Factorizare LU utilizand BMM (Block matrix multiplication)

Deadline 20.04.2008

Descriere:

Se cere realizarea unui program multithread in Python pentru realizarea factorizarii LU a unei matrice A (N*N), utilizand algoritmul lui Doolittle, iar pentru inmultirea de matrice din cadrul acestui algoritm, metoda BMM - Block matrix multiplication - descrisa si in laboratorul 6.

Se vor realiza doua versiuni ale programului: una seriala si una multithread. Dimensiunea matricei A ce va fi factorizata, va fi fie citita de la tastatura, fie dintr-un fisier, in variabila N. In functie de sistemul utilizat, timpii de rulare trebuie sa fie de ordinul zecilor de secunde (peste 20s si pana in 120s), pentru varianta seriala neoptimizata (fara BMM). In cazul sistemelor cu mai multe core-uri/procesoare, ar trebui sa observati un speed-up semnificativ atat prin adaugarea de threaduri. Pentru toate sistemele ar trebui sa observati o imbunatatire a performantelor la utilizarea BMM. Ca ordin de marime, matricele pot avea intre 1.000 si 1.000.000 de elemente. Initializarea se va face cu numerele reale de la 1 la dimensiunea matricei - pentru a putea usor verifica corectitudinea algoritmului implementat: A[i][j] = (i-1)*N+j, cu i = 1...N.

Se doreste explorarea optimizarilor pentru Cache utilizand metoda BMM si a paralelizarii utilizand threaduri, specific sistemului pe care realizati tema. Din acest motiv, alaturi de programul ce rezolva tema, sunteti rugati sa adaugati si o **documentatie** ce va contine cel putin:

1. Specificatiile hardware are procesorului si ierarhiei dumneavoastra de memorii (nivele de Cache si memoria principala): viteza/dimensiune/latenta/largime de banda, etc.
2. Rulati programul pentru mai multe valori ale lui N - dimensiunea matricei. N variaza cu cel putin 2 ordine de marime (de exemplu intre 100 si 1.000 puteti alege N = 100, 250, 500, 750, 1.000). Alegeti cel putin 5 dimensiuni pentru matricea A. Pentru N astfel ales, variati acum dimensiunea blocurilor din algoritmul BMM, si realizati grafice unde pe orizontala specificati dimensiunea blocului si pe verticala timpul de executie al programului. Comentati graficele si rezultatele astfel obtinute. Graficele pot fi realizate in Matlab/Gnuplot/Excel si datele trebuiesc furnizate si in format text. Daca ati utilizat scripturi (Matlab sau Gnuplot) pentru realizarea graficelor (recomandat), va rugam sa le atasati de asemenea temei.
3. Realizati tot atatea grafice ca la punctul 2, dar de data aceasta, pe verticala, in loc de timpul de executie, calculati performanta in GFLOPS, a programului rulat. Precizati si motivati de asemenea care este performanta maxima pe care o poate atinge sistemul de calcul pe care ati realizat tema. Comentati rezultatele astfel obtinute, si relatia dintre performanta obtinuta si cea teoretica a sistemului.

Notare (100 de puncte in total):

1. Implementarea corecta a programului de factorizare serial - 25p
2. Implementarea corecta a programului de factorizare multithread - 25p
3. Documentatie - Punctul 1 - HW Specs - 10p
4. Documentatie - Punctul 2 - Grafice + comentarii: dimensiunea blocului vs timp - 20p
5. Documentatie - Punctul 3 - Grafice + comentarii: dimensiunea blocului vs GFLOPS + performanta maxima teoretica a sistemului utilizat - 20p

Bonusuri:

1. Eficienta implementarii - 10p
2. Calculul r-ului (din laboratorul 6) pentru sistemul pe care ati realizat tema si mentionarea acestui parametru la punctul 1 a documentatiei - 10p

Observatii:

O implementare eficienta pentru vectori si matrice in Python este disponibila in pachetul Numerical Python, ce poate fi gasit la http://numpy.scipy.org/.