

Algorytmy Równoległe i Rozproszone

Część VIII - Systemy rozproszone, c. d.

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	Wybór lidera	2
2	Demony	5
3	Łamanie symetrii	6
4	Bizantyjscy generałowie	7

1 Wybór lidera w cyklu c. d.

Znamy prosty algorytm dla cyklu, w którym wysłanych jest $O(n^2)$ wiadomości. Pokażemy teraz metodę, która gwarantuje $O(n \log n)$ komunikatów.

Przez k -sąsiedztwo wierzchołka p_i rozumiemy zbiór wierzchołków w odległości nie większej niż k od p_i . Dla cyklu k -sąsiedztwo zawiera dokładnie $\min n, 2k + 1$ wierzchołków.

Notatki

Algorytm przebiega etapami.

W l -tym etapie każdy z wierzchołków próbuje zostać liderem w swoim 2^l -sąsiedztwie. Tylko wierzchołki którym się to uda, kontynuują działanie w kolejnym etapie.

Każda wiadomość zawiera trzy pola:

- id - identyfikator nadawcy,
- l - numer etapu,
- hc - licznik przesłań.
- f - znacznik powrotu

Notatki

Na początku każdej fazy każdy aktywny wierzchołek rozsyła w obu kierunkach wiadomości z własnym identyfikatorem, numerem etapu, licznikiem przesłań ustawionym na 0 i nieustawioną flagą powrotu. Tak jak w poprzednim algorytmie przekazywana dalej jest tylko otrzymana wiadomość o identyfikatorze większym niż własny. Jeśli $l = 2^{hc}$ ustawiany jest f , a wiadomość odsyłana z powrotem.

Notatki

Lemat 1 Dla $l > 1$ liczba aktywnych procesorów jest mniejsza lub równa od $n/2^{l-1}$.

Ponadto każda z dwóch wiadomości wysłanych przez tymczasowego lidera przechodzi odległość $2 * 2^l$, co w sumie daje $2 * 2 * 2^l * n / 2^{l-1} = 8n$ wiadomości w każdej fazie.

Notatki

Wybór lidera w grafie pełnym

Idea algorytmu

Każdy wierzchołek „budzi się”, po czym próbuje zwerbować jak najwięcej wierzchołków do swojego „królestwa”. Mniejsze królestwo rozpada się, napotykając większe.

Notatki

Podamy pseudokod dla wierzchołka p_i używając następujących komunikatów:

- **Collect(j,s)**: wiadomość do p_j - mam już s wierzchołków w swoim królestwie, więc i ty przyłącz się do mnie.
- **Join(j,s)**: o wielki p_j przyłączam się do twego królestwa.
- **Check(j,s)**: sprawdź, czy twoje królestwo jest większe niż królestwo p_j z s wierzchołkami.
- **Ack(j,s)** Potwierdzenie, że królestwo p_j jest większe od naszego.

Notatki

Oraz zmiennych lokalnych:

- K - królestwo p_i w formie identyfikatora posiadacza, inicjalnie pusty (\perp).
- SK - rozmiar K , jeśli $K = i$
- $waiting$ - wierzchołek, który chce przeciągnąć p_i do swego królestwa.

Notatki

Algorytm 1: Wybór lidera w grafie pełnym

if procesor sam się obudził **then**

$K = i$

$KS = 1$

$waiting = i$

 wyślij $Collect(i, 1)$ do któregoś z sąsiadów

end if

Notatki

if odebrano $Collect(j, s)$ **then**

if $K = \perp$ **then**

$K = j$

 wyślij $Join(j, s)$ do p_j

else

$waiting = j$

 wyślij $Check(j, s)$ do K

end if

end if

if odebrano $Check(j, s)$ od p_l **then**

if $K < s$ **then**

$waiting = \perp$

$K = \perp$

 wyślij $Ack(j, s)$ do p_l

end if

end if

Notatki

```

if odebrano  $Ack(j, s)$  od  $p_l$  then
   $waiting = \perp$ 
  wyślij  $Join(j, s)$  do  $p_j$ 
end if
if odebrano  $Join(i, s)$  od  $p_l$  then
  if  $waiting \neq \perp$  then
     $KS = KS + 1$ 
    wyślij  $Collect(j, KS)$  do  $p_{l+1}$ 
  end if
end if

```

Notatki

2 Systemy z demonem

(ang. daemon, scheduler, adversary)

Demon „steruje systemem” wskazując wierzchołki do wykonania, kolejnego kroku obliczeń.

- Demon centralny
- Demon rozproszony
- R/W atomicity

Notatki

Algorytmy w modelu ze zmiennymi współdzielonymi i demonem

Niech p będzie warunkiem, a M akcją. Algorytm dla każdego wierzchołka u jest dany za pomocą reguł postaci:

```

R:   if  $p(u)$ 
       then  $M$ ,

```

Warunek p może dotyczyć stanu wierzchołka u oraz stanów wierzchołków z sąsiedztwa wierzchołka u , $N(u) = \{v \mid \{v, u\} \in E\}$. Jeśli warunek p dla wierzchołka u jest spełniony, to mówimy, że wierzchołek jest *aktywny*.

Notatki

Demon centralny

Demon centralny spośród wierzchołków aktywnych wybiera jeden, który jako kolejny wykona ruch, w ten sposób żadne dwa ruchy nie są wykonywane w tym samym czasie.

Notatki

Demon rozproszony

Demon rozproszony spośród wierzchołków aktywnych wybiera dowolny podzbiór tych, które jako kolejne wykonają ruch.

Notatki

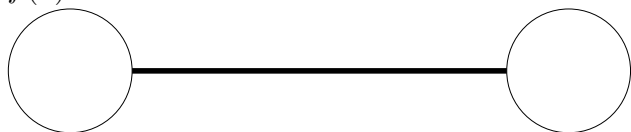
R/W atomicity

Przy konstruowaniu reguł pozwalaliśmy odczytać stany lokalne wszystkich wierzchołków w jednym kroku. Tu wprowadzamy ograniczenie do stanu tylko jednego sąsiada na ruch.

Notatki

3 Problem łamania symetrii

Rozpatrzmy graf jak na rysunku, w którym każdy wierzchołek v ma jedną zmienną lokalną f , inicjalnie $f(v) = 0$.



Algorytm 2: color

R1: if $\exists_{u \in N(v)} f(u) = f(v)$
then $f(v) + = 1$

Notatki

Ćwiczenie 1

Porównaj działanie algorytmu w modelu z demonem centralnym i rozproszonym.

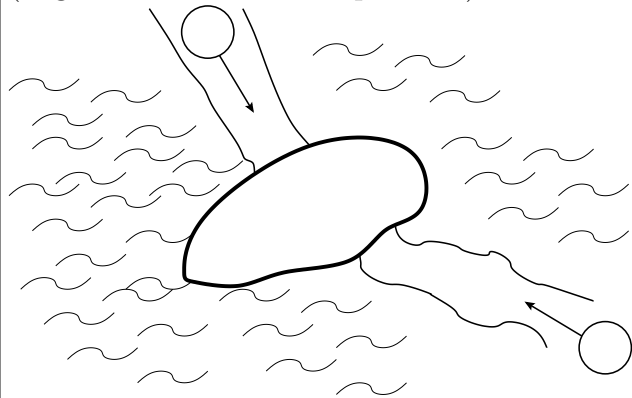
Ćwiczenie 2

Napisz odpowiedni algorytm dla modelu z przesyłaniem komunikatów.

Notatki

4 Problem Bizantyjskich generałów

(ang. coordinated attack problem)



Notatki

Dla dwóch procesorów p_i , jednobitowego wejścia: x_i oraz jednobitowego wyjścia y_i , gdzie $i = 1, 2$; nie istnieje algorytm, który komunikując się przez zawodne medium (wiadomość może być dostarczona w całości poprawnie, lub wcale) i spełnia trzy warunki

- Zgodność: $y_1 = y_2$;
- Poprawność: jeśli $x_1 = x_2 = 0$, to $y_1 = y_2 = 0$;
- Nie-trywialność: istnieje wykonanie, dla którego $y_1 = y_2 = 1$;

Notatki

Dowód nie wprost. Przypuśćmy, że taki algorytm istnieje. Rozpatrzmy wykonania, które prowadzą do zgodnej odpowiedzi $y_1 = y_2 = 1$ (nietrywialność). Spośród nich wybierzmy, te w której odebrano najmniej wiadomości. Odebrano co najmniej 1 wiadomość (poprawność i zgodność). Bez utraty ogólności możemy przyjąć, że ostatnią wiadomość otrzymał p_1 . Rozpatrzmy wykonanie, w którym ostatnia wiadomość zaginęła. Wtedy również $y_1 = y_2 = 1$ (zgodność). Sprzeczność.

Notatki