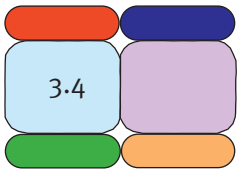
GRID COMPUTING: DA DOVE VIENE E CHE COSA MANCA PERCHÉ DIVENTI UNA REALTÀ?

Mauro Migliardi

Il Grid Computing rappresenta la frontiera della ricerca nel campo delle architetture di calcolo parallelo. Questo termine rappresenta la formulazione di un'idea, quella della condivisione delle risorse di calcolo, sviluppatasi negli ultimi tre decenni a seguito della rapida evoluzione delle tecnologie informatiche. Verrà descritta l'evoluzione di questo modello di calcolatore "non convenzionale" nelle soluzioni presenti e passate identificando quali sono i problemi ancora irrisolti.



1. INTRODUZIONE

Nel campo delle architetture di calcolo parallelo, il Grid Computing rappresenta oggi, senza alcun dubbio, la frontiera delle attività di ricerca. Tale fatto è testimoniato sia dal numero di progetti di ricerca dedicati a tale parola chiave [1], sia dal numero di grandi ditte di sistemi di calcolo, *software* e informatica in generale, che si sono dotate di una loro "soluzione Grid" [2, 3, 4, 5], sia dalle iniziative nazionali [6] o sovranazionali [7] connesse alla parola chiave "Grid". Tuttavia, questo termine, oggi così attuale, è, se si considerano i progetti di ricerca e l'evoluzione delle tecnologie informatiche nel loro complesso, solo la punta di un iceberg. Il termine "Grid", infatti, rappresenta la formulazione particolarmente fortunata di un'idea, quella della condivisione delle risorse di calcolo, sviluppatasi faticosamente negli ultimi tre decenni a seguito della rapida evoluzione delle tecnologie informatiche. Negli ultimi dieci anni, poi, a partire dai primi esperimenti di macchine parallele virtuali realizzate su reti locali sino a giungere a concezioni avveniristiche quali la potenza di calcolo su richiesta o l'integrazione dinamica di componenti simulativi sviluppati indipendentemente, un filo conduttore può essere individuato nella necessità di superare il problema dell'eterogeneità, sia a livello *hardware* che a livello software, al fine di fornire un'immagine integrata del sistema. Questo problema dell'eterogeneità, tuttavia, si presenta in nuove forme ogni qualvolta, trovata una soluzione accettabile per dominare il livello di eterogeneità precedentemente affrontato, si voglia superare la concezione precedente per aprire a nuove idee e possibilità.

Scopo del presente articolo è fornire una visione prospettica dell'evoluzione dei sistemi informatici dedicati al calcolo parallelo/distribuito. In quest'ottica, si forniranno al lettore una tassonomia del fenomeno Grid, una panoramica dell'evoluzione dei sistemi di condivisione delle risorse di calcolo collegata alle problematiche e agli sviluppi tecnologici che l'hanno guidata e un'analisi delle problematiche ancora aperte e che necessitano di trovare una soluzione prima di poter

osservare un effettivo avvento del Grid Computing nel tessuto economico e sociale del mondo tecnologicamente evoluto.

2. UNA TASSONOMIA DEL FENOMENO GRID

Secondo la definizione del progetto Globus, uno dei primi, più importanti e, dal punto di vista dei finanziamenti ottenuti, più fortunati progetti relativi al Grid Computing, "Le Grid sono ambienti persistenti che rendono possibile realizzare applicazioni software che integrino risorse di strumentazione, di visualizzazione, di calcolo e di informazione che provengono da domini amministrativi diversi e sono geograficamente distribuite[1]". La definizione di Grid si basa, quindi, sul concetto di condivisione di risorse di diverse tipologie e si presta, quindi, a categorizzare diverse tipologie di Grid: le *Grid computazionali* [21, 36], le *Grid di dati* [7, 8, 36], le *Grid di applicazioni e servizi* [36], le *Grid di strumentazione* [9, 18] e, infine, le *Grid di sensori* [10, 29, 37].
Una Grid computazionale è l'aggregazione di risorse di calcolo provenienti da domini di sicurezza e gestione differenti. Tale aggregazione è finalizzata a fornire a un insieme di utenti potenza di calcolo *on-demand*, in modo disaccoppiato dalla provenienza, cioè dai nodi che la stanno fisicamente fornendo. Si consideri, ad esempio, il caso di una società multinazionale con sedi sparse in tutto il mondo. Ognuna di queste sedi possiede, oggi, un parco di architetture di calcolo dimensionato o secondo un *worst-case*, dimensionato cioè per soddisfare non le necessità medie ma i picchi di richiesta, oppure dimensionato per soddisfare le necessità medie e, quindi, non in grado di gestire le situazioni di picco.
Una struttura Grid in grado di sopperire alle richieste di picco con potenza computazionale proveniente dalle altre sedi della società permetterebbe di dimensionare in modo più consono alle reali necessità quotidiane il parco macchine realizzando così un notevole risparmio. Allo stesso tempo, lo stesso meccanismo di raccolta della potenza di calcolo sull'intero insieme delle risorse di calcolo della società permette di realizzare un supercalcolatore disponibile dinamicamente nel momento del bisogno. Le Grid computazionali, rappresentando una linea evolutiva delle architetture di calcolo, hanno una storia particolarmente ricca e variegata, sono quindi quelle di cui maggiormente si parlerà nel presente articolo.
Le Grid di dati possono essere considerate una delle forme evolutive del Web. Infatti, come il Web, nascono per contenere grandi moli di dati distribuite in domini differenti per gestione e posizione geografica. A differenza del Web, dove, pur data la ricchezza delle informazioni presenti, mancano sia meccanismi espliciti di standardizzazione dei significati associati a queste informazioni, sia strumenti di elaborazione concorrente delle informazioni ottenute, le Grid di dati coniugano la ricchezza delle sorgenti con gli strumenti adatti a far sì che le informazioni presenti possano essere facilmente correlate e vengano, quindi, a dotarsi di un alto valore aggiunto rispetto al mero contenuto. Si consideri, a titolo di esempio, il caso delle basi di dati contenenti i casi clinici dei singoli ospedali nel mondo. Ognuno di queste basi di dati possiede, già presa singolarmente, un fortissimo valore in quanto permette di analizzare quali siano state nel passato le decisioni dei clinici. Tale valore però, aumenta in modo non lineare se, aggregando le basi di dati in una Grid di dati, si rende possibile analizzare, confrontare e correlare le decisioni prese da clinici di scuole diverse in presenza di patologie uguali o, quantomeno, comparabili.
Le Grid di applicazioni e/o servizi, rappresentano una delle visioni più futuribili nel campo del Grid Computing. Esse, infatti, non si limitano a essere una versione globalizzata all'intera Internet del concetto di *Application Service Provider* (ASP), cioè della possibilità di affittare un certo tempo di esecuzione di una specifica applicazione su di un *server* remoto, ma contemplano anche la aggregazione di componenti applicativi sviluppati indipendentemente al fine di realizzare nuove e originali applicazioni. Si consideri, a titolo di esempio, il seguente caso. La costruzione di una nave è un processo che ormai non coinvolge più il solo settore cantieristico in senso stretto, ma, al contrario, richiede l'intervento di un largo insieme di fornitori dedicati ai singoli sottosiste-

[1] Fonte: http://www.globus.org

mi. Lo sviluppo di tali sottosistemi, per esempio la parte motoristica, la parte di stabilizzazione o la parte dedicata alla sensoristica di navigazione, viene affidato esternamente all'ente cantieristico nominalmente responsabile della costruzione navale. In uno scenario tradizionale, ognuno di questi fornitori ha una visione limitata al sottosistema di cui è incaricato e si viene così a perdere la visione olistica del prodotto finale sino al momento in cui, ottenuti tutti i singoli sottosistemi dai diversi fornitori, l'ente cantieristico sarà in grado di assemblare il prodotto finito. Questo non permette di valutare all'interno del *loop* di progettazione gli effetti dovuti alle interazioni tra i diversi sottosistemi, e rischia, quindi, di portare a scoprire indesiderati effetti collaterali solo in fase di collaudo dell'intero sistema. Una soluzione a questa *empasse* è ottenibile tramite l'integrazione in una Grid di applicazioni di elementi simulativi relativi ai diversi sottosistemi. Tuttavia, elementi simulativi sviluppati secondo lo standard internazionale attuale, cioè (HLA) *High Level Architecture* [11], non supportano l'assemblaggio dinamico a *run-time* di un sistema completo, hanno un supporto limitato per l'esecuzione distribuita e non permettono di schermare completamente agli altri utenti i dettagli interni al componente. Al contrario, una Grid di applicazioni deve permettere a ciascun fornitore:

- di mantenere il pieno controllo del componente simulativo relativo al suo prodotto;
- di esporre solo un'interfaccia opaca;
- di assemblare l'intero sistema nave in una simulazione completa.

In una situazione di questo genere, ovviamente, una Grid di applicazioni sussume al suo interno sia una Grid computazionale che una Grid di dati. Infatti, le risorse di calcolo utilizzate per eseguire i diversi componenti applicativi sono aggregate dinamicamente a partire da domini di gestione diversi. Contemporaneamente, i dati su cui si trova a operare l'applicazione complessivamente costruita provengono da sorgenti differenti e sono correlate tramite l'esecuzione coordinata delle componenti applicative.
Una Grid di strumentazione consiste nella generalizzazione del concetto di accesso remoto ad apparati di costo elevato o per rendere fruibile in forma remota un *setup* sperimentale di elevato valore didattico. In una Grid di strumentazione, apparati gestiti da enti diversi possono essere integrati per aggregare esperimenti complessi non possibili in nessuno dei singoli siti coinvolti nella Grid o per condividere strumentazione didattica in modo da rendere disponibile a corsi di studi afferenti ad atenei diversi una struttura condivisa per la realizzazione di esperimenti di comprovato valore didattico.
Il concetto di Grid di sensori proviene, principalmente, dal mondo militare. In ambito militare, tale concetto sta alla base della moderna teoria del combattimento come attività incentrata sull'interconnessione delle componenti dell'esercito belligerante. In tale visione, infatti, si parla della Grid di sensori come del livello atto a fornire la visione dettagliata dello stato corrente agli agenti operativi e decisionali. Recentemente, tuttavia, l'idea di una Grid di sensori destinata a fornire una visione dettagliata e a prova di guasto della situazione del campo è penetrata anche in ambiti non militari quali quelli dell'agronomia e della salvaguardia del territorio. Si consideri, a titolo di esempio, il caso di un insieme di sensori all'infrarosso sparsi per una foresta con lo scopo di segnalare tempestivamente lo svilupparsi di focolai di incendio. L'integrazione delle informazioni provenienti da un grande numero di sensori permette di effettuare una migliore separazione del calore ambientale dal calore proveniente da un focolaio d'incendio e, quindi, garantisce una consistente diminuzione della sensibilità del sistema ai falsi positivi. Le Grid di sensori, sono il ramo del Grid Computing che maggiormente si avvicina al concetto di *Ubiquitous Computing*, infatti, tutte le attività di ricerca dedicate a questa tipologia di Grid pongono una fortissima attenzione alle problematiche legate alle reti *wireless*, all'instradamento autonomo e alle reti senza infrastruttura statica, cioè le cosiddette reti *ad hoc*.

3. UN PO' DI STORIA

Per valutare il livello di tumultuosità evolutiva delle tecnologie dell'informazione, si considerino, per esempio, questi due parametri:

- il numero di transistori e la frequenza di *clock* del singolo processore (Figura 1 **A** e **B**), a rappresentare la sua capacità elaborativa;

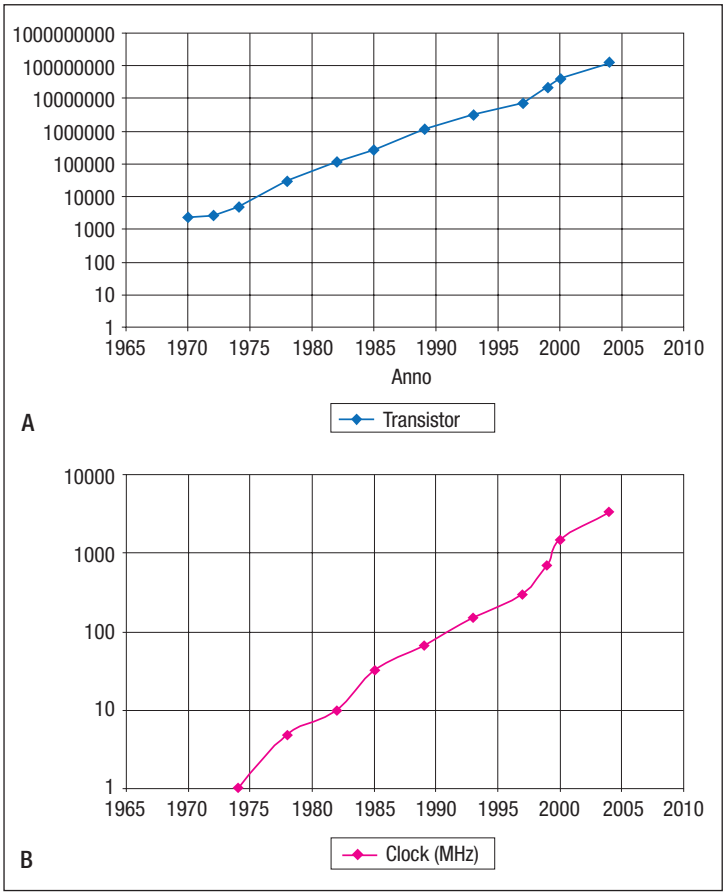


FIGURA 1

A Numero di transistor presenti su singola CPU. B Frequenza tipica di clock

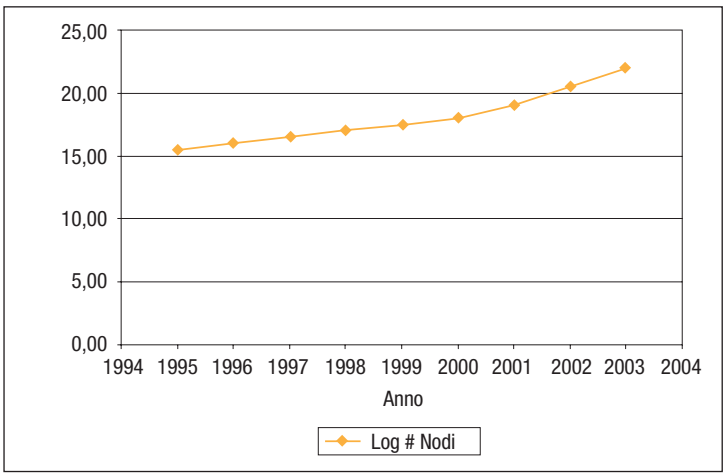


FIGURA 2

Numero di calcolatori interconnessi tramite Internet

il numero di calcolatori connessi a Internet a rappresentare la capacità di cooperare (Figura 2).

Questo aumento ha guidato la maccanica dell'evoluzione dei sistemi informatici in generale e dei meccanismi di condivisione delle risorse di calcolo in particolare.

Se si considera il fenomeno della condivisione delle risorse di calcolo, si possono identificare quattro "ere" (Tabella 1) caratterizzate da distinte metodologie e architetture dedicate al supercalcolo.

La prima era è caratterizzata dal concetto "un computer per molti utenti". In questa era il costo della singola risorsa di calcolo è tale da essere accessibile solo a entità grandi, in grado di ammortizzare l'investimento attraverso l'uso contemporaneo della risorsa da parte di numerosi utenti. In questa era, esisteva un ampio insieme di architetture dedicate al supercalcolo basate su soluzioni completamente *ad hoc* sviluppate sia da ditte specializzate che dai grandi venditori di hardware generico. Negli anni '80, la discesa dei costi dell'hardware porta all'inizio del decennio ad avere sistemi di lavoro dedicati a singoli utenti e alla fine del decennio ad avere computer personali anche per usi non strettamente tecnici (per esempio, *desktop publishing* e *office automation*). Contemporaneamente, il campo delle architetture di supercalcolo conosceva la stagione delle macchine SIMD (*Single Instruction Multiple Data*) [20] e una fioritura di ditte specializzate nella produzione di architetture di supercalcolo basate su tale paradigma. Tuttavia, già alla fine degli anni '80, l'attenzione di alcuni gruppi di ricercatori inizia a focalizzarsi sulla possibilità di sfruttare l'accresciuta potenza di calcolo disponibile su una singola *workstation* e la disponibilità di tecnologie a basso costo atte a permettere l'interconnessione in rete locale di tali workstation per la costruzione di macchine parallele virtuali [39].

Gli anni '90 sanciscono l'avvento del concetto di "Molti calcolatori per un singolo utente". La definitiva affermazione della legge di Moore [26, 34] nel campo dei *personal computer*, porta a rendere economicamente impraticabile ogni tentativo di sviluppare architetture di calcolo *ad hoc*. Infatti, come testimoniato dal fallimento di esperienze quali quella della LISP (*List Processing*) Machine [27, 30], il tempo richiesto dal ciclo di progettazione, test e produzione di un'architettura dedicata risulta essere nettamente superiore a quello richiesto dalle CPU *mainstream* per raggiungere prestazioni tali da permettere a sistemi software eseguiti su di esse di superare nettamente le

Era	Relazione tra calcolatori e utenti	Architettura di condivisione	Architetture di super calcolo
Anni '70	Uno a molti	Sistemi Time sharing	Supercalcolatore
Anni '80	Uno a uno	PC e Workstation	Supercalcolatore
Anni '90	Molti a uno	Clusterizzazione	COW
Terzo millennio	Molti a molti	Grid	Grid

TABELLA 1
Ere computazionali

prestazioni dei sistemi hardware dedicati. Allo stesso modo, l'enorme incremento prestazionale dimostrato con scadenza annuale dalle CPU mainstream esclude dal mercato tutte le architetture di supercalcolo basate su processori speciali, come le architetture SIMD. Contemporaneamente, gli anni '90 testimoniano il definitivo affermarsi come tecnologie mainstream di quelle che, a fianco dell'incredibile evoluzione delle CPU, rappresentano la seconda colonna portante dell'idea stessa di Grid: le reti di calcolatori e Internet.

3.1. Dal Supercalcolatore al Cluster di Workstation

Il passo successivo viene preannunciato dal colpo di coda della Thinking Machine Corporation. Nel 1993, dopo un decennio di ardito pionierismo nel campo delle architetture SIMD, per tentare di salvare un futuro che, sfumati i grandi finanziamenti militari per la guerra fredda, appare assai oscuro, tenta la produzione della Connection Machine 5 [35], un supercalcolatore basato sull'interconnessione tramite una struttura proprietaria di pochi, potentissimi[2] processori mainstream: gli SPARC prodotti da SUN. I costi di produzione del sistema restano comunque troppo alti a causa delle molte componenti altamente customizzate, quali per esempio la rete di interconnessione dei processori, ancora presenti nell'architettura, e l'esperimento risulta, quindi, economicamente fallimentare[3].

Nel 1994, presso l'università di Stanford, il progetto Beowulf inizia una nuova fase sottolineando con forza il concetto di architettura basata su *Components Off The Shelf* (COTS) o *Commodity Computing* (CC). Questo concetto di Commodity Computing, cioè di calcolo come servizio di base, è figlio dell'alleanza Intel/Microsoft che negli anni '90 porta alla realizzazione della visione "Un PC su ogni scrivania". Infatti, tra le caratteristiche che rendono particolarmente significativo il primo cluster Beowulf non vi sono *performance* di picco o peculiarità architetturali, ma bensì:

- l'uso come nodi computazionali di macchine comparabili con quelle destinate all'*office automation* (PC con processore Intel DX4);
- l'uso come tecnologia di interconnessione tra i nodi dello standard di rete progettato per l'*office automation* (*ethernet*);
- l'uso di un sistema operativo *open source* di uso generale, non specificamente progettato per il calcolo ad alte prestazioni (Linux o FreeBSD).

Nella seconda metà degli anni '90, nel mondo del calcolo ad alte prestazioni la ricerca verte in larga misura su architetture cluster, cluster di workstation (COW) e cluster di PC[4], cioè architetture caratterizzate dall'aggregazione statica di nodi di computazione dedicati completamente all'architettura cluster e interconnessi tramite un'interfaccia di rete di tipo standard.

È fondamentale notare che, tra la fine degli anni '80 e l'inizio degli anni '90, il processo che

[2] Si parla, ovviamente, in termini relativi ai primi anni '90.

[3] Nel vero senso della parola: nonostante la CM-5 abbia l'onore di apparire in un film di successo popolare quale *Jurassic Park*, la Thinking Machine Corporation dichiara fallimento nello stesso 1993.

[4] L'Università del Tennessee in Knoxville mantiene una lista dei 500 supercomputer più performanti al mondo che viene aggiornata due volte all'anno. Il primo cluster appare nella lista nell'anno 1995. Nel 2003 sette dei Top 10 sono architetture cluster.

Si parla di ***Single System Image*** (SSI) quando, in un ambiente costituito da risorse eterogenee e/o distribuite, si cerca di costruire a beneficio dell'utente un'interfaccia che unifichi la gestione e l'utilizzo di queste risorse mascherandone le differenti caratteristiche. È importante sottolineare come la definizione del concetto di SSI non dipenda da disomogeneità in termini di prestazioni, ma solo in termini di possibilità di utilizzo.
Un semplice esempio di SSI è quello fornito dalla condivisione delle risorse nei sistemi Microsoft. In tale versione di SSI, lo strato middleware incorporato nei sistemi Windows si occupa di mascherare la distribuzione e le disomogeneità delle risorse (per esempio, quelle derivanti da versioni diverse del sistema operativo) e le visualizza all'utente finale con un'interfaccia unificata. Tuttavia, la SSI fornita dal meccanismo di condivisione delle risorse non è assolutamente inficiata dalla presenza di sistemi più lenti in quanto l'unificazione cui si mira è meramente di interfaccia e non si applica alle prestazioni.

culmina con l'affermazione dei concetti COTS e CC segna un cambiamento sostanziale nella ricerca dedicata ai sistemi di calcolo. Se nel precedente ventennio tale ricerca era stata orientata principalmente all'hardware, essa diventa ora sostanzialmente orientata al software. Infatti, anche lo sviluppo di sistemi per il calcolo ad alte prestazioni risulta essere principalmente mirato alla definizione di strati software (middleware[5]) atti a mascherare l'eterogeneità delle risorse di calcolo costruendo in modo più o meno marcato una ***Single System Image***. Per i sistemi cluster di tipo Beowulf, il livello di SSI è dato fondamentalmente da:

- la possibilità di compilare un programma eseguibile sull'intero sistema cluster;
- la possibilità di schedulare l'esecuzione di un'applicazione su tutti o un sottoinsieme dei nodi del cluster tramite un sistema *batch*;
- la possibilità di monitorare ed eventualmente abortire le applicazioni in esecuzione.

3.2. Dal Cluster al Grid

L'inizio del passo successivo è segnato, nel 1996, da una fioritura di progetti mirati a definire sistemi per il *metacomputing*[6]. I sistemi di metacomputing possiedono molte caratteristiche in comune con i sistemi cluster, infatti, come questi ultimi, essi sono gruppi di nodo computazionali di tipo standard interconnessi tramite interfacce di rete di tipo standard. Tuttavia, esse possiedono in sè le caratteristiche seminali del concetto di Grid Computing, ovvero definiscono "sistemi formati dall'aggregazione dinamica di nodi computazionali non dedicati al metacomputer, interconnessi anche attraverso Internet e appartenenti a domini di sicurezza e gestione non omogenei". In tabella 2 sono schematizzate le principali differenze tra un sistema cluster e un sistema per il metacomputing.
Per due anni, il termine metacomputer o sistema di metacomputing viene utilizzato per rappresentare tutte le strutture di calcolo formate da nodi eterogenei distribuiti su Internet. Persino il progetto Globus, negli articoli pubblicati sino all'inizio dell'anno 1998, vie-

TABELLA 2
Principali differenze tra un sistema cluster e un sistema di Metacomputing

Sistemi Cluster	Sistemi di Metacomputing
Nodi computazionali dedicati completamente al cluster	Nodi computazionali aggregati dinamicamente su richiesta o base volontaristica
Nodi computazionali il più possibile omogenei fra loro	Nodi computazionali di qualunque genere
Rete di interconnessione dedicata al solo traffico dati delle applicazioni del cluster	Rete di interconnessione condivisa con traffico estraneo al metacomputer
Rete di interconnessione locale	Rete di interconnessione distribuita su Internet
Un singolo dominio di gestione e sicurezza	Più domini di gestione e sicurezza forniscono i nodi del metacomputer

[5] Secondo la base dati IEEE Explorer, il primo articolo scientifico relativo a middleware appare nel 1993. Nel 1994 gli articoli su middleware sono 7. Negli anni successivi circa 170 all'anno.

[6] Il termine metacomputing viene effettivamente coniato da L. Smarr, direttore NCSA, alla fine degli anni '80, tuttavia è solo nel 1997 che tale termine prende piede con il suo significato attuale. Nell'anno 1996 vengono pubblicati circa 150 articoli scientifici dedicati al metacomputing. Nel 1997, quasi 400 (Fonte: *Citeseer*).

Modello a scambio di messaggi
Il modello computazionale a scambio di messaggi viene introdotto negli anni '80 da C. A. R. Hoare. Esso si basa fondamentalmente su due primitive, *send* e *receive*, che permettono a processi strettamente sequenziali di comunicare e sincronizzarsi fra loro al fine di costruire un'applicazione concorrente o parallela di qualsivoglia complessità. Il modello a scambio di messaggi è quello più comunemente usato nel calcolo distribuito in quanto non richiede strutture complesse come la memoria condivisa, ma pone, invece, come unico vincolo architetturale la presenza di un canale di comunicazione tra tutte le componenti del sistema di calcolo.

ne descritto dai suoi autori "Un toolkit per un'infrastruttura di metacomputing" [22]. Tuttavia, tra l'inizio del 1998 [23] e la prima metà del 1999 [24], Ian Foster e Carl Kesselmann, i padri del progetto Globus, pubblicano la loro visionaria metafora di una "Griglia di distribuzione della potenza computazionale in cui, come i watt nella griglia di distribuzione dell'elettricità, la potenza computazionale può essere distribuita a chi ne fa richiesta senza badare alla sua provenienza". Questa futuristica visione ha un immediato successo e viene adottata dalla comunità scientifica sostituendo il termine metacomputing[7].

4. PROGRAMMARE IN PARALLELO

Sino a questo punto la nostra attenzione si è focalizzata principalmente sull'evoluzione delle architetture hardware che ha portato alla visione delle Grid. Si consideri ora il punto di vista complementare: cosa vuol dire scrivere programmi per una Grid computazionale?

4.1. Grid del Supercalcolo

Secondo la visione sponsorizzata dalla comunità del supercalcolo, una Grid computazionale consiste nell'interconnessione dei supercalcolatori disponibili nei diversi centri al fine di realizzare una struttura globale in grado di lanciare l'esecuzione dell'applicazione candidata su uno qualunque dei supercalcolatori parte della Grid che sia in grado di fornire le risorse (CPU, memoria e spazio disco) richieste dall'applicazione stessa. In tale visione, quindi, le applicazioni che devono essere scritte per una Grid computazionale sono sostanzialmente le stesse che venivano scritte per le architetture cluster in quanto la dinamicità del sistema Grid è, di fatto, limitata alla fase di scelta del supercalcolatore su cui lanciare l'esecuzione. In questa situazione, quindi, le tecniche di programmazione utilizzate sono quelle tradizionali dello scambio di messaggi e si basano, fondamentalmente, sull'uso delle librerie derivate dal modello CSP [19] quali PVM [39] e MPI [31]. Questa soluzione implica un modello di applicazione con un accoppiamento stretto tra le sue varie parti e, pur permettendo di ottenere livelli di prestazioni molto alti, pone vincoli tali da prevenire il completo utilizzo della natura dinamica di una Grid computazionale. Inoltre, a meno di accettare degradi prestazionali fortissimi, questo modello di applicazione richiede di ritagliare l'applicazione stessa sulle caratteristiche del sistema di calcolo che, quindi, non può essere soggetto alle variazioni dinamiche tipiche di una Grid computazionale. Infine, questo tipo di programmazione è estremamente complessa e richiede una conoscenza approfondita delle problematiche proprie del calcolo parallelo ad alte prestazioni. Quindi, pur essendo applicabile a una vasta gamma di tipologie di applicazione, la generazione di una nuova applicazione secondo il paradigma di scambio di messaggi o il trasporto di un'applicazione tradizionale allo stesso paradigma computazionale richiede uno sforzo di ingegnerizzazione o reingegnerizzazione estremamente elevato ed è affrontabile solo da un *pool* di programmatori esperti nelle problematiche proprie del calcolo parallelo ad alte prestazioni.
Per queste ragioni, questo tipo di soluzione, pur essendo già disponibile sul mercato, è di fatto appetibile solo per la grande industria, quella che possiede al suo interno uno staff IT di dimensioni tali da potersi permettere di addestrarne una parte in modo specifico e di averne una parte considerevole dedicata allo sviluppo e alla manutenzione di specifiche applicazioni.

[7] Nel 1999, il numero di articoli che parlano di metacomputing scende a 150, nel 2000 praticamente a 0. Contemporaneamente sale il numero di articoli relativi al Grid Computing. (Fonte: *Citeseer*).

4.2. Grid "Montecarlo"

Una seconda visione delle Grid computazionali è quella pionierizzata da progetti quali SETI@home [38] e realizzata al presente nei sistemi di ditte quali Entropia [12] o AVAKI [13] o dei grandi vendor di sistemi informatici citati all'inizio del presente articolo. In questa visione, un centro di controllo della Grid computazionale gestisce l'allocazione di parti dell'esecuzione globale dell'applicazione a nodi ottenuti dinamicamente in modo volontaristico, come nel caso di SETI@home, o identificando nodi disponibili all'interno di una intranet aziendale, come nel caso delle soluzioni commerciali. Le applicazioni generate in questo modo sono tutte di tipo Montecarlo, realizzate con un modello di computazione farmer-worker ad accoppiamento ridotto.

Questo tipo di soluzione permette la creazione di nuove applicazioni o il trasporto di applicazioni pre-esistenti verso piattaforme di tipo Grid computazionale con un costo di re-ingegnerizzazione piuttosto basso. Essa, infatti, non richiede una conoscenza estremamente approfondita delle problematiche proprie del calcolo parallelo ad alte prestazioni. Tuttavia, questa soluzione è adottabile solo per una specifica categoria di applicazioni, quelle appunto basate su simulazioni di tipo Montecarlo o, più in generale, quelle in cui è necessario eseguire una stessa sequenza di operazioni su di un enorme insieme di dati o casi di test che possono essere ripartiti in sottoinsiemi processabili in modo indipendente. L'indipendenza tra le diverse parti garantisce l'assenza di comunicazione tra i nodi che eseguono le operazioni. Questo permette di trascurare i problemi che potrebbero essere generati dall'eterogeneità delle interconnessioni tra i nodi della Grid computazionale e semplifica, quindi, sensibilmente l'applicazione.

Purtroppo, però, dati i vincoli sulla tipologia di applicazione utilizzabile con questa soluzione, essa non è generalizzabile a un ampio bacino di utenza ed è principalmente adottata per test di nuovi farmaci o ricerca di pattern all'interno di enormi moli di dati come nei casi del progetto SETI o dell'analisi del genoma umano.

4.3. Grid di Servizi

Una terza, nuova visione delle Grid computazionali, è quella di una Grid, orientata ai servizi, che utilizzi la tecnologia dei *Web Services*. Questa visione viene formulata da alcuni progetti di ricerca [8] ed è stata formalizzata nella prima metà dell'anno 2002 dal progetto *Open Grid Services Architecture* (OGSA), una *joint venture* tra IBM e il team del progetto Globus [14]. Tale visione si propone lo sviluppo di applicazioni capaci di utilizzare la tecnologia Grid-computing adottando un paradigma di computazione *multi-tier*, basato su servizi con interfacce standardizzate. Nel modello definito da OGSA, una Grid è sostanzialmente un'organizzazione virtuale in grado di permettere e coordinare l'uso di risorse condivise. Al fine di realizzare questa visione, OGSA definisce tre livelli di protocollo:

1. livello di connettività;
2. livello di controllo delle risorse;
3. livello di collettivizzazione delle risorse.

Il primo livello si occupa fondamentalmente di gestire la comunicazione tra le componenti della Grid e il controllo degli accessi, sia da parte di utenti finali che da parte di altre componenti della Grid, a ciascuna delle componenti stesse.

Il secondo livello, si occupa di definire le necessità di una richiesta ed effettuare un'associazione di tali necessità con le risorse atte a soddisfarle[8].

Il terzo livello, infine, si occupa di coordina-

Modello Farmer-Worker

Il modello computazionale Farmer-Worker viene comunemente utilizzato nel caso di applicazioni decomponibili in *task* di grana medio-grande (*Bag of Work*) tra di loro indipendenti, cioè che per il loro svolgimento non necessitano di comunicare con altri. Un Farmer genera un insieme di Bag of Work che vengono messe a disposizione dei Worker. Ogni Worker che si trova in stato ideale estrae una Bag of Work dall'insieme ed effettua l'esecuzione del task richiesto. Alla conclusione del task, il Worker restituisce i risultati ottenuti al Farmer e, trovandosi, quindi, ora nello stato ideale, torna a controllare se esiste altro lavoro da fare.

Questo meccanismo iterativo implementato dai Worker costituisce un naturale gestore del bilanciamento dei carichi in quanto un Worker più lento si limiterà a processare meno Bag of Work, mentre un Worker più efficiente se ne accaparrerà un numero maggiore.

Il modello Farmer-Worker può essere applicato sia ad applicazioni in cui esiste un solo passo di generazione delle Bag of Work, sia ad applicazioni in cui il Farmer, una volta raccolti i risultati ottenuti dal processamento di un insieme di Bag of Work, ne produca uno successivo (per esempio, applicazioni di simulazione a tempo discreto).

[8] Questo livello viene spesso definito "matchmaker", cioè paraninfo.

re l'utilizzo delle risorse condivise in modo da fornire un servizio di gestione collettiva di tali risorse.
Tale paradigma, facendo leva su tecnologie software mainstream quali Web Services, Java2 Enterprise Edition e Microsoft.NET, risulta essere più vicino alla sensibilità dei programmatori di applicazioni di tipo commerciale e presenta, quindi, una minor difficoltà di adozione da parte delle piccole medie imprese.
Una soluzione di questo genere, pur dovendo sacrificare una parte dell'ottimalità prestazionale raggiungibile solo con paradigmi di computazione dedicata alla computazione parallela quale quello di scambio di messaggi, permetterebbe di rendere fruibile la tecnologia Grid-computing anche a ditte medio-piccole non dotate di un gruppo di programmatori esperti nelle problematiche della computazione parallela. Una soluzione di questo genere permetterebbe, quindi, di:
- allargare il bacino di utenza della tecnologia Grid-computing rendendola disponibile non solo alla grande industria ma anche alle PMI;
- rendere disponibile alle PMI una potenza di calcolo decisamente superiore a quella correntemente a loro disposizione senza per altro richiedere grossi investimenti in hardware dedicato.

5. PROBLEMI APERTI

Purtroppo, il lavoro iniziato dal progetto OGSA è ben lungi dall'essere completato. Si consideri, a titolo di esempio, il caso del protocollo SOAP [9]. Questo protocollo, è il protocollo di riferimento per la trasmissione dati utilizzato dalle tecnologie Web Services e NET, tuttavia, le sue caratteristiche lo rendono del tutto inadatto a situazioni che richiedano alti livelli di prestazioni. Per questa ragione, il progetto OGSA ne suggerisce l'utilizzo soltanto per il livello di controllo delle risorse. Tuttavia, sebbene siano in corso diversi studi per identificare quali protocolli possano sostituire SOAP in ambienti ad alte prestazioni, tali studi sono ancora in fase sperimentale, quindi non esiste, ad oggi, una soluzione unificante che possa essere suggerita all'interno di OGSA. I modelli di programmazione per Grid computazionali disponibili sono, quindi, ad oggi, solo quello a scambio di messaggi e quello *farmer-worker* per applicazioni di tipo Montecarlo.
Un secondo problema aperto, è quello della gestione di un livello globale di eterogeneità all'interno delle risorse presenti in una Grid computazionale. Come spiegato in precedenza, il meccanismo di superamento del problema dell'eterogeneità e la costruzione di una SSI da super-imporre come colla unificante al di sopra delle diversità. Il progetto Globus, lo standard *de facto* per la costruzione di Grid di supercalcolo, ha definito un linguaggio di descrizione delle risorse che permette di descrivere in modo univoco le pur differenti caratteristiche dei nodi presenti in una Grid. Tuttavia, questa descrizione, seppure rappresenta una specie di SSI, si limita, di fatto, a demandare alle singole applicazioni la gestione dell'eterogeneità. Sono in corso esperimenti relativi alla costruzione estemporanea di Grid computazionali a partire da un insieme estremamente eterogeneo di risorse di calcolo, si consideri, per esempio, il caso di Flashmob1 [15] all'Università di San Francisco. Tuttavia, ad oggi, tutti i casi di successo per la costruzione di Grid computazionali provengono esclusivamente da due categorie:
- i casi in cui l'eterogeneità era resa innocua dalla natura stessa dell'applicazione (Grid montecarlo);
- i casi in cui l'eterogeneità viene limitata alla fase di identificazione delle risorse da usare al tempo dell'esecuzione (Grid di supercalcolo).

Allo stesso modo, risulta irrisolto il problema di garantire un livello di qualità del servizio in presenza di risorse che non solo sono eterogenee, ma cambiano la loro disponibilità nel tempo.
Infatti, se si escludono i due casi delineati sopra, ad oggi risulta praticamente impossibile garantire un livello di qualità del servizio alle applicazioni in mancanza di una quantificazione minima e massima delle risorse disponibili. In un ambiente dinamico come una Grid, un ulteriore livello di complessità è dato dalla necessità di scoprire le risorse che si rendono via via disponibili. La soluzione classica a questo tipo di problema è la realizzazione di un servizio di *directory*: tuttavia, la maggior parte dei servizi di directory oggi disponibili si basano su architetture con server centralizzato che,

ovviamente, mal si prestano ad una struttura potenzialmente globale come una Grid. Un esempio di directory globale e, in quanto completamente distribuita, estremamente scalabile, viene dal *Domain Name System* (DNS). Tuttavia, l'architettura del DNS è progettata per una frequenza di cambiamento dei dati presenti estremamente bassa, quindi non è in grado di supportare un sistema Grid in cui le risorse possono comparire e scomparire molto in fretta. Si consideri, a titolo di esempio, il caso di un nodo di computazione del sistema SETI@home. Esso è, di fatto, un PC che, essendo rimasto inattivo per alcuni minuti, ha fatto partire lo *screen saver*. Questo fatto, però, non fornisce alcuna garanzia sul prolungarsi della disponibilità della risorsa, al contrario è altrettanto probabile che il proprietario, disturbato dalla partenza dello *screen saver*, smetta di sognare ad occhi aperti e ricominci a utilizzare il suo PC. Per queste ragioni, le tecniche di scoperta delle risorse che oggi vengono utilizzate per aggregare le Grid di supercalcolo non si prestano a essere applicate a tipologie diverse di Grid e limitano, quindi, l'utilizzo di tale tecnologia a poche situazioni particolari.
Come dimostrato dal recente attacco [16] orchestrato ai danni del maggior sistema Grid degli Stati Uniti d'America, TeraGRID [17], il problema della sicurezza rappresenta uno dei maggiori punti di vulnerabilità di un sistema massicciamente distribuito e sottoposto a domini e regolamentazioni eterogenee come un sistema Grid. Ancora una volta, una soluzione proviene dal progetto Globus. Tale progetto ha, infatti, definito una struttura di controllo degli accessi alla Grid basata su *Public Key Infrastructure* (PKI) [32] e un sistema di certificati standard unico per l'intera Grid. Tuttavia, tale sistema si basa sulla reciproca fiducia tra le componenti costitutive della Grid. Questo fatto ha in sè un fattore di rischio e una limitazione di uso. Il fattore di rischio proviene dal fatto che, come dimostrato dall'attacco sopra citato, un sistema che assuma la affidabilità di ogni sua componente è sicuro solo come la più debole delle sue componenti. Quindi, nel caso di un sistema massicciamente distribuito tra diversi domini di gestione è praticamente impossibile garantire un effettivo livello di sicurezza. La limitazione proviene dal fatto che l'assunzione di fiducia completa di ogni componente una Grid in ogni altra componente taglia fuori qualsiasi meccansimo volontaristico di acquisizione di risorse e, quindi, limita grandemente i possibili usi della tecnologia. In particolare, ne limita la convergenza con il fenomeno del *peer-to-peer* con il quale, al contrario, condivide il concetto fondamentale di condivisione dinamica di risorse geograficamente distribuite.

7. CONCLUSIONI

In questo articolo, si è voluto rendere disponibile al lettore una visione prospettica del fenomeno Grid fornendogli una tassonomia di questo fenomeno, una descrizione di come la sua evoluzione sia profondamente legata all'evoluzione tumultuosa delle tecnologie informatiche e infine, una descrizione dei problemi irrisolti.
Per quanto il termine Grid sia oggi estremamente di moda, sono proprio problemi quali la gestione dell'elevatissimo livello di eterogeneità e la sicurezza che, restando irrisolti, rendono ancora improponibile un effettivo avvento del Grid in qualità di tecnologia pervasiva. Ad oggi, l'utilizzo della tecnologia Grid è limitato a grandi attori in grado di investire pesantemente in mano d'opera altamente specializzata, l'unica in grado di gestire e utilizzare i sistemi Grid di presente generazione.

Bibliografia

[1] AA.VV.: Enter the Grid website, http://enterthegrid.com/vmp/articles/contentsEnterTheGridResearch%20RepositoryResearch%20Projects.html

[2] AA.VV.: IBM Grid Computing, http://www-1.ibm.com/grid/

[3] AA.VV.: Sun Solutions: Grid Computing, http://www.sun.com/solutions/infrastructure/grid/

[4] AA.VV.: HP's grid solution - the grid stack, http://www.hp.com/techservers/grid/hp_grid_solution.html

[5] AA.VV.: Get on the Grid with Oracle 10g, http://www.oracle.com/events/grid/index.html?content.html

[6] AA.VV.: D-Grid, Iniziativa del governo tedesco per il Grid Computing, http://www.d-grid.de/

[7] AA.VV.: DataGRID Project Website, http://eu-datagrid.web.cern.ch/eu-datagrid/

[8] AA.VV.: The Globus Data Grid Initiative, disponibile on-line http://www.globus.org/datagrid/

[9] AA.VV.: sito progetto ISILab, http://isilab-esng.dibe.unige.it/

[10] AA.VV.: Has the military realized the power of grid computing?, disponibile *on-line* http://www.grid-today.com/02/1021/100580.html

[11] AA.VV.: Documentazione Ufficiale disponibile presso il sito del ministero della difesa Americano, https://www.dmso.mil/public/transition/hla/

[12] AA.VV.: Entropia website, www.entropia.com

[13] AA.VV.: AVAKI website, www.avaki.com

[14] AA.VV.: sito del progetto OGSA, http://www.globus.org/ogsa/

[15] AA.VV.: sito del progetto flashmobcomputing, http://www.flashmobcomputing.org/

[16] AA.VV.: Hackers hit supercomputing giants, disponibile on-line su http://www.cnn.com/2004/TECH/internet/04/15/hackers.supercomputers.ap/

[17] AA.VV.: Sito ufficiale del progetto TeraGrid, http://www.teragrid.org/

[18] Bagnasco A., Scapolla A.M.: *A grid of remote laboratory for teaching electronics*. Proc. of the 2nd International LeGE-WG Workshop, 3-4 Marzo, Parigi, Francia.

[19] C.A.R. Hoare: *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[20] Flynn M.: Some Computer Organizations and Their Effectiveness. *IEEE Trans. Comput*., Vol. C-21, 1972, p. 94.

[21] Foster I., Kesselmann C., Tuecke S.: *The Anatomy of the Grid, Enabling Scalable Virtual Organizations*. International J. Supercomputer Applications, 2001.

[22] Foster I., Kesselman C., Globus: A metacomputing intrastructure toolkit. *International Journal of Supercomputer Applications*, Vol. 11, n. 2, 1997, p. 115-128.

[23] Foster I., Kesselman C.: The Globus Project: A Status Report, Proceedings of the Seventh Heterogeneous Computing Workshop. *IEEE Computer Society Press*, March 1998, p. 4-19.

[24] Foster I., Kesselman C., (eds.): The Grid: *Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, 1999.

[25] Gannon D., et al.: Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications. *In Special Issue on Grid Computing, Journal of Cluster Computing*, Vol. 5, n. 3, 2002, p. 325-336. Kluwer Academic Publishers, 2002.

[26] Gelsinger P., Gargini P., Parker G., Yu A.: Microprocessors circa 2000. *IEEE Spectrum*, Ottobre, 1989.

[27] Graham P.: *Anatomy of a Lisp Machine*. AI Expert, December 1988.

[28] Gudgin M., Hadley M., Mendelsohn N., Moreau J., Frystyk Nielsen H.: *Simple Object Access Protocol* (1.2). Documentation available on line at http://www.w3.org/TR/SOAP/

[29] Kahn J.M., Katz R.H., Pister K.S.J.: *Mobile networking for smart dust*. Proc. Of MobiCom 99, Seattle, WA, USA, August 1999.

[30] Kogge P.M.: *The Architecture of Symbolic Computers*. McGraw-Hill 1991. ISBN 0-07-035596-7.

[31] Message Passing Interface Forum, MPI 2.0, disponibile on-line a http://www.mpi-forum.org/docs/docs.html

[32] Micali S.: *Enhanced Certificate Revocation System*. MIT/LCS/TM-542, 1995.

[33] Migliardi M., Kurzyniec D., Sunderam V.: *Standard Based Heterogeneous Metacompouting*. The Design of HARNESS II, Proc. of the Heterogeneous Computing Workshop of the International Parallel Distributed Processing Symposium 2002, Fort Lauderdale (FL) April 15-19, 2002.

[34] Moore G.: *Cramming more components onto integrated circuits*. Disponibile on-line, Electronics, Aprile, 1965.

[35] Ramachandramurthi S.: *A beginner's guide to the connection machine CM5, libro elettronico*. Disponibile on-line a http://csep1.phy.ornl.gov/cm5_guide/cm5_guide.html

[36] Saracco R.: Ubiquitous Computing. *Mondo Digitale*, Anno II, n. 7, Settembre 2003, p. 19-30.

[37] Shakkottai S., Srikant R., Shroff N.: *Unreliable Sensor Grids*. Coverage, Connectivity and Diameter, Proc. of INFOCOM 2003, S. Francisco, CA, USA, April 2003.

[38] Sullivan III T., Werthimer D., Bowyer S., Cobb J., Gedye D., Anderson D.: *A new major SETI project based on Project Serendip, data and 100,000 personal computers, Proc. of the Fifth Intl. Conf. on Bioastronomy*. IAU Colloq. n. 161, eds. C.B. Cosmovici, S. Bowyer, and D. Werthimer.

[39] Sunderam V.S., PVM: A Framework for Parallel Distributed Computing, Concurrency, Practice and Experience, 1990.

MAURO MIGLIARDI è nato a Genova nel 1966. Dopo esser stato uno dei principali ricercatori del progetto HARNESS per il meta-computing presso la Emory University di Atlanta, è ora ricercatore universitario presso l'Università di Genova. I suoi principali interessi di ricerca sono il meta-computing e il Grid-computing.
om@dist.unige.it