GUIDA ALL'UTILIZZO DEL MILANO DT SYSTEM

Il Milano DT gestisce il trasferimento tra due siti che possiamo identificare con due macchine (computer) remoti:

- Macchine

SENDER: macchina che spedisce i file, si configurano essenzialmente due situazioni:

1. tipicamente una macchina CERN (pcamsd0 o il "sostituto" pcamsd2
2. macchine di siti MC remoti (RMC) o di siti per la ricostruzione remota dei dati (RR) in funzione ad esempio del trasferimenti di dati MC o RECO dai vari siti di produzione al CERN

RECEIVER: parallelamente avrò due configurazioni:

1. CNAF se si tratta ad esempio della Master Copy di tutti i dati di AMS-02, ovvero una repository di back-up, o ASDC
2. CERN se si tratta di produzione locale di dati MC o RECO che dovranno essere inviati alla repository principale per essere valicati dal system manager (Vitaly Choutko)

- Directory

Per semplicità nei due computer in oggetto conserverò i nomi, cioè ad esempio avrò delle dir di questo tipo

a. src/SENDER/Xsrc (dove Xsrc varia in funzione del tipo di dato da trasferire)
b. src/RECEIVER/servers

LATO SENDER

Trasferimento root file e journal

File principale lancia.xml (o lancia_con_anche_local.xml) gestisce i vari perl (Finite States Automa FSA)

In ordine nel xml avrò i seguenti programmi perl

1. cerca_nuovi.pl:

   aggiorna il nostro DB (tabella ams_X in base al tipo di file) prendendo le informazioni da quello di Vitali (VCDB); trovato un file nuovo (del tipo X) lo mette in stato TO_BE_TRANSFERED;

2. trasferisci.pl (argomenti dir name):

   prende il path originale (la directory in cui fisicamente si trova il file) da VCDB; e trasferisce con globus_url_copy (o bbftp); si può usare: perl trasferisci.pl dir name_finto per tentare di mettere il file in stato TRANSFERED e size, senza in realtà trasferire veramente, se e' possibile che i file ci siano gia'.

3. controlla_size.pl (argomenti dir name size)

   controlla la dimensione (size) dei file utilizzando la connessione ssl, ritorna 0 se corretto (lo stato del file viene modificato in SIZE_OK) , se la size non è corretta ritorna 1 e lo stato del file viene aggiornato in TO_BE_TRANSFERED (in caso di errore ritorna 2 ma non ritrasferisce il file)

4. controlla_crc.pl (argomenti)

   prende adler32 (ovveroi checksum del file) dal valore riportato nel CASTOR SENDER (nel caso ad esempio del CERN); controlla adler32 (lo esegue nel RECEIVER o controlla il valore di CASTOR ad esempio al CNAF) utilizzando la connessione ssl, ritorna 0 se corretto, modificando lo stato del file in VALIDATED, oppure ritorna 1 e e lo stato del file viene aggiornato in TO_BE_TRANSFERED (in caso di errore ritorna 2 ma non ritrasferisce il file)

N.B.

- ✓ nel caso di lancia_con_anche_local.xml avremo:
  - o oltre al controlla_crc.pl viene fatto girare anche controlla_crc_local.pl per evitare lunghe latenze che a volte si verificano prima che i file siano messi su castor e quindi sia disponibile il loro checksum (si introduce quindi un nuovo stato, LOCAL_CRC, intermedio, nel caso controlla_crc.pl dia errore e ritorni 1)

- ✓ tutti i perl hanno uno sleep, ovvero un ritardo per evitare che due processi sullo stesso file si sovrappongano, creando quindi errori ad esempio quando il DB viene aggiornato.

- ✓ uso: python ../finitestates/finitestates.py lancia.xml

- ✓ iniziale_lancia.xml fa un trasferimento "finto", e serve all'inizio, quando alcuni file potrebbero essere gia' stati trasferiti, ma non vogliamo ritrasferirli.

- ✓ I file FSA (finitestates_Z.py) si trovano di solito nella directory src/SENDER/finitestates/ e vengono duplicati per ogni tipo di file (MC, RECO o RAWDATA ad esempio)

- ✓ I certificati per la connessione ssl si trovano nella dir srcX/certificati, ovvero ogni sottodir per il trasferimento di un certo tipo di file ha una copia dei certificati per la connessione

IMPOSTAZIONI

Nel file lib/Mc_Mover.pm ci sono tutti i parametri, molti da impostare, in funzione del tipo di file. I parametri per la connessione ssl sono anche nei file controlla_*.pl, sul lato SENDER (mentre in Nuovo_Protocollo.pl lato RECEIVER).

COME INIZIARE

lanciare:
perl cerca_nuovi.pl in automatico aggiorna il DB con i file da trasferire (ma non trasferisce)

Se alcuni file (o anche tutti) sono gia' stati trasferiti, per evitare di ritrasferirli, lanciare prima iniziale_lancia.xml

Poi far partire tutto:

python ../finitestates/finitestates_X.py lancia_dav.xml >/dev/null 2>/dev/null </dev/null &

COME CONTROLLARE LA CONSISTENZA

solo se c'e' il dubbio di aver perso qualche cosa! Con mysql, mettere status=TRANSFERED a tutti i file. Finitestates parte subito a fare il controllo di consistenza.

Info Varie

- Le informazioni importanti vengono salvate in

log/transfer_X.log (viene aperto da lancia.xml)
Nel log. Cercare ERROR e WARNING (da shell: more log/transfer.log | grep WARNING)

- resoconto.sh

scrive un po' di informazioni sugli ultimi file e su eventuali problemi sul DB
(da shell: sh resoconto.sh)

- Creare la tabella per il DB:

utilizzare il file crea_tabella.sql in questo modo:
mysql -u amsdes -p <crea_tabella.sql
oppure cancellare il contenuto se la tabella c'e' gia'

- Mantenimento del DB e ridondanza

Il file copia_DB_X_a_Milano.pl fa una copia della tabella ams_X a Milano, su amsmc10.mib.infn.it , la copia viene fatta una volta sola e successivamente il programma aggiorna_DB_da_Milano.pl aggiorna il DB con gli ultimi file trasferiti. In crontab vengono automaticamente eseguiti gli aggiornamenti. Esistono quindi almeno due copie dello stesso DB (ad esempio al CERN e a MILANO)

- CONTROLLARE DB

mysql -p
use mcmover;
select * from ams_X;

RECEIVER

File in
src/RECEIVER/servers/

File principale

Nuovo_Protocollo_Server_DT_bbftp.pl

Per l'utilizzo di bbftp deve essere running (/etc/rc.local o init.d)

si lancia così: bbftp –b .m 10

(in questo caso uso 10 stream parallele per trasferire)