Stato del Progetto AMS

Matteo Boschini (*), Andrea Favalli (**)

(*) *CILEA, Segrate*

(**) *Politecnico di Milano*

Abstract

Viene presentato lo stato del progetto CILEA-AMS, con particolare attenzione alle tecnologie di trasferimento dati utilizzate nella realizzazione di un *mock-up* dell'AMS-02 *Data Transfer Facility*.

Keywords

Scienze, Fisica, Networking, Trasferimento dati.

Introduzione

Il progetto AMS, cui il CILEA partecipa in maniera attiva dal 1997 in collaborazione con la sezione di Milano dell'I.N.F.N., è descritto in dettaglio altrove [1],[5].

Si intende qui dare alcuni aggiornamenti rispetto all'implementazione di un *mock-up* del sistema di trasferimento. Tale *mock-up* ha l'obiettivo di simulare, utilizzando i dati di AMS-01, il trasferimento e la gestione dei dati di AMS-02, che inizierà la presa dati a bordo della *International Space Station Alpha* nei primi mesi del 2005.

Il flusso dei dati

Il campione di dati atteso per AMS-02 ammonta a 200 Tbyte e sarà raccolto presso il CERN di Ginevra. Per ovvie ragioni, sarebbe impensabile che tutti i collaboratori accedano a tale set di dati; questo rende necessaria una copia di ridondanza del campione presso un'altra sede.

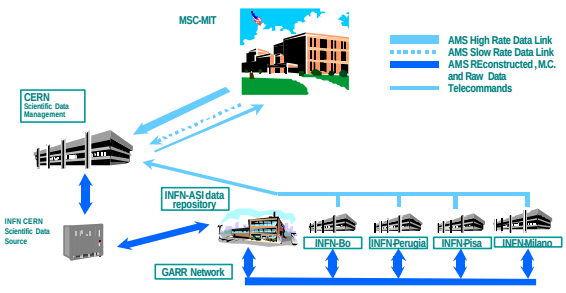

Fig. 1 - AMS Data/Commands Management

Questa master copy assolve inoltre il compito di backup dell'intero data-set. La master copy sarà implementata in Italia *all'AMS Italian Ground Segment Data Storage Facility (IGSDSF)* (ragionevolmente dislocata presso un centro di calcolo INFN).

Una problematica fondamentale, data l'entità del campione, e' legata al trasferimento dati: come già descritto altrove [5], i dati devono venire trasferiti appena disponibili, dal CERN al *IGSDSF*, in modo da permettere ai collaboratori un accesso quasi in tempo reale. Il sistema inoltre deve essere in grado di sostenere un rate medio di 16 Mbit/s e uno di picco di 32 Mbit/s per un periodo di 3 anni, minimizzando quanto possibile la necessità di interventi umani o di presidio. A tal fine si deve sviluppare un sistema di trasferimento semplice, affidabile, in grado di assicurare un alto livello di consistenza tra il campione dati del CERN e quello della master copy. L'architettura generale di tale sistema, si basa su architettura client/server integrata con un sistema di Data Base relazionale. Deve inoltre essere dotato di strumenti di monitoraggio o gestione.

Il sistema di trasferimento e book-keeping

Il primo prototipo di tale sistema [5], sviluppato in Perl5 [6], si basava su client/server TCP/IP e un RDBMS MySql [7] 3.23. Il trasferimento dati era affidato al protocollo OpenSsh/Sscp [8]. Sebbene questo primo prototipo sia stato in grado di sostenere un tra-

sferimento dati di 1.2 Tbyte, ha evidenziato alcuni problemi relativi al carico indotto dal protocollo *scp* sulle macchine coinvolte nel trasferimento [9][1] e una inconsistenza tra i due data set dell'ordine dell'1%.

Siamo quindi passati a valutare le caratteristiche di un altro protocollo di trasferimento che soddisfacesse i requisiti di sicurezza ma non caricasse inutilmente le CPU. Un simile protocollo, ampiamente usato nell'ambito della Fisica delle Alte Energie, è *bbftp* [10]. Si tratta essenzialmente di un ftp che cripta l'hand-shake e lo scambio *uid/passwd* ma non i dati; inoltre permette di cambiare a linea di comando sia la *TCP window size*[2] che il *socket_SND/RCV buffer size*, oltre al numero di stream paralleli da istanziare.

Abbiamo quindi anzitutto provato il sistema di trasferimento usando bbftp invece di scp; appurata la perfetta implementabilita' di bbftp[3]. Successivamente abbiamo modificato il codice sorgente di bbftp in modo da aumentare il livello di jogging e poter implementare un sistema di autenticazione client/server privato. A quel punto il sistema e' stato sottoposto ad un test di 20 settimane di trasferimento dati. In totale sono stati trasferiti quasi 3 Tbyte di dati.

Grazie al sistema di logging più dettagliato e ad una prima bozza di interfaccia Web per il monitoraggio, abbiamo anche trovato la causa dell'inconsistenza tra i DB: essa era legata ad un non corretto funzionamento del client che, in presenza di un alto numero di file da trasferire, aggiornava il DB in maniera errata. Abbiamo quindi modificato il client in modo che esegua un fork di sé stesso in N processi paralleli, dove N e' funzione dei file da trasferire. Grazie a questa soluzione i DB sono adesso perfettamente consistenti, e la percentuale di dati persi che hanno richiesto una ritrasmissione e' dello 0.3 %.

Un aspetto rilevante rivestono invece i parametri TCP ottimali per ottenere il massimo del throughput minimizzando il carico indotto sulle macchine. Ad un primo esame può sembrare infatti che aumentando il numero di stream TCP si riesca ad aumentare il throughput; ciò e' solo in parte vero (anzi può aumentare la congestione della rete e peggiorare in caso di un alto rate di collisioni, v. [11],[12]) e comunque l'inizializzazione degli stream paralleli induce un carico non trascurabile e rallenta il processo di trasferimento, specie se la quanitità di dati da spedire è modesta ma la spedizione va ripetuta molte volte.

Al fine di meglio comprendere questi comportamenti e trovare una configurazione ottimale, abbiamo studiato l'andamento del throughput in funzione della TCP window size e del numero di stream.

Come generalmente consigliato in letteratura [13], abbiamo anzitutto studiato le caratteristiche della rete in funzione dei parametri in esame. Solo successivamente abbiamo valutato le performance di bbftp: questo per disaccoppiare le caratteristiche di TCP dalla loro implementazione in bbftp.

Riportiamo qui sotto (vedi fig. 2) le misure del throughput ottenute con iperf [14] per la connessione CERN-INFN/MI (il cui throughput teorico massimo e' di 10 Mb/s[4]):

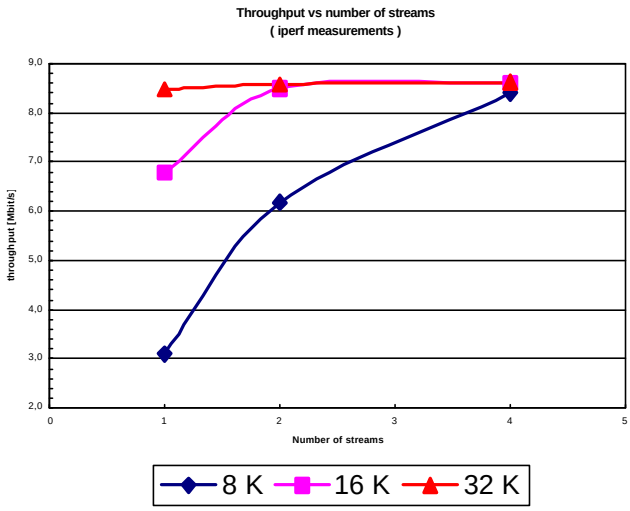

Fig. 2

Come si può vedere, con una TCP window size di 32 Kbyte, si ha già un throughput massimo con un solo stream; un simile comportamento

[1] Essenzialmente dovuto al fatto che *OpenSsh/scp* cripta non solo l'*hand-shake* inziale della connessione ma tutti i dati che durante connessione vengono scambiati. Questo e' piu' di quanto necessario al progetto, per cui e' sufficiente la criptazione *dell'hand-shake* e della coppia *uid/passwd*.

[2] Quantita' di dati che il cliente puo' spedire al server prima che quest'ultimo gli invii un ACK.

[3] Ricordiamo che l'intero sistema e' trasparente per quanto riguarda il protocollo di trasferimento che e' un semplice parametro di una funzione privata.

[4] La LAN del CERN su cui si trova l'host e' a 10 Mb/s.

si ha per diverse dimensioni del file trasferito. Questo risultato non deve stupire, essendo la connessione CERN-MI ad alta congestione pur essendo il tratto a più bassa banda passante essenzialmente senza traffico.
Riportiamo in fig. 3 i risultati ottenuti per bbftp:

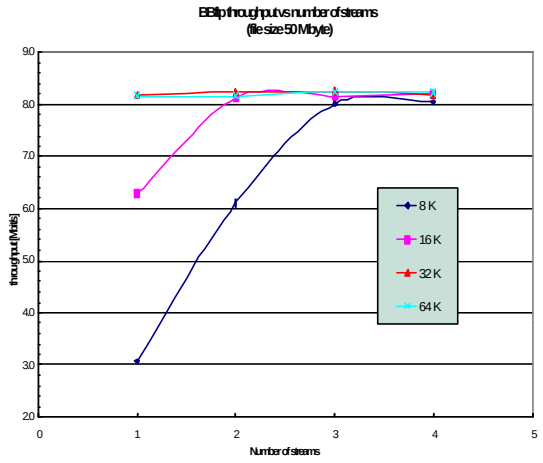

Fig. 3

Si può anzitutto osservare un ottimo accordo tra il comportamento di bbftp e quello di iperf, ad indicare una implementazione ottimale di TCP in bbftp.
Inoltre anche nel caso di bbftp, il throughput massimo si ha già con un solo stream, purché la TCP window size sia di 32 Kbyte. Questo risultato è particolarmente buono, essendo 32 Kbyte la dimensione di default per kernel 2.x GNU/Linux e reti da 10 Mbit/s (ovvero, le nostre condizioni).
Ricordiamo che la TCP window size ottimale è data da:

$$TWS\_opt = T\_BW * RTT,$$

dove T_BW è la banda passante nominale e RTT il Round Trip Time. Ovviamente esiste uno stretto legame tra RTT, T_BW e lo stato di congestione di una rete: non è detto quindi che una rete con T_BW molto alto permetta un buon throughput ad una fissata TCP window size. Tali parametri quindi varieranno nel tempo in funzione del carico della rete, e poiché il trasferimento dati non avverrà su rete dedicata ma sulla rete Geant/GARR-G, dobbiamo prevedere di dover variare la TCP window size ottimale in modo "adattivo"[5] e possibilmente ottimizzare il throughput senza dover ricorrere all'istanziazione di più di uno stream, che aumenterebbe il livello di congestione della rete stessa e aumenterebbe il carico sulla macchina.
Si rivela dunque necessario ripetere le misure per la rete che poi effettivamente sarà utilizzata. Purtroppo al momento questo non è possibile; abbiamo tuttavia iniziato a valutare le performance su LAN a 100 Mbit/s in condizioni di traffico congestionato e no. Alcuni risultati preliminari, riportati qui sotto, indicano come lo stato di congestione della rete influenzi molto la performance; in particolare, sulla LAN in questione in quasi totale assenza di traffico, e quindi con un RTT molto basso, la TCP window size ottimale a 1 stream è pari a 3 Kbyte (throughput = 73 Mbit/s), mentre con una window size di 32 Kbyte sono necessari 8 stream per ottenere lo stesso throughput.
Alcuni risultati preliminari relativi a queste misure sono riportati nel seguente grafico (vedi fig. 4):

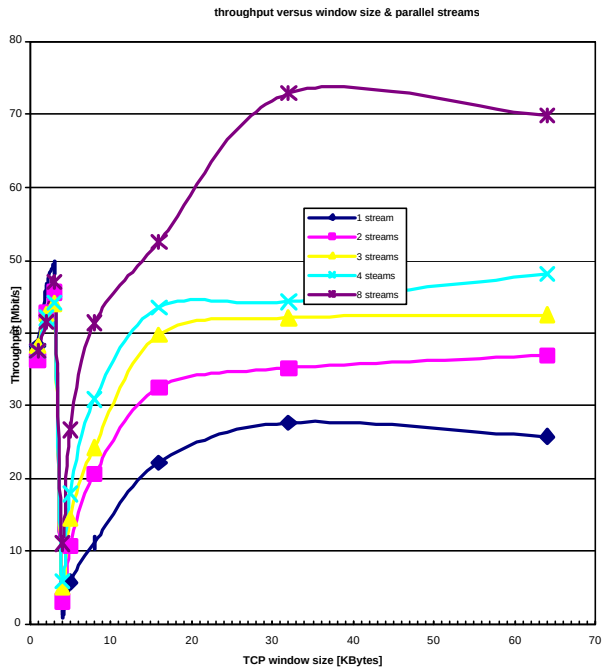

Fig. 4

Questo conferma l'ipotesi di dover ricorrere ad un window size "adattivo", essendo l'uso degli stream pesante a causa del carico indotto sulla

[5] Ovvero in funzione dello stato della rete in quel momento o in un periodo di tempo sufficientemente vicino.

macchina e comunque sconsigliato dagli RFC di TCP.

Conclusioni

In conclusione, dai risultati emersi e dai test di affidabilità, abbiamo una configurazione del sistema di trasferimento dati così fatto:

- Client/Server Perl5 con autenticazione SSL;
- Client in modalità fork;
- Bbftp con autenticazione privata;
- Bbftp a 1 stream e TCP window size "adattiva".

Il sistema così fatto è in funzione da circa 10 settimane e non ha mostrato alcun problema, confermando una affidabilità in termini di data loss dello 0.3% e di uso ottimale della banda passante disponibile.

Bibliografia

[1] M. Boschini, L. Trombetta "*Un DataBase Oracle per l'esperimento AMS*", Bollettino del CILEA, n. 64 Settembre 1998

[2] M. Boschini, L. Trombetta "*Il DataBase per AMS a un anno dal volo dello Space Shuttle STS-91*", Bollettino del CILEA, n. 68 Giugno 1999

[3] URL: http://ams.cern.ch

[4] M. Boschini e aa.vv. "*An Oracle (c) database for the AMS experiment*", NUCL PHYS B-PROC SUP 78, pagg. 727-731, August 1999;

[5] M. Boschini "*Stato del Progetto AMS*", Bollettino del CILEA, Giugno 2001

[6] URL: http://www.perl.org

[7] URL: http://www.mysql.com

[8] URL: http://www.openssh.org

[9] M. Boschini, A. Favalli "*Data Handling Tests*", AMS Note-2002-01-04, CERN Internal Reports

[10] URL: http://doc.in2p3.fr/bbftp/

[11] RFC1323

[12] URL: http://www.csm.ornl.gov/

[13] URL: http://archive.ncsa.uiuc.edu/

[14] URL: http://dast.nlanr.net/Projects/Iperf/