Algoritmi distribuiti

Thursday, 18 May 2023 08:38

- Sincronizzazione
  - Orologi fisici
    Se ognuno avesse un orologio, allora saremmo sincronizzati... Più o meno
    Siccome con 216.000 tic/ora
    Abbiamo un errore di $\pm$ 2 tic/ora
    Per via di questo errore, abbiamo problemi di sincronizzazione.
    Un modo per sistemare è un l'algoritmo di lamport's
    Perché? Senò i database, es, non sono aggiornati.
  - Orologi logici
    Il client manda un messaggio a molti
    Chi riceve lo mette in coda con timestamp associato
    Tutti inviano un ack a tutti con timestamp
    Ed una volta che tutti hanno l'ack, si può processare.
    Ogni ack ha un numero, e questo numero incrementa. Se notiamo che qualcuno ha un numero più grande di noi, allora aggiorniamo
- Mutua esclusione
  - Algoritmo centralizzato
    Abbiamo 1 che gestisce, ed ognuno richiede. Quando è libero, risponde, senò si aspetta.
    Commenti:
    - Non ci sono deadlock
    - 3 messaggi (request, grant, release)
    - Singolo punto failure (failure)
    - Coordiantore è un bootleneck = tutti devono passare per di li... Come può es
    - Si suppone safety
  - Algoritmo distribuito
    - SI suppone un ordinamento totale tra gli eventi (algoritmo di lamport, quello con gli ack)
    - "Devo andare in bagno, nessuno è in bagno?! "IO NO", "IO NO", ""

      L'ultimo non ha risposto, vuol dire che è in bagno
      Se io voglio andare prima, non rispondo. Così vado prima io e tu aspetti

aspetti
Qui si decide chi va prima con gli timestamp

Commenti:

- No starvation/deadlock
- 2(n-1) messaggi
- N nodi fallimento
  E se un nodo fallisce, rimane sempre dentro. Per prevenire, si aggiunge un tempo limite
- Conoscere partecipanti, ed lento performance

- Algoritmo token ring
  Tutti i processi si passano un token, e chi ha il token possiede la risorsa condivisa.
  Se non mi serve la risorsa, passi il token come se stesse giocando a "patata bollente"
  Vantaggi:
  - Semplice
  - No starvation
  - No deadlock

  Svantaggi:
  - Se X fallisce, la patata esplode.

- Sistema con coordinatore
  Abbiamo 1 algoritmo: algoritmi di elezione
  Praticamente, noi stiamo giocando a dnd, e dobbiamo scegliere chi mastera
  - L'algoritmo del bullo
    Chi ha l'id più grande, aka chi è più nuovo, diventa il coordinatore.
    Ed una volta che si comprende chi è il più grande, il coordinatore invia a tutti "io sono il coordinatore"
  - L'algoritmo dell'anello
    Si suppone che ogni processo è fisicamento ordinato in un anello
    Ed esso avviene inviando il proprio nome a tutti ed ognuno fà conoscere agli altri i propri vicini
    Se vedo che c'è il mio numero, io sono il coordinatore.
    Se un nodo non mi risponde, YEEET invio il messaggio al prossimo

- Tolleranza ai guasti
  - Definizione del problema
    - Disponibilità
      Pronto ad essere usato immediatamente (in questo momento è disponibile)
    - Reliability
      Un periodo di tempo in cui è disponibile (99.9% all'anno)
    - Safety

safe)
La situazione dove il sistema funziona correttamente in casi normali, e se succedono errori niente è catastrofico
- Manteneibilità
Il sistema è capace di autoripararsi (o che è facile da riparare)

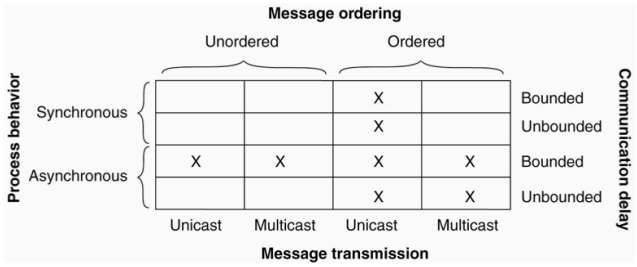
Diverse tipologie di fallimento:
- Crash failure, il server si interrompe ma la lavora correttamente fino all'interrizione
- Omission failure (mancanza invio/ricezione)
- AAAAAAAAAAA il prof ha cambiato troppo velocemente
Non ti preoccupare ale del passato! Io, ale del futuro risolvo tutto!
- TIming failure (la risposta del server è dopo a quando il client voleva)
- Response failure (La risposta del server è incorretta)
  - Valore
  - Stato (aka risponde con dei dati precedenti/di dopo)
- Il server potrebbe produrre errori casuali (arbitrari)

- Possibili soluzioni
  - Ridondanza, cioè se uno si rompe, chissene ho altri nodi!
  Le relazioni poliamorose funzionano così! Se la prima relaziona va male, ho la seconda relazione! (scherzo) (sono molto annoiato)
  [OK ALE DEL PASSATO MI HAI FATTO RIDERE]
  E quindi abbiamo 2 tipologie di gruppi:
    - Gruppi piatti (tutti conoscono tutti)
    - Gruppo a gerarchia (1 conosce tutti, gli altri conosce 1)
  [Non ho più concentrazione, pg. 35]
  [L'ale del futuro risolve tutto! Dandandan!]
  Prima di tutto bisogna comprendere che c'è stato un errore.
  Per farlo dobbiamo prima definire:
    - Dobbiamo definire se il server è sincrono oppure asincrono.
    Il server sincrono è quando tutti devono essere sincronizzati verso un valore c.
    Per ogni azione c incrementa, e chi ha un c inferiore al c di tutti, deve riprendersi.
    "Abbiamo 3 computer, ognuno ha un numero che diciamo essere 10.
    Io computer A voglio fare un operazione. Dico agli altri che voglio fare un operazione, faccio l'operazione, e poi tutti incrementano il proprio contatore a 11.
    Mettiamo caso entra computer C dopo un blackout, lui chiede agli altri "cosa avete fatto? Io sono all'operazione

numero 10" e loro lo aggiornano con l'operazione numero 11" -Io sul gruppo whatsapp

- La comunicazione è bounded, e lo sappiamo se c'è un time to live nei messaggi
- I messaggi sono ordinati oppure no
- La trasmissione del messaggio è fatto con unicast oppure multicast

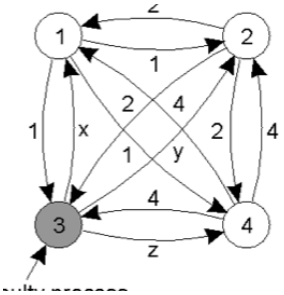
A seconda di queste affermazioni, possiamo definire se c'è stato un errore attraverso la seguente tabella:



In tutti i casi dove non abbiamo una x, per comprendere se c'è un errore bisogna seguire il "problema dei 3 reali ed 1 traditore"

Step 1: The generals announce their troop strengths (in units of 1 kilosoldiers) – (a)
Step 2: Results are collected by each general in vectors – (b)
Step 3: The vectors are sent to every other generals – (c)
Step 4: Each general examines the *ith* element of each of the received vectors
If any value has a majority, that value or UNKOWN is put into the result vector

In parole povere



1 Got(1, 2, x, 4)
2 Got(1, 2, y, 4)
3 Got(1, 2, 3, 4)
4 Got(1, 2, z, 4)

1 Got	2 Got	4 Got
(1, 2, y, 4)	(1, 2, x, 4)	(1, 2, x, 4)
(a, b, c, d)	(e, f, g, h)	(1, 2, y, 4)
(1, 2, z, 4)	(1, 2, z, 4)	(i, j, k, l)

Ogni nodo si scambia un messaggio a tutti, e poi confrontano ciò che è stato ricevuto. Sanno del traditore controllando il risultato; notano che il nodo 3 ha inviato risultati sballati.

Safe/Safety: tutti gli attori sono onesti e nessuno vuole violare la mutua esclusione
Security: si suppone che gli attori potrebbero non essere onesti