UMEÅ UNIVERSITET
Institutionen för datavetenskap
Erik Elmroth och Bo Kågström

Tentamen på kursen Parallella Beräkningar II

tisdagen den 19 december 1995
kl 9 - 15 i skrivsal 2

Tentamen kan maximalt ge 52 poäng. För betyget godkänd krävs 26 och för betyget väl godkänd 39 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 (2p + 2p + 2p + 2p = 8p)
Givet en hyperkub med *store-and-forward* nätverk.

a. Konstruera en algoritm för att utföra 1-till-alla personlig kommunikation, där alltså en processor skickar olika meddelande till varje annan processor. Antag att meddelandet till varje processor är av storlek m ord. Beskriv algoritmen på hög nivå. Illustrera gärna med något exempel (t ex 8 processorer) istället för att gå in på för mycket detaljer i själva algoritmen.

b. Beräkna den tid det tar att utföra operationen i uppgift a. Skulle algoritmen bli snabbare om nätverket vore av typ *cut-through*? (Motivera ditt svar.)

c. Kommunikationsoperationen alla-till-alla personlig kommunikation kan naturligtvis utföras genom att alla processorer utför 1-till-alla personlig kommunikation. Din uppgift är att konstruera en effektivare algoritm för alla-till-alla personlig kommunikation. Ge algoritmbeskrivning på samma nivå som i uppgift a.

d. Beräkna den tid det tar att utföra operationen i uppgift c. Skulle algoritmen bli snabbare om nätverket vore av typ *cut-through*? (Motivera ditt svar.)

Uppgift 2 (1p + 1p + 3p = 5p)
Denna uppgift behandlar analys av en algoritms skalbarhet.

a. Vad innebär det att en algoritm är skalbar och vad säger den asymptotiska isoeffektivitetsfunktionen om en algoritms skalbarhet?

b. Existerar den asymptotiska isoeffektivitetsfunktionen för alla algoritmer? Motivera ditt svar.

c. En parallel algoritm har arbetsmängden $W = n^3$ och tiden $T_p = \frac{n^3}{p} + pt_s + n^2 t_w$ på p processorer. Beräkna algoritmens asymptotiska isoeffektivitetsfunktion.

Uppgift 3 (3p + 3p + 3p = 9p)
I algoritmer för djupet-först-sökning i exempelvis trädstrukturer känner man inte på förhand till hur mycket arbete som kommer att krävas eller vilka delar av trädet som skall sökas igenom. Vid konstruktion av parallella djupet-först-algoritmer ställs man därför inför problem med lastbalansering och hur man skall kunna avgöra om algoritmen har terminerat.

a. Ge en beskrivning på hög nivå till hur lastbalansering kan ske i parallella djupet-först-algoritmer. Med andra ord, vad gör en processor som blir utan arbete? Vad gör de som redan har arbete? Ge exempel på olika sätt att utföra detta.

b. Förklara hur Dijkstras *token detection*- algoritm för att upptäcka terminering fungerar. Förklara algoritmen med hjälp av ett *illustrativt* exempel med 4 processorer.

c. Vid parallellisering av *bäst-först*-algoritmer för sökning i trädstrukturer ställs man inför andra problem vad gäller att lastbalansering och hantering av data som måste finnas tillgänglig för alla processorer. Ge en beskrivning till dessa problem, samt ge exempel på hur de kan lösas.

Uppgift 4 (3p)
Beskriv kortfattat idén bakom GEMM-based level 3 BLAS. Ange några fördelar med att utveckla ett bibliotek för level 3 BLAS utifrån detta koncept. Beskriv kortfattat hur level 3 BLAS kan tillämpas i samband med matrisberäkningar.

Uppgift 5 (1p + 6p + 2p + 4p = 13p)
Vid lösning av ett system av ekvationer $Ax = b$ så består det första steget i att beräkna en LU-faktorisering av A. Det andra steget innebär att lösa två triangulära system med framåt- respektive bakåtsubstitution. Denna uppgift behandlar konstruktion och analys av en parallell algoritm för lösning av ett triangulärt system $Ux = y$ där U har storlek $n \times n$ och är övertriangulär samt icke-singulär.

a. Ge en beskrivning (t ex i pseudokod) av en sekventiell algoritm för att lösa $Ux = y$, dvs elementet x_k kan uttryckas som $x_k = (y_k - \sum_{j=k+1}^{n} u_{kj} x_j)/u_{kk}$. Ange ett uttryck för algortitmens aritmetiska komplexitet uttryckt i antalet flyttalsoperationer.

b. Utgående från a-uppgiften ge en beskrivning (t ex i pseudokod) av en parallell algoritm för att lösa $Ux = y$, där U är schackbrädespartitionerad över ett 2-dim processornät av storlek $\sqrt{p} \times \sqrt{p}$ (*store-and-forward*-nätverk). Algoritmen ska så långt som möjligt använda pipelining mellan beräkningar och kommunikation. Beskriv i vilka sammanhang och hur detta sker. Använd gärna figurer för att åskådliggöra algoritmbeskrivningen.

c. Härled ett uttryck för den parallella exekveringstiden av algoritmen i b-uppgiften, innefattande både kostnader för aritmetik och kommunikation. Är den parallella algoritmen kostnadsoptimal (*cost optimal*)?

(Eventuellt användbara kommunikationstider för meddelanden av storlek m ord: granne-granne-kommunikation tar tiden $t_s + t_w m$, 1-till-alla broadcast i en ring av q processorer, t ex en processorrad i nätet, tar tiden $(t_s + t_w m)\lceil \frac{q}{2} \rceil$ och 1-till-alla broadcast i ett 2-dimensionellt nät med q processorer tar tiden $2(t_s + t_w m)\lceil \frac{\sqrt{q}}{2} \rceil$.)

d. Beskriv och visa att det går att konstruera en kostnadsoptimal parallell algoritm för att lösa $Ux = y$ med utnyttjande av p processorer (dvs lika många som i b-uppgiften).

Uppgift 6 (7p + 3p + 4p = 14p)
Givet en riktad och viktad graf med hörnen $V = \{v_1, v_2, \ldots, v_n\}$. Låt $D^{(k)}$ vara en matris i vilken element $d_{i,j}^{(k)}$ är längden på den kortaste vägen från v_i till v_j som endast går via hörn i mängden $\{v_1, v_2, \ldots, v_k\}$. 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.

Algoritmen tar matrisen $D^{(0)}$ som inparameter och den har slutligen beräknat matrisen $D^{(n)}$, där $d_{i,j}^{(n)}$ innehåller kortaste vägen mellan hörn v_i och hörn v_j. Huvudsteget är att kortaste vägen mellan varje par av hörn via hörnen $\{v_1, v_2, \ldots, v_k\}$ beräknas som minimum av kortaste vägen via hörnen $\{v_1, v_2, \ldots, v_{k-1}\}$ och kortaste vägen som dessutom går via v_k. (Observera att detta inte är den matrismultiplikationsbaserade algoritm som tagits upp på föreläsningarna.)

Algoritm Kortaste vägen: alla par (input $D^{(0)}$)
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            $d_{i,j}^{(k)} \leftarrow \min\{d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\}$

Figur 1: Algoritm för att beräkna kortaste vägen mellan alla par av hörn i en graf.

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 med *cut-through routing*, så att varje processorrad och varje processorkolumn utgörs av en äkta delkub.

b. Beräkna tiden T_p för den parallella algoritmen från uppgift a, dess uppsnabbning S, samt dess effektivitet.

(Eventuellt användbara kommunikationstider för meddelanden av storlek m ord: granne-granne-kommunikation tar tiden $t_s + t_w m$, 1-till-alla broadcast eller 1-till-alla ackumulering på hyperkub med q processorer tar tiden $(t_s + t_w m) \log q$.)

c. Avgör om (eller när) algoritmen är kostnadsoptimal, samt beräkna algoritmens isoeffektivitetsfunktioner med avseende på kommunikationsoverhead, parallellitet, samt totalt (*overall*).