UMEÅ UNIVERSITET
Institutionen för datavetenskap
Isak Jonsson efter material från
Erik Elmroth och Bo Kågström

Tentamen på kursen

Design och analys av algoritmer för parallelldatorsystem

torsdagen den 3:e juni 2003
kl 9 - 15 i Noliahallen

Tentamen kan maximalt ge 40 poäng. För betyget 3 krävs 20 poäng, för betyget 4 krävs 26 poäng och för betyget 5 krävs 32 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 (1+1+1+1+1+2+2=9p)
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 $T_p = n^2/p + pnt_s + nt_w$ 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.

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

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

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

e. Beräkna ett explicit uttryck för algoritmens asymptotiska isoeffektivitetsfunktion (*iso-efficiency function*).

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 $\tilde{n}$.

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 ytterligare större processorantal (och problemstorlekar skalade i enlighet med isoeffektivitetsfunktionen).

Uppgift 2 (4p + 2p + 2p = 8p)
Givet en parallelldator med ett 2-dimensionellt kvadratiskt *store-and-forward* kommunikationsnätverk med *wrap-around*.

a. Konstruera en algoritm för att utföra en *k-to-all broadcast*, dvs en operation där k processorer samtidigt gör broadcast till alla andra. Antag att $k \leq \sqrt{p}$ och att de k sändande processorerna befinner sig i olika rader i processornätet.

b. Härled tiden för att utföra operationen i uppgift **a**. Hur förhåller sig denna tid i det generella fallet till tiden för att utföra en vanlig broadcast från endast en processor.

c. Vilka modifieringar behöver göras för att algoritmen ska klara av fall då flera sändande processorer befinner sig i samma processorrad? Vilken skillnad ger detta för tiden att utföra operationen?

Uppgift 3 (4+6+2=12p)
För att uppnå hög prestanda krävs inte bara snabba processorer utan även att data finns tillgängligt när operationerna ska utföras. För att uppnå hög prestanda i ett program måste man därför ta hänsyn till systemets minneshierarki.

a. Beskriv hur man vanligtvis försöker organisera beräkningarna (i samband med täta matriser) för att utnyttja minneshierarkin på bästa sätt.

Definiera begreppen temporal lokalitet och spatial lokalitet och hur dessa påverkas av olika typer av blockningstekniker (för beräkningar och datalagring).

b. GEMM-operationen utför en matrismultiplikation av typen

$$C = \beta \cdot C + \alpha \cdot op(A)\,op(B),$$

där $op(X) = X$ eller X^T, och α, β är skalärer. I denna uppgift ska ni studera följande matrisproduktoperation:

$$C = C + A^T \cdot X \cdot B,$$

där matrisdimensionerna för matriserna C, A, X och B är godtyckliga, med bivillkoret att operationen är väldefinierad.

Härled ett uttryck för det minimala antalet operationer (additioner och multiplikationer) som krävs för att utföra denna operation.

Formulera en level-3 algoritm för att utföra matrisproduktoperationen. Motivera och diskutera algoritmens presumptiva prestanda och behov av temporalt arbetsminne.

c. Inom numerisk linjär algebra försöker man i biblioteket LAPACK uppnå god prestanda och god portabilitet. Beskriv kortfattat hur detta koncept är uppbyggt och förklara hur på vilket sätt man tänker sig uppnå både hög prestande och god portabilitet.

Uppgift 4 (2+3=5p)
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 **1−D** blockpartionerad, där varje processor har håller n/p kolumnder av $D^{(k)}$. Vilka fördelar och nackdelar har denna algoritm jämfört med en algoritm som opererar på en 2−D schackbrädespartionerad matris?

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

Uppgift 5 (4+2=6p)
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. 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.)

b. Härled uttryck för totala exekveringstiden T_p, uppsnabbningen S och effektiviteten E för algoritmen. Är algoritmen kostnadsoptimal?