Systemy Rozproszone

Zima 2014-2015

Dokumentacja projektowa

Temat:

Realizacja algorytmu Ricarta-Agrawali w środowisku rozproszonym przy założeniu awaryjności węzłów

Opiekun projektu:
dr inż. Artur Krystosik

Opis zadania

Zaimplementować bibliotekę udostępniającą zestaw funkcji realizujących wzajemne wykluczanie w dostępie do zasobu za pomocą rozproszonego algorytmu Ricarta-Agrawala. Działanie zaprezentować przy pomocy klientów umożliwiających zajmowanie i zwalnianie zasobu. System musi być odporny na awarie uczestników.

Algorytm

Ogólny opis

Każdy proces posiada swój zegar logiczny. Początkowo wszystkie zegary logiczne mają wartość 0 (start systemu).
Od razu po połączeniu się z procesem wysyłany jest komunikat INIT, który służy do synchronizacji zegarów logicznych (aby w razie padnięcia jednego z procesów i jego ponownego łączenia z siecią nie uzyskał on “niesprawiedliwie” pierwszego miejsca w kolejce do wejścia do sekcji krytycznej). Dopóki nie zostanie odebrana odpowiedź na INIT od wszystkich węzłów nie da się wejść do sekcji krytycznej (aż upłynie timeout na wypadek padnięcia któregoś z węzłów).
Jeżeli proces chce wejść do sekcji krytycznej, wysyła do wszystkich innych komunikat żądania ze swoim numerem oraz wartością zegara logicznego, oraz zapamiętuje czas logiczny swojego żądania.
Zakładamy, że przy odbieraniu jakiegokolwiek komunikatu następuje synchronizacja zegara u odbiorcy (tzn. czas odbiorcy = max(czas_odbiorcy, stempel_komunikatu) +1).
Jeżeli proces otrzyma żądanie to :

1. Jeśli nie chce wejść do sekcji krytycznej, wysyła zgodę żądającemu.
2. Jeśli chciał wejść to porównuje stempel czasowy żądania z czasem swojego żądania, jeśli jego jest wyższy(chciał później uzyskać dostęp) to wysyła żądającemu zgodę. Jeśli nie, dodaje żądającego do kolejki.
3. Jeżeli stempel żądającego jest taki sam jak czas żądania odbiorcy, to odbiorca wysyła zgodę jeśli proces żądający ma mniejszy numer ID od jego ID

Gdy proces otrzyma zgody od wszystkich procesów (lub upłynie timout), wchodzi do sekcji krytycznej.
Po wyjściu z sekcji krytycznej proces wysyła zgody do wszystkich węzłów, które miał w swojej kolejce.

W momencie nadania wiadomości **REQUEST** do pozostałych węzłów, aplikacja rozpoczyna odliczanie do **TIMEOUT =** liczba_klientów * (**MAX_SECTION_TIME** + 1sek) i najpóźniej po tym czasie wchodzi do sekcji. Po otrzymaniu każdego OK następuje zmniejszenie **TIMEOUT** oczekiwania o maksymalnie **MAX_SECTION_TIME** + 1s

Przykłady:
MAX_SECTION_TIME = 5s, liczba_klientów **=** 3, **TIMEOUT** = 18s

1) zaczynamy czekać 18s, po 3s przyszło OK, więc zamiast czekać jeszcze 15s, czekamy 12s
2) czekamy już 10s, po kolejnych 2s przyszło OK, więc zamiast czekać jeszcze 8s, czekamy 6s

pseudokod:

```
MAX_TIMEOUT = liczba_klientów *(MAX_SECTION +1)
Początkowo: TIMEOUT = MAX_TIMEOUT

Po otrzymaniu OK:

alreadyWaited = sysTime - startTime;
v = Integer(alreadyWaited / (MAX_SECTION +1))
v++;
TIMEOUT = TIMEOUT - v * (MAX_SECTION +1)
startTime = sysTime
```

gdy minie TIMEOUT….

```
if (sysTime - startTime  > TIMEOUT + 1)
{
     nie wchodz i wyslij ok do tych, ktore masz na liscie
}
```

Liczba klientów to liczba wszystkich założonych klientów, a nie aktualnie podłączonych.

Po otrzymaniu OK od wszystkich klientów, aplikacja ignoruje **TIMEOUT** i wchodzi do sekcji. Po wyjściu z sekcji wysyła OK do wszystkich serwerów, które miał na swojej liście oczekujących z parametrem **clock** równym wartości **clock** z wiadomości REQUEST.

Zgoda na wejście do sekcji

Przy wysyłaniu wiadomości OK w polu clock umieszczamy zegar logiczny żądania, z którym otrzymano daną wiadomość OK. Trzeba zapewnić, żeby dane OK zawsze dotyczyły bieżącego żądania (a nie np. żądania sprzed awarii i restartu węzła). W przeciwnym razie powinno zostać zignorowane przez proces, który je otrzyma.

Przykład:
A wysyła do B REQUEST z zegarem logicznym 7.
B odracza wysłanie OK, bo samo oczekuje na sekcję z mniejszym zegarem.
Po wyjściu z sekcji B odsyła OK do A z zegarem równym 7.

Synchronizacja zegara po nawiązaniu połączenia

Po nawiązaniu połączenia wysyłana jest wiadomość typu INIT z zegarem 1 służąca do synchronizacji zegarów, aby w razie kolejnego po restarcie podłączenia nie zaburzyć

instniejącej komunikacji. Na tę wiadomość należy odpowiedzieć wiadomością typu INIT ze swoim aktualnym zegarem. Dopóki nie otrzyma się od wszystkich połączonych węzłów wiadomości INIT, nie można wysłać żądania wejścia do sekcji. Ze względu na to, że nie wiadomo, ile węzłów jest aktualnie podłączonych (ilu wiadomości się spodziewać), wprowadzony został timeout *reconnection period* + 1s (opoźnienia w sieci), po którym po próbie nawiązania połączenia można wysłać żądanie.

Komunikacja

Od strony technicznej

Dla każdego węzła tworzony jest wątek odbierający - po odebraniu połączenia przez serwer oraz wątek wysyłający próbujący połączyć się z węzłem. Oznacza to, że komunikacja odbywa się dwoma jednokierunkowymi połączeniami. Dzięki temu przy nawiązaniu połączenia nie trzeba rozstrzygać kto, do kogo ma się przyłączyć.
Każdy serwer ma z góry ustalone ID, które jest wykorzystywane do identyfikacji węzła w sieci oraz roztrzygania kolejności dostępu do sekcji krytycznej w przypadku wystąpienia identycznych wartości zegara logicznego.

Format wiadomości (JSON):

```
{id: <serverId>, clock: <clock_value>, type: "<message_type>"}"\n"
```

Na końcu każdej wiadomości musi być umieszczony znak nowej linii.

Typy wiadomości (<message_type>):

- REQUEST - Oznacza chęć wejścia do sekcji
- OK - Zgoda na wejście
- INIT - Wiadomość używana do synchronizacja zegarów po podłączeniu węzła do sieci

ID Serwera w sieci

Do testów przyjęto następujący przydział ID dla poszczególnych implementacji:
1 - Java
2 - Python
3 - C#
4 - C++

Przypadki testowe

1. Realizacja algorytmu bez sytuacji wyjątkowych

Węzły muszą poprawnie realizować kroki algorytmu niedopuszczając do sytuacji, w której więcej niż jeden serwer jednocześnie okupuje sekcję krytyczną.

2. Realizacja algorytmu przy niekompletnej liczbie uczestników (dowolna liczba połączonych węzłów)

W przypadku, gdy w sieci znajduje się niekompletna liczba serwerów dostęp do sekcji powinien być zachowany. Niemożność otrzymania zgody od nieaktywnego węzłą nie możę całkowicie blokować dostępu do sekcji, Pozostałe węzły powinny same się uszeregować i kolejno wchodzić do sekcji po określonym czasie TIMEOUT

3. Podłączenie węzła do sieci, w momencie, w którym pozostałe węzły realizują już kroki algorytmu i ustawienie go w kolejce oczekujących na sekcję

Nie można dopuścić, aby nowopodłączony węzeł żądał dostępu do sekcji krytycznej z nieaktualnym zegarem logicznym. Należy zapewnić, że przed wysłaniem żądania serwer najpierw uspójni zegar logiczny z pozostałymi węzłami a dopiero potem zażąda dostępu do sekcji krytycznej. Jest to konieczne, aby wyeliminować możliwość “wepchnięcia się” nowopodłączonego serwera do kolejki oczekujących. Powinien on znaleźć się na końcu kolejki

4. Odłączenie od sieci węzła, który nie był w posiadaniu sekcji krytycznej

Odłączenie od sieci węzła, który nie był w posiadaniu sekcji krytycznej nie powinno w żaden sposób wpływać na pozostałe węzły. Kolejność dostępu do sekcji musi zostać zachowana a brak węzła musi zostać zastąpiony obliczeniem odpowiedniego czasu oczekiwania (TIMEOUT)

5. Odłączenie od sieci węzła, który był w posiadaniu sekcji krytycznej

W przypadku, gdy serwer został odłączony w momencie okupacji sekcji krytycznej i nie zdążył wysłać do pozostałych czekających zgody na wejście, pozostałe serwery muszą same, po określonym czasie, uzyskać dostęp do sekcji, zachowując kolejność wyznaczoną przed odłączeniem węzła. Czas oczekiwania na przyjście komunikatu potwierdzającego musi być skonfigurowany w taki sposób, aby pozostałe serwery mimo jego nieotrzymania wiedziały kiedy i w jakiej kolejności mogą wejść do sekcji.

6. Uśpienie komputera w czasie okupacji sekcji krytycznej

Podobnie jak w punkcie 5. Pozostałe serwery muszą nadal mieć możliwość wejścia do sekcji krytycznej mimo, że uśpiony serwer nie wysłał potwierdzenia i nie została wykryta jego awaria (w trakcie uśpienia procesu gniazda sieciowe nie są zamykane więc nie ma możliwości stwierdzenia, że proces po drugiej stronie umarł)

7. Wybudzenie procesu po uśpieniu

Algorytm musi również działać poprawnie w sytuacji, w której proces oczekujący na sekcję krytyczną zostanie uśpiony i jego okno czasowe wyznaczone na dostęp do sekcji minie. Nie można pozwolić, aby proces po wybudzeniu wszedł do sekcji krytycznej jeśli teoretyczny czas okupacji sekcji mógłby wykroczyć poza wyznaczone dla niego przez algorytm ramy czasowe. W przypadku wykrycia takiej sytuacji proces powinien zaniechać dostępu do sekcji oraz wysłać zgody do wszystkich czekających na niego procesów.