Politechnika Poznańska

Wydział Informatyki

Przetwarzanie Równoległe

Równoległe sumowanie komórek pamięci za pomocą wielu wątków przetwarzania

Autorzy:
Adam Szczepański
Mateusz Czajka

Prowadzący:
dr Rafał Walkowiak

11 lutego 2014

Spis treści

1 Informacje o projekcie

1.1 Dane autorów

Mateusz Czajka	106596
Adam Szczepański	106593

1.2 Historia projektu

1. Jest to pierwsza wersja projektu. Dokumentacja elektroniczna została przesłana w dniu 20 stycznia 2013.

2 Wstęp

2.1 Opis problemu

Głównym założeniem projektu było zapoznanie się biblioteką OpenMP na podstawie równoległego sumowania komórek tablicy. W ramach projektu zrealizowaliśmy 4 algorytmy sekwencyjne oraz 2 algorytmy zrównleglone. Celem zastosowania czterech różnych algorytmów sekwencyjnych było zbadanie wpływu sekcyjności pamięci podręcznej, wyprzdzającego pobrania danych do pamięcy podręcznej oraz kolejności uszeregowania pętli na czas realizacji zadania. W przypadku algorytmów zrównoleglonych badaliśmy wpływ kolejności uszeregowania pętli na końcowy rezultat.

Nasze badania podzieliliśmy na 3 spójne części. Kolejno badaliśmy

- rozmiar danych
- sekcyjność pamięci
- wyprzedzające pobranie

na czas realizacji problemu.

Sam problem sprowadzał się do zsumowania wartości komórek w tabeli. Dla zachowania czytelności w kolejności pętli zastosowaliśmy tablicę dwuwymiarową. Ponieważ sam problem jest prosty obliczeniowo zmuszeni byliśmy do stosowania maksymalnego rozmiaru tablicy tj $[2^{28} - wierszy]$ na $[2^4 - kolumn]$.

2.2 Punkt odniesienia (algorytm sekwencyjny - kolejność ij)

Punktem odniesienia dla wszystkich algorytmów był podstawowy algorytm sekwencyjny w którym sumowaliśmy elementy tablicy wierszami. Nazywany dalej sum_ij.

```
__declspec(noinline) int sum_ij() {
  int sum = 0;
  for (int i=0; i<ROWS; i++) {
    for (int j=0; j<COLS; j++) {
      sum += tab[i][j];
    }
  }
  return sum;
}
```

2.3 Badane algorytmy

2.3.1 Algorytm sekwencyjny - kolejność ji

Algorytm sekwencyjny ze zmienioną kolejnością pętli (sumujemy kolumnami). Nazywany dalej *sum_ji*.

```
__declspec(noinline) int sum_ji() {
  int sum = 0;
  for (int j=0; j<COLS; j++) {
    for (int i=0; i<ROWS; i++) {
      sum += tab[i][j];
    }
  }
  return sum;
}
```

2.3.2 Algorytm zrównoleglony - kolejność ij

Algorytm sumujący wierszami realizowany równolegle. Nazywany dalej *sum_par_ij*.

```
__declspec(noinline) int sum_par_ij() {
  int sum = 0;
  int i;
#pragma omp parallel for default(none) shared(tab) private(i) reduction(+:sum)
  for (i=0; i<ROWS; i++) {
    for (int j=0; j<COLS; j++) {
      sum += tab[i][j];
    }
  }
  return sum;
}
```

2.3.3 Algorytm zrównoleglony - kolejność jj

Algorytm realizowany równolegle ze zmienioną kolejnością pętli (sumujemy kolumnami). Nazwywany dalej sum_par_ji.

```
__declspec(noinline) int sum_par_ji() {
  int sum = 0;
  int j;
#pragma omp parallel for default(none) shared(tab) private(j) reduction(+:sum)
  for (j=0; j<COLS; j++) {
    for (int i=0; i<ROWS; i++) {
      sum += tab[i][j];
    }
  }
  return sum;
```

```
}
```

2.3.4 Algorytm na sekcyjność pamięci

Algorytm sekwencyjny w którym badaliśmy wpływ sekcyjności pamięci na czas wykonania zadania. Nazywany dalej *sum_sec*.

```
__declspec(noinline) int sum_sec() {
  int sum = 0;
  for (int j=0; j<COLS; j++) {
    for (int k=0; k<CACHE_LINES_ON_PAGE; k++) {
      for (int i=k; i<ROWS; i+=CACHE_LINES_ON_PAGE) {
        sum += tab[i][j];
      }
    }
  }
  return sum;
}
```

2.3.5 Algorytm na pobranie z wyprzedzeniem

Algorytm sekwencyjny w którym badaliśmy wpływ wyprzedzającego pobrania danych do pamięci podręcznej na czas realizacji zadania. Nazywany dalej *sum_pf*.

```
int tmp;

__declspec(noinline) int sum_pf() {
  int sum = 0;
  int i;
  for (i=0; i<ROWS-1; i++) {
    for (int j=0; j<COLS; j++) {
      sum += tab[i][j];
      tmp = tab[i+1][j];
    }
  }

  for (int j=0; j<COLS; j++) {
    sum += tab[i][j];
  }
  return sum;
}
```

3 Pomiary efektywności

3.1 Przyśpieszenie obliczeń równoległych

3.1.1 Pomiary

Tablica 1: Porównanie szybkości wybranych algorytmów wobec sum_ij

Algorytm	Czas wykonania w ms	Przyśpieszenie względem sum_ij
sum_ij	253	1.000
sum_ji	2362	0.107
sum_par_ij	87	2.908
sum_par_ji	1189	0.213

Warto zwrócić uwagę na fakt, że oprócz braku lub obecności zrównoleglenia znaczący wpływ na szybkość przetwarzania ma kolejność uszeregowania pętli. Zostało to szerzej omówione w sekcji 3.2. Przyśpieszenia dla poszczególnych uszeregowań pętli prezentują się następująco:

Tablica 2: Porównanie szybkości wybranych algorytmów (o kolejności pętli ij) wobec sum_ij

Algorytm	Czas wykonania w ms	Przyśpieszenie względem sum_ij
sum_ij	2362	1.000
sum_par_ij	87	2.908

Tablica 3: Porównanie szybkości wybranych algorytmów (o kolejności pętli ji) wobec sum_ji

Algorytm	Czas wykonania w ms	Przyśpieszenie względem sum_ji
sum_ji	2362	1.000
sum_par_ji	1189	1.987

3.1.2 Podsumowanie

Zrównoleglenie przetwarzania znacząco przyśpiesza jego czas przetwarzania. Znaczeny wpływ na wielkość wartości tego przyśpieszenia ma kolejność uszeregowania pętli. W przybliżeniu jest to:

- trzykrotne przyśpieszenie dla uszeregowania pętli ij
- dwukrotne przyśpieszenie dla uszeregowania pętli ji

3.2 Współczynniki braku trafień do pamięci

4 Wpływ rozmiaru danych

4.1 Wstęp

W celu sprawdzenia wpływu rozmiaru danych na czas realizacji zadania dokonaliśmy pomiarów czasów dla dwóch rozmiarów danych:

- Instancja A: $tab[2^{28}][2^4]$
- Instancja B: $tab[2^{24}][2^4]$

Spodziewaliśmy się, że czas obliczeń dla Instancji B będzie kilkukrotnie krótszy.

4.2 Wyniki

Wyniki pomiarów prezentują się następująco:

Tablica 4: Czas realizacji kodu dla poszczególnych funkcji w ms.

Problem	sum_ij	sum_ji	sum_sec	sum_pf	sum_par_ij	sum_par_ji
Instancja A	253	2352	19552	225	87	1189
Instancja B	62	59	4890	57	23	304
Stosunek A/B	4.081	4.003	3.998	3.947	3.783	3.911

4.3 Podsumowanie

Wyniki eksperymentu są bardzo zadowalające. W przypadku każdego algorytmu rozmiar danych ma liniowy wpływ na czas przetwarzania. Jest to zgodne ze złożonością algorytmu która jest liniowa wobec n gdzie n to $liczba\_wierszy * liczba\_kolumn$.

W każdym przypadku czas realizacji dla Instancji B był około 4 razy krótszy niż dla Instancji A. Średnia tych wartości wynosi 3.954.

5 Wpływ sekcyjności pamięci

6 Wyprzedzające pobranie danych

Spis rysunków

Spis tablic