Modelli di calcolo per algoritmi distribuiti

Corso di: Algoritmi paralleli e distribuiti
Prof.ssa: Rossella Petreschi
Anno accademico: 2004/2005
Autore: Danilo Carrabino

1. Eventi, ordini, e stati globali

L'entità matematica capace di rappresentare in modo astratto qualunque sistema distribuito è il grafo, dove i nodi rappresentano i nodi del sistema e gli archi rappresentano la rete di intercomunicazione.
Si vuole fornire un formalismo che riesca ad essere il più generale possibile, in modo tale da inglobare qualsiasi aspetto dei sistemi distribuiti sia in presenza di modelli sincroni, sia in presenza di modelli asincroni.
Il grafo G può rappresentare qualsiasi sistema distribuito e quindi si adatta perfettamente a descrivere la caratteristica fondamentale dei sistemi distribuiti che è lo scambio di messaggi.

Possiamo definire un *sistema sincrono* come una discretizzazione in base al tempo di un *sistema asincrono*. Infatti in un sistema sincrono c'è un clock globale che scandisce il tempo come un metronomo e che regola le azioni compiute dai nodi del sistema. Inoltre assumiamo che le computazioni a livello locale impiegano tempo zero all'interno di un intervallo dell'orologio globale.

In questo paragrafo la discussione ruota intorno al concetto di evento, ed è intesa in modo particolare alla descrizione delle computazioni che avvengono nel modello asincrono (cioè, esecuzioni di algoritmi asincroni).
In ogni modo, le condizioni sotto le quali viene definito il modello sincrono possono essere considerate un caso particolare delle condizioni per il modello asincrono, e perciò ciò che verrà detto sarà applicabile allo stesso modo ai modelli sincroni.

Eventi.

Con il concetto di evento intendiamo *l'unità fondamentale di una computazione distribuita,* che di volta in volta è l'esecuzione di un algoritmo distribuito.
Detto ciò una computazione distribuita è vista semplicemente come *un insieme di eventi,* che indichiamo con Ξ.
Un evento ξ è la 6-tupla

$$\xi = <n_i, t, \varphi, \sigma, \sigma', \Phi>,$$

dove

- n_i è il nodo in cui avviene l'evento;
- t è il tempo in cui avviene l'evento, dato dall'orologio locale di n_i;
- φ è il messaggio, se esiste, che ha scatenato l'evento in seguito alla sua ricezione da parte di n_i;
- σ è lo stato di n_i prima dell'avvenimento dell'evento;
- σ' è lo stato di n_i dopo l'avvenimento dell'evento;

- Φ è l'insieme di messaggi, se esistono, spediti da n_i come conseguenza dell'avvenimento dell'evento.

Questa definizione di evento è basata sulla premessa che il comportamento di ogni nodo durante la computazione distribuita può essere descritto come quello di una macchina a stati.
La computazione Ξ (l'insieme degli eventi) fa evolvere lo stato di ogni nodo ogni volta che si verificano degli eventi. Indichiamo con Σ_i la sequenza degli stati attraverso i quali passa n_i durante la computazione distribuita. Perciò il primo elemento di Σ_i è *lo stato iniziale* di n_i e l'ultimo elemento (che può non esistere se Ξ è non finito) è *lo stato finale* di n_i.

Inoltre questa definizione è abbastanza generale sia per comprendere il carattere reattivo delle computazioni distribuite (relativo quindi ad *eventi esterni*), sia per permettere la descrizione di *eventi interni*, che avvengono senza un'immediata *causa esterna* (causa esterna intesa come la ricezione di un messaggio o la spontanea iniziazione da parte dei nodi, che in definitiva può anche essere considerata come generata esternamente).
Quindi, al fine di poter descrivere sia eventi interni che eventi esterni (tra i quali l'iniziazione spontanea da parte dei nodi), abbiamo permesso che il messaggio di input φ associato ad un evento potesse anche essere assente.

Analizziamo il caso degli *algoritmi sincroni*.
Come già detto, le definizioni date fino ad ora sono essenzialmente valide allo stesso modo anche per gli algoritmi sincroni, ma vanno specificate alcune caratteristiche.
Poiché nel caso sincrono ci aiuta assumere che la computazione a livello locale in un intervallo dell'orologio globale impiega tempo zero, e poiché il comportamento dei nodi è "dettato" dall'orologio globale e non dalla ricezione di messaggi, possiamo assumere che ad ogni nodo avvenga esattamente un evento ad ogni pulsazione, con t inteso come multiplo della durata dell'intervallo.
Un evento tale non ha un messaggio di input ad esso associato (infatti il messaggio di input nella nostra definizione può non esistere), poiché, per assunzione, nel modello sincrono ogni messaggio è recapitato in un tempo minore della durata di un intervallo.
Oltre a questi ci possono anche essere eventi corrispondenti soltanto alla ricezione di messaggi.
Non ha senso, quindi, parlare di eventi interni, poiché ogni evento o ha un messaggio di input ad esso associato, o avviene in risposta ad un impulso, e in entrambi i casi c'è una causa esterna.

Ordini.

Gli eventi in Ξ sono fortemente intercorrelati, poiché i messaggi che un nodo spedisce in connessione con un evento sono ricevuti dal vicino di quel nodo in connessione con un altro evento.
Definiamo la relazione binaria denotata con $\prec$, sull'insieme degli eventi Ξ, come: se ξ_1 e ξ_2 sono eventi, allora $\xi_1 \prec \xi_2$ se e solo se vale una delle seguenti due condizioni:

(i) Sia ξ_1 che ξ_2 avvengono nello stesso nodo in tempi t_1 e t_2 ,tali che $t_1 < t_2$ e non accade nessun altro evento t tale che $t_1 < t < t_2$.
(ii) Gli eventi ξ_1 e ξ_2 avvengono in nodi vicini, ed esiste un messaggio φ spedito in connessione con ξ_1 e ricevuto in connessione con ξ_2.

La relazione appena descritta è aciclica. La prima condizione esprime la causalità tra gli eventi che avvengono nello stesso nodo, e la seconda fornisce la relazione causa-effetto di base che esiste tra nodi vicini.
La chiusura transitiva di $\prec$, $\prec^+$, è anti-riflessiva e transitiva, e quindi stabilisce *un ordine parziale* sull'insieme degli eventi Ξ.

Due eventi ξ_1 e ξ_2 sono definiti *eventi concorrenti* se non sono correlati da $\prec^+$ in modo che:

$$(\xi_1 , \xi_2) \in \Xi \text{ x } \Xi - \prec^+$$

$$(\xi_2 , \xi_1) \in \Xi \text{ x } \Xi - \prec^+.$$

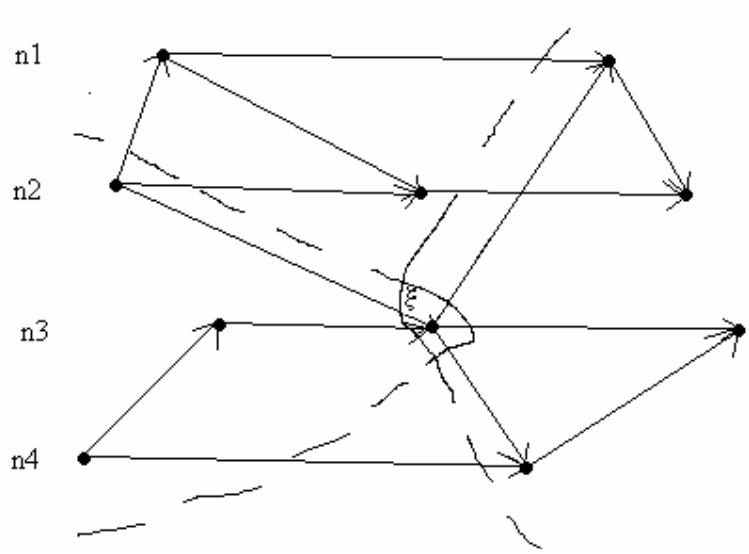
Con il termine eventi concorrenti, intendiamo eventi che non hanno alcun rapporto di causalità.

Il passato e il futuro di un evento possono essere espressi dalle seguenti relazioni:

$$Past(\xi) = \{\xi' | \xi' \in \Xi \text{ e } \xi' \prec^+ \xi\}$$

$$Future(\xi) = \{\xi' | \xi' \in \Xi \text{ e } \xi \prec^+ \xi'\}$$

Consideriamo il grafo delle precedenze, ovvero il grafo diretto aciclico $H = (\Xi, \prec)$, il cui insieme di nodi è l'insieme degli eventi e l'insieme degli archi è dato dalle coppie di eventi in relazione di $\prec$.
Gli archi orizzontali corrispondono alle coppie di eventi che verificano la condizione (i) e rappresentano lo stato dei nodi. Tutti i rimanenti archi sono di tipo (ii) e rappresentano i messaggi spediti tra i nodi.



Ordine parziale e ordine totale.
Abbiamo bisogno di estendere l'ordine parziale $\prec^+$,definito sull'insieme Ξ degli eventi della computazione distribuita, per ottenere un ordine totale consistente con $\prec^+$, cioè un ordine parziale che include esattamente una coppia (ξ_1 , ξ_2) o (ξ_2 , ξ_1) per ogni $\xi_1 , \xi_2 \in \Xi$.
Tale ordine totale così costruito non contraddice $\prec^+$, nel senso che contiene tutte le coppie di eventi già presenti in $\prec^+$ alle quali abbiamo aggiunto coppie di eventi concorrenti, cioè non correlate tramite la relazione $\prec^+$.

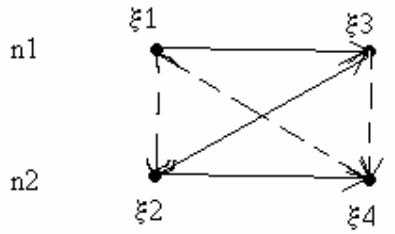
Dato un qualsiasi ordine totale < su Ξ, possiamo identificare $|\Xi| - 1$ coppie (ξ_1 , ξ_2) di *eventi consecutivi* tali che per ogni evento $\xi \neq \xi_1, \xi_2$ o $\xi < \xi_1$, o $\xi_2 < \xi$.

Stati globali.

Adesso si vuole puntare l'attenzione sugli aspetti riguardanti la misurazione globale in una computazione distribuita. Cioè vogliamo introdurre il concetto di *stato globale* o *snapshot* (fotografia della computazione distribuita).
Si vogliono fornire due definizioni di stato globale basate sul concetto di *stato del sistema*.
Lo **stato del sistema,** denotato con Φ_{ij}, è un insieme di n stati nodo(uno stato nodo per ogni nodo del sistema), dove lo stato del nodo n_i (σ_i)è estratto dalla sequenza degli stati che il nodo stesso attraversa, ed uno stato arco per ogni arco del sistema (m se il grafo è diretto, 2m se il grafo è indiretto), in cui lo stato di un arco $(n_i \rightarrow n_j)$ è l'insieme di messaggi che sono in transito da n_i a n_j in uno stato del sistema.

Riprendiamo la definizione di eventi consecutivi, perché associato ad ogni coppia (ξ_1 , ξ_2) di eventi consecutivi in <, c'è uno stato del sistema, *system_state*(ξ_1 ,ξ_2), con le seguenti caratteristiche:

- Per ogni nodo n_i , σ_i è lo stato risultante dall'avvenimento dell'evento ξ più recente in n_i ,tale che $\xi_1 \preccurlyeq \xi$ (cioè ξ può essere un evento che precede ξ_1 , oppure un evento concorrente, o anche $\xi_1 = \xi$).
- Per ogni arco $(n_i \rightarrow n_j)$, Φ_{ij} è l'insieme dei messaggi mandati in connessione con un evento ξ tale che $\xi_1 \preccurlyeq \xi$ e ricevuti in connessione con un evento ξ' tale che $\xi' \preccurlyeq \xi_2$.



Le frecce tratteggiate indicano le coppie di eventi concorrenti.
system_state(ξ_2 ,ξ_3) è tale che n_1 è nello stato in cui è stato lasciato all'occorrenza di ξ_1, n_2 è nello stato in cui è stato lasciato all'occorrenza di ξ_2, e un messaggio mandato in connessione con ξ_2 è in transito sull'arco da n_2 a n_1 per essere ricevuto in connessione con ξ_3. Quindi *system_state*(ξ_2 ,ξ_3) è uno stato globale.

Prima definizione di stato globale: uno stato del sistema Ψ è uno stato globale se e solo se o in Ψ tutti i nodi sono nel loro stato iniziale (e quindi tutti gli archi sono vuoti), o tutti i nodi sono nel loro stato finale (e tutti gli archi sono vuoti), o esiste un ordine totale consistente con $\prec^+$, in cui esiste una coppia di eventi consecutivi (ξ_1 ,ξ_2) tale che Ψ = *system_state*(ξ_1 ,ξ_2).

Seconda definizione di stato globale: si considera una partizione dell'insieme degli eventi Ξ in due sottoinsiemi Ξ_1 e Ξ_2 .
Definiamo *system_state*(Ξ_1 ,Ξ_2) il sistema in cui σ_i è lo stato in cui n_i è stato lasciato all'evento più recente di Ξ_1 avvenuto in n_i , e Φ_{ij} è l'insieme dei messaggi spediti su $(n_i \rightarrow n_j)$ in connessione con gli eventi in Ξ_1 e ricevuti in connessione con gli eventi in Ξ_2 .
Uno stato del sistema Ψ è uno stato globale se e solo se Ψ = *system_state*(Ξ_1 ,Ξ_2) per qualche partizione (Ξ_1 ,Ξ_2) di Ξ tale che:

$$\text{Past}(\xi) \subseteq \Xi_1 \text{ per ogni } \xi \in \Xi_1$$

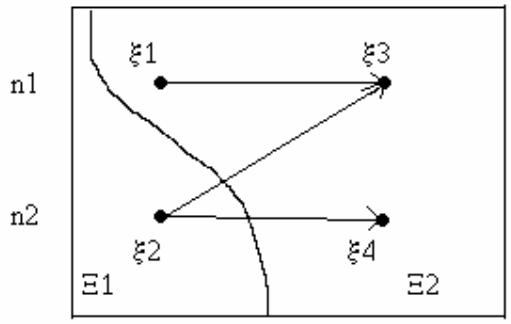
e in modo equivalente

$$\text{Future}(\xi) \subseteq \Xi_2 \text{ per ogni } \xi \in \Xi_2$$

Il caso in cui tutti i nodi sono nello stato iniziale è descritto da $\Xi_1 = \varnothing$ e il caso in cui tutti i nodi sono nello stato finale è descritto da $\Xi_2 = \varnothing$.

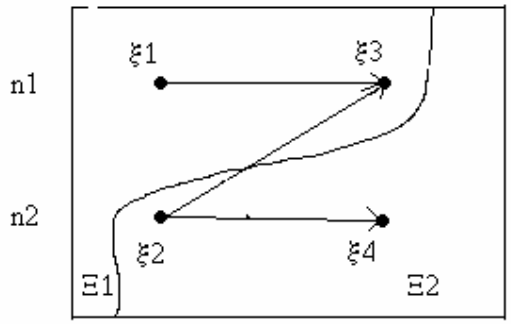
Queste due definizioni di stato globale sono tra loro equivalenti.
Riprendendo il concetto di grafo H delle precedenze, la partizione (Ξ_1, Ξ_2) induce in H un taglio (un insieme di archi) comprendente archi che vanno da eventi in Ξ_1 a eventi in Ξ_2 e archi che fanno il percorso inverso.

Def: un taglio non contiene alcun arco da Ξ_2 a Ξ_1 se e solo se *system_state*(Ξ_1, Ξ_2) è uno stato globale.



(a)

Poiché il taglio in (a) non contiene alcun arco da Ξ_2 a Ξ_1 allora *system_state*(Ξ_1, Ξ_2) è uno stato globale.



(b)

Poiché invece il taglio in (b) contiene un arco che va da Ξ_2 a Ξ_1 allora *system_state*(Ξ_1, Ξ_2) non può essere uno stato globale.

Le definizioni di stato globale si applicano sia a sistemi sincroni, sia a sistemi asincroni perché nella definizione non c'è nulla che preclude la possibilità che uno stato globale possa essere descritto in associazione col valore dell'orologio globale.

Analizziamo le definizioni di passato e futuro di uno stato globale.
Dato Ψ stato globale definiamo:

$$Past(\Psi) = \bigcup_{\xi \in \Xi_1} [\{\xi\} \cup Past(\xi)] = \Xi_1$$

$$Future(\Psi) = \bigcup_{\xi \in \Xi_2} [\{\xi\} \cup Future(\xi)] = \Xi_2$$

Quindi, per concludere, diciamo che uno stato globale Ψ_1 viene prima nella computazione Ξ rispetto ad uno stato globale Ψ_2 se e solo se Past(Ψ_1) $\subset$ Past(Ψ_2).

2. Complessità delle computazioni distribuite

Analizzare la complessità di qualsiasi computazione è un modo per esprimere quantitativamente l'uso che la computazione stessa fa delle risorse di cui ha bisogno per essere portata a termine.
In base al tipo di computazione le risorse possono includere il numero di cicli del processore, il numero dei processori, il numero dei messaggi spediti e altre quantità.
E' quindi importante determinare quali sono le risorse sulle quali definire appropriate misure di complessità per una computazione.

Ad es. per gli algoritmi sequenziali la risorsa fondamentale è il tempo (a volte è anche lo spazio), mentre per le computazioni parallele continua ad essere importante il tempo, ma diventa di fondamentale importanza anche il numero di processori sui quali vengono lanciati gli algoritmi.

La complessità degli algoritmi distribuiti è basata sull'assunzione che le due risorse fondamentali, su cui effettuare misurazioni, sono la *comunicazione*(lo scambio di messaggi) e il *tempo*.
Le misure della complessità sono espresse al caso peggiore, nello stile asintotico, come funzioni di n ed m, i nodi e gli archi di G.
Un buon punto di partenza per stabilire misure di complessità degli algoritmi distribuiti è quello di considerare la comunicazione come la risorsa predominante. Ciò non significa che la complessità temporale cessa di essere rilevante, ma dovrebbe esser preso in considerazione soltanto il tempo direttamente correlato allo scambio dei messaggi.

Complessità temporale e di comunicazione per sistemi sincroni e asincroni.
Se si considera come risorsa dominante la *comunicazione*, allora la complessità di un algoritmo distribuito è espressa con due misure.

1. La prima misura è la *complessità dei messaggi*, che è data dal numero di messaggi spediti tra nodi vicini durante la computazione nel caso peggiore, cioè il massimo numero di messaggi spediti quando sono considerate tutte le

possibili variazioni (potrebbe anche non essere possibile eseguire variazioni) sulla struttura di G, così come tutte le possibili esecuzioni dell'algoritmo. Una misura più accurata, la *bit complessità*, può essere utile per calcolare differenze rilevanti tra gli algoritmi quando le lunghezze dei messaggi dipendono da n ed m.

2. La seconda misura attraverso cui possiamo esprimere la complessità dell'algoritmo è la *complessità temporale*, che è data dal tempo speso per la comunicazione durante la computazione nel caso peggiore (massimo su tutte le possibili strutture di G e su tutte le esecuzioni dell'algoritmo).

Modello sincrono.
Per il modello sincrono, l'assunzione che il costo della comunicazione è quello dominante si adatta perfettamente, perché avevamo assunto che la computazione locale costava tempo zero.
La complessità nel modello sincrono si riduce essenzialmente a contare il numero di pulsazioni che trascorrono durante la computazione. Potrebbe sembrare una contraddizione il fatto che vengano contate anche quelle pulsazioni che passano senza che ci sia comunicazione.
Ma nel modello sincrono i messaggi sono tanto importanti quanto la loro assenza, poiché abbiamo assunto che, quando un messaggio viene spedito da un nodo ad un suo vicino, questo arriva sempre in tempo costante. Quindi si può ottenere informazione anche dall'assenza di comunicazione.

Modello asincrono.
Assumiamo, come per il modello sincrono, che la computazione locale impieghi tempo zero, e che il tempo per comunicare un messaggio ad un sottoinsieme dell'insieme dei nodi vicini sia O(1).
La complessità temporale sarà data quindi dal numero di messaggi nella più lunga catena causale della forma "ricevi un messaggio e spedisci un messaggio come conseguenza", calcolata su tutte le possibili esecuzioni dell'algoritmo e su tutte le possibili strutture di G.

2 osservazioni:

1. la complessità temporale non può mai essere maggiore della complessità di messaggi, perché ogni messaggio contato per la prima misura è anche contato per la seconda. L'utilità della complessità temporale è che considera soltanto i messaggi che accadono "in modo sequenziale" uno dopo l'altro, cioè messaggi che sono causalmente legati l'uno all'altro. Quindi per il caso asincrono la complessità temporale potrebbe essere ottenuta dalla complessità di messaggi tagliando via i messaggi che sono "concorrenti" a quelli nella più lunga catena causale ricezione-spedizione.

2. l'assunzione di O(1) per la spedizione di messaggi ai nodi vicini è valida se ogni messaggio ha a sua volta lunghezza O(1), ma comunque è già compito della bit complessità catturare la lunghezza variabile dei messaggi.

Riprendiamo il grafo H delle precedenze che riassume le dipendenze causali tra gli eventi nella computazione.
Etichettiamo tutti gli archi di H che corrispondono a messaggi con 1, gli altri con 0.
Questo riflette le assunzioni fatte, cioè un messaggio impiega tempo costante ad essere spedito, e le computazioni locali hanno costo zero.
Fissato il grafo G e il grafo H, la complessità temporale sarà la lunghezza del più lungo cammino diretto in H, dove la lunghezza di ogni arco è proprio la sua etichetta.
Prendendo il valore massimo su tutte le possibili variazioni di G e su tutte le possibili esecuzioni dell'algoritmo otteniamo il risultato voluto.

Misure locali e globali.
In alcuni casi l'assunzione che la computazione locale viene eseguita in tempo zero è troppo forte. Quindi si introduce una nuova misura chiamata complessità temporale locale per calcolare la computazione locale quando è stato ricevuto un messaggio.

Confronto tra modelli sincroni e modelli asincroni.

Il modello sincrono come detto è un caso particolare del modello asincrono in cui viene discretizzato il tempo tramite un clock globale.
Questa è una caratteristica fondamentale che rende il modello sincrono molto più forte rispetto a quello asincrono.
Se analizziamo il problema del consenso, sappiamo che nel caso sincrono il problema è risolvibile sia per guasti bizantini che per guasti fail-stop, sotto opportune ipotesi, mentre nel caso asincrono il problema è non risolvibile anche se si verifica un solo guasto(in questo caso è importante notare la potenza degli algoritmi probabilistici che invece rendono possibile la soluzione anche nel caso asincrono).

Ogni algoritmo che risolve un problema per il caso asincrono può essere trasformato in un algoritmo per sistemi sincroni senza modificare né la complessità temporale, né la complessità di messaggi.
Infatti un algoritmo costruito per sistemi asincroni, gira su tutte le possibili variazioni del tempo, e quindi in particolare nella variazione che coincide col modello sincrono.
Molto meno banale è invece trasformare un algoritmo costruito per il modello sincrono, in un algoritmo per il modello asincrono.
Intanto bisogna supporre che nel sistema non ci siano guasti di alcun tipo.
Inoltre nel caso asincrono ci sarà una comunicazione tra i nodi molto più elevata perché vengono a cadere delle importanti assunzioni sul tempo e quindi aumenteranno sia la complessità temporale che quella di messaggi.

2 conclusioni importanti sono quindi: il modello sincrono possiede le caratteristiche che permettono agli algoritmi sincroni di ottenere migliori performance rispetto agli algoritmi asincroni riguardo la complessità temporale e di messaggi.
L'interesse principale nel modello sincrono deriva dalla possibilità di ottenere alla fine un algoritmo per sistemi asincroni da un algoritmo originariamente progettato per sistemi sincroni.

Bibliografia.

Il materiale nel paragrafo 1 "Eventi, ordini, e stati globali" è basato su Lamport (1978) e su Chandy e Lamport (1985), e il concetto di stato globale può essere reperito in Bracha e Toueg (1984).
La maggior parte delle misure di complessità introdotte nel paragrafo 2 "Complessità delle computazioni distribuite" sono degli standard, e possono essere trovati in Lamport e Lynch (1990).