UMEÅ UNIVERSITET
Institutionen för datavetenskap
Granat, Karlsson, Kågström

Design och analys av algoritmer
för parallelldatorsystem – VT06
Laboration 2
12 mars 2007

Parallell Matrismultiplikation

Sista inlämningsdag: 2007-04-19.

Laborationen, som får lösas individuellt eller parvis, består av två delar som tillsammans behandlar implementation och analys av en algoritm för matrismultiplikation. Det står fritt att välja en av följande algoritmer: en enkel blockalgoritm (avsnitt 8.2.1 i kursboken), Cannons algoritm (avsnitt 8.2.2) och Foxs algoritm (information om algoritmen kan sökas på nätet). I laborationen läggs stor vikt vid analys av algoritmens prestanda.

Uppgifterna skall dokumenteras väl och alla resultat skall följas av diskussioner som visar att ni förstår resultaten. Även implementationen skall dokumenteras väl och testkörningar som visar att programmet räknar rätt skall bifogas. Ange sökväg till körbara filer och ge instruktioner till hur programmet körs.

Implementation av algoritm för matrismultiplikation

Implementera algoritmen för matrismultiplikation modifierad så att den beräknar $C = \beta C + \alpha AB^T$, där A, B och C är reella $m \times m$ matriser och α och β är reella tal. Notera att ni ska räkna på transponatet av B. Implementationen kan utvecklas i Linuxlabben med hjälp av LAM (en MPI-implementation). Se `http://www.cs.umu.se/kurser/TDBC20/HT03/lam.html` för en introduktion. Processorkonfigurationen betraktas som ett 2-dimensionellt nät med p processorer. De avslutande experimenten och prestandatesterna skall göras på Sarek eller Seth vid HPC2N.

Ni kan förutsätta att det 2-dimensionella processornätet är kvadratiskt. Ni får också förutsätta att matriserna A, B och C är kvadratiska av dimensioner som går jämnt ut på antalet processorer i aktuell dimension av processornätet samt att varje matriselement samt α och β är av typen `double`. För att minska kodningsarbetet finns skelett till program tillsammans med Makefile på `http://www.cs.umu.se/~larsk/lab2skel.tar.gz`. Skelettprogrammet är konstruerat för att beräkna $C = \beta C + \alpha AB$ (dvs B utan transponering). Modifiera testprogrammet så att korrekta feltester görs för det fall ni ska implementera.

Observera att inledningsvis ska nod (i, j) hålla blocken A_{ij}, B_{ij} och C_{ij}. Dvs, all omdistribuering av data skall utföras i samband med matrismultiplikationen (och alltså inkluderas i tidsmätningarna).

För att erhålla bra prestanda på varje enskild processor skall de lokala matrismultiplikationerna utföras genom anrop till BLAS-rutinen `DGEMM` i något optimerat BLAS-bibliotek. På Sarek rekommenderas GotoBLAS (`libgoto-2`) och på Seth rekommenderas ATLAS (`libatlas`).

Analys av matrismultiplikationsalgoritmen

Gör tester och redovisa tider, prestanda (t.ex. i Mflops/sekund), uppsnabbning och effektivitet för diverse olika problemstorlekar, t.ex. $m = 500, 1000, 1500, \ldots, m_{max}$, där m_{max} är den största problemstorleken som går att lösa med en processor. Det är fritt att välja andra problemstorlekar, men försök undvik storlekar som är en potens av två eftersom det kan uppkomma cachekonflikter som påverkar prestandan negativt.

I kursboken finns uttryck för tiden T_p på p processorer för både Cannons algoritm och den enkla matrismultiplikationen. För Foxs algoritm kan ni göra en egen analys alternativt leta på nätet. Modifiera uttrycket så att det bättre stämmer med er implementation. T.ex. måste kostnader för transponeringen tas med och om implementationen inte gör mottagning och sändning samtidigt måste konstanterna för kommunikationen modifieras på motsvarande sätt.

Tag experimentellt fram ett approximativt värde på tiden t_a för att utföra en flyttalsoperation utifrån testerna på 1 processor och det största kvadratiska problemet. Sätt in t_a och aktuella värden för kommunikationsparametrarna t_s och t_w (dessa måste ni bestämma på lämpligt sätt, t.ex. med programmet ping-pong som kan hittas på HPC2Ns supportsida) i uttrycket och beräkna teoretiska resultat för samma problem som tidigare har testkörts. Redogör för skillnaden mellan de teoretiska körtiderna och de praktiska och diskutera troliga orsaker till eventuella avvikelser. Utifrån testkörningarna på något litet antal processorer, t.ex. fyra, gör en anpassning av er tidsmodell (T_p) mot praktiskt uppmätta tider för dessa två fall:

- Betrakta t_a, t_w, t_s som okända.
- Betrakta endast t_w, t_s som okända. Detta motiveras av att t_a kan bestämmas mer exakt än kommunikationsparametrarna.

Kommentera i bägge fallen de eventuella skillnaderna mot de tidigare använda värdena.

Uttnyttja ert detaljerade uttryck för T_p för att göra följande skalbarhetsanalys. Utgå från den verkliga exekveringstiden på 4 processorer för en problemstorlek som ger en effektivitet kring $E = 90\%$. Beräkna hur stort problemet måste vara för betydligt fler processorer (t.ex. $p = 9, 16, 25, 36$) för att effektiviteten $E \approx 90\%$ skall erhållas. (Avrunda problemstorlekarna så att matriserna går jämnt ut på det aktuella processornätet.) Gör tester på dessa problemstorlekar och redovisa resultatet. Försök förklara eventuella avvikelser i effektiviteten.

För att beräkna hur stora problem som krävs ska ni använda två metoder:

1. Använd den asymptotiska isoeffektivitetsfunktionen för att bestämma hur en ökning i processorantal påverkar problemstorleken.
2. Använd det detaljerade T_p ni tagit fram och ställ upp ett uttryck för effektiviteten som ni sedan löser för m, antingen analytiskt eller numeriskt.

Eftersom punkt 2 antar effektivitetsjämförelser mellan olika problemstorlekar kan ni använda följande relation för att approximera E_p:

$$E_p \approx \frac{P_p(m_p)}{pP_1(m_1)}$$

där m_1 är antalet processorer och storlek på problemet för en processor och p, m_p är antalet processorer och storlek för det större antalet processer. Funktionen $P_p(m)$ är prestandan (mätt i t.ex. Mflops/sekund) för problemstorlek m och processorantal p. Formeln antar att $P_1(m_1) = P_1(m_p)$, dvs att för m_1 har praktisk peak nåtts och prestandan för det större problemet antas därför vara densamma.

Jämför hur väl överens de två olika metoderna är vad gäller problemstorlekar och kommentera uppmätta effektivitetsvärden. Använd formeln ovan för effektivitetsjämförelser, utgå från $m_1 = m_{max}$.