Aleksander Bielawski
Szymon Jabłoński

Dokumentacja wstępna projektu z przedmiotu UXP1A
System komunikacji między-procesowej dla zadań typu map-reduce

Treść zadania

Przedmiotem zadania jest opracowanie systemu komunikacji między-procesowej dla zadań typu map-reduce. Typowo obliczenia w tym modelu odbywają się w systemie rozproszonym z komunikacją poprzez pliki (DFS – rozproszony system plików), na potrzeby tego projektu przyjmiemy jednak „obliczenia" zlecane procesom wykonującym się na jednej maszynie (wykorzystujemy potencjalną wielo-procesorowość maszyny, przy czym zakładamy, że system operacyjny optymalnie wykorzysta procesory, tj. nie zajmujemy się przydziałem zadań do fizycznych procesorów).

System powinien realizować następujące funkcje:

- tworzenie i zarządzanie pulą procesów obliczeniowych (i procesem nadzorcy),
- przydzielanie procesom roboczym zdań do fazy „map", tj. przekazywanie danych, zlecenie zadania dla wskazanej porcji danych, przygotowanie do odbioru danych wynikowych z fazy „map",
- przydzielanie procesom roboczym zdań do fazy reduce", tj. przekazywanie danych, zlecenie zadania dla wskazanej porcji danych, przygotowanie do odbioru danych wynikowych z fazy „reduce"

Przedmiotem projektu jest przede wszystkim stworzenie infrastruktury komunikacyjnej opisanej wyżej, należy też przygotować prosty program testujący, nie musi on realizować algorytmu map-reduce, ale powinien funkcjonować w zakresie komunikacji, przekazywania danych i synchronizacji podobnie do programów m-r.

Interpretacja treści zadania

Formą realizacji zadania będzie biblioteka statyczna języka C pozwalająca na wykonanie algorytmu map-reduce dla zadanego zbioru plików wejściowych i plików wyjściowych, oraz zadanych funkcji map i reduce.

Założenia:

- Każdemu plikowi wejściowemu przypisywane jest dokładnie jedno zadanie mapowania.
- Plików pośrednich jest tyle, ile jest plików wejściowych.
- Zadanie mapujące zaznacza procesowi-nadzorcy zrzucenie swoich wyników do pliku pośredniego wtedy i tylko wtedy, gdy skończy całą swoją pracę (zmapuje cały swój plik wejściowy).
- Zrzucone przez zadanie mapujące wyniki są partycjonowane na tyle partycji, ile jest zadań redukujących (tj. poszczególne partycje przypisane są do poszczególnych zadań redukujących).

- Sposób partycjonowania danych jest dostarczany przez użytkownika biblioteki.
- Plików wyjściowych jest tyle, ile jest zadań redukujących.

Opis funkcjonalny – „black-box”

Wejście:

- Nazwy plików wejściowych.
- Nazwy plików wyjściowych.
- Funkcja map.
- Funkcja reduce.

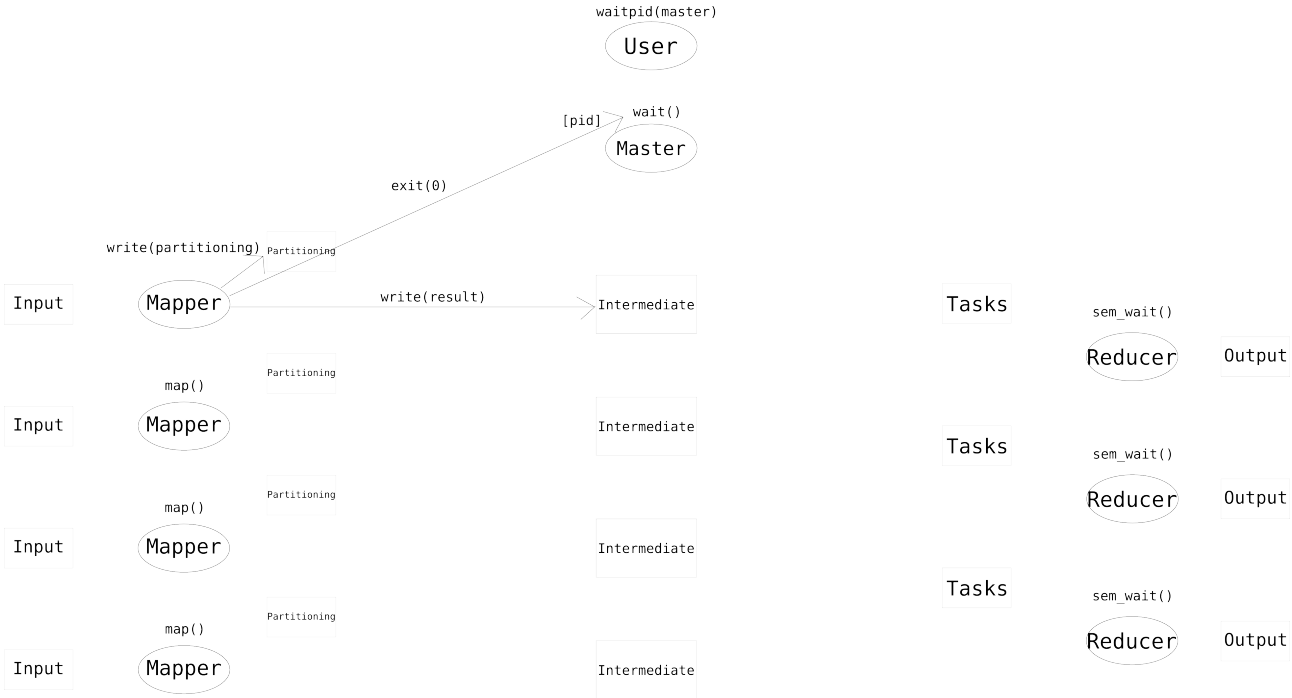
Wyjście:

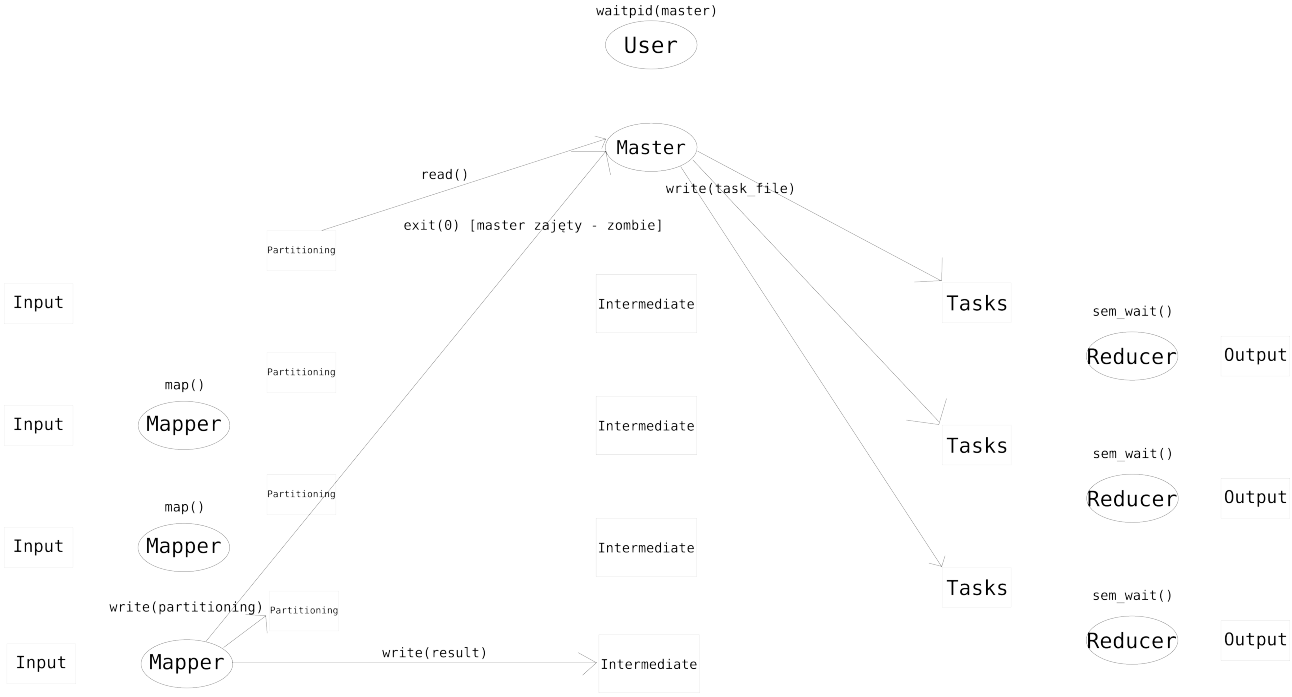
- Pliki wyjściowe z wynikami.
- Ewentualny kod błędu przy niepowodzeniu.

Stosowane protokoły komunikacyjne

Zadanie projektowe zakłada realizację komunikacji przy pomocy plików i semaforów. Do komunikacji między-procesowej wykorzystywane są 2 rodzaje plików:

- plik z opisem partycjonowania (Partitioning) – przypadający na każdy proces mapujący. Plik wypełniany jest przez proces mapujący w trakcie zadania mapowania. Określa jakie są położenia i rozmiary partycji (także pustych) dla każdego procesu redukującego w wynikowym pliku pośrednim. Plik ten odczytywany jest przez proces-nadzorcę po zakończeniu procesu mapującego.
- plik z opisem położenia istotnych dla danego procesu partycji (Tasks) – przypadający na każdy proces redukujący. Plik uzupełniany jest przez przez proces-nadzorcę dla każdego procesu redukującego przy każdej obsłudze zakończenia procesu mapującego. Dopisywana jest wtedy do niego informacja o tym, który plik pośredni jest gotowy, oraz położenie w pliku i rozmiar partycji przeznaczonej dla danego procesu redukującego.

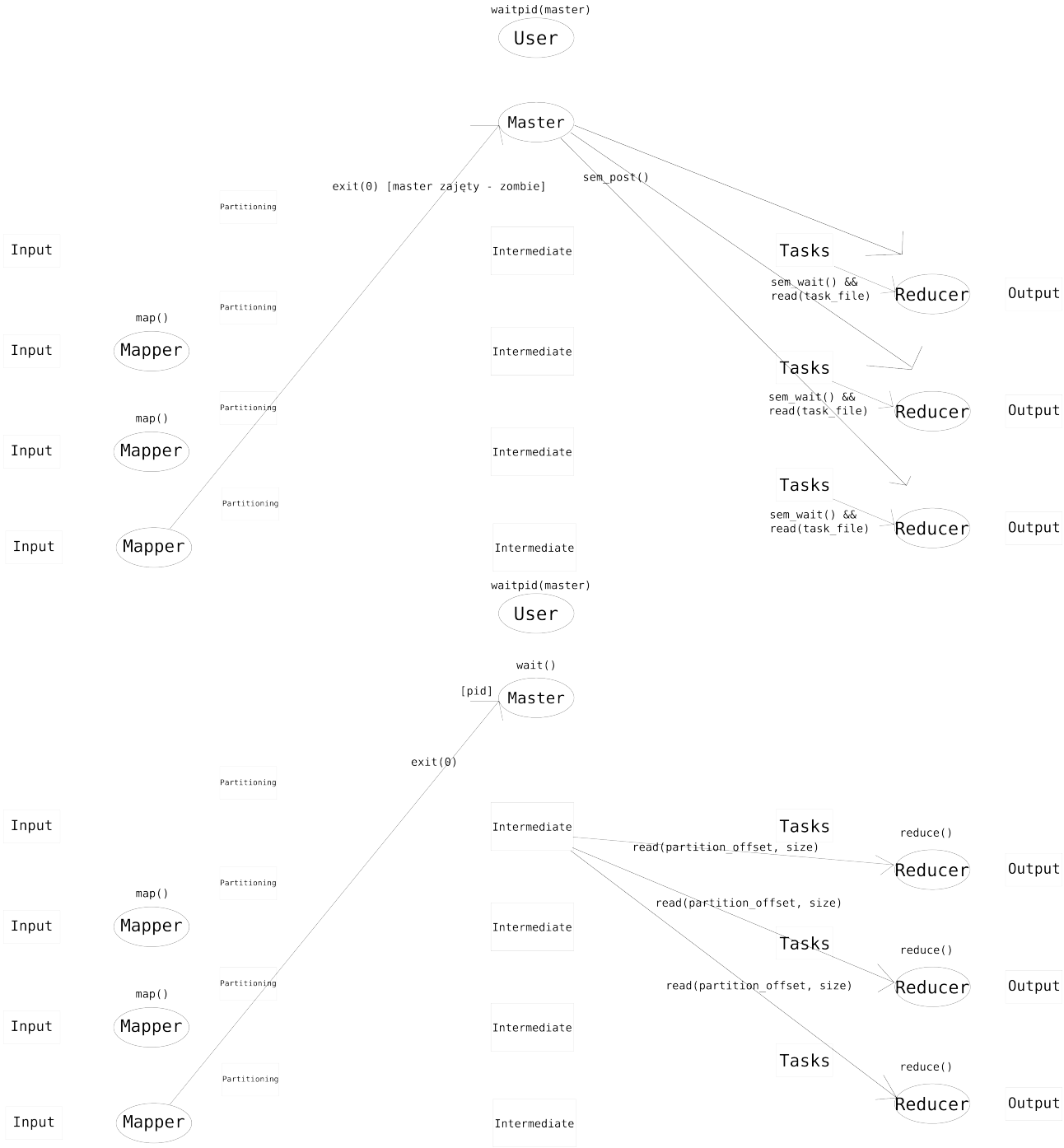




Podział na moduły i komunikacja między nimi

Podział na moduły jest identyczny z podziałem ról procesów:

- Nadzorca – tworzy procesy i pośredniczy w wymianie informacji między jednymi rodzajami procesów a drugimi. Kontroluje także czy i w jaki sposób kończą się procesy robocze. Zarządza także czasem życia plików roboczych (Partitioning, Intermediate, Tasks). Przekazuje poszczególnym procesom redukującym informacje o gotowych plikach pośrednich i istotnych dla tych procesów partycji w nich zawartych.
- Mapper – Komunikuje się z Nadzorcą poprzez wysłanie kodu z jakim zakończono proces oraz pośrednio z procesami redukującymi poprzez pozostawiany plik z partycjonowaniem. Realizuje wykonanie fazy „Map”.
- Reducer – Komunikuje się z Nadzorcą za pomocą pliku roboczego. Nadzorca dopisuje do pliku informacje o każdej gotowej partycji (tożsamości pliku z wynikami pośrednimi oraz położeniu w nim porcji danych). Do zakomunikowania pojawienia się nowej partycji Reducerowi Nadzorca używa dodatniej operacji na semaforze, na którym dany Reducer wykonuje operację ujemną przez przystąpieniem do odczytania informacji o partycji z pliku. Przekazuje także informacje Nadzorcy poprzez kod zakończenia procesu. Realizuje wykonanie fazy „Reduce”.
- User – program użytkownika-klienta biblioteki.


waitpid(master)
User
Master
exit(0) [master zajęty - zombie]
sem_post()
Partitioning
Input
Intermediate
Tasks
sem_wait() &&
read(task_file)
Reducer
Output
map()
Mapper
Partitioning
Input
Intermediate
Tasks
sem_wait() &&
read(task_file)
Reducer
Output
map()
Mapper
Partitioning
Input
Intermediate
Tasks
sem_wait() &&
read(task_file)
Reducer
Output
Mapper
Partitioning
Input
Intermediate
waitpid(master)
User
wait()
[pid]
Master
exit(0)
Partitioning
Input
Intermediate
Tasks
reduce()
read(partition_offset, size)
Reducer
Output
map()
Mapper
Partitioning
Input
Intermediate
read(partition_offset, size)
Tasks
reduce()
read(partition_offset, size)
Reducer
Output
map()
Mapper
Partitioning
Input
Intermediate
Tasks
reduce()
Reducer
Output
Mapper
Partitioning
Input
Intermediate