Distribuované algoritmy

<small>Z ČWUT</small>

Obsah

Asymetrické a symetrické algoritmy, metody interakce procesů

- asymetrické
    - procesy spolu pouze komunikují
    - spolupráce se omezuje na předávání výsledků
- symetrické - všechny prvky programu jsou stejné
    - procesy jsou si podobné, výsledek je produktem spolupráce
    - symetrický multiprocesing
    - difuzní algoritmy (hard-beat)
        - periodický charakter: proces dostane vstup od sousedů, vypočte a pošle výsledek sousedům
        - např. Floyd - Fulkerson
    - typy symetrie
        - strict - stejné chování i kód
        - text - stejný kód

        - slabá - procesy mají delší čas odlišné chování (token - pasing)

- interakce procesů
    - systém Helios - jednoduché mapování programů na procesy: programy si vyměňují data přes standardní vstup a výstup
    - broadcast
    - RPC
    - sdílená paměť
        - central server
        - migration - po požadavku stanice na proměnnou tato putuje na stanici (migruje)
        - read-replication - hodnotu si každá stanice po prvním přečtení čte i modifikuje u sebe (cache), při změně hodnoty invalidace
        - full replication - kopii má každá stanice, modifikace se distribuuje všude

Kauzalita v distribuovaném systému, logický čas (Lamportovy hodiny)

- zprávy mohou chodit v různém pořadí
- kauzální uspořádání - je vidět, jak mohly události proběhnout - řídí se vztahem mezi nimi
    - úplné - navíc všechna pozorování stejná - s přenášenými informacemi se přenáší inf. o kauzálních vazbách - logický čas
    - je zavedena relace *a* předchází *b*, značíme $a \rightarrow b$:
        - *a* a *b* jsou události stejného procesu, *a* nastala před *b* nebo
        - *a* je odesláním zprávy jedním procesem, *b* je příjem této zprávy jiným procesem nebo
        - existuje *c* tak, že *a* předchází *c* a *c* předchází *b*
        - nejsou-li *a* a *b* v uvedené relaci, jsou souběžné

Lamportovy hodiny (skalární logický čas)

Pokud událost *a* předchází události *b*, pak lokální čas procesu Pi odpovídající události *a* musí být menší než lokální čas procesu Pj odpovídající události *b*.

Podmínku lze nejjednodušeji splnit zavedením čítače v každém procesu. Čítač bude inkrementován mezi každými dvěma událostmi příslušného procesu Pi. Čítače různých procesů Pi a Pj se budou vzájemně synchronizovat tak, že při každém:

1. odeslání se časové razítko nastaví na hodnotu vlastního časovače
2. příjmu se nastaví časové razítko příchozí události na maximum z příchozího a vlastního časovače, navíc se synchronizují vlastní hodiny (ne dozadu)

- čítače procesů se inicializují náhodně
- inicializují-li se shodně - pak úplné uspořádání podle čísel procesů

Vektorový logický čas

- lokální čítač - vektor
    - počet složek jako procesů
    - proces inkrementuje pouze svou položku

Algoritmy výlučného přístupu a výběru

Algoritmy výlučného přístupu

distribuovaný semafor (Lamport)

- proces chce vstoupit do kritické sekce - pošle ostatním požadavek, ostatní pošlou potvrzení

- po vystoupení z kritické sekce se všem pošle zpráva
- zprávy nesou časové známky
- podmínky
    - komunikační kanály tvoří úplný graf
    - zachování pořadí zpráv
- proces si požadavky ukládá do seřazené fronty (podle čas. razítek)
    - je-li jeho záznam na řadě - může do kritické sekce
- 3(n-1) zpráv

Ricart - Agarwala

- na každý požadavek na vstup do kritické sekce proces odpoví buď hned nebo odpověď pozdží až po vykonání svého požadavku
- proces může rozhodnout, kterým požadavkům má dát přednost před vlastním
- do 2(n-1) zpráv
- modifikace negativní potvrzování
    - v sítích se zaručeným časem doručení
    - rovnou souhlas - mlčení
    - pozdržovací odpověď pouze jednobitová, plná odpověď až jako souhlas
- kde to efektivně dovoluje přenosové médium - broadcast

Carvalho - Roucairol

- udržuje přehled o poskytnutých pověřeních
- do 2(n-1) zpráv
- proces může, pokud nebylo starší pověření

Ricart - Agarwala (Token Passing)

- předávání jediného globálního pověření
- registrují se požadavky ostatních

~~* dostane-li proces peška a existuje-li starší žádost - pešek se pošle, jinak proces může vstoupit~~

- Procesy si předávají pověření (TOKEN). Požaduji-li prostředek, odešlu REQUEST a čekám na TOKEN. Až jej obdržím mohu vstoupit do kritické sekce. Po vystoupení odešlu dalšímu kdo mě žádal.

Logický kruh - cyklické předávání pověření

Klient / server s distribuovaným výběrem serveru

Algoritmy výběru

Například mám centrální řízení. Otázka, jak zvolit, kdo bude řídit:

- všichni vysílají, dostane-li proces zprávu s vyšším ID, přestane a pošle odpověd s nejvyšším ID, které dostal
    - poslední vysílající, který dostane svoje ID, se stává serverem
    - až $2N^2$ zpráv
- Chang - Roberts
    - modifikace předchozího
    - odesílání zpráv po logickém kruhu
    - $0{,}5N(N\text{-}1)$
- Hirschberg - Sinclair
    - jako Chang - Roberts, ale posílání v obou směrech

Zablokování při sdílení a komunikaci, detekce a prevence

K deadlocku může dojít splněním:

- použití prostředku je výlučné (ne více procesů)
- proces může používat prostředek v době žádosti o jiný
- přidělený prostředek nelze procesu násilně odejmout
- v grafu závislosti může vzniknout cyklus

Sdílení

Lometovy algoritmy - předcházení

- apriorní metoda
- prostředky mají pořadí, procesy o ně mohou žádat jen v tomto pořadí
- požadavky o přidělení se opatřují časovými razítky
- transakce - oblast výpočtu, kde proces může používat 1 nebo více než 1 prostředek (od místa, kde používá první do místa, kde uvolňuje poslední)
- vstup do transakce - potenciální žádost o prostředky
- takový vstup může být odepřen
- rozhoduje buď server nebo si každý proces vede lokální graf, musí se zabránit cyklům (silnější - méně efektivní využití prostředků)

- jinak: Rozdělení na úseky, ve kterých je vyžadováno sdílených prostředků a ve kterých ne. Každý proces před vstupem do kritické sekce předběžně žádá o přidělení. Na jednom z procesů se vše registruje a vytváří předběžný graf závislostí. Jestliže by se v něm po přidělení objevil cyklus není prostředek přidělen.

Detekce a zotavení

- aposteriorní - optimistické
- proces musí být schopen vrátit svůj výpočet před začátek transakce
- při detekci deadlocku se jeden proces vybere jako oběť - jeho výpočet se vrátí
- požadavky se opět opatřují časovými razítky
- jedna možnost - má-li mladší transakce již přiděleno a starší chce - starší je pozastavena do uvolnění prostředku, jinak ukončíme žádající transakci
- alternativa - starší transakce nikdy nečeká (mladší ano), mladší může být ukončena, má-li již přiděleno

Komunikace

- závislost - čekání na zprávu
- pasivní proces - čeká na zprávu, aktivní - nečeká
- množina procesů, od kterých proces čeká zprávu - množina závislosti

Zablokování na množině procesů:

- každý proces je pasivní
- pro množinu závislostí platí, že je podmnožinou uvažovaných (všech) procesů

Chandy - Misra - Hass

- zjištění, zda je proces součástí množiny zablokovaných
- test - začíná žádostí
- pokud libovolný proces je pasivní a tuto žádost ještě nedostal, rozešle ji
- dostane-li odesílatel stejný počet zpráv jako je procesů - zablokování

Detekce ukončení výpočtu

Dijkstra-Scholten

- ~~eviduje rozdíly mezi odeslanými / očekávanými a přijatými požadavky a odpověďmi~~
- z kořenového uzlu je odeslán požadavek na ukončení výpočtu

- dostane-li startovní uzel všechny očekávané odpovědi na testovací zprávu - konec

Dijkstra - Feijen - Van Gasteren

- proces, který je první v rámci uspořádání a už skončil, pošle bílou zprávu
- proces, který dostane bílou zprávu a neskončil, pošle černou, pokud už skončil, bílou zprávu
- dostane-li pův. proc. zpět bílou - konec výpočtu
- proces čekající na bílého peška nesmí poslat žádnou zprávu, kterou by mohl někoho zaúkolovat, dokud odpověď nepřijde. V opačném případě, byť by přišel bílý pešek, není platný.

Misra

- logický kruh nahrazuje cyklem obsahujícím všechny komunikační hrany

---

- Stránka byla naposledy editována v 23:51, 7. 2. 2007.
