ATLAS-Italia
Tier-3 Task Force

Rapporto preliminare
V1.1, 6 maggio 2010

1 Introduzione

Questo rapporto preliminare viene pubblicato a circa due mesi dalla costituzione della task force. Il suo scopo è di presentare le informazioni finora raccolte e l'indirizzo attuale dei lavori, al fine di generare una discussione nella comunità ATLAS-Italia e poterne tenere conto nel seguito dei lavori e per il rapporto finale.

La task force è stata costituita all'inizio di marzo 2010 con lo scopo di definire un modello italiano per i Tier-3 di ATLAS, per quanto riguarda la loro funzionalità, la dimensione e le esigenze operative.

Questa attività deve complementare ed estendere il lavoro del gruppo di lavoro sui Tier-3 recentemente formato per tutto ATLAS e coordinato da Massimo Lamanna. L'attività ATLAS globale è partita con un workshop al CERN il 25-26 gennaio 2010[1]; altre informazioni in merito sono disponibili nella relativa pagina wiki[2].

La task force ha analizzato in dettaglio i seguenti aspetti dei Tier-3:

- Tipologia del lavoro svolto localmente
- Risorse necessarie in relazione al numero di persone attive
- Configurazione dei siti
    - Tier-3 co-locato con un Tier-2
    - Tier-3 integrato in INFN-Grid
    - Tier-3 per solo accesso locale

Le raccomandazioni contenute in questo rapporto, e nel rapporto finale, dovrebbero permettere di ottimizzare la dimensione e la configurazione dei Tier-3 per i primi anni di presa data di ATLAS.

2 Tipologia di lavoro svolto in un Tier-3

Bisogna innanzi tutto distinguere il lavoro "locale", essenzialmente interattivo o con un ciclo di sviluppo rapido, dal lavoro "globale" che viene normalmente eseguito inviando jobs batch alla Grid. Ci sono due attività principali che richiedono capacità di calcolo locale per essere efficienti: lo sviluppo del software per ATLAS e l'analisi dei dati.

Per lo sviluppo del software per ATLAS, cioè del software che gira nell'ambito di Athena e che diventerà parte dei release ufficiali, è necessario avere un'installazione locale del release base sul quale si lavora, accesso a un po' di dati di test e (se si lavora su dati reali) accesso al database e ai file di calibrazione.

La catena completa di un'analisi di fisica in generale comprende:

a) la simulazione di piccoli campioni di eventi di segnale e di fondo per il processo che si vuole studiare,
b) la generazione in grande scala di eventi simulati,
c) la selezione di eventi reali che potenzialmente corrispondono al processo in questione,
d) l'analisi finale degli eventi selezionati.

Le attività a) e d) vengono tipicamente svolte localmente (Tier-3), mentre le attività b) e c) sono naturalmente svolte sulla Grid (Tier-2 e, per le simulazioni, anche Tier-1). L'attività a) necessita un'installazione completa di Athena, similmente allo sviluppo del software discusso poco sopra. L'attività d) invece necessita solamente l'accesso ai dati selezionati (normalmente ntuple che devono venire trasferite nel Tier-3 per essere analizzati) e un'installazione locale efficiente per l'analisi (di solito basata su Proof).

Possiamo quindi considerare solo due gruppi di attività nel resto di questo rapporto:

- attività basate su Athena che necessitano un'installazione completa del software di ATLAS e accesso al database e ai file di calibrazione;
- attività basate su Proof che necessitano di un accesso veloce ai dati per l'analisi.

3 Risorse di calcolo necessarie

Per quanto detto sopra, distinguiamo i siti che intendono supportare attività basate su Athena dai siti che si limitano a supportare la fase finale dell'analisi basata su Proof. Per sviluppare software in Athena e testarlo prima di sottomettere jobs alla Grid serve una configurazione completa del sito ma non una capacità di calcolo enorme; come ordine di grandezza, dovrebbero bastare per il 2010 e 2011 2 TB di disco e 1 macchina (8 cores) per ogni utente del Tier-3.

Per l'analisi finale con Root o Proof bisogna prevedere abbastanza disco per metterci i files da analizzare (tipicamente ntuple), e capacità di trasmissione dei dati e di calcolo corrispondenti. Ad esempio, con 1 fb^{-1} di luminosità integrata avremo per l'analisi del top 400 GB di ntuple per i dati reali, e probabilmente altrettanti per i dati simulati. Considerando che normalmente si vogliono tenere più di una versione dei dati, ma anche che in ogni sito di solito lavorano più persone alla stessa analisi, si può indicare in 2 TB di disco per ogni utente un ordine di grandezza ragionevole e sufficiente fino al 2011. Per quanto riguarda la CPU, una macchina con 8 cores con Proof può analizzare fino a 80-100 MB/s di dati in ingresso; questo corrisponde a 4-5000 secondi per i 400 GB di cui sopra. Supponendo che ogni utente voglia analizzare un campione completo in un'ora, e che lo voglia fare 2 volte al giorno ma con un po' di codice di analisi che rallenta la velocità di processamento, per il cluster Proof necessitano almeno 2 macchine (8 cores) per utente ma non in uso esclusivo; una macchina per utente in media dovrebbe bastare.

Una componente importante di un Tier-3 è la rete interna, che definisce la banda passante fra i dischi e le CPU. Come detto sopra, una macchina con Proof può assorbire fino a circa 100 MB/s di dati, che saturano la porta Ethernet da 1 Gb/s. Chiaramente lo storage deve poter fornire dati a molto più di una singola macchina, per cui l'infrastrutture di rete interna deve essere basata su link a 10 Gb/s ed avere una

topologia ben progettata per eliminare eventuali colli di bottiglia (con dati dello stesso tipo distribuiti su parecchi disk server).

4 Configurazione dei siti

4.1 Siti che supportano Athena

4.1.1 Installazione del software

I Tier-3 che sono co-locati con un Tier-2 possono evidentemente utilizzare la stessa installazione del software del Tier-2. Gli altri Tier-3 devono provvedere un'area software di qualche centinaio di GB (a seconda del numero di release che si vogliono tenere) e possono scegliere fra due modi di installazione del software[3]:

- Utilizzare il sistema automatico che viene usato per i Tier-1 e i Tier-2 in produzione
- Utilizzare manualmente sw-mgr per installare solamente le release che interessano la comunità locale

In entrambi i casi deve essere definito un alias $VO_ATLAS_SW_DIR su tutte le macchine (interattive e batch), che punti all'area software. Nei Tier-1 e Tier-2 l'area software viene di solito implementata in NFS, e questa soluzione è raccomandabile anche per i Tier-3.

4.1.2 Configurazione dello storage

Il sistema di storage scelto dal Tier-1, dal Tier-2 di Milano e dai Tier-3 di Genova, Roma3 e Trieste è GPFS. L'esperienza fatta finora si riferisce a GPFS con StoRM come interfaccia SRM per la Grid.

Il vantaggio fondamentale di un file system di rete parallelo (come GPFS o Lustre) rispetto all'uso di uno storage element come DPM o dCache consiste nell'accessibilità locale dei files via interfaccia Posix. In questo modo tutti i dati disponibili in un certo sito sono visibili interattivamente, in batch locale e (per la parte registrata nel catalogo della Grid), da job Grid.

I Tier-3 co-locati con un Tier-2 dovrebbero avere già definito le aree di storage, mappate a fileset in GPFS; gli altri dovrebbero definirle come segue:

- SCRATCHDISK per mettere l'output dei job Grid che girano sul sito (pochi TB per iniziare)
- HOTDISK per conservare i files di calibrazione in formato POOL/ROOT che vengono usati da Athena (1 TB dovrebbe durare almeno fino alla fine del 2010)
- LOCALGROUPDISK per metterci files (o piuttosto i datasets) che si vogliono conservare per periodi prolungati (questa è la "home directory" di ogni utente Grid; 2 TB per utente come linea guida)
- Tutto il resto dello storage GPFS, che è invisibile alla Grid, non contiene files catalogati ed è per uso locale (almeno 2 TB per utente locale)

LOCALGROUPDISK non è strettamente necessario ma può essere utile. In questo spazio tutti gli utenti locali possono scrivere dataset registrati nel catalogo centrale e che sono visibili a tutto ATLAS. Questo spazio deve essere gestito con i tools di DDM per garantire

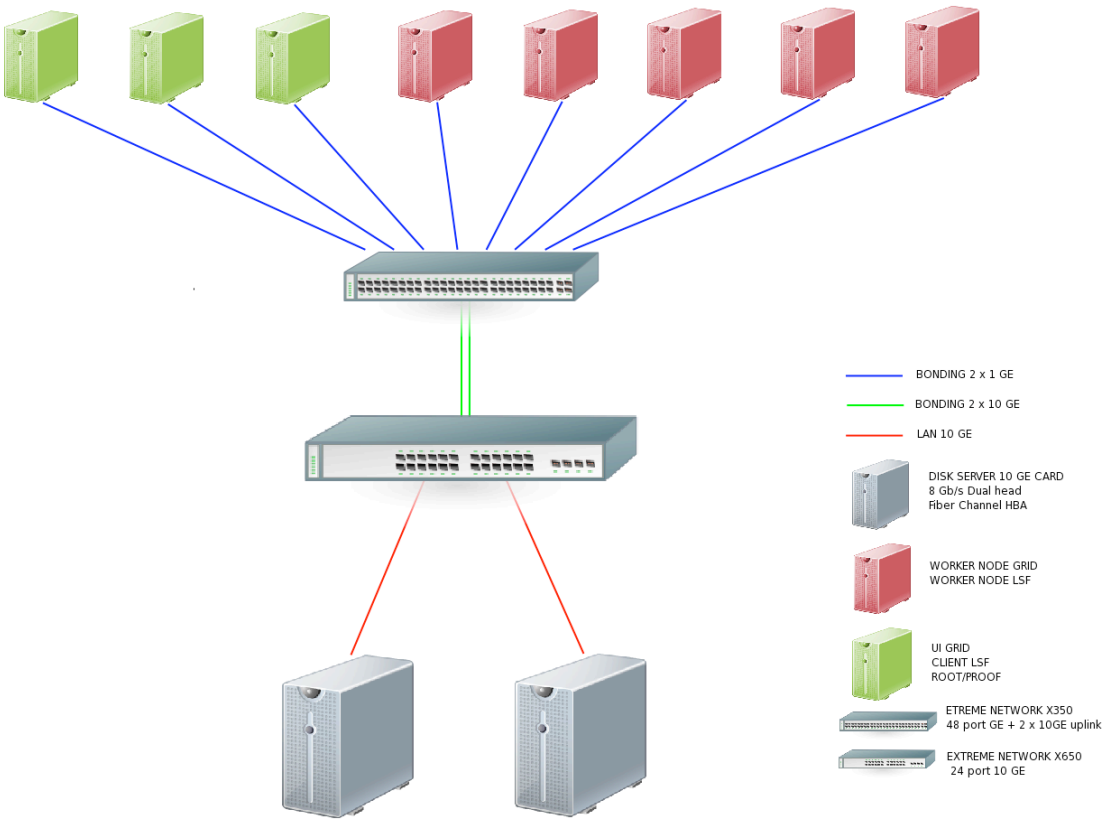
la consistenza fra quanto vi è effettivamente sul disco e la registrazione nei cataloghi; la gestione è interamente una responsabilità locale. I tools di DDM d'altra parte facilitano il trasferimento di questi dataset fra il LOCALGROUPDISK e tutti gli altri siti di ATLAS. Se LOCALGROUPDISK non è definito, per importare ed esportare dati bisogna usare dq2-get/dq2-put, avendo come destinazione (o sorgente) il file system locale (GPFS, NFS o qualsiasi altro); in questo caso, i dati importati, o prodotti localmente, non sono visibili al di fuori del sito in questione, ma possono essere utilizzati localmente sia in batch che interattivamente.

4.1.3 Configurazione dell'hardware

Si dà in questa sezione l'esempio della configurazione installata a Genova, come prototipo di sito integrato fra il Tier-3 e la farm locale. In questo modo si può razionalizzare il lavoro di gestione, pur dedicando le risorse necessarie alla porzione di infrastruttura dedicata al Tier-3.

4.1.3.1 Rete locale

La connettività locale tra i WN ed i server di disco e' realizzata utilizzando uno switch Extreme Networks Summit X650-24t (24 porte a 10 GE in rame) come backbone, e switch di rack Extreme Networks Summit X350-48t (48 porte GE con doppio uplink 10 GE in rame). La connessione tra gli switch è realizzata configurando un port trunk 2 * 10 GE gestito tramite protocollo LACP con algoritmo di selezione della porta basato sugli header dei pacchetti ai livelli 2 e 3 (MAC address + IP address), per garantire il massimo load balancing tra i singoli link.



Tutti gli switch sono dotati di architettura non-blocking, quindi la sola limitazione al throughput complessivo di questa configurazione è data dal doppio uplink degli switch

di rack, che forniscono 20 Gbps di aggregato verso i disk server ai 48 potenziali WN connessi alle porte GE (oversubscription 2.4:1).

Questa infrastruttura, così come quella relativa allo storage, è condivisa con il resto delle infrastrutture di calcolo presenti in sezione: farm Grid, farm locale e tutti i volumi di storage di servizio o di esperimento gestiti dal Servizio Calcolo.

4.1.3.2 Worker Node

I nodi dedicati al Tier-3 sono attualmente divisi staticamente in nodi per l'interattivo:

- 3 nodi biprocessori quad-core Intel Xeon E5520 a 2.27 GHz con hyperthreading abilitato (16 core virtuali per nodo), 24 GB di RAM, 250 GB di HD locale SATA, doppia scheda di rete GE integrata

e nodi per il batch:

- 3 nodi biprocessori dual-core AMD Opteron 2218 a 2.6 GHz (4 core per nodo), 8 GB di RAM, 250 GB di HD locale SATA. doppia scheda di rete GE integrata
- 2 nodi biprocessori quad-core Intel Xeon E5410 a 2.3 GHz (8 core per nodo), 16 GB di RAM, 250 GB di HD SATA, doppia scheda di rete GE integrata

L'accesso all'infrastruttura avviene attraverso i nodi per l'interattivo, che, oltre all'accesso interattivo (root, Proof), sono configurati per poter sottomettere job batch sia attraverso la GRID (sono user interface di INFN-GRID) che direttamente sul batch system locale, gestito tramite LSF (sono client del cluster LSF).

Il cluster LSF è quello utilizzato per l'infrastruttura GRID locale, con una opportuna configurazione di LSF per gestire lo sharing delle risorse fra i job GRID general purpose, i job GRID ed i job locali degli utenti del Tier-3.

I nodi per il batch sono configurati in modo identico ai WN della infrastruttura GRID; questo permette una gestione omogenea dei server di calcolo e la possibilità di realizzare – tramite LSF – uno sharing delle risorse flessibile e modificabile dinamicamente a seconda delle esigenze.

4.1.3.3 Storage

Tutti i volumi utilizzati (per il Tier-3 e non) sono esportati verso i disk server via Storage Area Network basata su Fibre Channel, e gestiti tramite file system GPFS.

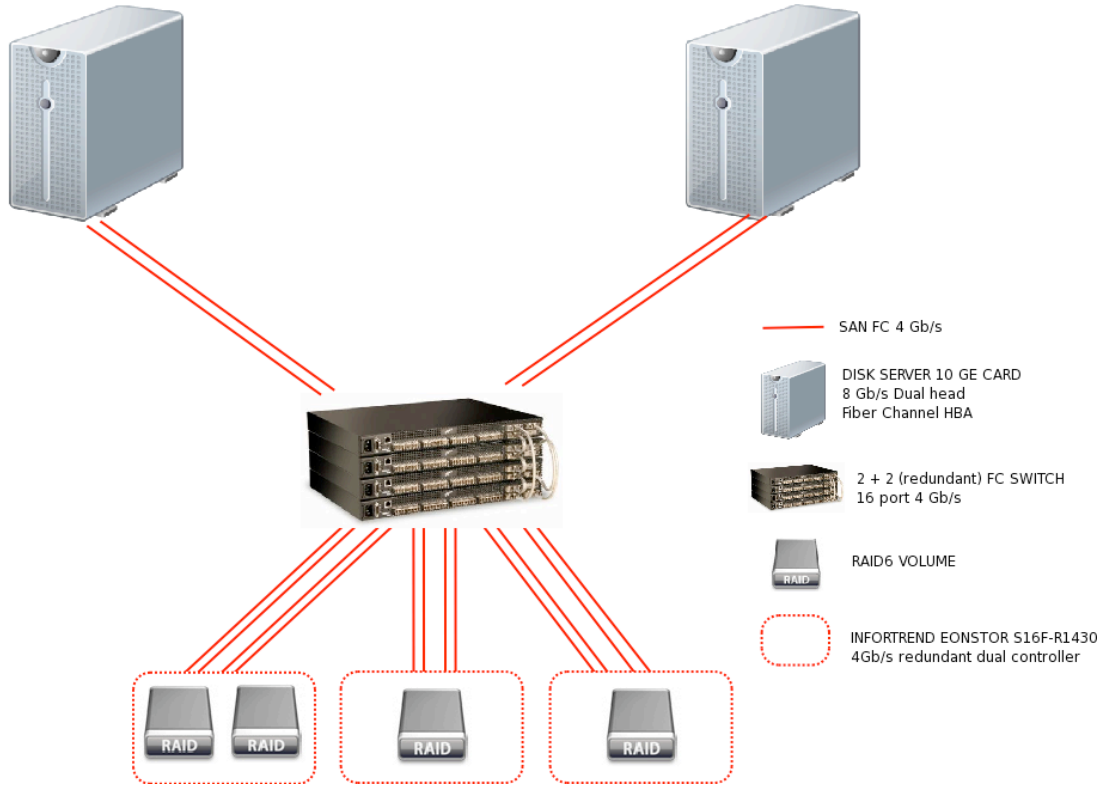
I volumi dedicati al Tier-3 sono quattro raid set configurati in RAID6 ed esportati da tre controller Infortrend Eonstor S16F-R1430 ciascuno con doppio controller a due canali FC a 4 Gbps. La SAN è costituita da due switch FC SanBox 5600 in configurazione di ridondanza.

I volumi sono configurati in un unico file system, in cui sono configurati i fileset ATLASSCRATCHDISK (4 TB), ATLASHOTDISK (1 TB), ATLASLOCALGROUPDISK (4 TB) per i token space di ATLAS, a cui si aggiunge un fileset ATLAS_GEN (9 TB) per lo spazio disco locale inaccessibile allo storage element. Le dimensioni dei fileset sono gestite tramite la quota di GPFS, e quindi modificabili dinamicamente, così come è aumentabile dinamicamente lo spazio disco totale del file system sempre grazie alla flessibilità di GPFS.

Tutto lo spazio disco descritto è accessibile direttamente da tutti i nodi (interattivi e batch) via Posix. Le aree relative agli space token di ATLAS sono ovviamente accessibili anche via SRM tramite l'SRM (StoRM) del sito grid locale, che gestisce automaticamente gli ACL per consentire l'accesso locale in modo controllato.

Alle aree dati si aggiunge un'area dedicata al software di ATLAS; il software viene importato tramite il sistema automatico operante via GRID utilizzato anche per i Tier-1 e Tier-2.

Il volume per il software è anch'esso configurato su GPFS, ma viene esportato verso i WN e le user interface via NFS per motivi di prestazioni; il meccanismo di export utilizzato è un Clustered NFS, costituito da due server, che integra le caratteristiche di fail-over ereditate da GPFS.



4.1.3.4 Disk Server

I volumi del Tier-3 sono esportati via GPFS da due disk server biprocessori Intel Xeon L5520 a 2.2 GHz (8 core totali), dotati di 24 GB di RAM. I server sono connessi tramite un'interfaccia 10 GE in rame direttamente allo switch di backbone. La connessione alla SAN avviene tramite una scheda FC dual-head 8 Gbps per porta (utilizzate a 4 Gbps per via della infrastruttura della SAN) configurate in modalità fail-over e load-balancing, garantendo quindi un throughput teorico di 800 MB/s per server, ampiamente superiore alle esigenze previste per questa fase.

I due server esportano ciascuno due dei quattro volumi raid configurati sul file system GPFS (NSD nella terminologia GPFS); anch'essi sono configurati per esportare gli NSD assegnati all'altro server dinamicamente in caso di failure, realizzando così un sistema ad alta affidabilità.

I due disk server sono anche configurati come GridFTP server per esportare gli space token di ATLAS anche tramite protocollo GridFTP, potendo così sfruttare appieno la banda di accesso al disco anche con questo protocollo.

4.1.4 Modello di lavoro

Il modello di lavoro qui proposto è possibile a causa del facile accesso diretto (via Posix) a tutti i files nello storage element. L'area con il software è anche condivisa fra macchine interattive e batch. Si può quindi sviluppare codice interattivamente, testarlo con piccoli jobs mandati al batch locale, provarlo ulteriormente in ambiente Grid (ma sempre locale), prima di mandare produzioni private o di gruppo (tipicamente selezioni di eventi) che devono essere eseguiti sulla Grid.

L'output dei job locali chiaramente deve essere salvato nell'area non-Grid dello storage; l'output dei job Grid locali può essere salvato ovunque, previa registrazione nei cataloghi LFC e DDM se si usa uno degli space token di ATLAS. L'output dei job Grid remoti può essere salvato sullo SCRATCHDISK dei sito dove girano i job e recuperato in un secondo tempo, o inviato direttamente a uno degli space token locali e registrato.

Nello schema generale del data management di ATLAS, tutti gli utenti possono scrivere in SCRATCHDISK ma solo gli utenti locali in LOCALGROUPDISK; tutti però possono accedere in lettura a tutti i dati, ovunque siano. Il trasferimento di dati da e allo storage gestito centralmente avviene utilizzando i tools di ATLAS/DDM dq2-get e dq2-put, che garantiscono la consistenza fra storage e cataloghi.

Tutti i Tier-2 hanno anche una componente Tier-3. Per i Tier-2 con StoRM/GPFS (solo Milano per il momento) vale il discorso fatto sopra per Genova. Agli altri Tier-2, con storage element DPM, il modello di lavoro deve essere adattato al fatto che DPM non dà accesso con il protocollo Posix al suo file system, ma bisogna utilizzare il protocollo rfio.

4.1.5 Test del sistema

I test stanno iniziando ora (inizio maggio 2010) a Genova, in coordinamento con Trieste e Roma3. Si prevedono i seguenti test iniziali:

- Sottomissione di job Athena in modo batch locale (test di funzionalità).
- Sottomissione di job Athena in modo Grid (test di funzionalità).
- Sottomissione di test automatizzati HammerCloud (test di throughput fra storage e CPU farm).

4.2 Siti che supportano Proof

Si dà in questa sezione l'esempio della configurazione installata a Milano e dei test effettuati finora. Per una descrizione completa del sistema testato a Milano e dei risultati dettagliati dei test si rimanda all'Appendice A: Test di PROOF.

4.2.1 Configurazione dello storage

La configurazione di storage che abbiamo utilizzato per i nostri test prevede l'utilizzo di GPFS come file system distribuito. E' stata creata all'interno dello Storage Element di Milano un'area "utenti", gestita da GPFS, al di fuori degli space tokens di ATLAS, dove sono state salvate le ntuple usate per il test. Altri file systems dovrebbero avere

prestazioni più o meno equivalenti ma non sono (ancora) stati testati. File system basati su DPM (presenti nei Tier-2 di Roma1, Napoli e Frascati) sono di utilizzo più complesso per un uso interattivo, e anche loro non sono ancora stati testati.

4.2.2 Configurazione dell'hardware

Il cluster sul quale sono state effettuati i test è composto da tre macchine, situate presso la sezione di Milano dell'INFN, con la seguente configurazione:

- Dual Intel Xeon E5440 2.83Ghz
- 16GB RAM
- Scientific Linux 5.4
- ROOT 5.26/00b

Le macchine sono collegate ad una rete con una banda di 1Gb/s tramite un unico switch. Da questa rete è visibile un cluster GPFS che viene utilizzato dagli utenti sia per le proprie home che per accedere direttamente allo Storage Element locale di ATLAS.

4.2.3 Modello di lavoro

L'utilizzo di PROOF all'interno di un Tier-3 può essere effettuato sia interattivamente (nelle versioni proofLITE su singolo nodo multi-core e/o nella versione "completa" in cui si ha a disposizione un cluster di diversi nodi) che utilizzando il sistema locale per la sottomissione di jobs sulle risorse disponibili. Finora è stata testata solamente la prima tipologia di utilizzo con entrambe le versioni di PROOF.

All'interno del sistema di calcolo e distribuzione dati di ATLAS, PROOF è pensato per agire sui D3PD, o comunque su ntuple di ROOT, che l'utente ha prodotto con i job di analisi distribuita all'interno del framework di Athena girando sui dati (in formato AOD o ESD generalmente) nei vari Tier-2 e che si è poi copiato nello Storage Element locale usando i vari tools di GRID (dq2-get e sottoscrizioni principalmente).

La dimensione di queste ntuple è variabile, sia in funzione del tipo di analisi che l'utente vuole fare che del periodo che l'esperimento sta vivendo, ma è ragionevole pensare che in situazioni standard essa si mantenga entro qualche decina di kB/evento. Il formato più comune è quello basato sulla classe TTree contenente, evento per evento, le varie grandezze fisiche utili all'analisi (energia mancante, quadri momenti di leptoni e getti, etc.) su cui girare il proprio codice per selezionare gli eventi di interesse, produrre gli istogrammi e salvarli su un file di ROOT.

4.2.4 Test del sistema

Le ntuple utilizzate per il test sono ntuple di ROOT prodotte a partire da AOD Monte Carlo di QCD e top per cercare di riprodurre uno scenario realistico con due diversi tipologie di eventi. Si tratta rispettivamente di 15M di eventi di QCD (dimensione 250 B/ev.) e di 2M di eventi di top (circa 500 B/ev). Data la differente dimensione delle ntuple, le prestazioni di PROOF sono state analizzate ponendo attenzione non tanto al rate di eventi letti (in Hz), quanto piuttosto alla velocità di lettura (in MB/s) per cercare di normalizzare le prestazioni tra i diversi tipi di ntuple.

Il codice di analisi che gira sulle ntuple è stato scritto in 4 versioni:

1) Codice completo che legge tutti i branches dell'ntupla, li analizza e produce un centinaio di istogrammi di controllo;
2) Codice iper-semplificato che legge tutti i branches dell'ntupla ma ne utilizza solo un paio in un semplice loop , producendo solo qualche istogramma;
3) Codice iper-semplificato come al punto 2) che legge solo i 2 branches dell'ntupla che effettivamente utilizza producendo solo qualche istogramma;
4) Codice iper-semplificato come al punto 2) che legge solo i 2 branches dell'ntupla che effettivamente utilizza ma effettua molto calcolo (ciclo for da 20000 operazioni algebriche) producendo solo qualche istogramma;

Per ognuna di queste 4 versioni sono stati monitorati il numero di eventi processati al secondo (in Hz) e la velocità di lettura (in MB/s). I numeri che ora verranno mostrati si riferiscono al processamento dei 15 M di eventi di QCD (250 B/ev). Le ntuple di top danno risultati simili se si utilizza la velocità di lettura come parametro di analisi.

Il primo test effettuato ha riguardato la scalabilità di PROOF Lite e di PROOF "versione cluster" con il numero di nodi a disposizione. Per valutare eventuali problematiche legate alla lettura dallo storage o alla configurazione della rete, abbiamo utilizzato il codice iper-semplificato di tipo 3, riducendo al minimo il calcolo che deve essere svolto nella macro di analisi. Dettagli dei test e dei loro risultati sono riportati nell'Appendice A: Test di PROOF.

Alla luce dei test effettuati possiamo arrivare a evidenziare i seguenti punti da tenere in considerazione per l'uso di PROOF interfacciato ad un sistema di storage gestito da GPFS.

1) L'uso di PROOF Lite risulta scalare bene (entro un 10% dall'idealità) con il numero dei cores fino a circa 5-6 cores. Aumentando ulteriormente il numero dei cores nella versione PROOF "cluster" ci si discosta sempre più dall'idealità, arrivando a un fattore oscillante tra 15.8 e 19.8 per i diversi tipi di codice testati nella versione a 24 cores.
2) Con una macro di analisi di media complessità, la velocità di lettura dallo storage non risulta essere limitata né dalla velocità intrinseca del disco né da quella dell'unico switch di rete presente nella configurazione utilizzata. Inoltre, la parallelizzazione dell'analisi realizzata da PROOF può garantire una velocità di lettura nettamente superiore a quella che garantirebbe lo switch della rete nella versione "single core".
3) La scalabilità di PROOF migliora con la complessità del codice di analisi (e quindi con il consumo di CPU).
4) Dal confronto tra i 4 tipi di codice emerge come gran parte della differenza tra una macro complessa e una iper-semplice sia dovuta alla lettura e all'allocazione in memoria dei vari branches dell'ntupla piuttosto che alla complessità del calcolo richiesto dalla macro. E' pertanto opportuno, soprattutto con ntuple molto grandi, leggere solamente i branches effettivamente utilizzati nel codice di analisi.

4.3 Siti integrati Athena/Proof

Essendo Proof un tool essenziale per l'analisi parallela di ntuple, che normalmente costituisce la parte finale di qualsiasi analisi, tutti i siti Grid dovranno anche supportare accesso interattivo per Root e Proof. Come detto nella sezione 4.1.4, l'integrazione fra parte interattiva e parte batch/Grid in un sito è molto più agevole se lo storage è basato su di un file system con protocollo Posix (come GPFS o Lustre) piuttosto che su uno storage element come DPM.

In questo campo, un lavoro interessante e potenzialmente molto utile ancora da fare è lo studio del cosiddetto "proof on demand". L'idea base è di poter fornire proof slaves sui worker nodes del sistema batch, lanciando un job batch che configura il worker node su cui gira come un proof slave, si connette al proof master (che invece sarà una delle macchine interattive dedicate) e contribuisce al carico di lavoro di Proof. Quando il lavoro è terminato, il job finisce e il worker node ridiviene disponibile per un altro job batch.

La gestione di un sistema come quello qui descritto è tutt'altro che ovvia, per cui una fase estesa di test di configurazioni e performance deve essere prevista a breve termine.

5 Conclusioni (preliminari)

La task force si è finora concentrata sulla definizione della funzionalità dei Tier-3 e della configurazione hardware e software necessaria. Test di performance del cluster Proof a Milano sono stati eseguiti con risultati positivi.

Prima del rapporto finale rimangono da effettuare un certo numero di test:

- Test di funzionalità e performance dei Tier-3 "Grid-enabled":
    - Definizione di dataset e configurazioni per i test comuni
    - Sottomissione di job Athena in modo batch locale (test di funzionalità)
    - Sottomissione di job Athena in modo Grid (test di funzionalità)
    - Sottomissione di test automatizzati HammerCloud (test di throughput fra storage e CPU farm)
- Test di sistema integrato Grid/locale in tutti i siti
    - Performance di Proof con dati in input nello storage element, particolarmente per i siti con DPM

Inoltre va ancora definito il modello di supporto per i siti e per gli utenti. Più a lungo termine, sarebbe interessante studiare "proof on demand".

6 Appendice A: Test di PROOF

6.1 Introduzione

Parallel ROOT Facility (PROOF) è un plugin di xrootd che fornisce degli strumenti per l'analisi interattiva con ROOT su sistemi paralleli e distribuiti. Il sistema è stato pensato inizialmente per jobs di durata medio-bassa per i quali il tempo di attesa su di un sistema batch tradizionale sarebbe troppo elevato.

PROOF permette di eseguire analisi parallele su catene di *TTree* sfruttando la non correlazione della struttura dati di eventi sottostante (parallelismo imbarazzante).

PROOF viene rilasciato con una versione "Lite" che permette di sfruttare le potenzialità di PROOF su di una singola macchina multi-core.

6.2 Architettura di PROOF

L'architettura di PROOF è composta da più strati: client, master di primo livello, masters di secondo livello (opzionali) e nodi worker. Il master di primo livello rappresenta il punto d'ingresso al cluster PROOF. Il suo scopo principale è di ricevere le richieste dai client, di distribuire codice e il lavoro ai workers e di fondere i risultati parziali al termine dell'analisi e quindi di inviare il risultato al client. Un master di primo livello può delegare alcuni dei suoi ruoli, come la fusione dei risultati parziali, ai master di secondo livello aumentando le performance complessive del sistema. In PROOF-lite è presente solo un master di primo livello e una serie di worker che vengono eseguiti sulla stessa macchina multi-core. In seguito considereremo esclusivamente il caso con un master di primo livello e nessun master di secondo livello.

Il caso d'uso generico prevede che l'utente si connetti da una sessione ROOT ad un master che a sua volta provvede ad istanziare un certo numero di slave server sui worker. Ogni utente possiede una *sandbox* locale sui master e sui worker che viene usata come cache e per salvare dei file di log della sessione PROOF corrente.

L'insieme di eventi da processare viene suddiviso in pacchetti composti dall'URL del dataset e da un intervallo da considerare. Tali pacchetti vengono inviati dal master agli slave. La dimensione di un pacchetto e lo slave assegnato ad esso dipendono dalla potenza di calcolo e dalla topologia del dataset.

Non appena uno slave finisce di processare un pacchetto ne richiede uno nuovo al master, questo comportamento permette di bilanciare il carico degli slave. Quando la macro è completata i risultati parziali vengono inviati dagli slave al master che provvede a fonderli in un unico risultato e a rispedire il tutto al client.

Questo richiede che i risultati parziali generati dagli slave possano essere "fusi"; è possibile implementare un'interfaccia di merging che permette, anche ai tipi definiti dall'utente, di essere utilizzati trasparentemente in PROOF.

Il sistema inoltre prevede la possibilità di monitorare il risultato finale mano a mano che viene costruito e di caricare su worker dei pacchetti contenenti delle librerie specifiche.

6.3 Selector framework

Un *TSelector* è un tipo che incapsula il codice specifico di un'analisi che lavora su di un certo *TTree*. La classe base fornisce dei metodi ridefinibili che permettono di eseguire del codice prima e dopo l'analisi di un *TTree* e del codice che viene eseguito ad ogni evento. Dato un *TTree* è possibile creare il corrispondente TSelector utilizzando il metodo *TTree::MakeSelector*.

PROOF usa un *TSelector* per esprimere una computazione parallela. Questa scelta permette di scrivere codice che funziona sia in una sessione ROOT standard che su di un sistema distribuito.

Il flusso di esecuzione della generica sessione di PROOF, dal punto di vista del selettore, è il seguente:

1. Sulla macchina client viene invocato il metodo *Begin*; questo metodo provvede ad inizializzare la sessione client e a preparare un'eventuale lista di oggetti di input da inviare al master assieme al codice per l'analisi;

2. Ogni slave crea un'istanza del selettore impostando la lista di oggetti in input a quella ricevuta dal master. A questo punto viene chiamato il metodo *SlaveBegin*;

3. Lo slave riceve i pacchetti da processare dal master e processa gli eventi in essi puntati con il metodo *Process*;

4. Lo slave prima di terminare, quando non c'è più lavoro disponibile, invoca il metodo *SlaveTerminate* per ripulire il nodo e provvede ad inviare la sua lista di oggetti di output al master che provvederà a fondere le liste ottenute dai vari slave in un'unica lista da inviare al client;

5. Il metodo *Terminate* viene invocato sul client

6.4 Gestione risorse e bilanciamento del carico

Il bilanciamento del carico viene gestito dal *packetizer*. Questo componente usa un algoritmo adattivo per evitare il sovraccarico dei nodi e per ogni sessione viene creata un'istanza specifica sul master. Un architettura di tipo pull è implementata in maniera tale che gli slave chiedono un nuovo pacchetto solo quando hanno finito di elaborare quello corrente. Per sfruttare al meglio la località dei dati sui worker, l'algoritmo adattivo assegna dei pacchetti di dati locali ad uno slave solo se il tempo di elaborazione di tutti i dati locali supera il tempo di esecuzione previsto del job sul cluster; se questo non è il caso allora allo slave viene assegnato un pacchetto di dati remoti appartenente eventualmente ad altri nodi che potrebbero in un secondo momento diventare dei "colli di bottiglia".

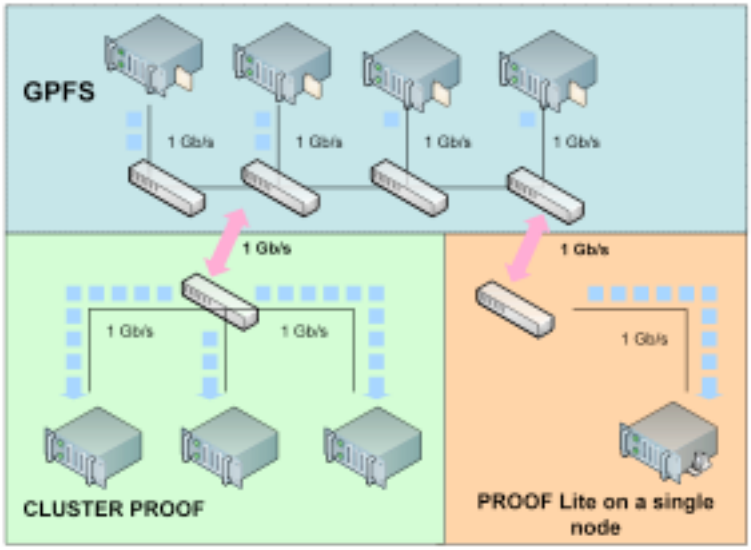
Lo *scheduler* di PROOF ha il compito di assegnare un certo numero di *slaves* ad un job specifico e ha anche l'autorità di rifiutare o mettere in coda un job se le risorse di calcolo disponibili non sono sufficienti.

6.5 Configurazione dello storage

La configurazione di storage che abbiamo utilizzato per i nostri test prevede l'utilizzo di GPFS come filesystem distribuito. E' stata creata all'interno dello Storage Element di

Milano un'area "utenti", gestita da GPFS, al di fuori degli spacetokens di ATLAS, dove sono state salvate le ntuple usate per il test.

In figura è mostrato uno schema grafico della configurazione usata per mettere in comunicazione lo storage con GPFS e il "cluster" dove è installato PROOF.



Nella parte alta della figura è stato schematizzato lo Storage Element gestito da GPFS, in basso a sinistra il cluster PROOF costituito dai 3 nodi di calcolo e infine in basso a destra la comunicazione in caso di singola macchina con l'installazione di PROOF Lite.

Come si può notare, ogni collegamento di rete è realizzato attraverso degli switch da 1 Gigabit/s, e ciò potrebbe essere una limitazione alla parallelizzazione operata da PROOF.

Innanzitutto è stata verificata la velocità di lettura I/O di GPFS separatamente rispetto a PROOF, e i risultati hanno mostrato che il sistema è sostanzialmente limitato dalla portata dello switch di rete (in questo caso 1 Gigabit/s, ovvero circa 118 MB/s) ed è sostanzialmente indipendente dalla dimensione dell'ntupla da analizzare.

In particolare, la configurazione utilizzata, che prevede un unico switch verso lo Storage Element da 1 Gigabit/s (condiviso anche da altri nodi della farm), grazie alla parallelizzazione operata da PROOF, sarebbe in grado di garantire al massimo 3x118 MB/s = 354 MB/s in caso di rete esclusivamente dedicata a questa attività.

Come mostrano i test prestazionali descritti nelle sezioni successive, l'accesso ad uno Storage Element distribuito, non costituisce una fonte di rallentamento per PROOF, visto che il tempo medio di analisi del singolo evento è decisamente maggiore della velocità di comunicazione I/O con lo storage stesso. Semmai il "collo di bottiglia" può essere costituito dallo switch di rete, sebbene il problema venga fortemente ridimensionato dalla capacità di parallelizzazione di PROOF, almeno per la configurazione a 3 nodi che abbiamo testato.

6.6 Configurazione dell'hardware

Il cluster sul quale sono state effettuati i test è composto da tre macchine, situate presso la sezione di Milano dell'INFN, con la seguente configurazione:

- Dual Intel Xeon E5440 2.83Ghz
- 16GB RAM
- Scientific Linux 5.4
- ROOT 5.26/00b

Le macchine sono collegate ad una rete con una banda di 1Gb/s tramite un unico switch. Da questa rete è visibile un cluster GPFS che viene utilizzato dagli utenti sia per le proprie home che per accedere direttamente allo Storage Element locale di ATLAS.

Per gestire l'autenticazione dei client sul cluster si è scelto di usare una o più macchine come User Interface (UI) dalle quali sono visibili i master. Nel setup corrente le macchine del cluster fungono anche da UI e sono bilanciate sotto IP pubblico comune. I permessi per accedere alle UI per il cluster di PROOF sono i medesimi di quelli per accedere alla UI di GRID. Le tre macchine dedicate a PROOF utilizzate per i test possono sia fungere da master che da *slave*:

xpd.worker worker t2-wn-[09-11] repeat=8

Ogni macchina ha un proprio spazio locale utilizzato per le sandbox delle sessioni.

La gestione delle risorse è impostato in maniera tale da assegnare la metà dei worker ad una sessione utente per un massimo di quattro sessioni concorrenti in maniera tale da non sovraccaricare i nodi perdendo i benefici del calcolo parallelo. I worker vengono assegnati con una politica round-robin e le richieste aggiuntive vengono accodate in una coda con politica FIFO in attesa che una delle sessioni in esecuzione passi in uno stato di idle:

xpd.schedparam selopt:roundrobin wmx:12 queue:fifo mxrun:4

6.7 Modello di lavoro

L'utilizzo di PROOF all'interno di un Tier-3 può essere effettuato sia interattivamente (nelle versioni proofLITE su singolo nodo multi-core e/o nella versione "completa" in cui si ha a disposizione un cluster di diversi nodi) che utilizzando il sistema locale per la sottomissione di jobs sulle risorse disponibili. Finora è stata testata solamente la prima tipologia di utilizzo con entrambe le versioni di PROOF.

All'interno del sistema di calcolo e distribuzione dati di ATLAS, PROOF è pensato per agire sui D3PD, o comunque su ntuple di ROOT, che l'utente ha prodotto con i job di analisi distribuita all'interno del framework di Athena girando sui dati (in formato AOD o ESD generalmente) nei vari Tier-2 e che si è poi copiato nello Storage Element locale usando i vari tools di GRID (dq2-get e sottoscrizioni principalmente).

La dimensione di queste ntuple è variabile, sia in funzione del tipo di analisi che l'utente vuole fare che del periodo che l'esperimento sta vivendo, ma è ragionevole pensare che in situazioni standard essa si mantenga entro qualche decina di KB/evento. Il formato più comune è quello basato sulla classe TTree contenente, evento per evento, le varie grandezze fisiche utili all'analisi (energia mancante, quadri momenti di leptoni e getti,

etc.) su cui girare il proprio codice per selezionare gli eventi di interesse, produrre gli istogrammi e salvarli su un file di ROOT.

6.8 Test del sistema

Le ntuple utilizzate per il test sono ntuple di ROOT prodotte a partire da AOD Monte Carlo di QCD e top per cercare di riprodurre uno scenario realistico con due diversi tipologie di eventi. Si tratta rispettivamente di 15M di eventi di QCD (dimensione 250 B/ev.) e di 2M di eventi di top (circa 500 B/ev). Data la differente dimensione delle ntuple, le prestazioni di PROOF sono state analizzate ponendo attenzione non tanto al rate di eventi letti (in Hz), quanto piuttosto alla velocità di lettura (in MB/s) per cercare di normalizzare le prestazioni tra i diversi tipi di ntuple.

Il codice di analisi che gira sulle ntuple è stato scritto in 4 versioni:

1) Codice completo che legge tutti i branches dell'ntupla, li analizza e produce un centinaio di istogrammi di controllo;
2) Codice iper-semplificato che legge tutti i branches dell'ntupla ma ne utilizza solo un paio in un semplice loop , producendo solo qualche istogramma;
3) Codice iper-semplificato come al punto 2) che legge solo i 2 branches dell'ntupla che effettivamente utilizza producendo solo qualche istogramma;
4) Codice iper-semplificato come al punto 2) che legge solo i 2 branches dell'ntupla che effettivamente utilizza ma effettua molto calcolo (ciclo for da 20000 operazioni algebriche) producendo solo qualche istogramma;

Per ognuna di queste 4 versioni sono stati monitorati il numero di eventi processati al secondo (in Hz) e la velocità di lettura (in MB/s). I numeri che ora verranno mostrati si riferiscono al processamento dei 15 M di eventi di QCD (250 B/ev). Le ntuple di top danno risultati simili se si utilizza la velocità di lettura come parametro di analisi.

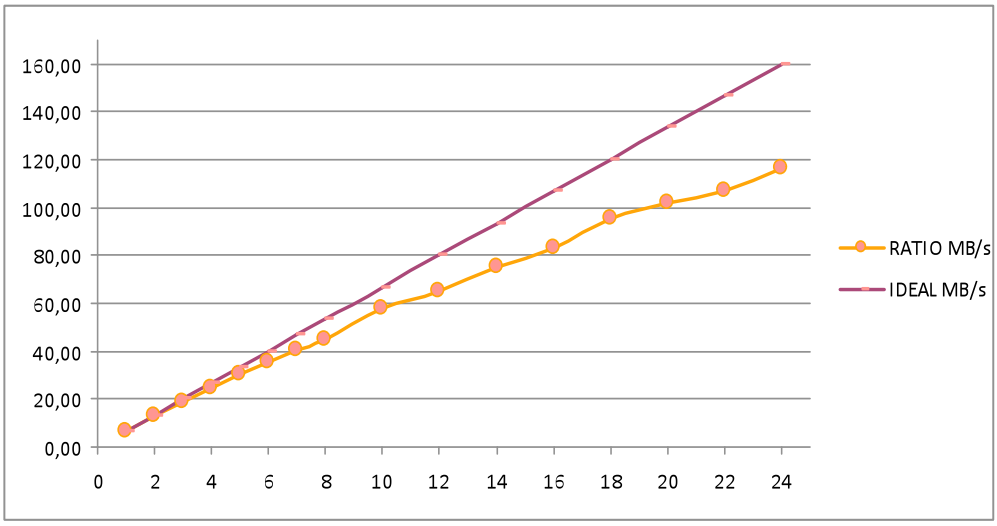
6.8.1 Scalabilità

Il primo test effettuato ha riguardato la scalabilità di PROOF Lite e di PROOF "versione cluster" con il numero di nodi a disposizione.

In tabella e nel grafico, sono visibili le prestazioni per il codice di tipo 1:

	Nr. Cores	FREQ. KHz	FREQ/IDEALE	VELOCITA' I/O MB/s
PROOF LITE	**1**	**28,00**	**1,00**	**6,68**
PROOF LITE	**2**	**54,00**	**1,93**	**12,87**
PROOF LITE	**3**	**78,67**	**2,81**	**18,76**
PROOF LITE	**4**	**103,00**	**3,68**	**24,56**
PROOF LITE	**5**	**127,50**	**4,55**	**30,40**
PROOF LITE	**6**	**149,50**	**5,34**	**35,64**
PROOF LITE	**7**	**168,75**	**6,03**	**40,23**
PROOF LITE	**8**	**187,25**	**6,69**	**44,64**
CLUSTER	**10**	**241,53**	**8,63**	**57,59**
CLUSTER	**12**	**274,53**	**9,80**	**65,45**
CLUSTER	**14**	**316,98**	**11,32**	**75,57**
CLUSTER	**16**	**349,73**	**12,49**	**83,38**
CLUSTER	**18**	**400,10**	**14,29**	**95,39**
CLUSTER	**20**	**426,73**	**15,24**	**101,74**

CLUSTER	**22**	**448,55**	**16,02**	**106,94**
CLUSTER	**24**	**488,98**	**17,46**	**116,58**



Come si può notare, l'idealità nella scalabilità viene mantenuta fino a circa 5-6 cores nel caso di PROOF Lite, dove si scende al di sotto del 90% della velocità di lettura ideale.

Il passaggio dalla versione Lite a quella "clustered" consente comunque di migliorare le prestazioni, arrivando ad un miglioramento nella velocità di lettura fino ad un fattore 17.46 rispetto al singolo core, anche se va valutato attentamente l'effettivo guadagno nell'aumento del numero di cores allorché ci siano più utenti in contemporanea ad utilizzare il cluster.

6.8.2 Velocità di lettura dallo storage

Per valutare eventuali problematiche legate alla lettura dallo storage o alla configurazione della rete, abbiamo utilizzato il codice iper-semplificato di tipo 3, riducendo al minimo il calcolo che deve essere svolto nella macro di analisi.

In tabella si possono vedere i risultati ottenuti in condizioni analoghe al test del punto precedente:

	CORES	FREQ. KHz	FREQ/IDEALE	VELOCITA' I/O MB/s
PROOF LITE	**1**	**55,67**	**1,00**	**13,27**
PROOF LITE	**2**	**105,00**	**1,89**	**25,03**
PROOF LITE	**3**	**151,67**	**2,72**	**36,16**
PROOF LITE	**4**	**202,25**	**3,63**	**48,22**
PROOF LITE	**5**	**246,33**	**4,43**	**58,73**
PROOF LITE	**6**	**304,50**	**5,47**	**72,60**
PROOF LITE	**7**	**344,00**	**6,18**	**82,02**
PROOF LITE	**8**	**385,00**	**6,92**	**91,79**
CLUSTER	**10**	**479,75**	**8,62**	**114,38**

CLUSTER	**12**	**542,00**	**9,74**	**129,22**
CLUSTER	**14**	**578,50**	**10,39**	**137,93**
CLUSTER	**16**	**625,75**	**11,24**	**149,19**
CLUSTER	**18**	**727,75**	**13,07**	**173,51**
CLUSTER	**20**	**778,50**	**13,99**	**185,61**
CLUSTER	**22**	**842,00**	**15,13**	**200,75**
CLUSTER	**24**	**879,75**	**15,80**	**209,75**

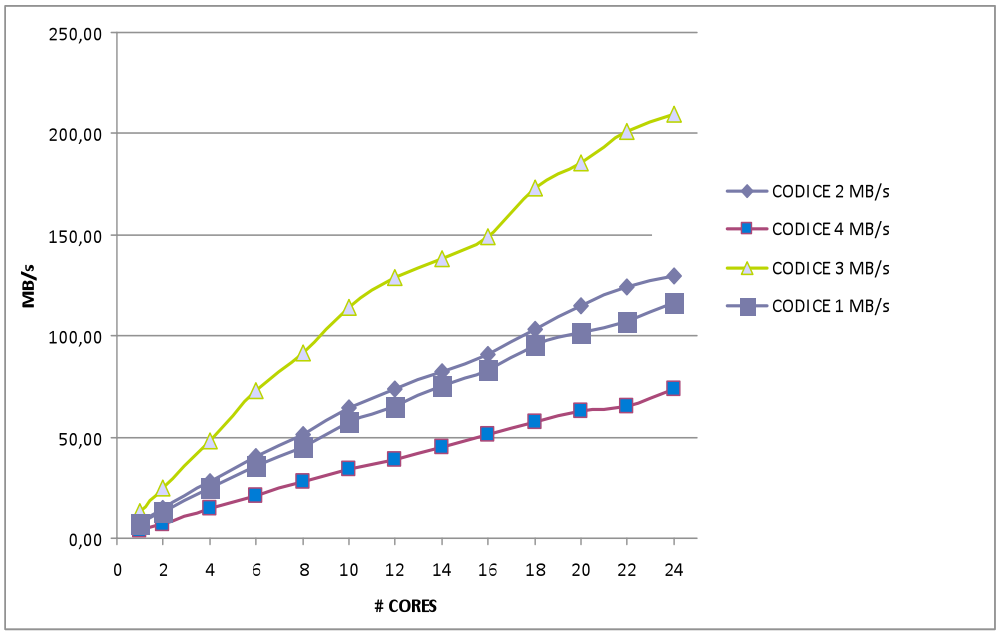
In questo test, la velocità di lettura non è dominata dalle operazioni di calcolo che normalmente vengono fatte all'interno di una macro di analisi ed è perciò ancora maggiormente visibile la capacità di parallelizzazione da parte di PROOF, che riesce a superare il limite di 118 MB/s del singolo switch di rete, raggiungendo una velocità di lettura di oltre 200 MB/s.

E' quindi possibile concludere che, nel caso realistico di macro di analisi mediamente complesse, la configurazione testata non ha nel singolo switch di rete un vero "collo di bottigilia".

Inoltre, la scalabilità con il numero di cores a disposizione è leggermente peggiore rispetto al caso precedente (con 24 cores si ha un fattore 15.8 contro 17.5) , giustificando l'idea che il guadagno di parallelizzare il codice è tanto più marcato quante più operazioni di calcolo vengono effettuate nella macro di analisi. Questo comportamento è ancora più chiaro se si prende in considerazione il test fatto con il codice di tipo 4, che arriva ad un fattore di 19.8 con la configurazione a 24 cores.

6.8.3 Confronto fra i vari tipi di codice

Di particolare interesse è anche il confronto prestazionale tra i 4 diversi tipi di codice sopra descritti.



La velocità di lettura è ovviamente maggiore per il codice di tipo 3 (pochi branches da leggere e pochissimo calcolo da fare), seguito dal codice di tipo 2 (tutti i branches letti

ma calcolo uguale al codice di tipo 3) che ha prestazioni similari al codice di tipo 1 (tutti i branches letti e calcolo tipico di una macro realistica di analisi) e infine il codice di tipo 4 (pochi branches letti e molto calcolo da svolgere).

In aggiunta alle conclusioni dei punti precedenti, si può notare un ulteriore importante aspetto legato alla gestione della memoria da parte di PROOF. La lettura di tutti i branches dell'ntupla, sebbene non vengano usati, rallenta notevolmente la velocità di esecuzione, tanto da portarla praticamente a livello del caso di analisi realistica (codice 1). La differenza dunque tra il codice di tipo 3 e il codice di tipo 1 è quasi esclusivamente dovuta alla lettura dei branches e alla loro allocazione in memoria piuttosto che al loro effettivo utilizzo.

Un suggerimento può dunque essere quello di leggere soltanto i branches effettivamente utilizzati nell'analisi, soprattutto nei casi in cui si abbia a che fare con ntuple con diverse centinaia di branches.

6.9 Conclusioni

Alla luce dei test effettuati possiamo arrivare a evidenziare i seguenti punti da tenere in considerazione per l'uso di PROOF interfacciato ad un sistema di storage gestito da GPFS.

1) L'uso di PROOF Lite risulta scalare bene (entro un 10% dall'idealità) con il numero dei cores fino a circa 5-6 cores. Aumentando ulteriormente il numero dei cores nella versione PROOF "cluster" ci si discosta sempre più dall'idealità, arrivando a un fattore oscillante tra 15.8 e 19.8 per i diversi tipi di codice testati nella versione a 24 cores.
2) Con una macro di analisi di media complessità, la velocità di lettura dallo storage non risulta essere limitata né dalla velocità intrinseca del disco né da quella dell'unico switch di rete presente nella configurazione utilizzata. Inoltre, la parallelizzazione dell'analisi realizzata da PROOF può garantire una velocità di lettura nettamente superiore a quella che garantirebbe lo switch della rete nella versione "single core".
3) La scalabilità di PROOF migliora con la complessità del codice di analisi (e quindi con il consumo di CPU).
4) Dal confronto tra i 4 tipi di codice emerge come gran parte della differenza tra una macro complessa e una iper-semplice sia dovuta alla lettura e all'allocazione in memoria dei vari branches dell'ntupla piuttosto che alla complessità del calcolo richiesto dalla macro. E' pertanto opportuno, soprattutto con ntuple molto grandi, leggere solamente i branches effettivamente utilizzati nel codice di analisi.

7 Referenze

[1] http://indico.cern.ch/conferenceTimeTable.py?confId=77057&showDate=all&showSession=all&detailLevel=contribution&viewMode=parallel#all

[2] https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasTier3

[3] Maggiori informazioni in https://twiki.cern.ch/twiki/bin/view/Atlas/Tier3SoftwareWorkingGroup; il tool di installazione si trova in https://atlas-install.roma1.infn.it/atlas_install.