Sabou Alexandru Mihai
Grupa 236

Proiectare paralela

Algoritm Dijkstra – MPI

1. Sablon de proiectare folosit

Pentru implementarea paralela a algoritmului lui Dijkstra (algoritm de drum minim in graf orientat), am ales sa folosesc sablonul de proiectare master/slave. Problema se va divide in doua componente:

- **Master:** aici se va descompune problema in mai multe subprobleme (aproape identice sau chiar identice), ce vor fi distribuite componentelor slave, dar si de combinarea rezultatelor partiale furnizate de componentele slave pentru a produce rezultatul dorit.
- **Slave:** se preia de la master informatii necesare rezolvarii subproblemei, se afla rezultatul partial al acestei subprobleme, dupa care se transmite inapoi la master.

Pentru acest laborator voi aborda problema de algoritm minim in graf folosind algoritmul lui Dijkstra. Adiacenta nodurilor o voi reprezenta prin liste de adiacenta, astfel incat fiecarui nod ii va fi atribuita o lista de noduri adiacente, precum si costul/ponderea/distanta de la acel nod la nodul vecin. Deoarece voi folosi o coada cu prioritati, intotdeauna primul nod vecin din lista de adiacenta va fi cel cu distanta cea mai mica.

Am constatat ca acest algoritm se poate paraleliza prin impartirea listelor de adiacenta in bucati. Cu alte cuvinte, daca privim adiacenta ca o matrice de adiacenta, fiecare componenta slave va primi un anumit numar de coloane din matrice (impartit in mod egal sau cat mai egal). Componenta slave, va determina nodul adiacent cu cea mai mica distanta, precum si actualizarea vectorului de “parinti” si a vectorului de distante, informatii ce vor fi transmise la componenta master, pe care le va pregati pentru obtinerea rezultatului final.

2. Granularitate

Prin granularitate se intelege dimensiunea minima a unei unitati secventiale dintr-un program exprimat in numar de instructiuni. O unitate secventiala este acea parte din program in care nu au loc sincronizari sau comunicari cu alte procese.

Conform lucrarii de laborator anterioare, cat si ce va fi prezentat in aceasta lucrare in continuare, putem spune ca granularitatea este mare, iar gradul de paralelism poate fi incadrat la coarse-grained parallelism. Procesul parinte trimite asincron datele la procesul fiu si procesul parinte prelucreaza rezultatul.

3. Scalabilitate

Scalabilitatea masoara modul in care se schimba performanta unui anumit algoritm in cazul in care sunt folosite mai multe elemente de procesare. Un indicator important pentru

aceasta este numarul maxim de procesoare care pot fi folosite pentru rezolvarea unei probleme.

In cazul algoritmului lui Dijkstra, pe masura ce dimensiunea grafului creste (atat numarul de noduri, cat si numarul de muchii), este necesara si cresterea numarului de procesoare, intrucat mai multe operatii vor fi executate in paralel. Insa, pentru dimensiuni mai mici este posibil ca timpul de executie secventiala sa fie mai bun decat timpul de executie in paralel. De asemenea daca se folosesc mai multe procesoare, va scadea performanta algoritmului.

4. Analiza performantei

A) Complexitate timp

Pentru grafurile dense, in lucrarea de laborator anterioara, am dedus ca Ts(V) = V^2 unde V este numarul de arce din graf. Pentru grafurile de dimenisuni mai mici (ca numar de muchii si noduri), am dedus ca timpul de executie secvential este Ts(E,V) = E + V^2. Dar, deoarece algoritmul lui Dijkstra va fi testat pe grafuri de dimensiuni mari, aleg in continuare sa folosesc Ts(V) = V^2.

Daca notam cu t0 = timpul de creeare a unui proces, atunci Ts(V) = t0 * V. Prin urmare Tp(V) = (t0 * V) / p + C, unde p este numarul de procese iar C este timpul de comunicare dintre procesul master si cel slave.

B) Acceleratie

Accelerarea notat cu Sp, este definita ca raportul dintre timpul de executie al celui mai bun algoritm serial cunoscut, executat pe un calculator monoprocesor si timpul de executie al programului paralel echivalent, executat pe un sistem de calcul paralel.

Pentru algoritmul lui Dijkstra, Sp = Ts(V) / Tp(V). Facand raportul, obtinem ca rezultatul poate fi incadrat intr-un interval (0, t0)