DISTRIBUTED SYSTEMS

Insieme di processori connessi in rete.

Vantaggi

- Scalabilità
- Modularità ed eterogeneità
- Condivisione dati e risorse
- Struttura geografica
- Basso costo

Caratteristiche

- Assenza di clock condiviso
- Assenza di memoria condivisa
- Assenza di rilevatore di errori

Teorema CAP

Qualsiasi sistema connesso in rete con dati condivisi deve avere almeno 2 di queste 3 proprietà:

- (C) **Consistency**: tutti i dati nei vari nodi del sistema non presentano incongruenze. Se un dato è stato aggiornato in un nodo, tutti gli altri dovranno essere aggiornati immediatamente.
- (A) **Availability**: a qualunque richiesta deve corrispondere una risposta (costante disponibilità di dati), anche se parte del sistema non è funzionante.
- (P) **Partition tolerance**: il sistema deve continuare a funzionare anche se parte di esso non è funzionante.

Latenza

Ritardo nella comunicazione. Nel teorema CAP viene apparentemente ignorata, ma influenza molto la **partition tolerance**.

Problemi e limiti della computazione distribuita

- Programmazione OOP
- Programmazione procedurale

Modello di computazione distribuita

Consiste in un **sistema asincrono** composto da un insieme di processi, i quali:

- Sono composti a loro volta da stati ed eventi che alterano gli stati stessi.
- Inviano messaggi attraverso canali unidirezionali, con buffer infiniti e privi di errori.
- Ritardi arbitrari (ma finiti)
- Nessuna assunzione sull'ordinamento dei messaggi.

Modelli utilizzati - Interleaving

Si assume ordinamento totale tra gli eventi (tempo fisico).

Modelli utilizzati - Happened before

Si assume ordinamento parziale tra gli eventi (tempo logico).

- **Causalità diretta**: se ($e \lessdot f$) oppure ($e \rightsquigarrow f$), allora $e \rightarrow f$:
  - ($e \lessdot f$) vuol dire che e avviene prima di f, nello stesso processo.
    - **Esempio**:
  - ($e \rightsquigarrow f$) vuol dire che e avviene prima di f, in processi diversi, e comunicano tramite messaggi.
    - **Esempio**: e è l'invio di un messaggio nel processo P1, f è la ricezione del messaggio nel processo P2.
- **Transitività**: se esiste g tale che ($e \rightarrow g$) e ($g \rightarrow f$), allora $e \rightarrow f$.

Modelli utilizzati - Happened before - Meccanismi implementativi

- Logical clocks: una funzione C di un processo Pi assegna un numero $Ci(a)$ ad ogni evento a, secondo la **clock condition**:
  - Se ($a \rightarrow b$) allora ($Ci(a) \rightarrow Cj(b)$), con a evento del processo Pi e b evento del processo Pj. E' inoltre sempre vero che se ($a \rightarrow b$) allora ($Ci(a) < Cj(b)$), quindi se un evento è happened before un altro, allora il valore del clock logico del primo evento è minore del secondo. Ma **non è vero il contrario**
- Vector clock: risolvono la limitazione dell'affermazione precedente. Un vettore VC, di dimensione uguale al numero di processi considerati, viene assegnato ad ogni evento.
  - ($a \rightarrow b$) $\iff$ ($VC(a) < VC(b)$)

Modelli utilizzati - Potential causality

L'idea di base è che non tutti gli eventi di un sistema concorrente siano direttamente correlati tra loro. Alcuni di essi possono essere ad esempio causalmente in relazione oppure indipendenti/concorrenti.

- Concorrenza tra eventi: ($e \parallel f$) = not ($e \rightarrow f$) and not ($f \rightarrow e$).