Communication-Avoiding Parallel Sparse-Dense Matrix Multiplication — The MPI Programming Assignment

Mateusz Kitlas, nr ind.: 306262

1. Proces 0 wczytuje do pamięc rozmiaru |A| całą macierz.
---------- bariera
2.Proces 0 broadcastuje metadane:

```
struct Mpi_meta_init{
  int row_no_max; //maksymalna liczba wierszy. Do algoruytmu inner
  int nnz_max; //maksymalna liczba nnz spośród wszystkich mini macierzy A
  int side; //rozmiar boku macierzy
};
```

Niestety, na studentsie nie jest dostępna funkcja MPI_Ibrcst, więc użyłem synchronicznej.

3. Proces 0 generuje mini macierze A. Każdy blok ma mallocowaną maksymalną pamięć, której rozmiar jest funkcją **mpi_meta_init**, jednak przesyłane dane mają rozmiar zależny od **nnz**, **row_no**.
4. Proces 0 wysyła do kazdego procesu jego porcję danych, samu zachowując swoją.

---------- bariera

5. Każdy proces wysyła asynchronicznie do **c** poprzednich procesów (poruszamy się po torusie) odpowiednie bloki.

6. Każdy proces odbiera asynchronicznie do **c** kolejnych procesów odpowiednie bloki.
7. Każdy proces czeka na odebranie swoich bloków.
---------- bariera

8. Każdy proces **j** oblicza C odbierając najnowsze bloki od procesu **(j+c)%p** i przesyłając najstarsze do **(j-c)%p**. Wszystko asynchronicznie. Aby odebrać blok nr **n** MPI_Wait() jest wywoływany dopiero po obliczeniu **c** bloków.

9. Po przeliczeniu **p** bloków, następuje zwolnienie pamieci macierzy B, pozmienienie B na C i zaalokowanie nowej pamięci dla C. Kroki od 8 do 9 dą wykonywane **exponent**.

10.Proces 0 w zależności od parametrów wywołania programu odbiera od każdego procesu asynchronicznie blok macierzy C (w postaci *gęstej*) lub za pomocą MPI_Reduce wylicza **ge**.

Optymalizacje

1. Zmienny rozmiar przesyłanych bloków.
2. Klasa **Sparse** potrafi iterować (oczywiście w liniowym czasie) po danych bez żadnego preprocesingu, więc nie jest wymagana żadna dodatkowa pamięć ani czas na przetworzenie przesyłanych bloków.

Prystosowałem wszzystkie struktury tak, aby była możliwa łatwa implemetacja algorytmu inner ABC; mam zamiar to dosłać jak najszybciej.

Dołączyłem własną testerkę.

- id_test.sh mnoży A przez macież identycznościową
- test.sh uruchamia testy dołączone do zadania
- diff_numbers.py porównuje pliki z double'ami z dokładnością do 0.01.