

Algorytmy Równoległe i Rozproszone

Część VII - Systemy rozproszone, wstęp

Łukasz Kuszner
pokój 209, WETI
<http://www.sphere.pl/~kuszner/>
kuszner@sphere.pl

Oficjalna strona wykładu
<http://www.sphere.pl/~kuszner/ARiR/>

2005/06

Spis treści

1	Przykład	1
2	Algorytmy rozproszone a scentralizowane	2
3	System tranzycji	5
4	Operacje atomowe	6
5	Sprawiedliwość	7
6	Jednorodność	7
7	Typy komunikacji	8
8	Model all-port	11

1 Przykład

Rozproszony algorytm dla problemu: „wybór lidera”

Przypuśćmy, że mamy rozproszony system asynchroniczny, w którym jednostki obliczeniowe tworzą pierścień.

Problem polega na skonstruowaniu algorytmu, który pozwoli wyróżnić jeden z procesorów. Oczekujemy, że po zakończeniu działania algorytmu dokładnie jeden procesor zasygnalizuje „jestem liderem”, a wszystkie pozostałe „nie jestem liderem”.

Notatki

Przypuśćmy, że każdy z procesorów ma unikalny identyfikator. Procesory tworzą pierścień, możemy więc mówić o lewym i prawym sąsiedzie. Inicjalnie każdy z procesorów wysyła wiadomość do swojego lewego sąsiada, w której zapisuje swój identyfikator.

Następnie każdy procesor odbierając wiadomość od prawego sąsiada przesyła ją dalej, jeśli zapisany na niej identyfikator jest większy od jego własnego identyfikatora i kasuje, jeśli jest mniejszy.

Procesor, który otrzymał z powrotem swój własny identyfikator ogłasza się liderem i wysyła wiadomość kończącą działanie algorytmu.

Notatki

2 Algorytmy rozproszone a scentralizowane

Algorytmy rozproszone różnią się od sekwencyjnych przede wszystkim tym, że sterowanie przebiega równocześnie w wielu elementach systemu.

Natomiast trzy najważniejsze różnice pomiędzy algorytmami rozproszonymi i scentralizowanymi algorytmami równoległymi odnoszą się do:

- wiedzy o stanie systemu,
- ram czasowych,
- przewidywalności działania.

Notatki

Brak wiedzy o stanie globalnym

W algorytmach ze sterowaniem centralnym decyzje mogą być podejmowane na podstawie aktualnego stanu całego systemu. Mimo, że zwykle nie możemy mieć dostępu do całego stanu systemu w ramach jednej operacji maszynowej, to program może przetwarzać sekwencyjnie wszystkie potrzebne dane by podjąć decyzję na podstawie wszystkich potrzebnych informacji. Stan systemu nie jest modyfikowany w czasie obliczeń i możemy mieć pewność integralności danych.

Inaczej w algorytmach rozproszonych. Tu przy podejmowaniu decyzji możemy brać pod uwagę jedynie dane widoczne lokalnie. Jeśli uda się w jednym miejscu uzyskać w sposób pośredni dane o całym systemie, to nie wiemy, czy stan odległych jednostek w tym czasie się jeszcze nie zmienił.

Notatki

Graf systemu

W związku z tym abstrahując od konkretnej struktury fizycznej systemu rozproszonego w naturalny sposób kojarzymy z nim *graf systemu*, w którym zbiór wierzchołków odpowiada jednostkom obliczeniowym, a zbiór krawędzi bezpośrednim połączeniom komunikacyjnym.

W będziemy zakładać, że graf systemu jest spójny. Gdyby graf był niespójny, to poszczególne składowe nie miałyby na siebie żadnego wpływu, można by więc rozważać je oddzielnie.

Notatki

Brak uporządkowania w czasie

W algorytmach ze sterowaniem centralnym każda para zdarzeń może być uporządkowana w czasie, dla każdej pary zdarzeń możemy powiedzieć, że jedno zdarzyło się przed drugim. W algorytmach rozproszonych zdarzenia mogą dowolnie nakładać się na siebie w czasie.

Notatki

Niedeterminizm

Typowo dla algorytmu sekwencyjnego na podstawie danych wejściowych i algorytmu możemy odtworzyć prawidłowy przebieg obliczeń. W przypadku systemów rozproszonych nawet jeśli każda z jednostek wykonuje algorytm deterministyczny, to losowość jest wpisana w strukturę systemu. Przykładowo nie jesteśmy w stanie przewidzieć kolejności odbierania wiadomości, czasu działania poszczególnych jednostek, czy opóźnień związanych z przesyłaniem danych. W związku z tym typową sytuacją jest, że dla identycznego stanu początkowego i identycznych danych wejściowych możemy otrzymać różne przebiegi obliczeń.

Notatki

3 System tranzycji

System, który zmienia swój stan poprzez ciąg dyskretnych kroków (tranzycji, zdarzeń) może być opisany jako *system tranzycji*.

Definicja 1 *System tranzycji (ang. transition system)* S jest trójką $S = (\mathcal{C}, \rightarrow, \mathcal{I})$, gdzie \mathcal{C} jest zbiorem wszystkich możliwych konfiguracji, \rightarrow jest dwuargumentową relacją tranzycji w \mathcal{C} , a \mathcal{I} zbiorem wszystkich możliwych, konfiguracji początkowych $\mathcal{I} \subset \mathcal{C}$.

Mimo, iż formalnie \rightarrow jest podzbiorem iloczynu kartezjańskiego $\mathcal{C} \times \mathcal{C}$, to dla wygody piszemy $\gamma \rightarrow \delta$ zamiast $(\gamma, \delta) \in \rightarrow$.

Notatki

Stan i konfiguracja

W dalszej części bliskoznaczne terminy konfiguracja i *stan* będziemy używać dla odróżnienia globalnej konfiguracji całego systemu i lokalnego stanu pojedynczego procesu. Tak więc konfigurację systemu możemy rozumieć jako sumę mnogościową lokalnych stanów poszczególnych procesów.

Notatki

Wykonania

Definicja 2 *Niech $S = (\mathcal{C}, \rightarrow, \mathcal{I})$ będzie systemem tranzycji. Wykonaniem (ang. execution) S nazywamy maksymalną sekwencję $E = (\gamma_0, \gamma_1, \gamma_2, \dots)$ gdzie $\gamma_0 \in \mathcal{I}$ oraz dla wszystkich $i \geq 0$ zachodzi $\gamma_i \rightarrow \gamma_{i+1}$.*

Notatki

Takie spojrzenie na system rozproszony może się kojarzyć z modelem sekwencyjnej maszyny jednoprotocowej. Rzeczywiście użyliśmy podobnego języka, jednak w istocie jest to tylko pozorne uproszczenie, gdyż analizując konkretny problem dokładamy jeszcze szereg dodatkowych warunków na relację tranzycji i poprawność wykonań. Głównym ograniczeniem jest fakt, że pojedynczy *ruch* (zmiana stanu jednego procesu) odbywa się na podstawie ograniczonej wiedzy o konfiguracji.

Notatki

4 Operacje atomowe

W systemach rozproszonych mamy do czynienia z równoległym wykonaniem operacji. Analizując poprawność algorytmu napotykamy na poważne trudności w wypadku jednoczesnego dostępu dwóch procesów do jednego, współdzielonego obiektu. Na przykład, jeśli jeden proces dokonuje operacji zapisu do zmiennej współdzielonej a w tym samym czasie drugi proces odczytuje zapisane tam dane, to wynik takiej operacji mógłby być trudny do przewidzenia. Typowo rozwiązujemy ten problem zakładając *atomowość* pewnych operacji (ang. atomicity assumptions).

Notatki

Operacja atomowa, to taka, która jest niepodzielna z punktu widzenia pozostałych procesów. Przyjmowane założenia o atomowości operacji mogą się bardzo różnić. Z jednej strony im operacje atomowe są prostsze tym łatwiej będzie je odnieść do systemów rzeczywistych, ale tym trudniej będzie przeprowadzić analizę algorytmu. Z drugiej strony przyjęcie złożonych operacji atomowych znacznie ułatwi analizę, może się jednak okazać, że analiza opierająca się na takich założeniach ma niewiele wspólnego z praktyką.

Notatki

5 Sprawiedliwość

O wykonaniu powiemy, że jest *sprawiedliwe* (ang. weakly fair), jeśli każde zdarzenie, które jest możliwe w ciągu kolejnych konfiguracji „stanie się” w skończonym czasie. O *silnej sprawiedliwości* mówimy, gdy w poprzednim zdaniu można opuścić słowo „kolejnych”.

W niektórych przypadkach sprawiedliwość odgrywa istotną rolę dla poprawności samego algorytmu, jednak za lepsze uważa się algorytmy lub protokoły, które nie nakładają żadnych ograniczeń na sprawiedliwość wykonań.

Notatki

Przykład

Rozpatrzmy dla przykładu system w którym procesy rywalizują w dostępie do pewnego zasobu. Jeśli proces *A* wielokrotnie żąda dostępu do zasobu, to w systemie silnie sprawiedliwym można założyć, że w skończonym czasie zasób ten zostanie przydzielony. W systemie sprawiedliwym może się zdarzyć, że za każdym razem zgłoszenie zostanie odrzucone, chyba że proces *A* domaga się zasobu w sposób nieprzerwany. Wtedy również można założyć, że w skończonym czasie zasób zostanie przydzielony. Bez wiedzy o sprawiedliwości systemu takich założeń poczynić nie możemy.

Notatki

6 Jednorodność

Systemy rozproszone możemy podzielić na

- Systemy jednorodne (ang. uniform),
- Systemy semi-jednorodne (ang. semi-uniform),
- Niejednorodne.

Notatki

7 Typy komunikacji

Procesy mogą komunikować się poprzez *przesyłanie wiadomości* (wymianę komunikatów, ang. message passing) lub *zmiennie współdzielone* (ang. shared variables). W naszym modelu komunikację poprzez zmiennie współdzielone rozumiemy w ten sposób, że ruch odbywa się wyłącznie na podstawie stanu procesu oraz fragmentów stanów procesów sąsiednich w grafie systemu. Te fragmenty interpretujemy jako zmiennie współdzielone.

Notatki

W wypadku komunikacji poprzez przesyłania wiadomości dla pojedynczego procesu dopuszczamy dodatkowe typy ruchów: wyślij (ang. send) i odbierz (ang. receive). Wykonanie ruchu typu wyślij wiąże się z umieszczeniem porcji danych w kanale komunikacyjnym, podobnie wykonanie ruchu typu odbierz wiąże się z odebraniem danych z kanału. Wykonanie każdego ruchu odbywa się wyłącznie na podstawie stanu procesu i odebranych wiadomości.

Notatki

Własności medium komunikacyjnego

Niezawodność

Medium *niezawodne* (ang. reliable), to takie dla którego każda wysłana wiadomość zostanie odebrana dokładnie raz. Typowe błędy medium zawodnego, to: przekłamanie wiadomości, utrata wiadomości, powielenie wiadomości, dostarczenie wiadomości nigdy niewysłanej.

Notatki

Własność *fifo*

Medium ma własność *fifo*, jeśli zachowany jest porządek wysyłanych wiadomości, tzn. jeśli wiadomości m_1 i m_2 zostały wysłane przez nadawcę p do odbiorcy q i wiadomość m_1 została wysłana przed m_2 , to kolejność odbioru wiadomości będzie taka sama.

Notatki

Pojemność

Pojemność medium (ang. capacity), to ilość informacji, która może być jednocześnie przesyłana. Jeśli model pomija to ograniczenie, to mówimy o medium z nieograniczoną pojemnością (ang. unbounded capacity). Powyższe własności określają model z komunikacją poprzez wymianę wiadomości.

W wypadku komunikacji poprzez zmienne współdzielone zakładamy poprawność operacji zapisu/odczytu, o ile sam proces działa poprawnie. Domyślnie zakładamy, że zmienna ma ograniczony rozmiar, chyba, że zostanie zaznaczone inaczej.

Notatki

Topologia sieci

Typ komunikacji możemy też rozróżnić ze względu na parametry grafu systemu, czyli inaczej mówiąc jego *topologię*. Podstawowe typy topologii to:

1. Pierścień (ang. ring) – graf systemu jest cyklem,
2. Drzewo (ang. tree) – graf systemu jest drzewem,
3. Gwiazda (ang. star) – graf systemu jest gwiazdą,
4. Klika (ang. clique) – graf systemu jest grafem pełnym, czyli możliwa jest komunikacja pomiędzy każdą parą procesów.

Notatki

Błędy

Błędy w systemach rozproszonych mogą być zaklasyfikowane do jednej z pojemnych kategorii:

- *przerwanie działania* (ang. crash failures) – proces przestaje działać,
- *błędy bizantyjskie* (ang. Byzantine failures) – proces nazywamy *bizantyjskim* (ang. Byzantine process); jeśli wykonuje on nieprawidłowe operacje, niezgodne ze swoim algorytmem. Komunikaty wysyłane przez proces bizantyjski mogą mieć dowolną zawartość;
- *błędy tymczasowe* (ang. transient failures) – przekłamania lub zagubienie pojedynczej porcji informacji, opóźnienia i inne; po ustaniu przyczyny błędu proces działa dalej poprawnie.

Notatki

8 Model synchroniczny all-port

W modelu *all-port* zakładamy:

1. gwarantowaną niezawodność obliczeń i komunikacji,
2. jednostkowy czas przesłania i odebrania komunikatów do wszystkich wierzchołków sąsiednich w grafie systemu,
3. pomijalnie mały czas obliczeń wykonywanych lokalnie
4. pełną synchronizację wszystkich wierzchołków – w każdym kroku algorytmu wszystkie wierzchołki jednocześnie wykonują obliczenia lokalne w czasie zerowym, a następnie wymieniają się komunikatami w czasie jednostkowym – taki krok wykonany równoległe przez wszystkie wierzchołki nazywamy *rundą*,

Notatki

5. brak globalnej wiedzy o systemie, znane są tylko: lokalny stan wierzchołka, sąsiedzi w grafie systemu,
6. struktura systemu nie zmienia się w trakcie działania algorytmu,
7. każdy wierzchołek ma swój unikalny identyfikator,
8. rozmiar przesyłanych komunikatów nie jest ograniczony,
9. liczba operacji obliczeniowych wykonywanych lokalnie nie jest ograniczona

Notatki

Model obliczeń all-port jest modelem bardzo silnym. Większość przyjętych tu założeń nie wydaje się realistyczna, jednak jest to model użyteczny i często stosowany do analizy algorytmów rozproszonych. Nie ma pełnej zgody, co do wszystkich założeń modelu, co za chwilę omówimy dokładniej.

Notatki

Niezawodność

W punkcie 1 przyjęliśmy, że kanały komunikacyjne są niezawodne, z własnością *fifo* i nieograniczoną pojemnością. Zapewnienie poprawnej komunikacji jest oddzielnym zagadnieniem i zakładamy, że jest realizowane w niższej warstwie dostępnego protokołu.

Notatki

Jednostkowy czas przesłania komunikatu

Czas przesłania komunikatu w systemach rzeczywistych różni się w zależności od jego długości, rodzaju połączenia, czy aktualnych warunków panujących w systemie. Precyzyjne uwzględnienie tych wszystkich czynników wydaje się bardzo trudne, dlatego przyjmujemy czas jednostkowy.

Notatki

Czas obliczeń lokalnych

Czas wykonywania prostych obliczeń jest niewspółmiernie mały w stosunku do czasu potrzebnego na komunikację. Dlatego czas ten nie jest brany pod uwagę w analizie i porównaniach. Więcej wątpliwości budzi założenie przyjęte w punkcie 9. Można nałożyć Postulować wielomianowy czas wykonania w zależności od rozmiaru danych. Przyjęcie takiego ograniczenia eliminuje pewne patologie.

Notatki

Bez tego ograniczenia można na przykład uzasadnić, że każdy problem algorytmiczny jest rozwiązywany w czasie proporcjonalnym do średnicy grafu systemu: Najpierw dane o całym grafie zbieramy w jednym wierzchołku, następnie bez wpływu czasu znajdujemy tam rozwiązanie optymalne po czym rozsyłamy odpowiedź do wszystkich węzłów sieci. Z drugiej strony, jeśli wielomian jest wysokiego stopnia to czas wielomianowy może i tak być zbyt długi do zastosowań. Ponadto typowo autorzy prac zwykle podkreślają prostotę i niską złożoność obliczeń lokalnych.

Notatki

Synchronizacja i pomiar czasu

Przez pomiar czasu w modelu all-port rozumiemy wyłącznie liczbę rund jaka jest potrzebna do wykonania algorytmu.

Założenie o pełnej synchronizacji systemu jest wygodne przy analizie złożoności obliczeniowej algorytmu. Natomiast często nie jest konieczne dla jego poprawności. Wielu autorów podkreśla, że ich algorytmy działają w systemie asynchronicznym. Powstał też pomysł zdefiniowania tak zwanej *rundy asynchronicznej*. Wykorzystanie w dowodzie poprawności założenia o pełnej synchronizacji osłabia wynik.

Notatki

Model one-port

Dodatkowym rygiorem może być uniemożliwienie w ciągu jednej rundy wymiany komunikatów z więcej niż jednym sąsiadem. W tym wypadku mówimy o modelu *one-port*.

Można ograniczyć model tak, by w ciągu jednej rundy każdy wierzchołek mógł wysłać do wszystkich sąsiadów tylko jedną, tę samą wiadomość (ang. broadcast).

Notatki

Zmienne globalne

Wiedza lokalna jest jedną z podstawowych różnic między systemami rozproszonymi i równoległymi. Przyjęcie w każdym z wierzchołków wiedzy o jakimkolwiek parametrze systemu, takim jak liczba wierzchołków lub maksymalny stopień, osłabia uzyskany wynik. Nie znaczy to jednak, że założenia takie są z gruntu nierealistyczne. Wiedza o niektórych parametrach może być znana a priori, co może wynikać na przykład ze specyfikacji sprzętu użytego do budowy systemu, a w niektórych wypadkach można przyjąć wartości szacunkowe.

Notatki

Struktura statyczna

W modelu all-port przyjmujemy, że system się nie zmienia w trakcie działania algorytmu. Ani nie przybywa, ani nie ubywa zarówno procesów jak i kanałów komunikacyjnych. Model, w którym zakłada się strukturę dynamiczną, jest oczywiście mniej restrykcyjny, a więc wynik uzyskany w takim modelu jest mocniejszy.

Notatki

Identyfikatory

Ze względu na możliwość występowania symetrii konieczne jest zastosowanie mechanizmów do ich łamania. Założenie przyjęte w modelu (punkt 8) o istnieniu unikalnych identyfikatorów jest założeniem silnym. Słabsza wersja zakłada unikalność tylko w sąsiedztwie każdego wierzchołka.

Notatki