Algorytm - Wskrzeszanie smoków

Sebastian Michoń 136770, Marcin Zatorski 136834

1 Wstęp

1.1 Założenia

1. Biura o liczności b i szkielety o liczności s są zasobami nierozróżnialnymi.
2. Procesy wiedzą jaką rolę pełni każdy inny proces.
3. Proces o specjalizacji x będzie nazywany "proces-x" (np. proces o specjalizacji głowa to proces-głowa)
4. Zlecenia są rozróżnialne i mają przypisane id (numerowane od 1), zgodnie z kolejnością generacji.
5. Liczby procesów o poszczególnych specjalizacjach będą oznaczane przez: procesy-głowy: n_g, procesy-tułowie: n_t, procesy-ogony: n_o.

2 Zmienne

z_p Liczba otrzymanych wiadomości *ZLECENIE*

z_c Liczba otrzymanych wiadomości *REQ_ZLECENIE* o priorytecie większym

cnt Liczba otrzymanych wiadomości *REQ_ZLECENIE* o priorytecie mniejszym

QUEUE_ZLECENIE, QUEUE_BIURO, QUEUE_SZKIELET Kolejki procesów oczekujących na wiadomość *ACK*

QUEUE_OFFER Kolejka procesów od których otrzymaliśmy wiadomość *OFFER*.

3 Wiadomości

ZLECENIE Informacja o nowym zleceniu

REQ_ZLECENIE Informacja o chęci dostępu do zlecenia

ACK_ZLECENIE Zaakceptowanie dostępu do zlecenia

OFFER Oferta przyłączenia do grupy, zawiera id zlecenia

REQ_BIURO Prośba o dostęp do biura

ACK_BIURO Zaakceptowanie dostępu do biura

KONIEC_BIURO Informacja o końcu pracy w biurze

REQ_SZKIELET Prośba o dostęp do szkieletu

ACK_SZKIELET Zaakceptowanie dostępu do szkieletu

START_SZKIELET Rozpoczęcie pracy ze szkieletem

KONIEC_SZKIELET Koniec pracy ze szkieletem

Wiadomości *REQ_ZLECENIE*, *REQ_BIURO*, *REQ_SZKIELET* zawierają priorytet procesu - parę: zegar Lamporta, identyfikator procesu. Wiadomości *ACK_ZLECENIE*, *ACK_BIURO*, *ACK_SZKIELET* zawierają informację o wiadomości *REQ*, na którą są odpowiedzią.

4 Stany

`START` Stan początkowy procesu, zanim zacznie ubieganie się o zlecenie.

`CZEKAJ_ZLECENIE` Stan oczekiwania na sekcję krytyczną zleceń.

`MAM_ZLECENIE` Stan po otrzymaniu zlecenia, przed przydzieleniem do grupy.

`CZEKAJ` Stan po dobraniu do grupy, czekanie na skończenie pracy w biurze i rozpoczęcie pracy ze szkieletem

`CZEKAJ_BIURO` Oczekiwanie na otrzymanie dostępu do biura

`PRACA_BIURO` Praca w biurze

`CZEKAJ_SZKIELET` Oczekiwanie na otrzymanie dostępu do szkieletu

`PRACA_SZKIELET` Praca ze szkieletem i wskrzeszanie smoka

5 Wskrzeszanie

5.1 Dostęp do zleceń i dobór do grupy

1. Generator co jakiś czas wysyła wiadomość *ZLECENIE* do wszystkich procesów poza samym sobą.
2. Pozostałe procesy zaczynają w stanie `START`.

   Reakcje na wiadomości w stanie `START`:

   (a) *REQ_SZKIELET* Odpowiedz komunikatem *ACK_SZKIELET*

   (b) *REQ_BIURO* Odpowiedz komunikatem *ACK_BIURO*

(c) *REQ_ZLECENIE* Odpowiedz komunikatem *ACK_ZLECENIE*. Zwiększ wartość z_c o 1.

(d) *OFFER* Dodaj proces do kolejki QUEUE_OFFER.

(e) *ZLECENIE* Zwiększ wartość z_p o 1

(f) *ACK_BIURO*, *ACK_SZKIELET*, *ACK_ZLECENIE* Zignoruj

3. Proces w stanie **START** wysyła wiadomość *REQ_ZLECENIE* do wszystkich procesów o tej samej specjalizacji i przechodzi do stanu **CZEKAJ_ZLECENIE**.

   Reakcje na wiadomości w stanie **CZEKAJ_ZLECENIE**:

   (a) *REQ_ZLECENIE* Wyślij *ACK_ZLECENIE*. Jeśli odebrana prośba ma priorytet mniejszy to zwiększ *cnt* o 1. W przeciwnym wypadku zwiększ z_c o 1.

   (b) *ACK_ZLECENIE* Jeśli jest to odpowiedź na ostatnią wysłaną wiadomość *REQ_ZLECENIE* zwiększ licznik ACK o 1, w p.p. zignoruj.

   (c) Pozostałe jak w stanie **START**.

4. Po otrzymaniu $n_x - 1$ odpowiedzi *ACK_ZLECENIE* proces przechodzi do stanu **MAM_ZLECENIE**. Id zlecenia jest równe z_c+1 (to zlecenie może jeszcze nie być wygenerowane). Następnie proces aktualizuje zmienne: $z_c \mathrel{+}= cnt + 1$, $cnt = 0$.

   n_x liczba procesów o tej samej specjalizacji

   z_c liczba otrzymanych wiadomości *REQ_ZLECENIE* o priorytecie większym.

5. Reakcje na wiadomości w stanie **MAM_ZLECENIE**:

   (a) *REQ_ZLECENIE* Dodaj proces do QUEUE_ZLECENIE.

   (b) *ACK_ZLECENIE* Zignoruj

   (c) *OFFER* Dodaj proces do kolejki QUEUE_OFFER.

   (d) *KONIEC_BIURO*, *START_SZKIELET*, *KONIEC_SZKIELET* Dostarcz wiadomości, w kolejności odebrania, po przejściu do stanu **CZEKAJ**. Może się zdarzyć, jeśli jedna z wiadomości *OFFER* lub *ZLECENIE* jeszcze nie doszła do procesu.

   (e) Pozostałe jak w stanie **START**.

6. Proces po przejściu do stanu **MAM_ZLECENIE** wysyła wiadomość *OFFER* z id zlecenia do wszystkich procesów o innej specjalizacji.

7. Po otrzymaniu dwóch wiadomości *OFFER* z id równym id otrzymanego zlecenia oraz jeśli $z_p >= id$ zlecenia proces przechodzi do stanu **CZEKAJ**.

5.2 Dostęp do biur

1. Zamiast przejścia do stanu **CZEKAJ** proces-ogon wysyła do wszystkich innych procesów-ogonów komunikat *REQ_BIURO* i przechodzi do stanu **CZEKAJ_BIURO**.

   Reakcje na wiadomości w stanie **CZEKAJ_BIURO**:

(a) *REQ_BIURO* Jeśli wysłana prośba ma priorytet mniejszy, wyślij *ACK_BIURO*, w przeciwnym przypadku dodaj proces do QUEUE_BIURO

(b) *ACK_BIURO* Jeśli jest to odpowiedź na ostatnią wysłaną wiadomość *REQ_BIURO* zwiększ licznik ACK o 1, w p.p. zignoruj

(c) Pozostałe jak w stanie **START**

2. Po otrzymaniu co najmniej $n_o - b$ odpowiedzi *ACK_BIURO* proces przechodzi do stanu **PRACA_BIURO**.

   Reakcje na wiadomości w stanie **PRACA_BIURO**:

   (a) *REQ_BIURO* Dodaj proces do QUEUE_BIURO.

   (b) *ACK_BIURO* Zignoruj

   (c) Pozostałe jak w stanie **START**.

3. Po pewnym czasie proces kończy pracę i wysyła *ACK_BIURO* do wszystkich procesów w QUEUE_BIURO. Następnie wysyła *KONIEC_BIURO* do procesów w swoim zespole i przechodzi do stanu **CZEKAJ**.

4. Reakcje na wiadomości w stanie **CZEKAJ**:

   (a) *KONIEC_BIURO* Jeśli odbiorcą jest proces-tułów zacznij staranie się o dostęp do szkieletu. W przeciwnym przypadku zignoruj.

   (b) *START_SZKIELET* Przejdź do stanu *PRACA_SZKIELET*.

   (c) Pozostałe jak w stanie **START**.

5.3 Dostęp do szkieletów

1. Proces-tułów po otrzymaniu komunikatu *KONIEC_BIURO* wysyła do wszystkich innych procesów-tułowiów komunikat *REQ_SZKIELET* i przechodzi do stanu **CZEKAJ_SZKIELET**.

   Reakcje na wiadomości w stanie **CZEKAJ_SZKIELET**:

   (a) *REQ_SZKIELET* Jeśli wysłana prośba ma priorytet mniejszy, wyślij *ACK_SZKIELET*, w przeciwnym przypadku dodaj proces do QUEUE_SZKIELET

   (b) *ACK_SZKIELET* Jeśli jest to odpowiedź na ostatnią wysłaną wiadomość *REQ_SZKIELET* zwiększ licznik ACK o 1, w p.p. zignoruj

   (c) Pozostałe jak w stanie **START**

2. Jeśli otrzyma co najmniej $n_t - s$ odpowiedzi *ACK_SZKIELET* wysyła do procesów w swoim zespole wiadomość *START_SZKIELET* i przechodzi do stanu **PRACA_SZKIELET**.

   Reakcje na wiadomości w stanie **PRACA_SZKIELET**:

   (a) *REQ_SZKIELET* Dodaj proces do QUEUE_SZKIELET.

   (b) *ACK_SZKIELET* Zignoruj

   (c) *KONIEC_SZKIELET* Zwiększ licznik zakończonych części o 1

(d) Pozostałe jak w stanie **START**.

3. Po pewnym czasie proces kończy pracę i wysyła wiadomość *KONIEC_SZKIELET* do innych procesów w swojej grupie.

4. Jeżeli proces otrzymał dwie wiadomości *KONIEC_SZKIELET* i sam ją wysłał to przechodzi do stanu **START**.

5. Proces-tułów przed przejściem do stanu **START** wysyła wiadomości *ACK_SZKIELET* do procesów w kolejce *QUEUE_SZKIELET*.

6 Inne pomysły i optymalizacje

1. Po wyjściu ze stanu **PRACA_SZKIELET** tylko proces-tułów może czekać na wiadomości *KONIEC_SZKIELET* - wtedy również nie musi ich wysyłać.

2. Po otrzymaniu zlecenia z QUEUE_OFFER można usunąć wiadomości o id mniejszym od otrzymanego.

3. Kolejkę QUEUE_OFFER można zastąpić wysyłaniem wiadomości *OFFER_ACC*.

4. Przy dostępie do zleceń można dodać tablicę LAST_RECEIVED z czasem otrzymania ostatniej wiadomości. Proces czeka wtedy, aż $n_x - 1$ wartości będzie większych lub równych niż czas wysłania *REQ_ZLECENIE*. Gdyby wszystkie procesy zaczęły się ubiegać, jeden z nich otrzyma *REQ_ZLECENIE* o niższych priorytetach i nie musi czekać na *ACK_ZLECENIE*. Proces wiedząc, że wysłał *REQ_ZLECENIE* o niższym priorytecie nie musi wtedy wysyłać *ACK_ZLECENIE* - tutaj można też dodać tablicę LAST_SEND.

6.1 Złożoność Komunikacyjna

Przez złożoność komunikacyjną w tym zadaniu rozumiana jest liczba komunikatów potrzebna do wykonania uprzednio wydanego zlecenia. Kluczowym spostrzeżeniem w obliczaniu złożoności komunikacyjnej jest zauważenie, że każdy komunikat można przypisać do jakiegoś zlecenia.:

ZLECENIE - Przypisana do właśnie wysyłanego zlecenia

ACK_... - Przypisana do zlecenia, które jest przypisane do requesta, na które to ACK odpowiada

REQ_ZLECENIE - Przypisana do zlecenia wykonywanego przez specjalistę po odebraniu odpowiedniej ilości ACKów.

OFFER Przypisana do zlecenia, które zostało przyjęte.

Pozostałe komunikaty są przypisywane do obecnie wykonywanego zlecenia przez specjalistów (związane ze szkieletem i biurem - start i koniec pracy, requesty). Dzięki takiemu przypisaniu na gruncie teoretycznym (na poziomie algorytmu te przypisanie jest niewidoczne) możliwa jest analiza tego, ile komunikatów powstaje w związku z pojedynczym zleceniem. Warto też zauważyć następującą równość: $n = n_o + n_t + n_g + 1$ - liczba wszystkich procesów to liczba procesów o 3 specjalizacjach plus generator zleceń.

1. Dostęp do zleceń i dobór do grupy:

   $n - 1$ - wysyłanie zleceń przez generator - ZLECENIE

   $2(n_g - 1 + n_t - 1 + n_o - 1) = 2n - 8$ - przyjmowanie i odbieranie REQ/ACK_ZLECENIE

   $(n_o + n_t) + (n_g + n_o) + (n_g + n_t) = 2n - 2$ - wysyłanie OFFERA do procesów o różnej specjalizacji.

   Wszystkie procesy w sumie mają złożoność komunikacyjną $5n - 11$

2. Dostęp do biur:

   $2(n_o - 1)$ przyjmuje i odbiera REQ/ACK_BIURO od innych procesów-ogonów

   2 wysłanie komunikatu KONIEC_BIURO do pozostałych specjalistów

   W sumie $2n_o$

3. Dostęp do szkieletów:

   $2(n_t - 1)$ przyjmuje i odbiera REQ/ACK_SZKIELET od innych procesów-szkieletów

   2 wysłanie komunikatu START_SZKIELET do pozostałych specjalistów przez proces-tułów

   $3 * 2$ wysłanie komunikatu KONIEC_SZKIELET do pozostałych specjalistów przez każdego specjalistę

   W sumie $2n_t + 6$

4. W sumie komunikacyjna - liczba komunikatów związanych z pojedynczym zleceniem - wynosi $5n + 2n_o + 2n_t - 5$.

6.2 Złożoność Czasowa

Złożoność czasowa dla pojedynczego zlecenia jest liczona jako liczba rund, w których wysyłane są komunikaty przy pomijalnie małym czasie lokalnego przetwarazania i jednostkowym czasie wysyłania komunikatu.

1. Dostęp do zleceń i dobór do grupy:

   1 - wysyłanie *ZLECENIE* przez generator i *REQ_ZLECENIE* przez specjalistów

   2 - odbieranie i wysyłanie *ACK_ZLECENIE* przez specjalistów

   3 - wysłanie *OFFER* przez specjalistów z nr zlecenia

W sumie 3.

2. Dostęp do biur:

   4 wysyłanie *REQ_BIURO*

   5 wysyłanie *KONIEC_BIURO* do kompanów po zakończeniu przcy w biurze przez ogon

   6 wysyłanie *ACK_BIURO* do oczekujących procesów

   W sumie 3.

3. Dostęp do szkieletów:

   7 wysyłanie *REQ_SZKIELET* przez proces-tułów

   8 wysyłanie *START_SZKIELET* przez proces-tułów do kompanów po zdobyciu odpowiedniej ilości ACKów

   9 wysyłanie *KONIEC_SZKIELET* przez proces do kompanów po zakończeniu pracy ze szkieletem.

   10 wysyłanie *ACK_SZKIELET* przez proces-tułów do pozostałych procesów-tułowiów

   W sumie 4.

4. W sumie złożoność czasowa dla pojedynczego zlecenia wynosi 10.

5. Złożoność będzie się komplikowała przy analizie m zleceń przy k specjalistach poszczególnych typów: O ile procesy 4-10. muszą być wykonane po przyjęciu zlecenia, o tyle procesy 2-3. nie muszą (1. musi zostać wykonany - należy wysłać zlecenie, aby proces-ogon mógł przejść do biura). Przy $k = 2$ przez 10 rund wykonywane jest pierwsze zlecenie, od 2.-10. rundy wykonywane jest 2. zlecenie - zlecone w 2. rundzie, po wysłaniu *REQ_ZLECENIE* przez 2 proces i otrzymaniu odpowiedzi równocześnie dostając zlecenie - dalej natomiast zlecenia będą wysyłane, ale nikt ich nie będzie przyjmował - aż do 11. rundy, kiedy obydwie trójki procesów wyjdą ze szkieletu i znowu wejdą w cykl zlecenia, przyjmując 2 z pozostałych, ciągle generowanych zleceń. Złożoność będzie wynosić $10\lceil\frac{m}{2}\rceil$. Analogicznie, dla 3 procesów o każdej specjalizacji złożoność wyniosłaby $10\lceil\frac{m}{3}\rceil$. Później, procesy będą kończyć asynchronicznie, bo 4. trójka procesów nie pójdzie dalej po wysłanie *OFFER*, w oczekiwaniu na *ZLECENIE*. Kolejne procesy będą się kończyły w rundach: $10-10-10-11-20-20-20-21-30\ldots$; analogicznie, dla $k \in < 4; 9 >$ proocesy kończą się w rundach $10-10-10-11-...-10+k-3-20-20-20$. Dla $k \in < 10; \infty)$ zlecenia będą się kończyły w rundach $10-10-10-11-\cdots-19?-20-21-22-23\ldots$. (Niekoniecznie 19, ale wtedy będzie zamiast 19/18 będą 20-tki) Reasumując:

   $$k \in \{2,3\} \Rightarrow o = 10\lceil\tfrac{m}{k}\rceil$$

   $$k \in < 4; 9 > \Rightarrow o = 10\lceil\tfrac{m}{k}\rceil + \begin{cases} 0 \iff m \ mod \ k - 2 <= 0 \\ m \ mod \ k - 2 \text{ otherwise} \end{cases}$$

$k \in < 10; \infty > \Rightarrow o = 10 + max(m, 2) - 2$

Gdzie o to liczba rund - złożoność czasowa.

6.3 Pojemność kanału

Konieczna pojemność kanału komunikacyjnego może być potencjalnie nieskończona - po pierwsze, ponieważ proces może nie zareagować na nieskończoną liczbę zleceń od generatora zleceń. Po drugie, jeśli jakiś proces np. wskrzesza smoka przez dłuższy czas i nie odpowiada na komunikaty, pozostałe procesy mogą wysyłać mu *REQ_SZKIELET*, nie dostawać odpowiedzi i i tak wchodzić do sekcji krytycznej (bo s>1), wychodzić z niej i ponawiać proces w nieskończoność.