Algorytm - Wskrzeszanie smoków

Sebastian Michoń 136770, Marcin Zatorski 136834

1 Wstęp

Celem zadania było opracowanie programu rozwiązującego problem rozproszonej sekcji krytycznej. W zadaniu są dwa rodzaje procesów. Generator zleceń wysyła co jakiś czas zlecenia do innych procesów. Zlecenia są rozróżnialne i mają przypisane id (zaczynając od 1), zgodnie z kolejnością generacji. Pozostałe procesy to profesjonaliści o trzech możliwych specjalizacjach: głowa, tułów lub ogon. Do wykonania zlecenia potrzeba po jednym specjaliście z każdej specjalizacji. Przed wykonaniem zlecenia jeden ze specjalistów (ogon) musi wykonać pracę w biurze. Biur jest b, i są one nierozróżnialne. Po wykonaniu pracy w biurze można wykonać zlecenie - wskrzesić smoka. Potrzeba do tego jednego ze szkieletów. Szkieletów jest s i są one nierozróżnialne. Szkielety nie znikają po wskrzeszeniu smoka. Za dostęp do szkieletów odpowiada proces-tułów.

Proces o specjalizacji x będzie nazywany “proces-x” (np. proces o specjalizacji głowa to proces-głowa). Liczby procesów o poszczególnych specjalizacjach będą oznaczane przez: procesy-głowy: n_g, procesy-tułowie: n_t, procesy-ogony: n_o.

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 ***TASK***

z_c Liczba otrzymanych wiadomości ***REQ_TASK*** o priorytecie większym

cnt Liczba otrzymanych wiadomości ***REQ_TASK*** o priorytecie mniejszym

OFFERS Zbiór procesów od których otrzymano wiadomość *OFFER*, razem z ich id zlecenia

QUEUE_OFFICE Kolejka komunikatów *REQ_OFFICE*, na które jeszcze nie wysłano odpowiedzi

QUEUE_SKELETON Kolejka komunikatów *REQ_SKELETON*, na które jeszcze nie wysłano odpowiedzi

3 Wiadomości

TASK Informacja o nowym zleceniu

REQ_TASK Informacja o chęci dostępu do zlecenia

ACK_TASK Zaakceptowanie dostępu do zlecenia

OFFER Oferta przyłączenia do grupy, zawiera id zlecenia

REQ_OFFICE Prośba o dostęp do biura

ACK_OFFICE Zaakceptowanie dostępu do biura

END_OFFICE Informacja o końcu pracy w biurze

REQ_SKELETON Prośba o dostęp do szkieletu

ACK_SKELETON Zaakceptowanie dostępu do szkieletu

START_SKELETON Rozpoczęcie pracy ze szkieletem

END_SKELETON Koniec pracy ze szkieletem

Wiadomości *REQ_TASK*, *REQ_OFFICE*, *REQ_SKELETON* zawierają priorytet procesu - parę: zegar Lamporta, identyfikator procesu. Wiadomości *ACK_TASK*, *ACK_OFFICE*, *ACK_SKELETON* zawierają informację o wiadomości *REQ*, na którą są odpowiedzią.

4 Stany

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

WAIT_TASK Stan oczekiwania na sekcję krytyczną zleceń.

HAS_TASK Stan po otrzymaniu zlecenia, przed przydzieleniem do grupy.

HAS_TEAM Stan po znalezieniu pozostałych specjalistów koniecznych do wskrzeszenia smoka.

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

WAIT_OFFICE Oczekiwanie na otrzymanie dostępu do biura

WORK_OFFICE Praca w biurze

FINISH_OFFICE Zakończenie pracy w biurze

WAIT_SKELETON Oczekiwanie na otrzymanie dostępu do szkieletu

WORK_SKELETON Praca ze szkieletem i wskrzeszanie smoka

FINISH_SKELETON Zakończenie pracy ze szkieletem

5 Wskrzeszanie

5.1 Dostęp do zleceń i dobór do grupy

1. Generator co jakiś czas wysyła wiadomość *TASK* do wszystkich procesów poza samym sobą.

2. Pozostałe procesy zaczynają w stanie **START**, natychmiast przechodzą z niego do **WAIT_TASK**.

   Reakcje na wiadomości w stanie **START**:

   (a) *REQ_SKELETON* Odpowiedz komunikatem *ACK_SKELETON*

   (b) *REQ_OFFICE* Odpowiedz komunikatem *ACK_OFFICE*

   (c) *REQ_TASK* Odpowiedz komunikatem *ACK_TASK*. Zwiększ wartość z_c o 1.

   (d) *OFFER* Dodaj id procesu do zbioru ofert OFFERS razem z id zlecenia, którego dotyczy oferta.

   (e) *TASK* Zwiększ wartość z_p o 1

   (f) *ACK_OFFICE*, *ACK_SKELETON*, *ACK_TASK* Zignoruj

3. Proces w stanie **WAIT_TASK** wysyła wiadomość *REQ_TASK* do wszystkich procesów o tej samej specjalizacji.

   Reakcje na wiadomości w stanie **WAIT_TASK**:

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

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

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

4. Po otrzymaniu $n_x - 1$ odpowiedzi *ACK_TASK* proces przechodzi do stanu **HAS_TASK**. 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_TASK* o priorytecie większym.

5. Reakcje na wiadomości w stanie **HAS_TASK** i **HAS_TEAM**:

   (a) *REQ_TASK* Wysyłane jest *ACK_TASK*, z_c jest zwiększane o 1.

   (b) *ACK_TASK* Zignoruj

   (c) *END_OFFICE*, *START_SKELETON*, *END_SKELETON* Dostarcz wiadomości, w kolejności odebrania, po przejściu do stanu **IDLE**. Może się zdarzyć, jeśli jedna z wiadomości *OFFER* lub *TASK* jeszcze nie doszła do procesu.

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

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

7. Po otrzymaniu dwóch wiadomości *OFFER* z id równym id otrzymanego zlecenia proces przechodzi do stanu **HAS_TEAM**.

8. Jeśli w stanie **HAS_TEAM** $z_p >= id$ proces przechodzi do stanu **IDLE**.

5.2 Dostęp do biur

1. Zamiast przejścia do stanu **IDLE** proces-ogon wysyła do wszystkich innych procesów-ogonów komunikat *REQ_OFFICE* i przechodzi do stanu **WAIT_OFFICE**.

   Reakcje na wiadomości w stanie **WAIT_OFFICE**:

   (a) *REQ_OFFICE* Jeśli wysłana prośba ma priorytet mniejszy, wyślij *ACK_OFFICE*, w przeciwnym przypadku dodaj proces do QUEUE_OFFICE

   (b) *ACK_OFFICE* Jeśli jest to odpowiedź na ostatnią wysłaną wiadomość *REQ_OFFICE* 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_OFFICE* proces przechodzi do stanu **WORK_OFFICE**.

   Reakcje na wiadomości w stanie **WORK_OFFICE**:

   (a) *REQ_OFFICE* Dodaj proces do QUEUE_OFFICE.

   (b) *ACK_OFFICE* Zignoruj

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

3. Po pewnym czasie proces kończy pracę i przechodzi do stanu **FINISH_OFFICE**. Następnie wysyła *ACK_OFFICE* do wszystkich procesów w QUEUE_OFFICE. Wysyła również *END_OFFICE* do procesu-tułowia.

4. Reakcje na wiadomości w stanie **IDLE** i **FINISH_OFFICE**:

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

(b) *START_SKELETON* Przejdź do stanu *WORK_SKELETON*.

(c) Pozostałe jak w stanie **START**.

5.3 Dostęp do szkieletów

1. Proces-tułów po otrzymaniu komunikatu *END_OFFICE* wysyła do wszystkich innych procesów-tułowiów komunikat *REQ_SKELETON* i przechodzi do stanu **WAIT_SKELETON**.

   Reakcje na wiadomości w stanie **WAIT_SKELETON**:

   (a) *REQ_SKELETON* Jeśli wysłana prośba ma priorytet mniejszy, wyślij *ACK_SKELETON*, w przeciwnym przypadku dodaj proces do QUEUE_SKELETON

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

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

2. Jeśli proces-tułów otrzyma co najmniej $n_t - s$ odpowiedzi *ACK_SKELETON* wysyła do procesów w swoim zespole wiadomość *START_SKELETON* i przechodzi do stanu **WORK_SKELETON**.

   Reakcje na wiadomości w stanie **WORK_SKELETON**:

   (a) *REQ_SKELETON* Dodaj proces do QUEUE_SKELETON.

   (b) *ACK_SKELETON* Zignoruj

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

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

3. Po zakończeniu pracy proces-tułów przechodzi do stanu **FINISH_SKELETON**. Pozostałe procesy w zespole wysyłają do niego wiadomość *END_SKELETON* i przechodzą do stanu **START**.

4. Jeżeli proces-tułów otrzymał dwie wiadomości *END_SKELETON* to przechodzi do stanu **START**. Wysyła też wiadomości *ACK_SKELETON* do procesów w kolejce *QUEUE_SKELETON*.

6 Inne pomysły i optymalizacje

6.1 Złożoność Komunikacyjna

Złożoność komunikacyjna została obliczona jako liczba komunikatów potrzebnych do wykonania jednego zlecenia. Każdy komunikat jest powiązany z jednym zleceniem, a zatem każdemu komunikatowi można przypisać dokładnie jedno zlecenie:

TASK - Przypisana do wygenerowanego zlecenia

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

REQ_TASK - Przypisana do zlecenia wykonywanego przez specjalistę po odebraniu odpowiedniej ilości komunikatów *ACK_TASK*.

OFFER Przypisana do zlecenia, które zostało przyjęte.

Pozostałe komunikaty są powiązane z zleceniem wykonywanym przez dany zespół. Liczba komunikatów wysłanych w ramach zlecenia to:

1. Dostęp do zleceń i dobór do grupy:

   $n-1$ - wysyłanie zleceń przez generator - *TASK*

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

   $(n_o + n_t) + (n_g + n_o) + (n_g + n_t) = 2n - 2$ - wysyłanie *OFFER* 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_OFFICE* od innych procesów-ogonów

   1 wysłanie komunikatu *END_OFFICE* do procesu-tułowia przez proces-ogon.

   W sumie $2n_o - 1$

3. Dostęp do szkieletów:

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

   2 wysłanie komunikatu *START_SKELETON* do pozostałych specjalistów przez proces-tułów

   2 wysłanie komunikatu *END_SKELETON* do procesu-tułowia przez proces-ogon i proces-głowa.

   W sumie $2n_t + 2$

4. W sumie komunikacyjna - liczba komunikatów związanych z pojedynczym zleceniem - wynosi $5n + 2n_o + 2n_t - 10$.

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 transmisji komunikatu.

1. Dostęp do zleceń i dobór do grupy:

   1 - wysyłanie *TASK* przez generator i *REQ_TASK* przez specjalistów

   2 - odbieranie i wysyłanie *ACK_TASK* przez specjalistów

   3 - wysłanie *OFFER* przez specjalistów z nr zlecenia

W sumie 3.

2. Dostęp do biur:

   4 wysyłanie *REQ_OFFICE*

   5 wysyłanie *END_OFFICE* do procesu-tułowia po zakończeniu pracy w biurze przez ogon

   6 wysyłanie *ACK_OFFICE* do oczekujących procesów

   W sumie 3.

3. Dostęp do szkieletów:

   7 wysyłanie *REQ_SKELETON* przez proces-tułów

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

   9 wysyłanie *END_SKELETON* przez procesy do procesu-tułowia po zakończeniu pracy ze szkieletem.

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

   W sumie 4.

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

6.3 Pojemność kanału

Konieczna pojemność kanału komunikacyjnego może być potencjalnie nieskończona - choćby dlatego, że proces może nie zareagować na nieskończoną liczbę zleceń od generatora zleceń.