UMEÅ UNIVERSITET
Institutionen för datavetenskap
Mikael Rännar (Erik Elmroth,
Robert Granat och Bo Kågström)

Tentamen på kursen

Design och analys av algoritmer för parallelldatorsystem

onsdagen den 1 juni 2005
kl 9 - 15 i Skrivsal 7

Tentamen kan maximalt ge 44 poäng. För betyget 3 krävs 22 poäng, för betyget 4 krävs 29 poäng och för betyget 5 krävs 35 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+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 $T_p = n^2/p + pt_s + pnt_w$ 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 (*isoefficiency 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 decimaler.

Vad kan sägas om detta värde jämfört med det som beräknades i deluppgift 1f. Förklara eventuella skillnader. Vad kan sägas om utvecklingen av effektiviteten för ytterliggare större processorantal (och problemstorlekar skalade i enlighet med isoeffektivitetsfunktionen).

Uppgift 2 *Biblioteksprogramvara* (4+2+2=8p)
I denna uppgift skall du berätta vad du känner till om programvarubiblioteket ScaLAPACK.

a. Redogör för *mjukvaruhierarkin* i ScaLAPACK. Var noga med att skilja på komponenter som opererar *globalt* respektive *lokalt*. Beskriv kortfattat vad de olika komponenterna innehåller för typ av rutiner.

b. Grundläggande i ScaLAPACK är att man endast har en typ av *processortopologi* och ett enhetligt sätt att *distribuera data*. Vilken topologi och vilken typ av datadistribution avses?

c. I ScaLAPACK utnyttjas en *objektorienterad ansats* för att beskriva hur datastrukturerna är distribuerade över de ingående processorerna. Beskriv hur detta är konstruerat.

Uppgift 3 *Principer för design av parallella algoritmer* (5+1+2+2+2=12p)
Denna uppgift handlar om grundläggande principer för design av parallella algoritmer. Svara så utförligt och noggrant som du kan.

a. En seriell algoritm beskrivs ofta som en sekvens av steg för att lösa ett givet problem. Ange (minst) fem saker som tillkommer vid icke-trivial design av en parallell algoritm för att lösa ett givet problem. Ge exempel för att illustrera din framställning.

b. En parallell algoritm kan ibland illustreras som en ett antal *uppgifter* (tasks) i en *beroendegraf* (dependency graph). Redogör för följande:

I. Vad representerar noderna och kanterna i en beroendegraf?

II. Vad är den *kristiska vägen* (critical path) för en beroendegraf? Hur definieras *längden av den kritiska vägen*?

III. Visa hur man kan utnyttja längden av den kritiska vägen för att beräkna *genomsnittlig grad av parallellitet* (average degree of concurrency) för den parallella algoritmen och dra slutsatser om sambandet mellan den kritiska vägen och graden av parallellitet.

c. Storleken på de olika uppgifterna i förhållande till hela problemet brukar kallas för *kornigheten* (granularity). Graden av parallellitet kan ofta ökas genom att man ökar finkornigheten. Ange två saker som sätter gränser för hur mycket finkornigheten kan ökas.

Uppgift 4 *Lastbalansering* (2+2+2=6p)
Redogör kortfattat hur följande tre, i kursboken beskrivna, 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

b. Global Round Robin

c. Random Polling

Uppgift 5 *Parallell grafalgoritm* (6+3=9p)
Givet en riktad och viktad graf med hörnen $V = \{v_1, v_2, ..., v_n\}$. Låt $D^{(k)}$ vara en matris i vilken element $d_{i,j}^{(k)}$ är längden av den kortaste vägen från v_i till v_j som endast går via hörn i mängden $\{v_1, v_2, ..., v_k\}$. I matrisen $D^{(0)}$ innehåller element $d_{i,j}^{(0)}$ kostnaden för kanten $\{v_i, v_j\}$ om den existerar, annars värdet ∞. Kortaste vägen mellan alla par av hörn kan beräknas med algoritmen 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.

Algoritmen tar matrisen $D^{(0)}$ som inparameter och den har slutligen beräknat matrisen $D^{(n)}$, där $d_{i,j}^{(n)}$ innehåller kortaste vägen mellan hörnen v_i och v_j. Huvudsteget är att kortaste vägen mellan varje par av hörn via hörnen $\{v_1, v_2, ..., v_k\}$ beräknas som minimum av kortaste vägen via hörnen $\{v_1, v_2, ..., v_{k-1}\}$ och kortaste vägen som dessutom går via v_k. (Floyd's Algorithm)

a. Konstruera en effektiv parallell motsvarighet till algoritmen i Figur 1 då matrisen $D^{(0)}$ är schackbrädespartitionerad över ett 2-dim logiskt processornät av storlek $\sqrt{p} \times \sqrt{p}$. Antag att n är en multipel av $\sqrt{p}$. Låt processor $P_{s,t}$ beteckna processorn på position (s, t) i det logiska nätet. Antag vidare att det logiska 2-dimensionella nätet är inbäddat i en hyperkub.

b. Härled uttryck för tiden T_p för den parallella algoritmen från uppgift a, dess uppsnabbning S_p, samt dess effektivitet E_p.