PRZETWARZANIE ROZPROSZONE
Wskrzeszanie Smoków Inc.

Bartosz Malcherek 132697
Michał Świderski 136816

30 kwietnia 2020

1 Opis problemu

Są dwa rodzaje procesów: jeden generuje co pewien czas zlecenie. O zlecenie ubiegają się profesjonaliści o jednej z trzech możliwych specjalizacji (głowa, ogon, tułów). Do realizacji zlecenia potrzeba trzech profesjonalistów o różnych specjalizacjach - należy zapewnić, by nie doszło do zakleszczeń! Dodatkowo, profesjonaliści muszą wypełnić robotę papierkową (robi to jeden z nich) przy jednym z b biurek w gildii Wskrzeszania Smoków. Następnie profesjonaliści zdobywają dostęp do jednego z s szkieletów smoków i rozpoczynają wskrzeszanie. Należy zapewnić, by profesjonaliści dzielili się w miarę równo pracą.

2 Założenia

a. Wyróżniamy 2 rodzaje procesów:

   - generator zleceń, tylko 1 proces.
   - profesjonalista, dowolna ilość procesów.

b. Generator zleceń co pewien losowy czas generuje zlecenie i wysyła je wszystkim profesjonalistom.

c. Każdy profesjonalista jest osobnym procesem, który posiada następujące dane:

- SPECIALIZATION - swoją specjalizację, przypisywaną przy tworzeniu procesu. Zakładamy, że rozkład specjalizacji jest równomierny i przydział do nich jest uzależniony od numeru procesu.
- JOBS_DONE - ilość podjętych prac.
- STATE - stan procesu.
- DESK_COUNT - liczba biurek (B).
- SKELETON_COUNT - liczba szkieletów (S).
- SPECIALIST_COUNT - liczba specjalistów (N).
- COLLEAGUES - tablica partnerów wskrzeszania.
- JOB_MAP - mapa zadań.
- DESK_QUEUE_ACK - liczba przysłanych ACK na żądanie zajęcia biurka.
- SKELETON_QUEUE_ACK - liczba przysłanych ACK na żądanie wzięcia szkieletu.
- JOB_TIMEOUT - liczba prac, których specjalista nie będzie próbował podjąć.
- LAST_REQUESTED_JOB - UUID ostatnio szukanej pracy.

3 Wiadomości

A. NEW_JOB - wiadomość wysyłana wszystkim specjalistom przez generator zadań. Zawiera informacje o nowym zadaniu (JOB_UUID).

B. REQUEST_JOB - komunikat wysyłany do wszystkich specjalistów przez innego specjalistę. Zgłasza chęć wykonania zadania JOB_UUID. Zawiera również wartość JOBS_DONE w celu ustalenia priorytetu i zapobieganiu głodzenia procesów.

C. ACK_JOB - komunikat wysyłany przez specjalistę do procesu, od którego otrzymaliśmy REQUEST_JOB. Wyraża on akceptacje przypisania zadania.

D. REJECT_JOB - komunikat wysyłany przez specjalistę do procesu, od którego otrzymaliśmy REQUEST_JOB. Wysłanie tego komunikatu oznacza, że nasz proces również stara się zająć to zadanie, ale ma do niego niższy priorytet.

E. HELLO - komunikat wysyłany przez specjalistę, który sukcesem zakończył poszukiwanie nowego zadania. Ma na celu znalezienie partnerów w wykonywanej pracy, zawiera JOB_UUID.

F. REQUEST_DESK - wiadomość do wszystkich specjalistów, wysłana w celu zgłoszenia się po dostęp do biurka.

G. ACK_DESK - wiadomość do specjalisty, od którego otrzymaliśmy REQUEST_JOB. Oznacza zgodę na zajęcie zasobu.

H. RELEASE_DESK - informacja o zwolnieniu biurka

I. REQUEST_SKELETON - wiadomość do wszystkich specjalistów wyrażająca chęć zajęcia szkieletu.

J. ACK_SKELETON - zgoda na zajęcie szkieletu.

K. SKELETON_TAKEN - informacja o zabraniu szkieletu z magazynu. Celem wiadomości jest dekrementacja licznika SKELETON_COUNT u specjalistów, z powodu nieodnawialności szkieletów.

L. START - informuje specjalistów z drużyny o rozpoczęciu wskrzeszania.

M. FINISH - informuje specjalistów z drużyny o zakończeniu wskrzeszania.

4 Stany specjalistów i ich reakcje na wiadomości

Reakcja specjalisty na otrzymanie komunikatu NEW_JOB we wszystkich stanach, z wykluczeniem AWAITING_JOB:

- Jeśli danego JOB_UUID nie ma w JOB_MAP, to jest wpisywany z wartością 0
- Jeśli dane JOB_UUID istnieje to nie zmieniamy jego wartości

Dodatkowo w każdym stanie, po otrzymaniu komunikatu SKELETON_TAKEN dekrementujemy licznik pozostałych szkieletów SKELETON_COUNT

4.1 AWAITING_JOB

Początkowy stan specjalisty. W tym stanie czeka na nadesłanie zadania przez generator, jeżeli w jego tablicy zleceń nie ma zadania, którego mógłby się podjąć.

1. Jeśli otrzyma NEW_JOB:
   - Jeżeli JOB_UUID otrzymanej pracy nie znajduje się w JOB_MAP, to wpisuje dane JOB_UUID do niej, dekrementuje parametr JOB_TIMEOUT i jeżeli będzie on <= 1, to przechodzi do stanu REQUESTING_JOB
   - Jeżeli JOB_UUID znajduje się w JOB_MAP to pozostaje w obecnym stanie i czeka na następne zadania
2. Jeśli otrzyma REQUEST_JOB:
   - Jeżeli specjalizacja procesu, który wysłał żądanie jest taka sama jak nasza - to w swoim JOB_MAP przy danym JOB_UUID wpisuje wartość -1, aby uniknąć starania się o nie w przyszłości, oraz odpowiada ACK_JOB
   - Jeżeli żądanie jest na inną specjalizację, to dopisuje to zadanie do JOB_MAP z wartością 0, oraz dekrementuje parametr JOB_TIMEOUT i jeżeli będzie on <= 1, to przechodzi w stan REQUESTING_JOB (poczta pantoflowa między specjalistami, którzy mogą poinformować się o nowej pracy, zanim nadejdzie ona z generatora). Procesowi, który nadesłał zlecenie odpowiada ACK_JOB
3. Jeżeli otrzyma REJECT_JOB z JOB_UUID = LAST_REQUESTED_JOB, to inkrementuje JOB_TIMEOUT
4. Jeśli otrzyma jakikolwiek inny REQUEST odpowiada na niego pasującym komunikatem ACK
5. Pozostałe wiadomości ignoruje

4.2 REQUESTING_JOB

Stan poszukiwania nowej pracy. Na wejściu do niego specjalista wysyła REQUEST_JOB, w którym zawiera JOB_UUID wybranego zadania, oraz swoją specjalizację.

1. Jeśli otrzyma REJECT_JOB dla szukanego JOB_UUID, to zmienia jej wartość w JOB_MAP na -1, czyści tablicę COLLEAGUES, oraz wraca do stanu AWAITING_JOB

2. Jeśli otrzyma ACK_JOB dla szukanego JOB_UUID, to inkrementuje jej wartość w JOB_MAP. Jeśli ta wartość wyniesie N - 1, będzie to oznaczało, że wszyscy specjaliści zgodzili się na przyjęcie przez nas tego zlecenia. W tym przypadku wysyłamy wszystkim komunikat HELLO i przechodzimy w stan AWAITING_COLLEAGUES oraz inkerementujemy JOBS_DONE.

3. Jeśli otrzyma HELLO to dopisuje nadawcę do tablicy COLLEAGUES

4. Jeśli otrzyma REQUEST_JOB:

   - Jeżeli JOB_UUID tego zapytania jest inne niż to, którego sam szuka, specjalista odpowiada ACK_JOB. Dodatkowo jeżeli specjalizacja procesu, który wysłał zapytanie jest taka sama jak nasza, to zmieniamy wartość JOB_MAP dla danego zadania na -1
   - Jeżeli JOB_UUID jest takie samo jak szukane przez tego specjalistę -> sprawdzamy kto ma lepszy priorytet. Najważniejszymn czynnikiem w sprawdzaniu priorytetu jest parametr JOBS_DONE -> wygrywa zawsze proces z niższą jego wartością:
     - Jeśli wygrał ten proces, wysyłamy REJECT_JOB
     - Jeśli wygrał nadawca, wysyłamy ACK_JOB, czyścimy tablicę COLLEAGUES, ustawiamy JOB_MAP[JOB_UUID] na -1 i przechodzimy w stan AWAITING_JOB
     - W przypadku braku możliwości ustalenia priorytetu dostępnymi sposobami, wygrywa proces który ma niższy pid.

5. Jeśli otrzyma jakikolwiek inny REQUEST odpowiada na niego pasującym komunikatem ACK

6. Pozostałe wiadomości ignoruje

4.3 AWAITING_COLLEAGUES

Stan w którym specjalista czeka, aż do jego zadania zgłoszą się wszyscy specjaliści.

1. Jeśli otrzyma HELLO -> dopisuje nadawce do tablicy COLLEAGUES. Jeżeli ilość elementów tablicy wyniesie 3 - proces z najniższym numerem procesu przechodzi w stan AWAITING_DESK, pozostali dwaj specjaliści przechodzą w stan AWAITING_START.
2. Na wszystkie komunikaty typu REQUEST odpowiada pasującym ACK
3. Pozostałe wiadomości ignoruje

4.4 AWAITING_DESK

Stan oczekiwania na biurko w celu zrobienia papierkowej roboty. Na wejściu w ten stan proces wysyła REQUEST_DESK do wszystkich procesów.

1. Jeśli otrzyma ACK_DESK -> inkrementuje DESK_QUEUE_ACK, oraz jeśli (N - DESK_QUEUE_ACK - 1) < DESK_COUNT, to wchodzimy do sekcji krytycznej, oraz przechodzimy do stanu PAPER_WORK
2. Jeśli otrzyma REQUEST_DESK:
   - Jeżeli priorytet nadawcy jest niższy niż tego procesu -> zapisuje go w lokalnej kolejce oczekujących
   - Jeżeli priorytet nadawcy jest wyższy od tego procesy -> wysyła ACK_DESK
3. Jeśli otrzyma jakikolwiek inny REQUEST odpowiada na niego pasującym komunikatem ACK
4. Pozostałe wiadomości ignoruje

4.5 PAPER_WORK

Stan wykonywania papierkowej roboty przez losowy okres czasu. Po jego zakończeniu wysyłamy ACK_DESK do wszystkich procesów w kolejce oczekujących i przechodzimy do stanu ACQUIRE_SKELETON. Wszystkie żądania REQUEST_DESK zapisujemy w kolejce oczekujących, na pozostałe REQUEST odpowiadamy pasującym ACK. Pozostałe komunikaty ignorujemy.

4.6 ACQUIRE_SKELETON

Stan oczekiwania na szkielet. Podczas wejścia do tego stanu specjalista wysyła komunikat REQUEST_SKELETON.

- Jeśli otrzyma ACK_SKELETON -> inkrementuje licznik SKELETON_QUEUE_ACK, oraz sprawdza, czy (N - SKELETON_QUEUE_ACK - 1) < SKELETON_COUNT. Jeżeli zachodzi ten warunek wysyłamy komunikat SKELETON_TAKEN, następnie ACK_SKELETON procesom w kolejce oczekujących
- Jeśli otrzyma REQUEST_SKELETON:
  - Jeżeli priorytet nadawcy jest niższy niż tego procesu -> zapisuje go w lokalnej kolejce oczekujących
  - Jeżeli priorytet nadawcy jest wyższy od tego procesy -> wysyła ACK_SKELETON
- Jeśli otrzyma jakikolwiek inny REQUEST odpowiada na niego pasującym komunikatem ACK
- Pozostałe wiadomości ignoruje

4.7 AWAITING_START

Stan oczekiwania procesów na skończenie przez jednego ze specjalistów papierkowej roboty i zdobycia szkieletów.

- Jeśli otrzyma dowolny REQUEST odpowiada pasującym ACK
- Jeśli otrzyma START przechodzi do stanu REVIVING

4.8 REVIVING

Stan wskrzeszania smoka.

- Jeśli otrzyma dowolny REQUEST odpowiada pasującym ACK
- Jeśli jest procesem z najniższym rankiem z ekipy wskrzeszającej, to po losowym okresie czasu potrzebnym do wskrzeszenia smoka wysyła FINISH do procesów w tablicy COLLEAGUES i przechodzi do stanu

AWAITING_JOB po uprzednim zresetowaniu wszystkich zmiennych do stanu początkowego (poza ilością szkieletów)

- Jeśli nie jest procesem z najniższym rankiem w tablicy COLLEAGUES -> oczekuje na komunikat FINISH i przechodzi do stanu AWAITING_JOB po uprzednim zresetowaniu wszystkich zmiennych do stanu początkowego (poza ilością szkieletów)

5 Rozwiązywanie dostępu do sekcji krytycznej

Rozwiązanie tego problemu, będzie oparte na algorytmie Ricakrta-Agrawali'ego, który zostanie dostosowany do zapewnienia większej niż 1 ilości miejsc w sekcji krytycznej. W celu przypisania komunikatom priorytetu zostanie użyty zegar Lamporta -> każdy komunikat będzie posiadał własny timestamp. W przypadku braku możliwości rozstrzygnięcia priorytetu w ten sposób, preferowany będzie proces z niższym numerem.