ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI SCIENZE

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

SECURE LARGE DATA TRANSFER

Tesi in
VIRTUALIZZAZIONE E INTEGRAZIONE DI SISTEMI

Relatore:
Chiar.mo Prof. Ing.
VITTORIO GHINI

Correlatore:
Prof. Ing.
STEFANO DICIOTTI

Presentata da:
MATTEO PAOLUCCI

Sessione IV

Anno Accademico 2024 – 2025

Ai miei genitori,
alla mia famiglia

Indice

Elenco delle figure

Introduzione

Trasferire grandi moli di dati in modo sicuro è ancora oggi uno dei problemi più complessi e cruciali dell'informatica moderna. Non si tratta di una sfida puramente tecnica: dietro ogni trasferimento si celano scoperte scientifiche, decisioni strategiche e ricerche che possono cambiare la nostra comprensione del mondo.

Le sorgenti di questi dati sono molteplici e in continua crescita. Tre ambiti in particolare ne rappresentano le manifestazioni più estreme ed esigenti:

- Fisica delle alte energie;
- Ricerca scientifica distribuita;
- Ambito militare e intelligence.

0.1 Fisica delle alte energie

La fisica delle alte energie riguarda esperimenti come l'LHC (Large Hadron Collider), sviluppato dal CERN a partire dal 2008. In questi esperimenti le singole collisioni producono enormi moli di dati che poi dovranno essere analizzati ed elaborati. Nel caso dell'LHC in quattro punti distinti, ciascuno corrispondente ad un esperimento specifico (ATLAS, CMS, ALICE, LHCb), delle particelle subatomiche vengono fatte scontrare ad altissime velocità [CKW16]. Questi scontri generano fino a 140 terabyte di dati ogni giorno [Bra19].

0.2 Ricerca scientifica distribuita

Con l'avvento dei supercomputer e delle reti di calcolo distribuite, è nata la necessità di gestire il trasferimento di grandi moli di dati. Non si tratta di un'esigenza recente: già negli anni novanta, per esempio, il progetto I-WAY rese necessaria la collaborazione tra diversi supercomputer dislocati su 17 siti nel Nord America. [FK97]. In questo esperimento vennero già identificate 4 principali classi di applicazioni:

- *Desktop supercomputing*: queste applicazioni fornivano agli utenti remoti (programmatori, utenti finali e risorse) capacità grafiche molto avanzate;
- *Strumenti intelligenti*: queste applicazioni prevedevano il collegamento di strumenti avanzati ai supercomputer per l'elaborazione in tempo reale dei dati;

- *Ambienti collaborativi*: un insieme di applicazioni che permetteva di far interagire utenti remoti con simulazioni di supercomputer;
- *Supercomputing distribuito*: queste applicazioni affrontavano la richiesta di risolvere problemi non risolvibili con un singolo supercomputer.

0.3 Ambito militare e intelligence

In ambito militare, il trasferimento di ingenti volumi di dati ha storicamente rappresentato una sfida significativa. Tuttavia, per ragioni di sicurezza nazionale, le informazioni rese disponibili al pubblico risultano inevitabilmente limitate e parziali.

Tra le prime applicazioni riconducibili alla gestione massiva di dati in ambito difensivo si annoverano i sistemi di acquisizione e analisi delle immagini satellitari di ricognizione in formato digitale, sviluppati a partire dalla metà degli anni settanta. Si ritiene che tali sistemi abbiano richiesto capacità computazionali dedicate per l'elaborazione dei dati acquisiti e la produzione di informazioni di intelligence. Il National Reconnaissance Office (NRO), agenzia del Dipartimento della Difesa degli Stati Uniti preposta alla gestione delle tecnologie e delle immagini satellitari di ricognizione, fu responsabile del lancio, nel 1976, del satellite KH-11, il primo a impiegare tecnologia di imaging digitale elettro-ottico [Bat24].

In epoca più recente, nel 2014, è stato realizzato lo Utah Data Center (UDC), struttura gestita dalla National Security Agency (NSA), la cui capacità operativa consentirebbe, secondo le informazioni disponibili, la gestione e l'analisi in tempo reale di flussi informativi di qualsiasi natura e dimensione.

Nel contesto europeo, con particolare riferimento al Regno Unito, il Defence Information Infrastructure (DII) costituisce una rete militare integrata gestita dal Ministero della Difesa britannico. A partire dal 2000, tale infrastruttura ha progressivamente sostituito i sistemi preesistenti, con l'obiettivo di realizzare un'architettura unificata in grado di ottimizzare le comunicazioni e l'efficienza nel trasferimento dei dati. Già nel 2005, il sistema comprendeva circa 150.000 terminali, serviva oltre 300.000 utenti e collegava più di 2.000 siti operativi della difesa [20008].

Questa tesi si propone di analizzare gli strumenti e i protocolli che nel corso dei decenni hanno affrontato queste sfide: dalla nascita del grid computing con ARPANET e i metacomputer, fino ai moderni sistemi di autenticazione federata e containerizzazione. L'obiettivo non è solo descrivere le tecnologie esistenti, ma confrontarle su un'infrastruttura di test reale, per fornire un quadro concreto e replicabile delle loro prestazioni e caratteristiche di sicurezza.

Parte I

Contesto e fondamenti

Capitolo 1

Origini del calcolo distribuito

Il primo passo fatto per connettere calcolatori distanti, per poter condividere risorse, ebbe luogo nel 1969 con il progetto ARPANET, anche se inizialmente questo passo fu compiuto per comunicare più che per poter distribuire lavori di calcolo su insiemi di più calcolatori. Per arrivare al concetto di grid computing si dovrà aspettare fino alla metà degli anni novanta quando venne presentato il progetto I-WAY, anche se allora si parlava di «metacomputers» [FK97]. In poco tempo si iniziarono a sviluppare altre infrastrutture adibite al calcolo distribuito come Teragrid (2001), Datagrid (2001) e EGEE (2006). Verso la fine del decennio il concetto di «cloud computing» iniziò a diventare popolare.

1.1 ARPANET

La prima spinta che generò la rete internet attuale avvenne il 29 agosto 1949 quando l'Unione Sovietica detonò la prima bomba nucleare. Il governo statunitense già nel 1948 iniziò a studiare una rete di stazioni radar per la difesa aerea e nel 1949 istituì l'*Air Defense System Engineering Committee (ADSEC)* coordinato da George Valley, un professore del Massachusetts Institute of Technology (MIT) per migliorare il sistema di radar già progettato.

Inizialmente i dati, provenienti dai radar che controllavano i settori aerei, venivano scritti a mano. Da queste informazioni, i comandanti della difesa aerea tattica determinavano le azioni da intraprendere. Nel 1950 l'ADSEC decise di automatizzare il processo attraverso il calcolatore Whirlwind del MIT. La prima dimostrazione significativa avvenne il 20 aprile 1951 quando un bombardiere venne intercettato da un caccia utilizzando dati radar trasmessi tramite linee telefoniche e analizzati dal Whirlwind.

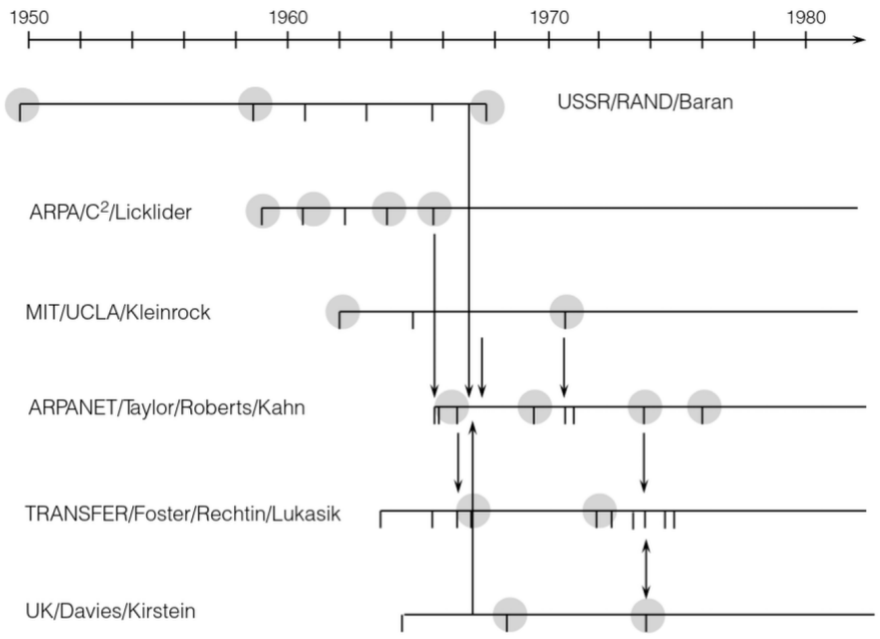

Figura 1.1: Sviluppo cronologico di ARPANET. I punti evidenziano i momenti in cui sono intervenuti individui, organizzazioni e casualità. [Luk11]

Un altro evento saliente della Guerra Fredda fu il lancio del primo satellite russo, lo Sputnik, il 4 ottobre 1957, che spinse gli statunitensi a credere che i russi fossero già in grado di lanciare missili intercontinentali carichi di bombe atomiche. La risposta degli Stati Uniti fu la creazione dell'Advanced Research Projects Agency (ARPA) il 7 febbraio 1958, sotto il comando del Segretario alla Difesa. Lo scopo affidato all'ARPA era quello di «essere responsabile della direzione o dell'esecuzione di progetti avanzati nel campo della ricerca e dello sviluppo».

Nell'ottobre del 1962 venne assunto J.C.R. Licklider (Lick) che in seguito si rivelò una risorsa fondamentale nello sviluppo di ARPA. Di lì a poco Lick espanse il programma di ricerca sul comando e controllo presso l'SDC trasformandolo in un programma di informatica di alta qualità basato su Centri di eccellenza, tra cui il MIT. In quel tempo si fece strada il concetto di «time-sharing», ovvero la possibilità di consentire a più utenti di accedere simultaneamente alle risorse di un medesimo calcolatore tramite terminali dedicati. Nonostante i calcolatori fossero già molto veloci, emerse una criticità rilevante, in termini di velocità del processo, con lo scambio di dati tra calcolatori diversi.

È noto che Lick e alcune altre persone parteciparono alla *Great Intergalactic Network*, che oggi si presume fosse un precursore di ARPANET. Anche l'UCLA (University of California, Los Angeles) condusse un progetto sperimentale per studiare la possibilità di collegare in rete calcolatori separati affinché funzionassero come un processore parallelo virtuale. Inoltre all'UCLA in quegli anni c'erano tre mainframe dell'IBM che in parte vennero impiegati sempre per supportare ARPA.

Tra i tanti contributi di Lick e del suo successore, Sutherland, ci fu quello di concepire le reti di calcolatori come reti general-purpose, mentre a quel tempo esistevano solo reti special-purpose o specifiche per singole

applicazioni. Per Lick, più nello specifico, la «rete» era:

- la rete di hardware, cavi e software;
- la rete di calcolatori, in cui ciascuno di essi eseguiva una funzione specifica per raggiungere un fine più grande;
- la rete di persone, con le loro abilità, esperienze e risorse di informazioni che potevano essere utilizzate per risolvere problemi intellettualmente complessi e non risolvibili dal singolo individuo.

In seguito ai sorvoli degli U-2 sopra l'URSS nel 1959, che rivelarono la presenza di missili balistici intercontinentali sovietici, e alla crisi di Cuba del 1962, il centro di ricerca RAND assunse Paul Baran, il quale formalizzò alcuni fondamenti della comunicazione distribuita. Nel 1964, la RAND pubblicò il quinto report della serie, che sintetizzava undici concetti di routing esaminati tra il 1952 e il 1961.

Un'altra figura che contribuì alla creazione di ARPANET fu Len Kleinrock, uno studente laureato nel dipartimento di Ingegneria Elettrica e Informatica del MIT. Nel 1961 la sua tesi, «Information Flow in Large Communication Nets» (Flusso di informazioni nelle grandi reti di comunicazione), esaminava concetti di scalabilità, valutazione delle prestazioni, controllo, instradamento e aspetti correlati nelle reti distribuite. In seguito lavorò con Baran per assistere ARPA nella progettazione di ARPANET.

Il passo successivo avvenne quando Bob Taylor si unì ad ARPA nel 1965 e, di lì a poco, assunse Larry Roberts che, nel giugno del 1966, stava lavorando a un'implementazione di rete, una connessione a pacchetti, tra un calcolatore TX-2 situato al Laboratorio Lincoln e un Q32/PDP-1 dell'SDC di Santa Monica.

Arrivati a questo punto era necessario ottenere dei finanziamenti per poter realizzare il progetto. Nel maggio del 1968 venne stipulato un contratto per realizzare 20 «image message processors» (IMPs). Si trattava di dispositivi che dovevano essere interposti a dei calcolatori «time-shared», per realizzare una rete di calcolatori [Luk11].

Infine, nel dicembre del 1969 venne realizzata la prima implementazione di ARPANET con nodi situati a UCLA, SRI, UCSB e Utah net, per poi aggiungere altri 16 nodi alla rete e ulteriori 7 tra l'aprile e il luglio del 1970. Dal punto di vista informatico va ricordato che nel dicembre del 1970 venne rilasciato NCP (Network Control Protocol), precursore dello stack TCP/IP. Nel 1972 Kahn effettuò la prima dimostrazione pubblica di ARPANET all'International Computer Communication Conference (ICCC). Da quel momento la rete si espanse rapidamente incorporando centri di ricerca e sviluppo militari, la NASA e altri centri di ricerca.

In quegli anni Kahn riscontrò una serie di limitazioni di NCP:

- non poteva indirizzare reti (e macchine) oltre un primo nodo IMP;
- per l'affidabilità si appoggiava ad ARPANET; in concreto per non perdere pacchetti bisognava gestire il problema a livello di host da entrambe le parti della comunicazione.

Per queste motivazioni Kahn iniziò a progettare quello che poi diventò TCP/IP (Transmission Control Protocol/Internet Protocol), il tutto seguendo quattro regole critiche:

- ogni rete distinta deve essere autonoma e non possono essere richieste modifiche interne a nessuna di tali reti per collegarla a Internet;

- le comunicazioni avvengono con una politica «best effort» (miglior sforzo/massimo impegno). Se un pacchetto non raggiunge la destinazione finale, va ritrasmesso dalla fonte in breve tempo;
- per collegare le reti vengono utilizzate delle scatole nere, che furono in seguito denominate *gateway* e *router*. I *gateway* non conservano alcuna informazione sui singoli flussi di pacchetti che li attraversavano, mantenendoli così semplici ed evitando complicati adattamenti e ripristini da varie modalità di guasto;
- non vi è alcun controllo globale a livello operativo.

TCP/IP venne adottato in ambito militare nel 1980, mentre per gli altri ciò avvenne il 1° gennaio 1983 [LCC+09], quando ARPANET venne divisa in due parti. ARPANET continuò a fare ricerca sul networking, mentre Milnet rimase per gli utenti militari.

La commercializzazione di Internet[1] avvenne a partire dagli anni ottanta. Un momento saliente fu nel settembre del 1988, quando nacque la prima fiera Interop. Cinquanta aziende superarono la selezione e cinquemila ingegneri, provenienti da potenziali organizzazioni clienti, parteciparono per verificare l'effettiva interoperabilità dei sistemi. I fornitori si impegnarono a garantire la piena compatibilità dei propri prodotti con quelli della concorrenza, e la dimostrazione ebbe esito positivo. Nel frattempo, nel 1989, ARPANET fu chiusa definitivamente [Luk11]. Infine Internet raggiunse il grande pubblico il 6 agosto 1991 quando Tim Berners-Lee rese il World Wide Web disponibile al pubblico.

1.2 I Metacomputer e le reti di calcolo distribuito

I primi esemplari di reti di calcolo distribuito si ebbero con:

- il progetto Condor (1988): fu un sistema di schedulazione delle attività mirato a massimizzare l'utilizzo dei calcolatori all'interno di una rete [LLM];
- il progetto Utopia (1993): fu un precursore del moderno IBM Spectrum LSF (Load Sharing Facility)[2].

A partire dagli anni novanta, con l'avvento del World Wide Web, iniziarono a nascere svariati progetti di questo tipo: Charlotte, ParaWeb, Popcorn, e SuperWeb per citarne alcuni.

Il successivo importante impulso al progresso fu dato dalla creazione di reti ad alta velocità, come i banchi di prova gigabit statunitensi. Queste reti resero possibile l'integrazione delle risorse in più siti, un approccio definito «metacomputing» nel 1992 da Catlett e Smarr. Il metacomputer veniva definito come «a network of heterogeneous, computational resources linked by software in such a way that they can be used as easily as a personal computer» (una rete di risorse informatiche eterogenee collegate tramite software in modo tale da poter essere utilizzate con la stessa facilità di un personal computer) [SC92].

L'esperimento I-WAY del 1995, che coinvolse circa 50 gruppi di applicazione nella dimostrazione di applicazioni innovative su reti di ricerca nazionali, stimolò lo sviluppo dell'infrastruttura I-Soft, precursore

[1] Il termine Internet deriva da una risoluzione approvata all'unanimità dalla FNC nel 24 ottobre 1995 [LCC+09].

[2] Un LSF è una piattaforma di gestione del carico di lavoro, un programma di pianificazione delle attività, per il calcolo distribuito ad alte prestazioni (HPC)

sia del Globus Toolkit che del National Technology Grid. Anche il libro The Grid: Blueprint for a New Computing Infrastructure ebbe un effetto catalizzatore [FK22].

Le comunità scientifiche del tempo iniziarono a considerare il Grid Computing come la soluzione ai problemi di federazione delle risorse. In Europa i fisici che progettavano il Large Hadron Collider (LHC) si resero conto che avevano bisogno di un sistema federato di risorse per poter gestire i petabyte di dati che avrebbe prodotto l'esperimento. Da qui nacque il progetto Data Grid, nel 2000, per gestire i dati di LHC: era necessario gestire una rete di calcolatori situati su più siti fisici, in quanto il CERN collabora con fisici, centri di ricerca e università di tutto il mondo [HJMS+00]. Progetti simili negli Stati Uniti furono Particle Physics Data Grid e Grid Physics Network che portarono alla creazione di Open Science Grid, mentre in Europa si formò prima EGEE (Enabling Grids for E-sciencE) e poi EGI (European Grid Infrastructure). A livello internazionale venne creata la Worldwide LHC Computing Grid (WLCG), tuttora esistente, prima denominata LCG che coinvolge 170 centri di computazione distribuiti in 42 paesi diversi.

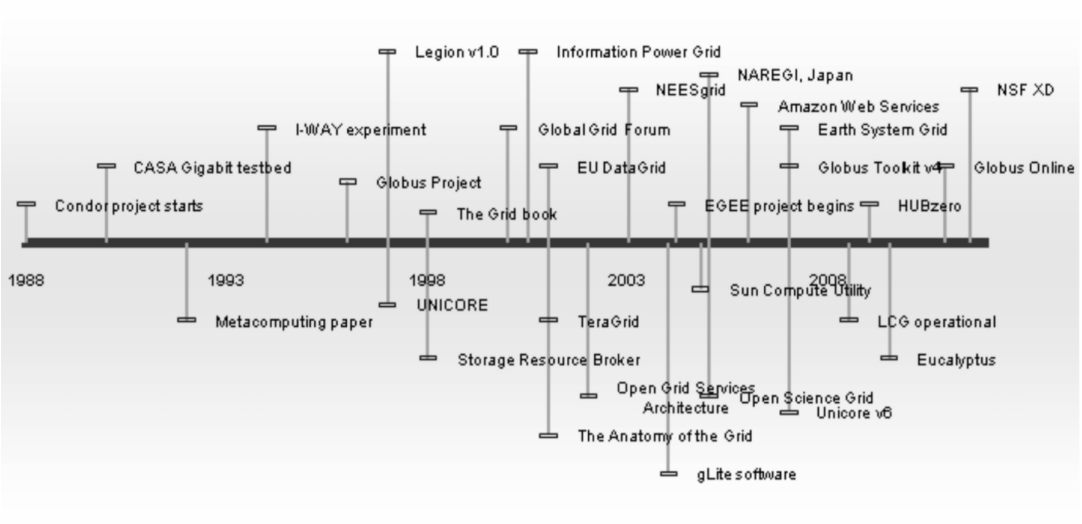

Figura 1.2: Linea temporale dello sviluppo del Grid Computing

Capitolo 2

Strumenti e protocolli per il trasferimento dati su larga scala

Il trasferimento efficiente e sicuro di grandi volumi di dati rappresenta una delle sfide centrali nel contesto del grid computing e delle infrastrutture di calcolo distribuito. La crescente produzione di dati scientifici, in particolare nell'ambito della fisica delle alte energie e degli esperimenti condotti presso il CERN, ha reso necessario lo sviluppo di strumenti e protocolli capaci di gestire flussi di dati dell'ordine dei petabyte, garantendo al contempo affidabilità, sicurezza e prestazioni elevate.

In questo capitolo si analizzeranno tre tecnologie fondamentali per il trasferimento dei dati su larga scala:

- Globus Toolkit: un insieme di moduli che hanno la finalità di localizzare risorse, comunicare, unificare l'interfaccia di accesso alle informazioni, fornire autenticazione, creare processi di elaborazione e accedere ai dati.
- GridFTP: il protocollo che estende FTP (File Transfer Protocol) e permette di sfruttare link di comunicazione superiori a 10 Gbit/s.
- FTS (File Transfer Service): il servizio responsabile della distribuzione dei dati dell'LHC sull'infrastruttura WLCG. Pianifica i trasferimenti, fornisce autenticazione, gestisce le connessioni simultanee, riprova automaticamente i trasferimenti non riusciti e consente agli utenti di monitorare lo stato di avanzamento e l'esito dei propri trasferimenti.

2.1 Globus toolkit

Il Globus toolkit è stato introdotto per la prima volta nel 1997 per supportare elaborazioni su reti di calcolo distribuito, allora chiamate «Metacomputer» (1.2). Fin dall'inizio si componeva di vari moduli che risolvevano specifici problemi:

Servizio	Nome	Descrizione
Gestione risorse	GRAM	Localizza e alloca le risorse
Comunicazione	Nexus	Servizi di comunicazione unicast e multicast
Informazione	MDS	Fornisce accesso distribuito a informazioni sullo stato e la struttura
Sicurezza	GSI	Servizi di autenticazione e autorizzazione
Gestione processi	GEM	Costruisce, localizza e crea una cache dei processi
Controllo	HBM	Monitora la salute e lo stato dei componenti del sistema
Accesso remoto	GASS	Fornisce accesso remoto tramite interfacce sequenziali o parallele

Tabella 2.1: Servizi principali forniti dal Globus Toolkit

- *Localizzazione e allocazione delle risorse*: i meccanismi di localizzazione delle risorse sono necessari perché, in generale, non è possibile aspettarsi che le applicazioni conoscano l'esatta ubicazione delle risorse richieste, in particolare quando il carico e la disponibilità delle risorse possono variare. L'allocazione delle risorse comporta la pianificazione delle risorse e l'esecuzione di qualsiasi inizializzazione necessaria per la successiva creazione di processi, l'accesso ai dati e così via;

- *Comunicazione*: il modulo fornisce meccanismi di comunicazione di base. Tali meccanismi devono consentire l'implementazione efficiente di un'ampia gamma di metodi di comunicazione, tra cui il passaggio di messaggi, la chiamata di procedura remota, la memoria condivisa distribuita, lo streaming e il multicast. Infine, questi meccanismi devono tenere conto di parametri come jitter, affidabilità, latenza e banda disponibile;

- *Servizio informativo unificato sulle risorse*: questo componente fornisce un'interfaccia unificata per ottenere informazioni in tempo reale per quanto riguarda lo stato della griglia e della sua struttura. Permette ai componenti di fornire e ricevere le informazioni precedentemente descritte;

- *Autenticazione e sicurezza*: il modulo è responsabile di fornire meccanismi di autenticazione intercambiabili per validare l'identità degli utenti e delle risorse. Questi meccanismi costituiscono elementi fondamentali per altri servizi di sicurezza, quali l'autorizzazione e la sicurezza dei dati, che richiedono la conoscenza dell'identità delle parti coinvolte in un'operazione;

- *Gestione dei processi*: questo componente inizia l'elaborazione di una risorsa, una volta che quest'ultima è stata localizzata e allocata, impostando tutti i parametri necessari all'esecuzione, quali l'avvio e il passaggio degli argomenti necessari all'eseguibile, l'integrazione di nuovi processi all'interno di una elaborazione già avviata e la gestione della terminazione di un processo;

- *Accesso ai dati:* questo componente è responsabile di fornire accesso remoto ultraveloce ai dati persistenti come i file. Deve anche fornire l'accesso ai database remoti [FK97].

2.2 GridFTP

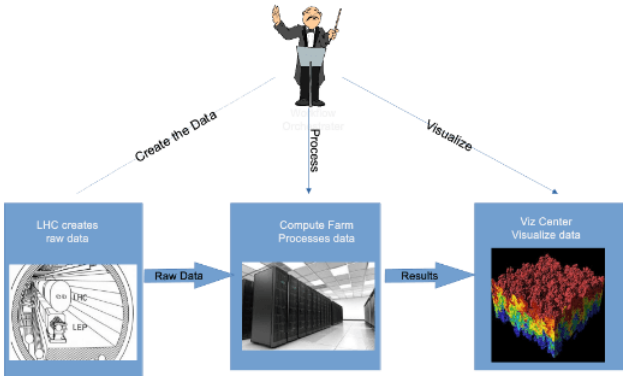

Figura 2.1: Il processo di gestione dei dati all'interno di una *grid*

Sin dall'inizio del grid computing sono stati adottati una serie di protocolli di trasferimento per tipi di storage diversi. Con il tempo, emerse l'esigenza di un protocollo di trasferimento univoco, più veloce che fornisse il supporto ai vari tipi di storage esistenti.

GridFTP è un'estensione del protocollo FTP classico, sviluppata nell'ambito del progetto Globus a partire dai primi anni 2000. È stato per quasi vent'anni il protocollo dominante per il trasferimento di dati scientifici su larga scala, in particolare nell'infrastruttura WLCG.

Le caratteristiche fondamentali che doveva fornire e tuttora fornisce GridFTP sono:

- *Third-party transfer o managed file transfer (MFT)*: fornisce un sistema di orchestrazione nella pipeline del processo di ricerca in modo che non sia necessario al software conoscere effettivamente che dati e che tipi di dato esso stia manipolando. Va notato che il sistema non effettua il trasferimento di per sé, ma fornisce al mittente e al destinatario i dati necessari al trasferimento e alla gestione dei processi;

- *Sicurezza*: il sistema fornisce un trasferimento sicuro e affidabile. Anche se il sistema non necessariamente deve trasferire dati riservati, è comunque importante che enti terzi malevoli o errori di vario genere non pregiudichino il trasferimento dei dati;

- *Trasferimento efficiente su WAN*: un altro problema di FTP è dato dal fatto che basandosi su TCP non funziona molto bene su reti ad alta latenza. Questo problema nasce dall'operatività *AIMD* (*Additive-Increase-Multiplicative-Decrease*) che è un concetto fondamentale di TCP e permette una gestione delle congestioni di rete in modo quasi ottimale. In pratica ogni volta che si perde un pacchetto durante un trasferimento il numero di pacchetti mandati all'interno del trasferimento TCP viene dimezzato, mentre se non ci sono perdite il numero di pacchetti viene incrementato solo linearmente. Ne consegue che le reti ad alta latenza hanno dei problemi, in termini di efficienza di trasferimento, rispetto a reti a bassa latenza. Nello specifico quando c'è una perdita di pacchetto nelle reti ad alta latenza è necessario aspettare un tempo molto più alto affinché il throughput di trasferimento ritorni ottimale.

Da qui nascono i **flussi di trasferimento paralleli**, introdotti da GridFTP. In questo modo anche se avviene la perdita di un pacchetto si ha un effetto molto più contenuto nella velocità di trasferimento. Inoltre, utilizzando più stream di trasferimento, si possono sfruttare in modo più efficiente le architetture multi-core dei processori odierni e dei file system paralleli;

- *Trasferimento parziale*: rispetto a FTP, GridFTP supporta il trasferimento parziale dei file e il recupero del trasferimento in seguito;
- *Negoziazione automatica delle dimensioni del buffer*: può automaticamente stabilire il buffer necessario al trasferimento e ciò semplifica le operazioni dell'utente e migliora la qualità del servizio;
- *Trasferimento a strisce (striped transfer)*: GridFTP permette di trasferire un file partendo da pezzi dello stesso file situati in vari nodi di rete. In questo modo si può ulteriormente migliorare la velocità di trasferimento all'interno di una comunicazione GridFTP [CSH+05].

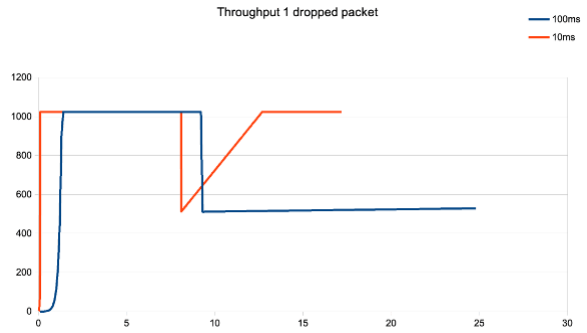

Figura 2.2: L'impatto della latenza in un trasferimento TCP a seguito di perdita di un pacchetto

Negli ultimi anni GridFTP è stato progressivamente sostituito da protocolli più moderni, innanzitutto perché il progetto Globus Toolkit è stato dismesso come software open-source, e la manutenzione del server GridFTP (globus-gridftp-server) si è ridotta. Parallelamente, HTTP/WebDAV e XRootD si sono dimostrati sufficientemente funzionali per i casi d'uso di LHC, con il vantaggio di essere più standard, più facili da integrare con infrastrutture moderne (bilanciatori di carico, CDN, autenticazione basata su *token*) e meglio supportati. Il passaggio da autenticazione X.509 pura verso token OAuth2/OIDC ha ulteriormente favorito protocolli basati su HTTP.

Ad esempio, FTS3, tramite la libreria GFAL2, supporta in modo trasparente tutti questi protocolli, quindi la transizione è stata relativamente semplice nel caso degli esperimenti LHC: è stato sufficiente cambiare gli endpoint di storage da `gsiftp://` a `https://` o `root://` senza modificare la logica di gestione dei dati.

2.3 FTS3

FTS3 (File Transfer Service, versione 3) è il servizio di trasferimento file sviluppato e gestito dal CERN, utilizzato come componente centrale dell'infrastruttura di computing distribuito del WLCG (Worldwide LHC Computing Grid) per gestire gli enormi volumi di dati prodotti dagli esperimenti LHC (ATLAS, CMS, ALICE, LHCb).

LHC produce petabyte di dati all'anno, i quali devono essere distribuiti in modo affidabile da un sito di storage a un altro, attraverso centinaia di centri di calcolo distribuiti in tutto il mondo (organizzati in livelli Tier-0, Tier-1, Tier-2). FTS3 è il motore che orchestra questi trasferimenti: acquisisce le richieste, le accoda, le pianifica, le esegue e ne monitora il completamento.

2.3.1 L'architettura

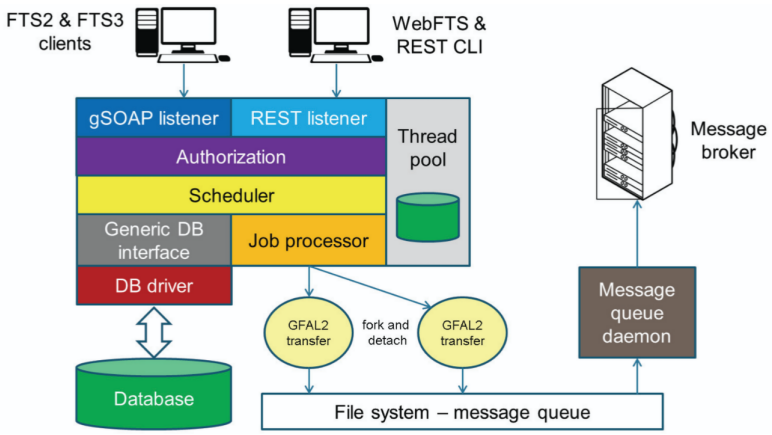

Figura 2.3: Architettura di FTS3

L'obiettivo dell'architettura di FTS3 è quello di fornire scalabilità, modularità, efficienza nel trasferimento, sicurezza e affidabilità.

I principali componenti di FTS3 (mostrati parzialmente nella figura 2.3) sono:

- *CLI client, WebFTS e REST CLI*: le interfacce utente attraverso le quali è possibile interagire con il sistema;
- *database centrale*: si occupa di organizzare i dati, in modo relazionale, dell'intero sistema; memorizza inoltre lo stato di tutte le attività, i trasferimenti individuali, le configurazioni dei canali («link»), le statistiche storiche e i parametri di ottimizzazione;
- *GFAL2*: è una libreria C che permette di creare un livello di astrazione nella gestione dei vari tipi di file system in un sistema distribuito;

- *ottimizzatore*: si occupa di migliorare la comunicazione tra due o più nodi della rete in modo automatico e trasparente; inoltre parallelizza i trasferimenti, evita carichi di lavoro eccessivi sulla rete e gestisce il numero di trasferimenti per canale;
- *sistema di monitoraggio*: permette di controllare i trasferimenti e le risorse all'interno della grid tramite diverse modalità di accesso (API REST, CLI, Web);
- *Scheduler*: è responsabile di organizzare i processi di trasferimento per quanto riguarda l'inizializzazione, la terminazione e la ripresa in caso di trasferimento parziale.

2.3.2 Funzioni principali

Le funzioni principali di FTS3 sono:

- *ottimizzazione adattiva*: rispetto a FTS2 dove andavano definiti vari parametri per il trasferimento, FTS3 impiega un ottimizzatore che si occupa automaticamente di gestire il trasferimento. Nello specifico, FTS3 regola in modo dinamico il numero dei trasferimenti paralleli per ogni coppia mittente-destinatario, osservando vari parametri della connessione tra cui throughput, tasso di errore e tempi di completamento. Inoltre, si passa da una topologia strutturata a una rete completamente interconnessa;
- *configurazione incentrata sulle VO*: le Virtual Organization (VO) sono delle unità logiche che possono avere configurazioni specifiche dedicate come ad esempio priorità, limiti di banda e numero massimo di trasferimenti attivi per link. Questo facilita la suddivisione delle risorse all'interno di una stessa infrastruttura;
- *trasferimento multi-hop*: permette di definire trasferimenti che passano per uno o più nodi intermedi prima di raggiungere il destinatario. In questo modo si possono superare i problemi dovuti all'assenza di un collegamento diretto o si possono sfruttare eventuali file di staging posizionati su nodi intermedi;
- *supporto alla replica multipla*: nel caso in cui un file sia disponibile su più nodi della rete, si può tentare il trasferimento da sorgenti diverse in caso di fallimento oppure si possono effettuare trasferimenti che utilizzano file presenti in siti diversi;
- *interfaccia REST*: la maggior parte delle operazioni fornite da FTS3 sono esposte da un'API RESTful. Questo permette ad esempio di commissionare trasferimenti e ottenere lo stato del sistema tramite sistemi di data management (Rucio[1], PhEDEx[2]) o tramite script di vario tipo che effettuano semplici chiamate HTTP;
- *meccanismo di recupero del trasferimento*: come in Globus e GridFTP, è presente un sistema per ritentare o recuperare i trasferimenti problematici. In caso di fallimento (timeout, errori di rete, storage temporaneamente non disponibile ecc.), FTS3 ritenta in modo automatico il trasferimento. Inoltre, se il protocollo di trasferimento lo supporta, FTS3 può riprendere e terminare un trasferimento parziale senza doverlo reinizializzare completamente;

[1] https://rucio.cern.ch/
[2] https://twiki.cern.ch/twiki/bin/view/CMSPublic/PhEDEx

- *supporto per i database Oracle e MySQL*: il backend di persistenza può essere sia Oracle sia MySQL/MariaDB, permettendo ai siti di scegliere in base alla propria infrastruttura esistente. Il CERN storicamente ha usato Oracle per le istanze centrali, mentre molti siti più piccoli preferiscono MySQL;
- *supporto per protocolli di accesso e trasferimento tramite plug-in GFAL2*: FTS3 non utilizza direttamente i protocolli di trasferimento, ma si appoggia su un sistema di plug-in, GFAL2. In pratica GFAL2 fornisce dei moduli che possono essere inseriti discrezionalmente in base alle necessità tecniche della rete. Inoltre, se viene pubblicato un nuovo protocollo è sufficiente scrivere un nuovo modulo e collegarlo;
- *riutilizzo della sessione*: quando più file devono essere trasferiti tra la stessa coppia mittente-destinatario, FTS3 riutilizza la connessione/sessione già stabilita anziché aprirne una nuova per ogni file. Questo riduce drasticamente l'overhead di handshake TLS/GSI e negoziazione, migliorando il throughput, soprattutto per trasferimenti di molti file piccoli.

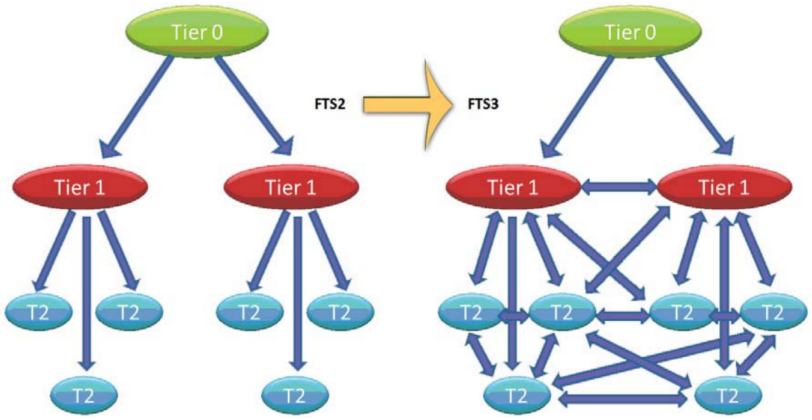

Figura 2.4: Passaggio da una topologia ad albero ad una completamente magliata

Capitolo 3

La sicurezza nei sistemi di grid computing

La sicurezza è un requisito imprescindibile nelle comunicazioni e, di conseguenza, anche nel grid computing, dove risorse distribuite geograficamente in siti diversi e sottoposte a domini amministrativi eterogenei devono cooperare per trasferire ed elaborare ingenti volumi di dati. In alcuni ambiti, come quello della ricerca medica, le esigenze di privacy e sicurezza sono imprescindibili. È necessario autenticare le varie entità coinvolte, garantire la riservatezza dei dati in transito, l'integrità dei trasferimenti e il controllo degli accessi alle risorse condivise.

Con il passare del tempo, queste griglie di calcolo si sono evolute e si sono adeguate a standard sempre più esigenti e rigorosi. Sin dagli anni '90, la comunità scientifica ha sostenuto sforzi per migliorare la sicurezza di questi sistemi. L'evoluzione è iniziata con modelli come la GSI (Grid Security Infrastructure), che ha integrato la cifratura e l'autenticazione nelle comunicazioni attraverso le chiavi pubbliche e i certificati. La GSI ha introdotto il concetto di delega delle credenziali tramite proxy certificate, che consentivano a un utente autenticato di delegare temporaneamente la propria identità a servizi e processi remoti, permettendo l'esecuzione di operazioni a catena senza richiedere l'intervento diretto dell'utente a ogni passaggio. In seguito, questo meccanismo si è rivelato fondamentale per abilitare attività complesse che richiedono il coinvolgimento di molti nodi intermedi, ciascuno appartenente a domini amministrativi diversi.

Per quanto riguarda il controllo degli accessi e l'autorizzazione, nell'ambito del calcolo distribuito, i progetti DataGrid e EGEE (1) hanno introdotto strumenti come il *Virtual Organization Membership Service* (*VOMS*) che ha permesso di associare agli utenti attributi e ruoli all'interno di organizzazioni virtuali (VO), consentendo politiche di autorizzazione più granulari rispetto alla semplice verifica dell'identità.

L'integrazione tra GSI e VOMS è stata un pilastro della sicurezza nel grid computing per oltre un decennio, soprattutto all'interno di WLCG (1). L'unica limitazione riscontrata è la necessità di ogni utente di ottenere, rinnovare e gestire i certificati dalle Certificate Authority (CA). Questo problema ha afflitto soprattutto realtà meno strutturate rispetto all'ambito della ricerca, dotate di meno risorse e disponibilità per la gestione di sistemi complessi,

A partire dal secondo decennio del duemila si è notata una graduale transizione a tecnologie più recenti come ad esempio OAuth 2.0 e OpenID Connect, che hanno tentato di fornire meccanismi basati su token che risultano più semplici e che aggirano alcune limitazioni del modello basato sui certificati [CVCG19].

3.1 Requisiti di sicurezza nel grid computing

Ogni sistema informatico, che può essere esposto a canali di comunicazione difettosi o utenti malevoli, deve fornire almeno le seguenti funzioni per essere sicuro:

- *autenticazione*: consente al destinatario di accertarsi che il messaggio sia effettivamente spedito dal mittente;
- *autorizzazione*: permette a un ente di garantire o revocare permessi di accesso e manipolazione delle risorse all'interno di un dominio;
- *riservatezza*: garantisce che il contenuto di una comunicazione non sia intelligibile se non al mittente e al destinatario;
- *integrità*: consente di verificare che i dati trasmessi non siano stati manipolati durante il transito tra il mittente e il destinatario;
- *non repudiabilità*: assicura che una transazione, di qualsivoglia genere, non possa essere negata da una o entrambe le parti coinvolte.

Fornire queste funzioni è già un problema non banale nei domini centralizzati, dove, ad esempio, non ci sono problemi di fiducia riguardo agli enti che hanno accesso alle risorse del dominio. Nel calcolo distribuito, fornire queste funzioni diventa più difficile perché si incontrano una serie di problemi, tra cui:

- *assenza di un'entità centrale*: non essendoci un'entità centrale a cui fare riferimento, nascono problemi di fiducia reciproca: chi determina quali permessi ha un certo utente che accede alle risorse dall'esterno dell'organizzazione? Come ci si può fidare di un utente esterno all'organizzazione? Chi garantisce la sicurezza dei dati durante il transito su una rete pubblica?
- *mancata centralizzazione dell'autenticazione*: quando si hanno sistemi distribuiti come quelli del grid computing, non è possibile avere un singolo database delle credenziali a cui fare riferimento per gestire le politiche di accesso e manipolazione delle risorse.

Da tali problematiche emergono diverse necessità tra cui:

- fornire un meccanismo di fiducia inter-dominio;
- delegare credenziali per comunicare tra più nodi;
- fornire un'autorizzazione basata su ruoli trasversali alle organizzazioni;
- fornire un meccanismo di protezione per i canali comunicativi non fidati.

3.2 La GSI e il modello dei certificati

Le necessità precedentemente elencate sono state parzialmente soddisfatte dall'introduzione di GSI (Globus Security Infrastructure) che ha introdotto vari meccanismi e tecnologie per migliorare la sicurezza nel calcolo distribuito. Prima di comprendere come GSI abbia risolto parzialmente questi problemi di sicurezza nei sistemi distribuiti, è opportuno illustrare come fornisca le funzioni necessarie per un sistema informatico sicuro:

- *autenticazione*: richiede l'autenticazione reciproca di tutte le parti che partecipano a una sessione. L'autenticazione è garantita da un'infrastruttura a chiave pubblica in cui ogni entità all'interno del sistema possiede un certificato conforme allo standard X.509, considerato attendibile da un'autorità di certificazione (CA). All'inizio di una comunicazione, ha luogo un handshake iniziale con lo scambio dei rispettivi certificati. Al fine di garantire uno scambio sicuro, GSI si affida ai servizi offerti dalle librerie di comunicazione OpenSSL per lo scambio dei certificati;

- *autorizzazione*: viene effettuata tramite l'uso di file di testo denominati grid-mapfiles, contenenti una mappatura tra i nomi dei soggetti dei certificati e gli account locali. Ogni sistema locale nella griglia ha il proprio grid-mapfile e un utente remoto è autorizzato ad accedere alle risorse solo se il grid-mapfile del sistema locale mappa il nome del soggetto a un ID locale;

- *riservatezza*: GSI non implementa direttamente meccanismi di riservatezza, ma si appoggia su TLS (Transport Layer Security). TLS è un protocollo crittografico che può essere aggiunto alla maggior parte dei protocolli di comunicazione per fornire sicurezza aggiuntiva;

- *integrità*: come per la riservatezza, GSI non fornisce direttamente funzioni di controllo dell'integrità, ma delega queste funzioni alle librerie OpenSSL;

- *non repudiabilità*: dal momento che si usa un sistema a chiave pubblica, si impiegano, in primo luogo, meccanismi di firma digitale. In secondo luogo, GSI fornisce un sistema di logging che traccia la maggior parte delle operazioni effettuate dagli utenti.

Una volta illustrate le funzioni fornite parzialmente da GSI [DGG02], si può osservare che alcune delle necessità di sicurezza dei sistemi distribuiti risultano già soddisfatte:

- *meccanismo di fiducia inter-dominio*: dal momento che l'autenticazione è fornita tramite certificati X.509, ne deriva l'adozione del modello di fiducia basato sulle autorità di certificazione (CA);

- *protezione dei canali di comunicazione non sicuri*: per quanto riguarda la sicurezza dei dati durante il trasferimento, come precedentemente illustrato, GSI si appoggia su TLS;

- *delega delle credenziali per la comunicazione tra più nodi*: durante il trasferimento dei dati tra più nodi sarebbe necessario che il mittente autorizzasse ciascun passaggio. Per risolvere questo problema, GSI fornisce i proxy certificate, ovvero certificati temporanei che fungono da delega dell'identità dell'utente;

- *autorizzazione basata sui ruoli*: GSI non soddisfa tale requisito, che è stato affrontato successivamente con l'introduzione del Virtual Organization Membership Service (VOMS), il quale estende il modello GSI. VOMS introduce il concetto di organizzazione virtuale (VO): un gruppo logico di utenti e risorse che collaborano verso un obiettivo comune, indipendentemente dall'istituzione di appartenenza. Un server VOMS rilascia agli utenti attributi da incorporare nei proxy certificate, contenenti informazioni sui ruoli e i gruppi all'interno della VO. In questo modo, ciascun ente che riceve tali certificati può verificare se autorizzare la transazione richiesta.

3.2.1 PKI e i modelli di fiducia

L'infrastruttura a chiave pubblica (Public Key Infrastructure, PKI) nasce dall'esigenza di garantire fiducia reciproca nell'ambito di organizzazioni distribuite. Il principio fondamentale consiste nel designare un ente esterno di cui tutte le parti coinvolte nel sistema possano fidarsi. Tale infrastruttura si compone principalmente di due elementi:

- *autorità di certificazione (CA)*: entità responsabile dell'emissione, della verifica, del rinnovo e della revoca dei certificati digitali. Un certificato include la chiave pubblica, o informazioni relative a essa, e può inoltre offrire una directory in cui memorizzare le chiavi pubbliche;
- *sistema di gestione*: il software utilizzato dalla CA per amministrare il ciclo di vita dei certificati.

Alcune delle funzionalità fornite da una PKI [Spe03] sono:

- *certificazione:* processo che associa un utente a un documento, in questo caso una chiave pubblica;
- *autenticazione:* una volta che un certificato è stato inserito all'interno del sistema, l'utente può accedere alle risorse senza dover effettuare continue richieste di autorizzazione;
- *validazione e scadenza:* processo che consente a ciascuna parte del sistema di verificare se il certificato sia ancora valido tramite la data di scadenza apposta dalla CA;
- *non repudiabilità:* meccanismo mediante il quale è possibile provare chi ha inviato un messaggio; il destinatario può esibire tale prova a una terza parte, la quale può verificare in modo indipendente la fonte;
- *firme digitali:* al momento dell'emissione, la CA genera due coppie distinte di chiavi pubbliche e private per ciascun utente o server. Una coppia viene utilizzata per cifrare e decifrare le informazioni, mentre l'altra viene impiegata dalle applicazioni client per apporre una firma digitale su un documento o una trasmissione;
- *cifratura dei messaggi:* attraverso questa infrastruttura è possibile cifrare qualsiasi tipo di messaggio o documento in modo trasparente per l'utente finale.

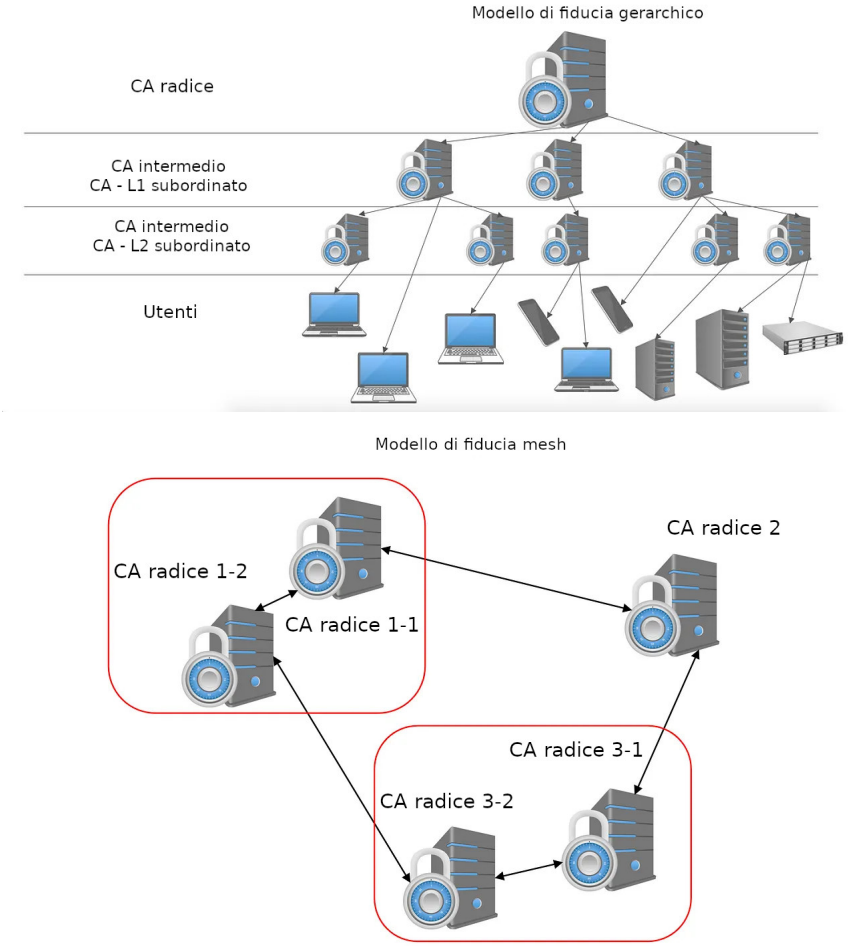

Figura 3.1: Alcuni modelli di fiducia per infrastrutture a chiave pubblica

3.2.2 I certificati X.509

Per comprendere i certificati X.509 utilizzati da GSI, è necessario prima introdurre lo standard X.500. Tale standard si basa su directory che contengono voci rappresentanti entità di vario tipo, quali computer, utenti e organizzazioni. Va precisato che lo standard non impone che a ciascuna voce sia associata una chiave pubblica.

Il metodo di ricerca per i sistemi conformi a X.500 si fonda su una proprietà denominata DN (Distinguished Name, nome distinto), un identificatore utilizzato per individuare le voci all'interno di una directory. Il DN deve essere univoco e organizzato secondo una struttura gerarchica. L'insieme delle voci, caratterizzate dai rispettivi DN, è noto come Directory Information Base (DIB). Il DN, affinché sia univoco, è costituito da vari attributi, tra cui: nome comune (CN), organizzazione (O), unità organizzativa (OU), località (L), stato o provincia (S) e paese (C).

Oltre al DN, ciascun certificato X.509 deve contenere:

- versione;
- numero seriale;
- identificatore dell'algoritmo di firma;
- nome dell'emittente;
- periodo di validità;
- chiave pubblica;
- firma della CA.

3.2.3 Proxy certificate e delega delle credenziali

I certificati proxy sono stati introdotti con GT2 (Globus Toolkit 2) [WSF+], nello specifico tramite un'estensione GSI ai certificati di identità X.509 che consente a un utente di assegnare dinamicamente una nuova identità X.509 a un'entità e quindi delegare alcuni dei propri diritti a tale identità. L'utente crea un certificato proxy emettendo un nuovo certificato X.509 firmato con le proprie credenziali anziché ricorrere a una CA.

Il problema dell'instaurazione di relazioni di fiducia tra entità appartenenti a domini amministrativi diversi viene affrontato da GSI attraverso i certificati proxy e servizi di autorizzazione come il Community Authorization Service (CAS). Il meccanismo si basa su un principio implicito: due entità che possiedono certificati proxy derivati dal certificato dello stesso utente sono considerate reciprocamente fidate, in quanto la loro legittimità è riconducibile a un'unica identità verificabile lungo la catena di fiducia. Questo consente agli utenti di creare dinamicamente domini di fiducia temporanei, delegando i propri certificati proxy ai servizi con cui intendono interagire. In questo modo, la collaborazione tra risorse distribuite su più organizzazioni non richiede accordi di fiducia diretti tra i singoli siti, ma si fonda sulla fiducia condivisa nell'identità dell'utente che ha avviato la delega.

3.2.4 Il ruolo dei VOMS

Una limitazione di GSI risiede nel fatto che le autorizzazioni sulle risorse presentano possibilità limitate, in quanto il meccanismo si basa sui grid-mapfile, tabelle che associano i DN agli utenti locali. Quando il numero di utenti diventa elevato, la gestione delle autorizzazioni può risultare onerosa e complessa.

Per risolvere tale problema è stato introdotto il VOMS (Virtual Organization Membership Service). Il VOMS è un servizio, basato sul concetto di organizzazione virtuale (VO), che mantiene un database per ciascuna VO, all'interno del quale sono registrati i dati di ogni membro dell'organizzazione. Una volta che il servizio VOMS è attivo, vengono utilizzati certificati proxy VOMS contenenti le informazioni specifiche degli utenti. In questo modo, prima di concedere l'accesso a una risorsa, il sistema richiede un ulteriore certificato che consente di autorizzare o negare la transazione in base alle politiche prestabilite.

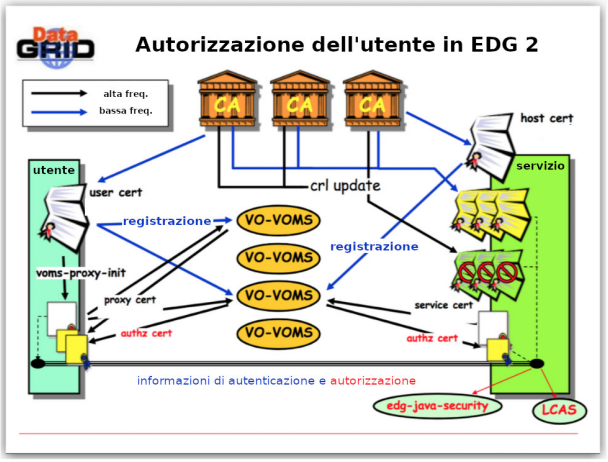

Figura 3.2: Autorizzazione dell'utente in EDG 2

Nonostante sia in corso una transizione verso nuove infrastrutture di autenticazione e autorizzazione (AAI) basate su token, la PKI rimane una componente fondamentale nell'ambito della sicurezza della computazione distribuita, oltre che nelle comunicazioni web in generale. I nuovi sistemi basati su OAuth 2.0 e OpenID Connect continuano infatti a ereditare i medesimi principi, quali gli attributi di ruolo e gruppo associati all'identità dell'utente, originariamente sviluppati per infrastrutture basate su certificati X.509.

3.3 La transizione verso modelli di autenticazione federata

Con lo sviluppo di internet e delle moderne tecnologie informatiche sono emerse delle limitazioni evidenti [CVCG19] nell'utilizzo di AAI basate su certificati X.509:

- *scarsa usabilità*: la gestione dei certificati era un processo macchinoso che frustrava la maggior parte degli utenti scientifici. Per un ricercatore il cui obiettivo è fare fisica o analisi medica, dover ottenere un certificato personale da una CA riconosciuta, installarlo correttamente, rinnovarlo periodicamente e generare proxy con il comando giusto rappresentava un ostacolo significativo;

- *incompatibilità con i browser e portali web*: i proxy certificate VOMS non funzionano nei browser, il che ha costretto a sviluppare workaround complessi per integrare VOMS con i Science Gateway e i framework sperimentali. In un'epoca in cui l'accesso ai servizi si sposta sempre più verso interfacce web, questo era un limite strutturale;

- *difficoltà di integrazione con le federazioni di identità*: il modello di autorizzazione dei servizi grid, essendo strettamente legato ai certificati X.509, rendeva difficile l'integrazione con federazioni di identità come eduGAIN, che è il sistema usato dalle università e dagli enti di ricerca europei per il single sign-on federato;

- *barriera verso i provider esterni*: i certificati X.509 rappresentavano un ostacolo nell'integrazione di risorse di calcolo e storage provenienti da partner esterni come cloud commerciali, cloud ibridi e centri HPC. Questa integrazione sta diventando cruciale per soddisfare le esigenze computazionali di HL-LHC, e il modello a certificati era troppo rigido per un ecosistema che include provider eterogenei che non adottano le convenzioni del grid tradizionale.

Il modello di autenticazione e autorizzazione basato su *token* si fonda su tre tecnologie complementari: OAuth 2.0, OpenID Connect e i JSON Web Token (JWT). OAuth 2.0 è il *framework* standard per l'autorizzazione delegata nei servizi HTTP e costituisce il componente architetturale principale. Esso definisce una serie di flussi autorizzativi attraverso i quali un'applicazione *client*, che può essere un browser, un'interfaccia a riga di comando o un altro servizio, ottiene un token di accesso da un server di autorizzazione centrale e lo presenta alle risorse per ottenere l'accesso.

OpenID Connect estende OAuth 2.0 con un livello di identità, definendo le modalità con cui le informazioni di autenticazione vengono trasmesse ai servizi, consentendo così l'implementazione di funzionalità legate all'identità dell'utente, come l'*accounting* e la tracciabilità. I token stessi sono codificati come JWT, un formato che consente di esprimere in modo sicuro un insieme di *claim*, asserzioni relative all'identità dell'utente, alle sue proprietà di autenticazione, ai suoi attributi e alle sue capacità, firmati digitalmente dal server emittente. Analogamente a quanto avveniva con VOMS, la firma dei JWT consente una verifica distribuita: qualsiasi servizio che possieda la chiave pubblica del server di autorizzazione può verificare autonomamente la validità e l'integrità del token, senza necessità di contattare il server emittente a ogni richiesta.

Nel modello proposto per le infrastrutture grid, il funzionamento è concettualmente simile a quello di VOMS, ma disaccoppiato dal meccanismo di autenticazione sottostante. Un servizio centrale di Identity and Access Management (IAM), con ambito definito per ciascuna VO, gestisce l'autenticazione degli utenti supportando molteplici meccanismi, dalle federazioni di identità accademiche come eduGAIN ai *social login* fino ai certificati X.509, e rilascia *token* di accesso contenenti le informazioni necessarie per l'autorizzazione presso le risorse. I servizi, a loro volta, verificano il *token* ricevuto e prendono decisioni di autorizzazione sulla base del suo contenuto, in modo analogo a come avveniva con gli attributi VOMS estratti dal proxy certificate. La delega delle credenziali, che nel modello GSI era realizzata attraverso la catena di *proxy certificate*, viene implementata tramite il meccanismo di *token exchange*: un servizio che necessita di interagire con un servizio a valle scambia il proprio *token* con uno nuovo, appropriato per la chiamata successiva, interagendo con il server IAM centrale. Una policy definita a livello di organizzazione virtuale determina quali servizi sono autorizzati a effettuare scambi di *token* e quali privilegi possono essere delegati lungo la catena, garantendo un controllo più granulare rispetto alla delega implicita dei *proxy certificate*. Infine, le computazioni di lunga durata sono supportate dai *refresh token*, credenziali a lunga scadenza che consentono di ottenere un nuovo *token* di accesso dal server di autorizzazione quando quello corrente scade, eliminando il problema della scadenza dei *proxy* che nel modello tradizionale poteva causare il fallimento di trasferimenti prolungati.

Parte II

Comparazione degli strumenti

Capitolo 4

L'infrastruttura di test

Questo capitolo descrive l'ambiente di testing utilizzato per la fase sperimentale di questo progetto. L'obiettivo è quello di fornire una panoramica completa dell'infrastruttura utilizzata, sia dal punto di vista hardware che da quello software. In questo modo sarà possibile, almeno in parte, replicare gli esperimenti condotti e confrontare i propri risultati con quelli qui ottenuti.

Per simulare uno scenario di test, per quanto semplice, rappresentativo di uno scenario reale, si è scelto di ricorrere all'utilizzo di tre macchine virtuali remote. Le macchine virtuali sono state fornite attraverso la piattaforma Google Cloud che ha permesso di disporre di risorse computazionali configurabili e scalabili. Un altro vantaggio, fornito dal cloud, è stato l'isolamento adeguato tra gli ambienti di esecuzione.

Per quanto riguarda il software di supporto sono stati utilizzati principalmente quattro programmi principali: *SSH*, *podman*, *git* e *tmux*. *SSH* ha consentito l'amministrazione remota e sicura delle istanze in formato *headless* (senza ambiente desktop). Podman è un motore di containerizzazione adeguato allo standard *OCI* (*Open Container Initiative*) che, a differenza del più conosciuto docker, si basa su un demone (un processo eseguito in background) che non necessita di privilegi di amministrazione.

Tmux ha permesso di effettuare sessioni di lavoro parallele per ogni singola macchina. Infine, *git* è stato utilizzato principalmente per sviluppare il container del client e del server *GCT*.

Nelle sezioni seguenti verranno descritti nel dettaglio le specifiche hardware delle macchine virtuali, la configurazione di rete adottata e gli strumenti software impiegati per il deployment e l'esecuzione dei test.

4.1 Podman

Podman, come Docker, è un motore software che supporta la containerizzazione [Arr22]. Un container è un ambiente isolato e relativamente leggero che impacchetta un'applicazione assieme a tutte le sue dipendenze (librerie, configurazioni, eseguibili). A differenza delle macchine virtuali, i container condividono il kernel del sistema operativo host (userspace), risultando molto più efficienti in termini di risorse e tempi di avvio. L'unica richiesta è appunto che il kernel del sistema operativo che andrà ad eseguire i container sia strutturato in modo apposito.

Le differenze principali tra containerizzazione e virtualizzazione sono:

- *necessità di risorse*: non dovendo incorporare l'intero kernel del sistema operativo e i relativi moduli per farlo funzionare, un container è molto più leggero rispetto ad una macchina virtuale. Inoltre i container fornendo servizi più specifici e mirati consumano meno memoria centrale per funzionare;
- *disponibilità delle immagini*: c'è da tenere in considerazione che la disponibilità delle immagini dei container è molto più alta rispetto a quella delle macchine virtuali in quanto sia Podman che Docker si appoggiano su dei repository (contenitori adibiti al software), chiamati image registries (registri delle immagini), che forniscono immagini di applicazioni in modo affidabile e veloce. Questi registri possono essere sia pubblici che privati e contengono sia le immagini delle aziende che sviluppano le applicazioni che le immagini personalizzate dagli utenti;
- *personalizzazione*: un altro punto di vantaggio dei container sta nel fatto che si possono creare dei container personalizzati a partire da container di base forniti dalle aziende che producono le applicazioni o i sistemi operativi. In questo modo si possono creare applicazioni ad hoc che non necessitano di lunghi processi di installazione e configurazione;
- *velocità di istanziazione*: i servizi che si appoggiano su container sono molto più veloci a reagire a un mutamento delle richieste esterne e delle risorse fornite in quanto un container è più leggero in termini di dimensioni e inoltre non deve inizializzare un intero sistema operativo per funzionare.

4.1.1 Riproducibilità delle build

Durante lo sviluppo di un servizio informatico è fondamentale poter ricreare ambienti di sviluppo identici su macchine diverse. La riproducibilità garantisce che il processo di costruzione di un container produca sempre lo stesso risultato, indipendentemente dal sistema su cui viene eseguito. Questo permette, in caso di errori durante la build, di escludere problematiche legate all'ambiente di esecuzione, come la presenza di dipendenze software non necessarie al servizio o incompatibilità dovute a versioni differenti di librerie specifiche.

4.1.2 I microservizi

Le architetture basate su microservizi suddividono le applicazioni in più servizi che svolgono funzioni dettagliate e fanno parte dell'applicazione nel suo complesso. Le applicazioni tradizionali hanno un approccio monolitico in cui tutte le funzioni fanno parte della stessa istanza. Lo scopo dei microservizi è quello di suddividere il monolite in parti più piccole che interagiscono in modo indipendente. Le applicazioni monolitiche si adattano bene ai container, ma le applicazioni basate su microservizi sono la soluzione ideale per questi ultimi.

Avere un contenitore per ogni singolo microservizio aiuta a ottenere importanti vantaggi, come ad esempio:

- scalabilità indipendente dei microservizi;

- responsabilità più definite per i team di sviluppo;
- potenziale adozione di diversi stack tecnologici nei diversi microservizi;
- più controllo sugli aspetti relativi alla sicurezza (come ad esempio servizi esposti al pubblico e servizi privati).

4.1.3 Le principali differenze con Docker

La differenza principale tra i due strumenti risiede nell'architettura: Podman adotta un approccio privo di demone, mentre Docker si basa su un demone centrale. In termini di sicurezza, tale distinzione è rilevante poiché il demone di Docker opera con privilegi di amministrazione, esponendo il sistema ad attacchi di privilege escalation che potrebbero compromettere l'intero host. In Podman, invece, il problema non sussiste in quanto per ogni container viene generato un processo utente con permessi limitati, riducendo così la superficie di attacco anche in caso di compromissione. Va inoltre osservato che i container Docker sono processi figli del demone principale: qualora quest'ultimo termini in modo anomalo, tutti i container associati cessano di conseguenza la propria esecuzione. Infine, va ricordato che il team di Docker ha sviluppato una versione rootless che, analogamente a Podman, non necessita né di un demone centrale né di privilegi di amministrazione per funzionare.

Un'ulteriore differenza riguarda il concetto di pod, integrato nativamente in Podman. Un pod è un insieme di container che condividono risorse di rete e di storage. La gestione dei pod facilita l'integrazione delle applicazioni all'interno di sistemi come Kubernetes, un orchestratore di container in grado di distribuire applicazioni su interi cluster di calcolatori.

Docker, dal canto suo, dispone di un sistema di orchestrazione dei container più essenziale, denominato Docker Compose, che consente di gestire più container all'interno di una singola macchina. Podman ha introdotto un sistema analogo che, tuttavia, non supporta pienamente le funzionalità offerte dall'orchestratore nativo di Docker.

4.2 Le macchine virtuali

Le macchine virtuali impiegate sono tre e sono quasi identiche se non per alcune differenze marginali. Le caratteristiche hardware delle macchine sono:

- Tipo di macchina virtuale: N2
- architettura del processore: x86/64;
- numero di processori: 32 vCPU;
- piattaforma del processore: Intel Broadwell;
- memoria centrale (RAM): 32 GB;
- dimensione della memoria di archiviazione: 6 TB;

- tipo di memoria di archiviazione: SSD[1];
- distribuzione linux: Rocky Linux 10.1 (Red Quartz);
- versione kernel: 6.12.0-124.31.1+2.1.el10_1_ciq.x86_64.

La principale differenza è il posizionamento fisico delle macchine virtuali:

- la prima macchina è situata a Parigi (Francia);
- la seconda macchina è situata a Council Bluffs (Iowa, Stati Uniti);
- l'ultima è situata a Eemshaven (Paesi Bassi).

La scelta di dislocare le macchine in regioni a distanza crescente consente di disporre di tratte con valori di latenza significativamente diversi, condizione necessaria per osservare l'effetto del meccanismo *AIMD* del protocollo TCP sulle prestazioni dei vari strumenti di trasferimento. Come illustrato nel capitolo 2 (2.2), nelle reti ad alta latenza il lento recupero della finestra di congestione a seguito di una perdita di pacchetto penalizza in modo sostanziale il throughput delle connessioni TCP singole: confrontare una tratta intra-europea con una intercontinentale permette di quantificare concretamente tale degrado.

4.2.1 Programmi di supporto

I principali programmi di supporto utilizzati sono:

- *podman*: motore di containerizzazione (4.1);
- *podman-compose*: orchestratore di container Podman (4.1.3);
- *tmux* (*terminal multiplexer*): programma che permette di gestire più sessioni di terminale all'interno di una singola finestra. Può dividere lo schermo in più pannelli e finestre, ognuno con una shell indipendente. Permette di creare Sessioni persistenti, cioè sessioni che continuano a funzionare in background anche se ci si disconnette (può risultare utile in contesti di accesso server a remoti via SSH). Infine, permette di creare più sessioni di lavoro a cui ci si può collegare e scollegare a piacimento, in base alla necessità;
- *git*: sistema di controllo delle versioni distribuito. Permette di tracciare le modifiche al codice sorgente nel tempo, facilitando la collaborazione tra sviluppatori. Ogni copia di un repository Git contiene l'intera cronologia del progetto, consentendo di lavorare offline. Le funzionalità chiave includono branching e merging, che permettono di sviluppare funzionalità in parallelo e integrarle successivamente.

[1] https://docs.cloud.google.com/compute/docs/disks/performance

4.3 Considerazioni sull'infrastruttura di rete

Le tre macchine virtuali impiegate comunicano tra loro all'interno di una rete *VPC* (*Virtual Private Cloud*). Una *VPC* è una rete virtuale logicamente isolata, ospitata all'interno dell'infrastruttura del provider cloud, che consente alle risorse di comunicare tra loro come se fossero connesse alla stessa rete locale, indipendentemente dalla loro collocazione geografica. Tra i vantaggi principali di questo approccio vi è la possibilità di definire regole di firewall centralizzate e di gestire il traffico tra le istanze senza dover esporre indirizzi IP pubblici, semplificando la configurazione e riducendo la superficie di attacco.

Le prestazioni di rete delle istanze Google Cloud dipendono da molteplici fattori [2] che è opportuno esplicitare affinché i risultati dei benchmark presentati nei capitoli successivi possano essere interpretati correttamente. Le macchine virtuali impiegate appartengono alla serie N2, per la quale è stata abilitata l'opzione denominata *Tier_1 networking performance*. Tale configurazione innalza il limite massimo teorico di banda in uscita per istanza fino a 100 Gbps, rispetto ai 32 Gb/s della configurazione standard prevista per la serie N2.

È importante precisare che questi valori rappresentano un tetto massimo raggiungibile e non una garanzia di throughput effettivo. La larghezza di banda per istanza è proporzionale al numero di vCPU allocate, con un rapporto indicativo di circa 2 Gb/s per vCPU. Con le 32 vCPU di ciascuna macchina virtuale utilizzata, il limite teorico di banda in uscita si attesta intorno ai 64 Gb/s, un valore che rientra ampiamente nella capacità massima offerta dalla configurazione *Tier_1*.

Tuttavia, il fattore più rilevante per il presente lavoro non è tanto il limite per istanza, quanto il modo in cui Google Cloud gestisce il traffico tra istanze situate in regioni geografiche differenti. La documentazione ufficiale della piattaforma distingue due categorie di traffico in uscita, ciascuna soggetta a limiti distinti:

Il traffico instradato all'interno di una rete *VPC* tra regioni diverse è soggetto a una quota di banda inter-regionale, denominata *Inter-region network egress bandwidth*, definita a livello di progetto e di coppia di regioni. Le tre macchine virtuali sono distribuite nelle regioni *europe-west9* (Parigi), *europe-west4* (Eemshaven) e *us-central1* (Council Bluffs): questo significa che i trasferimenti tra le due macchine europee e quelli intercontinentali verso gli Stati Uniti sono soggetti a quote potenzialmente differenti. È ragionevole attendersi che il traffico intra-europeo benefici di una latenza inferiore e, in condizioni favorevoli, di una banda effettiva più elevata rispetto a quello transatlantico.

Il traffico instradato verso destinazioni esterne alla *VPC* è soggetto a limiti ancora più stringenti. Per le istanze con *Tier_1* abilitato, la banda massima in uscita verso destinazioni esterne è limitata a 25 Gb/s per istanza, indipendentemente dal numero di vCPU. A ciò si aggiunge un limite per singolo flusso di connessione di 3 Gb/s per ogni quintupla univoca (indirizzo IP sorgente, porta sorgente, indirizzo IP destinazione, porta destinazione, protocollo): per avvicinarsi al limite complessivo è quindi necessario utilizzare connessioni multiple e parallele, una caratteristica che, come si vedrà, è supportata nativamente da alcuni degli strumenti analizzati.

Ulteriori fattori che possono ridurre la banda effettiva rispetto ai valori nominali includono: il tipo di driver di rete virtuale impiegato (*VirtIO* rispetto a *gVNIC*), la dimensione dei pacchetti trasmessi e il relativo

[2] https://docs.cloud.google.com/compute/docs/network-bandwidth

overhead protocollare, le impostazioni del sistema operativo guest relative all'*offloading di checksum* e alla segmentazione TCP (*TSO*), nonché le condizioni di congestione della rete lungo il percorso.

Queste considerazioni sono particolarmente rilevanti ai fini dell'analisi sperimentale. Le prestazioni misurate durante i trasferimenti riflettono non soltanto le capacità degli strumenti software analizzati, ma anche i vincoli intrinseci dell'infrastruttura di rete sottostante. In particolare, il confronto tra trasferimenti intra-europei (Parigi - Eemshaven) e trasferimenti intercontinentali (Francia - Stati Uniti) consente di valutare come ciascuno strumento reagisca a condizioni di rete significativamente diverse in termini di latenza e banda disponibile.

4.4 Strumenti utili al trasferimento dati

Per la figura (4.1) ci sono alcune aggiunte da fare:

- Acronimi presenti nella tabella:
  - OS: Open source
  - TP: Trasferimento Parallelo
  - RdT: Recupero del Trasferimento
- Note aggiuntive:
  - [1]alla data corrente
  - [2]alcune parti sono open source
  - [3] ci sono delle versioni a pagamento
  - [4] il supporto è parziale
  - [5] riceve gli aggiornamenti di sicurezza di ssh

	OS	Gratis	Livello sicurezza	Intefacce	Quantità di dati gestiti	TP	RdT	Self hosted	Ancora supportato[1]
Aspera	✗	✗	Alta	Web, CLI, APIs	TBs-EBs	✓	✓	✗	✓
bbcp	✓	✓	Bassa	CLI	GBs-PBs	✓	✓	✓	✗
fdt	✓	✓	Moderata	CLI	GBs-PBs	✓	✓	✓	✓
FTS	✓	✓	Alta	Web, CLI, APIs	TBs-EBs	✓	✓	✓	✓
gct	✓	✓	Moderata	CLI	GBs-PBs	✓	✓	✓	✓
Globus	✗[2]	✓[3]	Alta	Web, CLI, APIs	GBs-PBs	✓	✓	✗	✓
GridFtp	✓	✓	Moderata	CLI	TBs-EBs	✓	✓	✓	✓
MDTMftp	✓	✓	Bassa	Web, CLI	GBs-PBs	✓	✓	✓	✗
rclone	✓	✓	Alta	Web, CLI, API	MBs-TBs	✓	✓	✓	✓
rsync	✓	✓	Moderata	CLI	KBs-GBs	✗	✓[4]	✓	✓
scp	✓	✓	Moderata	CLI	KBs-GBs	✗	✗	✓	✗[5]
sftp	✓	✓	Moderata	CLI	KBs-GBs	✗	✓	✓	✓

Figura 4.1: Tabella comparativa degli strumenti

Capitolo 5

Risultati sperimentali

In questo capitolo vengono presentati e analizzati i risultati degli esperimenti condotti al fine di valutare le prestazioni di diversi strumenti di trasferimento dati in scenari di rete caratterizzati da elevata latenza. In particolare, sono stati messi a confronto cinque strumenti (4.4) ampiamente utilizzati, SCP, SFTP, rsync, rclone e GCT (Globus Community Toolkit), con l'obiettivo di evidenziare le differenze in termini di throughput e tempi di trasferimento al variare delle condizioni di rete.

Come noto, il protocollo TCP adotta il meccanismo di controllo della congestione AIMD (Additive Increase Multiplicative Decrease) (2.2), il quale riduce drasticamente la finestra di congestione in presenza di perdita di pacchetti e la incrementa in modo conservativo in assenza di congestione. Questo comportamento, efficace nelle reti tradizionali a bassa latenza, si rivela penalizzante nei contesti in cui il prodotto banda-ritardo (bandwidth-delay product) è elevato: il lento recupero della finestra di trasmissione impedisce di sfruttare appieno la capacità del collegamento. Gli strumenti che si affidano direttamente a una singola connessione TCP, come SCP, SFTP e rsync, ereditano integralmente questa limitazione.

GCT, sfruttando il protocollo GridFTP, supera questo vincolo grazie all'impiego di trasferimenti paralleli su più flussi TCP simultanei e, opzionalmente, di dimensioni ottimizzate dei buffer. Ciò consente di saturare il canale trasmissivo anche in presenza di latenze significative. I risultati sperimentali, descritti in dettaglio nelle sezioni successive, confermano questa analisi teorica: le prestazioni di GCT si mantengono elevate indipendentemente dalla latenza, mentre il degrado osservato per SCP, SFTP, rsync e rclone risulta tanto più marcato quanto maggiore è il ritardo di propagazione della rete.

5.1 Metodologia

Per effettuare gli esperimenti si è scelto di collocare tre macchine virtuali in tre regioni geografiche distinte (4.2), al fine di confrontare la velocità di trasferimento in funzione della latenza media. Per ridurre possibili errori dovuti a variazioni delle condizioni del sistema, i trasferimenti sono stati eseguiti in una sola direzione, affinché le caratteristiche dei canali di trasmissione risultassero il più uniformi possibile. In particolare, sono stati testati trasferimenti dall'Olanda alla Francia e dalla Francia agli Stati Uniti.

Sono stati considerati trasferimenti di file da 5 TB. Nei casi in cui il tempo di trasferimento risultava eccessivamente elevato, sono stati effettuati trasferimenti da 100 GB e successivamente stimati i tempi necessari al trasferimento di 5 TB.

Come indicato nell'introduzione del capitolo, gli strumenti scelti per gli esperimenti sono cinque: SCP, SFTP, rclone, rsync e GCT.

5.2 Scp

SCP (Secure Copy Protocol) è uno strumento di trasferimento file basato sul protocollo SSH. Consente di copiare file e directory tra un host locale e uno remoto, o tra due host remoti, garantendo la cifratura dei dati in transito tramite il canale sicuro offerto da SSH. SCP utilizza una singola connessione TCP per il trasferimento e non prevede meccanismi nativi di compressione, sincronizzazione incrementale o parallelismo. Tuttavia, proprio la dipendenza da una singola connessione TCP lo rende vulnerabile alle limitazioni imposte dal meccanismo AIMD in contesti di rete ad alta latenza.

5.3 SFTP

SFTP (SSH File Transfer Protocol) è un protocollo di trasferimento file che opera interamente all'interno di una sessione SSH, garantendo cifratura e autenticazione dei dati in transito. A differenza di SCP, SFTP offre un set più ampio di operazioni sul filesystem remoto, tra cui la navigazione delle directory, la rimozione e la rinomina dei file, e il ripristino di trasferimenti interrotti. Nonostante queste funzionalità aggiuntive, SFTP condivide con SCP la medesima architettura di trasporto, basandosi su una singola connessione TCP. Di conseguenza, anche SFTP risulta soggetto alle limitazioni prestazionali imposte dal meccanismo AIMD in scenari di rete caratterizzati da elevata latenza, come confermato dai risultati sperimentali presentati nelle sezioni successive.

5.4 Rsync

rsync è uno strumento utilizzato per la sincronizzazione e il trasferimento efficiente di file tra sistemi locali e remoti. La sua caratteristica principale è l'algoritmo di trasferimento incrementale, che analizza le differenze tra i file sorgente e quelli di destinazione e trasmette soltanto i blocchi modificati, riducendo significativamente la quantità di dati trasferiti nelle operazioni successive alla prima copia. rsync supporta la compressione dei dati durante il trasferimento e può operare attraverso un canale SSH per garantire la cifratura. Nonostante queste ottimizzazioni a livello applicativo, rsync si appoggia anch'esso a una singola connessione TCP, ereditando le stesse limitazioni prestazionali di SCP nelle reti ad alta latenza. Va precisato che solitamente rsync viene utilizzato per sincronizzare cartelle remote o per generare delle cartelle di backup.

5.5 Rclone

Rclone è un programma a riga di comando per gestire i file su archivi cloud. È un'alternativa ricca di funzionalità alle interfacce di archiviazione web dei fornitori di servizi cloud. Oltre 70 prodotti di archiviazione cloud supportano rclone, inclusi archivi di oggetti S3, servizi di archiviazione file aziendali e consumer, nonché protocolli di trasferimento standard. Rclone offre funzionalità avanzate come la sincronizzazione incrementale, la cifratura lato client e il trasferimento multi-thread verso i servizi cloud. Tuttavia, quando opera su protocollo SFTP, come in questo caso, il suo comportamento è sostanzialmente analogo a quello di SCP e rsync, in quanto il trasferimento avviene attraverso il protocollo TCP soggetto alle medesime limitazioni di AIMD in presenza di elevata latenza.

5.6 GCT

GCT (Globus Community Toolkit) è un fork del più celebre Globus Toolkit (2.1). In sintesi è un insieme di strumenti open source sviluppato per il trasferimento affidabile e ad alte prestazioni di grandi volumi di dati in ambienti distribuiti, come le infrastrutture di grid e cloud computing. Il componente centrale per il trasferimento dati è GridFTP (2.2). Grazie al parallelismo, GridFTP è in grado di mitigare efficacemente le limitazioni imposte dal meccanismo AIMD su singola connessione, distribuendo il carico su più flussi e consentendo di sfruttare l'intera banda disponibile anche in condizioni di elevata latenza.

5.7 I dati

Nelle tabelle seguenti i tempi di trasferimento sono espressi nel formato `GG:HH:MM:SS`, dove `GG` sta per giorni, `HH` sta per ore, `MM` sta per minuti e `SS` sta per secondi. Le colonne $\overline{v}$ e $\overline{RTT}$ indicano rispettivamente la velocità media di trasferimento e il *Round Trip Time* medio, ossia il tempo medio impiegato da un singolo pacchetto per compiere il percorso di andata e ritorno tra sorgente e destinazione.

Come è stato già fatto notare precedentemente, i tempi di trasferimento sono stati stimati per i file da 5TB, nel caso di SCP, SFTP, rsync e rclone.

	Strumento	Tempo di trasferimento	$\overline{v}$	$\overline{RTT}$
Paesi Bassi → Francia	SCP	00:00:11:05	150,37 MB/s	10,61 ms
	SFTP	00:00:09:56	167,79 MB/s	10,62 ms
	Rsync	00:00:07:00	238,09 MB/s	10,63 ms
	Rclone	00:00:14:31	114,81 MB/s	10,61 ms
	GCT	00:00:01:52	894,92 MB/s	10,56 ms
Francia → Stati Uniti	SCP	00:01:28:50	18,76 MB/s	98,08 ms
	SFTP	00:01:28:50	18,76 MB/s	99,10 ms
	Rsync	00:01:05:30	25,45 MB/s	99,01 ms
	Rclone	00:01:30:50	18,35 MB/s	99,09 ms
	GCT	00:00:15:24	108,25 MB/s	100,32 ms

Tabella 5.1: Tabella riassuntiva dei trasferimenti a 100 GB

	Strumento	Tempo di trasferimento	$\overline{v}$	$\overline{RTT}$
Paesi Bassi → Francia	SCP	00:09:14:10	150,37 MB/s	10,61 ms
	SFTP	00:08:16:39	167,79 MB/s	10,62 ms
	Rsync	00:05:50:00	238,09 MB/s	10,63 ms
	Rclone	00:12:06:00	114,81 MB/s	10,61 ms
	GCT	00:01:50:00	757,57 MB/s	10,66 ms
Francia → Stati Uniti	SCP	03:02:00:05	18,76 MB/s	98,08 ms
	SFTP	03:02:00:05	18,76 MB/s	99,10 ms
	Rsync	02:06:34:24	25,45 MB/s	99,01 ms
	Rclone	03:03:41:20	18,35 MB/s	99,09 ms
	GCT	00:02:40:00	545,26 MB/s	100,98 ms

Tabella 5.2: Tabella riassuntiva dei trasferimenti a 5 TB

5.8 Il confronto

Nel collegamento a bassa latenza (Paesi Bassi → Francia, RTT ≈ 10,6 ms), le differenze tra gli strumenti basati su singola connessione TCP sono contenute: rsync raggiunge il throughput più elevato tra questi con 238,09 MB/s, seguito da SFTP (167,79 MB/s), SCP (150,37 MB/s) e rclone (114,81 MB/s). GCT si colloca tuttavia su un ordine di grandezza superiore, raggiungendo 894,92 MB/s; circa 3,7 volte la velocità di rsync e circa 6 volte quella di SCP.

È nel collegamento ad alta latenza (Francia → Stati Uniti, RTT ≈ 99 ms) che il divario diventa drastico. I quattro strumenti a singola connessione TCP convergono tutti verso throughput compresi tra 18 e 25 MB/s: SCP e SFTP si attestano entrambi intorno a 18,76 MB/s, rclone scende a 18,35 MB/s, e rsync si distingue leggermente con 25,45 MB/s. Questo appiattimento delle prestazioni è coerente con il modello teorico: quando il prodotto banda-ritardo è elevato, il meccanismo AIMD diventa il collo di bottiglia dominante e le differenze implementative tra gli strumenti perdono rilevanza. GCT, al contrario, mantiene un throughput di 108,25 MB/s; oltre 4 volte superiore al migliore degli altri strumenti.

Il degrado percentuale conferma ulteriormente questa analisi. Passando dalla tratta a bassa latenza a quella ad alta latenza, SCP perde l'87,5% del throughput, SFTP l'88,8%, rsync l'89,3% e rclone l'84,0%. GCT subisce anch'esso una riduzione significativa (87,9%), ma partendo da una base molto più elevata, il throughput risultante resta ampiamente superiore a quello di tutti gli altri strumenti.

Le implicazioni operative di queste differenze emergono con chiarezza se si considerano i tempi stimati per il trasferimento di 5 TB sulla tratta ad alta latenza: SCP e SFTP richiederebbero oltre 3 giorni, rclone più di 3 giorni e 3 ore, rsync circa 2 giorni e 6 ore, mentre GCT completa il trasferimento in sole 2 ore e 40 minuti. In uno scenario reale di gestione di dati scientifici su scala di petabyte, questa differenza può tradursi in settimane o mesi di ritardo operativo.

In sintesi, i risultati confermano che per trasferimenti su reti ad alta latenza, l'utilizzo di strumenti basati su singola connessione TCP è fortemente penalizzante, indipendentemente dalle ottimizzazioni a livello applicativo. Il parallelismo offerto da GridFTP attraverso GCT si dimostra la soluzione più efficace per sfruttare la capacità disponibile del collegamento.

Conclusioni

Il grid computing è nato dall'esigenza di federare risorse di calcolo e archiviazione distribuite geograficamente, per rispondere a sfide scientifiche, tecniche e operative che un singolo sistema centralizzato non sarebbe stato in grado di affrontare. Dalla prima implementazione di ARPANET nel 1969, passando per i metacomputer degli anni novanta, fino alle moderne infrastrutture come WLCG, il percorso evolutivo del calcolo distribuito è stato caratterizzato da una costante tensione tra prestazioni, scalabilità e sicurezza.

In questa tesi si è ricostruito tale percorso, analizzando le origini storiche del calcolo distribuito e gli strumenti fondamentali che ne hanno accompagnato lo sviluppo, dal Globus Toolkit e il protocollo GridFTP fino al servizio FTS3 del CERN, soffermandosi in modo particolare sull'evoluzione dei modelli di sicurezza: dalla Grid Security Infrastructure e i certificati X.509 con delega tramite proxy certificate e VOMS, fino alla più recente transizione verso l'autenticazione federata basata su token OAuth2/OpenID Connect.

La parte sperimentale ha messo a confronto cinque strumenti di trasferimento su un'infrastruttura composta da tre macchine virtuali Google Cloud dislocate tra Paesi Bassi, Francia e Stati Uniti. I risultati hanno confermato in modo netto la previsione teorica: sulla tratta a bassa latenza le differenze tra gli strumenti a singola connessione TCP restano contenute, mentre sulla tratta intercontinentale il meccanismo AIMD livella le loro prestazioni verso valori compresi tra 18 e 25 MB/s, rendendo trascurabili le ottimizzazioni a livello applicativo. GCT, grazie al parallelismo di GridFTP, ha mantenuto un throughput superiore di oltre quattro volte rispetto al migliore degli altri strumenti, riducendo da giorni a ore il tempo necessario per trasferire 5 TB.

Questi dati hanno un'implicazione diretta per le infrastrutture di ricerca che operano su scala di petabyte: affidarsi a strumenti basati su singola connessione TCP per trasferimenti intercontinentali comporta un costo operativo che, accumulato su campagne di acquisizione dati prolungate, può tradursi in settimane di ritardo. Il parallelismo a livello di protocollo non è dunque un'ottimizzazione facoltativa, ma una necessità architetturale.

Va tuttavia osservato che l'ecosistema del trasferimento dati su larga scala è in una fase di transizione. Il Globus Toolkit non è più mantenuto come progetto open-source, e la comunità WLCG sta progressivamente adottando protocolli basati su HTTP/WebDAV e XRootD, più semplici da integrare con le infrastrutture moderne e nativamente compatibili con i modelli di autenticazione federata. La sfida per il futuro risiede nel coniugare le prestazioni dimostrate dal trasferimento parallelo con la semplicità e l'interoperabilità che questi nuovi protocolli offrono. I risultati di questo lavoro suggeriscono che qualsiasi soluzione che ambi-

sca a sostituire GridFTP dovrà necessariamente incorporare meccanismi di parallelismo analoghi, pena il ripresentarsi delle stesse limitazioni qui documentate.

Bibliografia

[20008] Uk government: National audit office report - the defence information infrastructure (dii), 2008.

[Arr22] Alessandro Arrichiello. *Podman for DevOps : containerization reimagined with Podman and its companion tools*. [first edition].. edition, 2022.

[Bat24] Aaron Bateman. Hunting the red bear: Satellite reconnaissance and the 'second offset strategy' in the late cold war. *The International History Review*, 47(3):535–551, September 2024.

[Bra19] Henry E. Brady. The challenge of big data and data science. *Annual Review of Political Science*, 22(1):297–323, May 2019.

[CKW16] P. Campana, M. Klute, and P.S. Wells. Physics goals and experimental challenges of the proton–proton high-luminosity operation of the lhc. *Annual Review of Nuclear and Particle Science*, 66(1):273–295, October 2016.

[CSH+05] European Grid Conference, Peter M. A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak. *Advances in grid computing - EGC 2005 : European grid conference Amsterdam, The Netherlands, February 14-16, 2005 : revised selected papers*, volume 3470 of *Lecture Notes in Computer Science*. Springer, Berlin, 2005 edition. edition, 2005.

[CVCG19] Andrea Ceccanti, Enrico Vianello, Marco Caberletti, and Francesco Giacomini. Beyond x.509: token-based authentication and authorization for hep. *EPJ Web of Conferences*, 214:09002, 2019.

[DGG02] M. Draoli, C. Gaibisso, and D. Giannelli. Deploying the globus security infrastructure in a production environment: Testing and evaluation. In *Electronic Workshops in Computing*. BCS Learning & Development, December 2002.

[FK97] Ian Foster and Carl Kesselman. Globus: a metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, June 1997.

[FK22] Ian Foster and Carl Kesselman. The history of the grid. 2022.

[HJMS+00] Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. *Data Management in an International Data Grid Project*, pages 77–90. Springer Berlin Heidelberg, 2000.

[LCC+09] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the internet. *ACM SIGCOMM Computer Communication Review*, 39(5):22–31, October 2009.

[LLM] M.J. Litzkow, M. Livny, and M.W. Mutka. Condor-a hunter of idle workstations. In *[1988] Proceedings. The 8th International Conference on Distributed*, pages 104–111. IEEE Comput. Soc. Press.

[Luk11] Stephen Lukasik. Why the arpanet was built. *IEEE Annals of the History of Computing*, 33(3):4–21, March 2011.

[SC92] Larry Smarr and Charles E. Catlett. Metacomputing. *Communications of the ACM*, 35(6):44–52, June 1992.

[Spe03] Tim Speed. *Internet security : a jumpstart for systems administrators and IT managers*. Digital Press, Amsterdam ; London, 1st edition. edition, 2003.

[WSF+] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for grid services. In *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, HPDC-03, pages 48–57. IEEE Comput. Soc.

Ringraziamenti

Ringrazio innanzitutto il correlatore Stefano Diciotti per la disponibilità e la competenza dimostrate durante l'intera fase di gestione del progetto che è culminato in questa tesi. Lo ringrazio per avermi offerto l'opportunità di lavorare in un ambito attuale, in continua evoluzione e cruciale per la risoluzione di problemi complessi che richiedono ingenti risorse di calcolo.

Ringrazio Alessandro Chiarini e BI-REX[1] per aver messo a disposizione l'infrastruttura necessaria ai test e per la costante disponibilità accordatami. Senza un'infrastruttura di tale livello, la componente sperimentale di questo lavoro non sarebbe stata realizzabile.

Ringrazio Ciro Barbone per avermi segnalato al correlatore Stefano Diciotti e per aver configurato i firewall utilizzati per alcuni test all'interno dell'Università di Cesena.

[1] https://bi-rex.it/

Appendice A

Configurazione e script di trasferimento

Per ciascuno strumento analizzato sono stati predisposti script *bash* dedicati, responsabili di due operazioni principali: la misurazione della latenza tramite `ping` e l'esecuzione del trasferimento vero e proprio. Di seguito si riportano lo script utilizzato per SCP, rappresentativo della struttura generale, e le configurazioni specifiche adottate per rsync, rclone e GCT.

```
#!/bin/bash
HOST="matteopaolucci99@10.128.0.34"
IP="${HOST##*@}"
FILE="../testfile_100g"
REMOTE_PATH="/home/matteopaolucci99/"
LOG="transfer_log.txt"

echo "Transfer started: $(date)" | tee "$LOG"

# Start latency monitoring in background
ping -i 1 "$IP" > latency_log.txt &
PING_PID=$!

# Transfer and capture time
START=$(date +%s%N)
scp "$FILE" "$HOST:$REMOTE_PATH" 2>&1 | tee -a "$LOG"
END=$(date +%s%N)

# Stop latency monitoring
kill $PING_PID

ELAPSED=$(( (END - START) / 1000000 ))  # milliseconds
```

```
echo "Duration: ${ELAPSED}ms" | tee -a "$LOG"
echo "Transfer ended: $(date)" | tee -a "$LOG"
```

Listato A.1: Script bash per trasferimento SCP

A.1 Rsync

Per rsync non è stato utilizzato il protocollo SSH per il trasferimento, ma il demone nativo (`rsyncd`). Nello specifico sono stati utilizzati comandi della forma
`rsync -avh -progress rsync://$IP/$REMOTE/$FILE`.

Il file `/etc/rsyncd.conf` è strutturato come segue:

```
use chroot = false

[transfer]
    path = /home/matteopaolucci99/rsync-dir
    read only = no
    comment = Transfer test
```

A.2 Rclone

Per rclone si è prima creato una remote sftp tramite `rclone config`, per poi effettuare i trasferimenti con `rclone copy "$FILE" $REMOTE -transfers $STREAMS` dove `STREAMS` è il numero di trasferimenti paralleli da effettuare.

A.3 GCT

Per effettuare i trasferimenti tramite GCT si è ricorsi al comando `globus-url-copy -vb -fast -p $STREAMS $SRC $DST`. Nello specifico, per i trasferimenti da 100 GB il parametro `STREAMS` è stato impostato a 4, mentre per quelli da 5 TB a 10.