06, SI2k
  - Pubblicare il dato sul sistema informativo del sito, o nella configurazione del sistema di accounting.
  - Ogni volta che ci sono delle variazioni nell'hardware del cluster.
- Sistema di accounting deve:
  - Recuperare il valore dal sistema informativo o dalla configurazione
  - Utilizzare il valore medio per normalizzare tutti i job (WCT*Bench)

Limiti del sistema attuale

- Normalizzazione utilizzando valori medi di benchmark
  - Cluster eterogeneo, valori non reali
- Inserimento manuale dei valori da parte dei site managers
  - Errore umano nell'inserimento del valore vero e proprio, o nel calcolare il valore medio
  - Mancato aggiornamento del valore dopo un cambiamento nel cluster

Obiettivi del progetto

- Normalizzazione puntuale
  - Normalizzare ogni singolo job con il valore di benchmark effettivo della macchina fisica su cui e` girato
    - Database contenente una mappatura hostname -> benchmark
  - Macchine virtuali
    - Lo stesso host-name (virtuale) puo` essere istanziato su differente hardware
- Database *dinamico*
  - Aggiornato automaticamente, in modo trasparente.
  - Semplice da gestire ed installare
    - Sensore(client) installato sui worker nodes (fisici o virtuali)

Database dinamico

- Associa ad un host name, ed un periodo temporale, una configurazione hardware
  - Ogni entry nel database avra` una data di inizio e di fine
- Tabella di match make
  - Associa una configurazione hardware/software un benchmark
  - Possibile inserire piu` benchmark diversi per ogni configurazione hardware

Tabella workernode

- HostName
  - Completo di dominio
- date-begin, date-end
  - identificano la finestra temporale in cui sono validi i dati dell'entry
- CPUmodel
  - Stringa univoca di identificazione del processore
- TotalCores
  - Numero di cores visti dal kernel (HT on/off)
- KernelVersion, OSversion, TotalRAM
- VM
  - Campo che distingue wn reali da quelli virtuali. Tenere conto di eventuale overhead della virtualizzazione nell'accounting.
- Popolata automaticamente

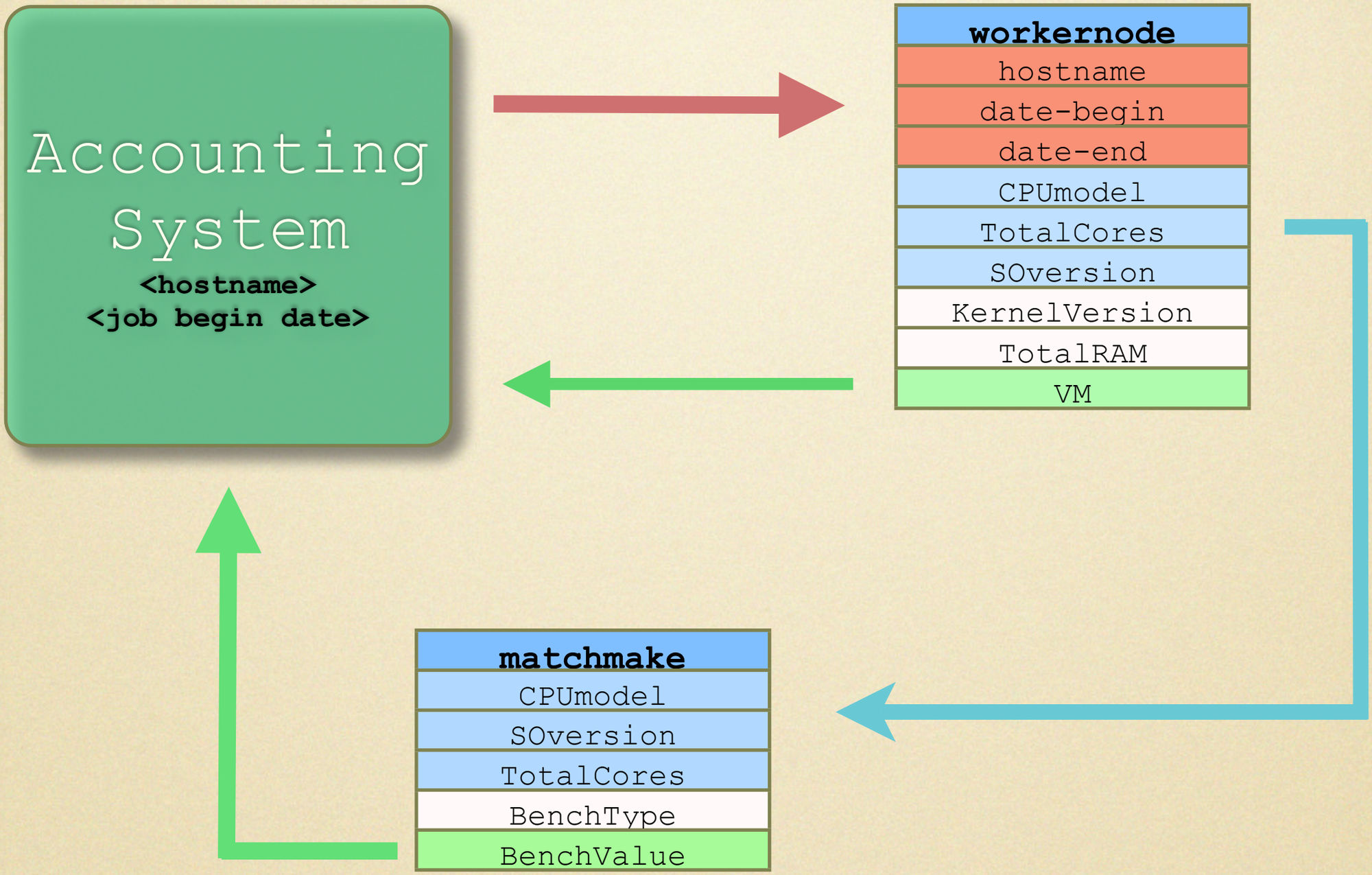
Tabella matchmake

- CPUmodel
- TotalCores
- SOversion
- BenchmarkType
  - Il tipo di benchmark contenuto nella riga della tabella (SI2k, HEP_SPEC..)
- BenchmarkValue
  - Il valore da utilizzare per la normalizzazione

Tabella matchmake 2

- Non compilata direttamente dai site managers
- Resa disponibile aggiornata su un server http al CNAF
  - Puo` essere recuperata con un *wget* dalle installazioni locali
  - Prevista fin dall'inizio la possibilita` di inserire da un form web configurazioni hardware da aggiungere alla tabella
- Prevista nel futuro la possibilita` di un'installazione completamente indipendente dal CNAF, in ogni caso personalizzata

Ottenere il benchmark




```
SELECT CPUmodel,TotalCores,SOversion, VM FROM workernode WHERE <hostname> = hostname AND date-
                begin <= <job begin date> AND  date-end >= <job begin date>;
```

Implementazione: client

- Script per la raccolta delle specifiche della macchina
- Invio dati su protocollo HTTP, utilizzando *curl*
  - Tool comunemente disponibile sulle installazioni standard
  - Invio di un messaggio BEGIN allo startup della macchina
  - Invio di un messaggio END allo shutdown della macchina
- Si preoccupa di verificare la ricezione dei dati da parte del server
  - Nel caso il server non risponda, il client ritentera` di inviare il messaggio fino a quando possibile
    - Macchine virtuali non permettono l'uso di file temporanei locali

Implementazione: client 2

- Macchine virtuali: CPU type
  - Utilizzare *libguestfs* per scrivere un file direttamente nel disco della macchina virtuale
    - Installazione sul dom0 di una serie di rpm
  - Montare il file system della macchina virtuale e scrivere un file

Implementazione: server

- Operazioni lato server:
  - Salvataggio messaggi ricevuti su storage locale
    - Invio di risposta OK al client
  - Verifica sintattica del messaggio ricevuto
  - Coerenza informazioni ricevute (data, stringa gia` ricevuta..)
  - Immissione informazioni nel database
  - Aggiornamento dei log files
- Server HTTP
  - Implementazione in python
  - Gestione di eventuali messaggi persi (correggendo il database)
  - Gestione della sicurezza: solo filtraggio degli IP

Implementazione: server 2

- Scalabilita`, affidabilita`: utilizzare un altro sistema di trasporto?
- Dimensioni delle tabelle:
  - Tabella workernode: 600MByte/mese per un sito delle dimensioni del Tier-1 totalmente virtualizzato
  - Tabella matchmake: pochi KByte
- Prevedere fino dall'inizio la possibilita` di ridondare i server
  - Possibilita` di un database multisito

Installazione del tool

- RPM del server
  - Script per la generazione dell'RPM del client da installare sui WN
    - Configurazione del server comprende la generazione dell'RPM del client
    - RPM del client generato gia` contenente il file di configurazione completo, con indicazione del/dei server dove inviare i messaggi
      - Utilizzandolo su macchine virtuali potrebbe non essere possibile installare direttamente un RPM, ma sarebbe piu` comodo ottenere direttamente i files
- Installazione RPM+YAIM
  - Normale workflow per l'installazione software grid
- Possibilmente mantenere entrambi i metodi

Integrazione con DGAS

- Il database HLR contiene gia` le informazioni sull'hostname e sulla data/ora di inizio e fine del job
  - I due tool (database dinamico/DGAS) dovranno collaborare
  - Come e quando utilizzare i dati sulla normalizzazione?
- Tre possibilita`
  - Mantenere i dati separati e farne la join solo al momento dell'effettivo utilizzo dei valori normalizzati
  - Interrogare il database dinamico al momento dell'inserimento dello usage record nell'HLR
    - *Livello HLR*
  - Eseguire la query al momento della creazione dell'usage record, prima dell'inserimento nel HLR
    - *Livello sensore dgas*

Conclusioni

- Tool ancora in fase di progettazione
  - Fondamentale il contributo dei potenziali utilizzatori
  - Potra` migliorare la precisione e l'utilizzabilita` dei dati di accounting, ma non solo...
  - Un passo (obbligato?) verso il Cloud Computing
- **Stakeholders**:
  - Site managers
  - Team accounting
  - Referee
  - Vo Managers
  - Utenti grid

Fine

- Commenti/Critiche/Suggerimenti.
  - peter.solagna(at)pd.infn.it

Grazie per l'attenzione.