UMEÅ UNIVERSITET
Institutionen för datavetenskap
Robert Granat, Lars Karlsson
och Bo Kågström

Design och analys av algoritmer
för parallelldatorsystem – vt2006
22:e augusti 2006

Tentamen på kursen

Design och analys av algoritmer för parallelldatorsystem

måndagen den 22:e augusti 2006
kl 9 - 15 i Östra Paviljongen 7

Tentamen kan maximalt ge 60 poäng. För betyget 3 krävs 30 poäng, för betyget 4 krävs 40 poäng och för betyget 5 krävs 50 poäng. Poängtal anges inom parentes för varje uppgift. Uppgifterna är ej placerade i svårighetsordning.
Hjälpmedel:

- Miniräknare.

Skrivsalen får lämnas tidigast en halvtimme efter att skrivningen påbörjats.

OBS!!!
Skriv namn på varje blad och börja varje uppgift på nytt blad. Var noga med att redovisa alla beräkningar. Markera på försättsbladet vilka uppgifter du lämnar in lösning till och lägg lösningarna i nummerordning.
Skriv TYDLIGT!

Lycka Till!

Uppgift 1 - Skalbarhetsanalys (1+1+1+1+2+2+2=10p)
Kostnaden (mätt i tid) för en sekventiell algoritm för ett givet problem är $T_1 = n^2 t_a$ där vi för enkelhets skull antar $t_a = 1$. En parallell motsvarighet till algoritmen har kostnaden $T_p = n^2/p + p^{3/2} n t_s + n t_w$ då den exekverar på p processorer.

a. Förklara de olika komponenterna i uttrycket för T_p med avseende på aritmetisk kostnad, kostnad för kommunikation, antal paket som kommuniceras samt genomsnittlig storlek på paketen. (1p)

b. Beräkna explicita uttryck för den parallella algoritmens uppsnabbning (*parallel speedup*) och effektivitet (*parallel efficiency*). (1p)

c. Avgör om (och i så fall under vilka förutsättningar) algoritmen är kostnadsoptimal (*cost optimal*). (1p)

d. Förklara vad som avses med begreppet skalbarhet (*scalability*). Vad säger isoeffektivitetsfunktionen om en algoritms skalbarhet? (1p)

e. Beräkna ett explicit uttryck för algoritmens asymptotiska isoeffektivitetsfunktion (*isoefficiency function*). (2p)

f. Antag att $t_s = 100$ och $t_w = 10$. Beräkna med fyra decimalers noggrannhet effektiviteten då $p = 10$ och $n = 10^4$.

Hur stort ska, enligt iso-effektivitetsfunktionen, problemet vara för att samma effektivitet ska erhållas för tio gånger så många processorer (dvs $\tilde{p} = 100$)? Ange ditt svar som storlek för aktuellt n-värde, betecknat $\tilde{n}$. (2p)

g. Beräkna med samma antagande som ovan effektiviteten för $\tilde{p} = 100$ processorer och det värde på $\tilde{n}$ du beräknat ovan. Ange effektiviteten med fyra korrekta decimaler.

Vad kan sägas om detta värde jämfört med det som beräknades i ovanstående deluppgift? Förklara eventuella skillnader. Vad kan sägas om utvecklingen av effektiviteten för ytterliggare större processorantal (och problemstorlekar skalade i enlighet med isoeffektivitetsfunktionen)? (2p)

Uppgift 2 - Grafalgoritmer (5+5=10p)
Floyds algoritm för kortaste vägen mellan alla par är given i Figur 1.

procedure FLOYD_ALL_PAIRS
begin
  $D^{(0)} = A$
  for $k := 1$ to n
    for $i := 1$ to n
      for $j := 1$ do n
        $d_{i,j}^{(k)} := \min\left(d_{i,j}^{(k)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\right)$
end

Figure 1: Floyds algoritm för kortaste vägen mellan alla par av hörn i en viktad graf.

a. Konstruera en effektiv parallell motsvarighet då matrisen $D^{(k)}$ är $\mathbf{2-D}$ blockpartionerad, där varje processor håller $n/\sqrt{p}$ rader och kolumner av $D^{(k)}$. (5p)

b. Beräkna tiden T_p för den parallella algoritmen från uppgift a, dess uppsnabbning S_p, samt dess effektivitet, given en hyperkub med *cut-through routing*. (5p)

Uppgift 3 - Lastbalansering (3+3+3+1=10p)
Redogör kortfattat hur följande tre strategier för lastbalansering i parallella algoritmer för *djupet-först-sökning* (depth-first searching) fungerar. Ange också en fördel och en nackdel med varje metod.

a. Asynkron Round Robin (3p)

b. Global Round Robin (3p)

c. Random Polling (3p)

Vad får dålig lastbalans för effekt på den parallella prestandan vid djupet-först-sökning? (1p)

Uppgift 4 - Linjär algebra (5+5=10p)
I denna uppgift skall du konstruera en parallel algoritm för matris-vektormultiplikation.

a. Beskriv en parallell algoritm för operationen

$$y = Ax,$$

där matrisen $A \in R^{n \times n}$ är $\mathbf{1-D}$ kolumn-blockpartionerad och elementen i vektorerna x och y är jämnt fördelade över processorerna. (5p)

b. Visa att den parallella exekveringstiden blir densamma som i deluppgift a om A partitioneras rad-blockvis istället. (5p)

Uppgift 5 - Kollektiva kommunikationsoperationer (10p)
Givet ett nätverk med dubbelriktade länkar i form av ett balanserat binärt träd med p processorer, beskriv en algoritm för att utföra en alla-till-alla utsänding (*all-to-all broadcast*) som kostar $(t_s + t_w mp/2) \log p$ för meddelanden av storlek m. Antag att alla processorerna är löv-noder i trädet och att det kostar $t_s + t_w mk$ att utbyta två meddelanden av storlek m mellan två grann-noder i trädet över en länk som (helt eller delvis) delas av k stycken samtidiga meddelanden.

Uppgift 6 - Dynamisk programmering (2+4+4=10p)
Givet en sekvens $A =< a_1, a_2, \ldots, a_n >$ ges en delsekvens genom att ta bort några element ur A. (M.a.o. behöver delsekvensens element inte vara konsekutiva element från A, men de måste vara i samma ordning som i A.) Exempelvis är $< b, c, e >$ en delsekvens av $< a, b, c, d, e, f >$ medan $< b, c, g >$ och $< c, b, e >$ inte är det. Den längsta gemensamma delsekvensen av två sekvenser $A =< a_1, a_2, \ldots, a_n >$ och $B =< b_1, b_2, \ldots, b_m >$ är den längsta sekvens som är delsekvens av både A och B.

Låt $F[i,j]$ vara längden av den längsta gemensamma delsekvensen av $< a_1, a_2, \ldots, a_i >$ och $< b_1, b_2, \ldots, b_j >$. Då är $F[n,m]$ längden av den längsta delsekvensen av A och B. $F[n,m]$ kan härledas från

$$F[i,j] = \begin{cases} 0 & \text{om } i = 0 \text{ eller } j = 0 \\ F[i-1, j-1] + 1 & \text{om } i,j > 0 \text{ och } a_i = b_j \\ \max\{F[i, j-1], F[i-1, j]\} & \text{om } i,j > 0 \text{ och } a_i \neq b_j \end{cases}$$

a. Utgående från den rekursiva formuleringen ovan kan man beräkna $F[i,j]$ m h a dynamisk programmering (DP). Vilken klassificering får detta DP-problem utav följande fyra alternativ: *seriell monadisk*, *icke-seriell monadisk*, *seriell polyadisk*, *icke-seriell polyadisk*? (2p)

b. Konstruera en effektiv parallell algoritm för att beräkna $F[n,m]$ på ett två-dimensionellt nät med p processorer där $m, n > p$. (För att undvika krångel med specialfall kan du själv sätta och ange förutsättningar på relationer mellan p, m och n.) (4p)

c. Härled uttryck för totala exekveringstiden T_p, uppsnabbningen S_p och effektiviteten E_p för algoritmen. Är algoritmen kostnadsoptimal? (4p)