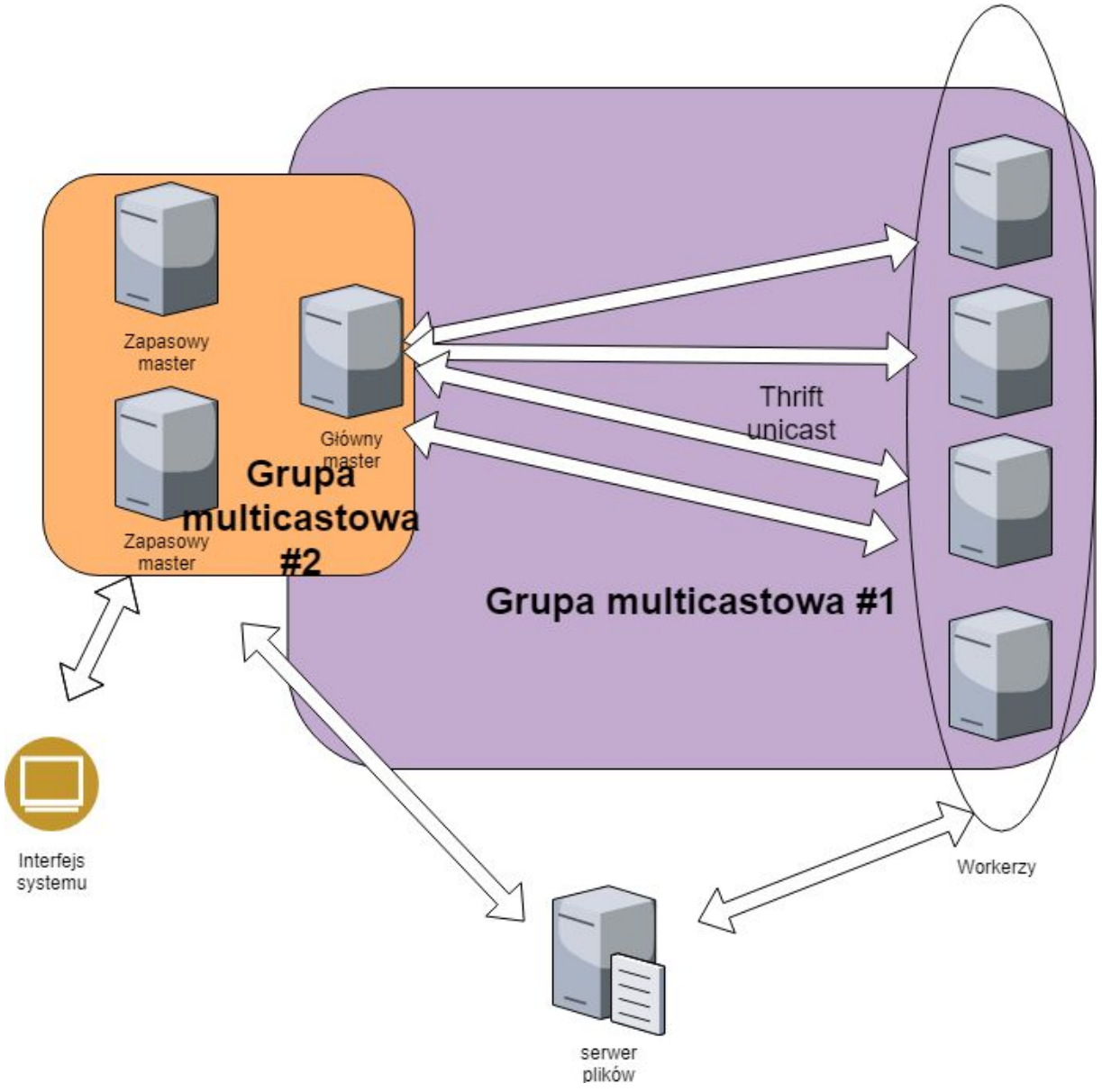
IntePiotr Czytański
Izabela Drozdowska
Krzysztof Dziurlikowski
Maciej Suchocki

Implementacja systemu do obliczeń Map-Reduce

Zarys tematu

Tematem naszego projektu jest realizacja systemu umożliwiającego wykonywanie obliczeń metodą Map-Reduce. Użytkownik końcowy, który ma do dyspozycji system, musi jedynie sformułować swój problem za pomocą pary funkcji – Map i Reduce w języku Python oraz przekazać dane, na których mają się wykonać dane funkcje.



Rysunek 1Poglądowy schemat

Objaśnienia:

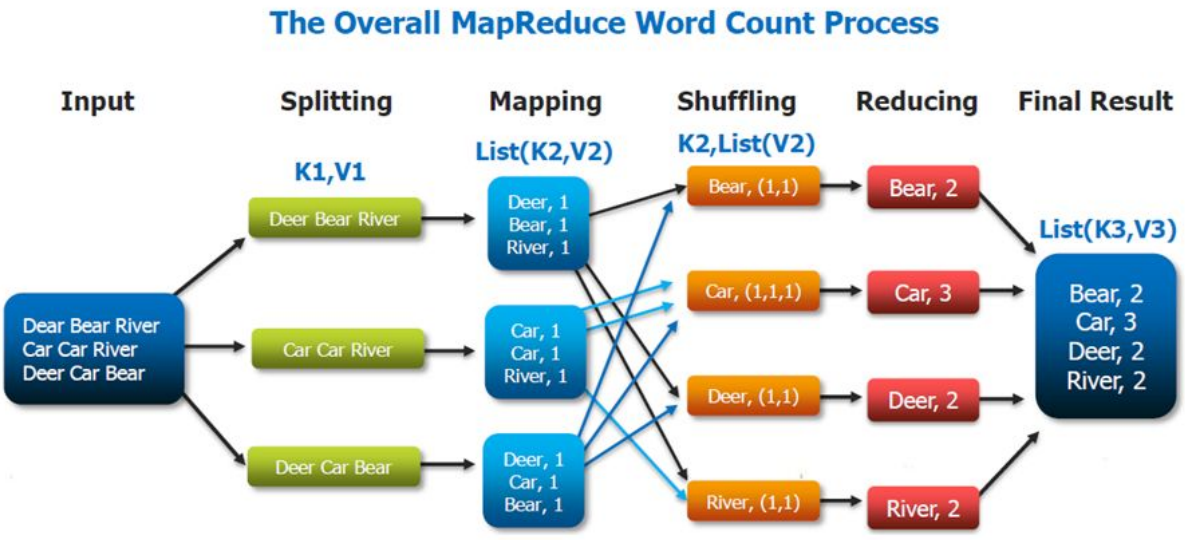
Pole fioletowe – grupa multicastowa #1 – jest to grupa służąca do zebrania przez mastera informacji o wszystkich węzłach typu worker gotowych do podjęcia pracy. Proces ten jest bliżej opisany w dalszej części dokumentacji.

Pole pomarańczowe – grupa multicastowa #2 – służy do wybrania głównego mastera na początku działania systemu oraz do synchronizacji stanu systemu między węzłami master w późniejszej fazie działania.

Podstawowe założenia

- Dane do przetwarzania są dostępne na serwerze plików, do którego dostęp ma każdy węzeł
- Wszystkie węzły są uwierzytelniane na podstawie certyfikatów wystawionych przez nasze CA
- Węzły master posiadają klucze publiczne innych węzłów master
- Zakładamy, że w pełni ufamy działaniom wszystkich węzłów posiadających nasze certyfikaty
- Po zebraniu przez węzeł master informacji o węzłach worker komunikacja między nimi przechodzi z transmisji multicastowej na transmisję unicastową realizowaną przez bibliotekę Thrift
- Węzły są konfigurowalne przez pliki konfiguracyjne znajdujące się wraz z plikiem wykonywalnym na każdej maszynie; znajdują się tam m.in. informacje o adresach grup multicastowych, dzięki temu w jednej sieci może być rozwiązywanych wiele problemów równocześnie

Schemat działania



Rysunek 2. Schemat procesu Map-Reduce Zródło: wikis.nyu.edu

Faza I – wybranie głównego węzła Master

Wszystkie włączone węzły typu master rozgłaszają się co określony czas w grupie multicastowej i zapisują informacje o innych węzłach. Nasłuchuje również interfejs użytkownika. Użytkownik przy pomocy interfejsu systemu wysyła żądanie rozwiązania zadania, które jest kierowane do węzła master o najniższym adresie ip – staje się on głównym masterem. Rozsyła on pakiety multicastowe (kilka, w niewielkich odstępach czasu) podpisane swoim kluczem prywatnym, zawierające komunikat nakazujący przejście do fazy II i zaprzestanie rozgłaszania się.

Faza II – przygotowanie

Główny master wysyła na grupę multicastową #1 komunikaty odpytujące o uruchomione węzły worker. Pakiety wysyłane są przez określony czas, lub aż do skompletowania zadanej liczby workerów. Węzły klienckie odpowiadają zestawieniem połączenia przez Thrift z węzłem master.

Faza III – przydział zadań

Główny master dzieli dane wejściowe na mniejsze fragmenty i przydziela je do węzłów worker. Przesyłane są do nich informacje o położeniu w systemie plików funkcji map, reduce, położenie danych wejściowych przeznaczonych dla danego workera oraz uporządkowana lista adresów ip workerów pracujących nad zadaniem.

Faza IV – Map + shuffling

Węzły worker pobierają kod funkcji Map, uruchamiają ją podając dane wejściowe. Następnie na podstawie wyniku obliczany jest węzeł worker, do którego powinny trafić dane (shuffling). Węzeł wybierany jest poprzez liczenie funkcji skrótu każdego klucza z danych wyjściowych, wynik wybieramy modulo liczba workerów – otrzymujemy numer w tablicy workerów. Następnie klucz i wartość wysyłana jest do workera, do którego powinny trafić dane związane z danym kluczem. W przypadku, gdy nie istnieje jeszcze połączenie thrift z danym węzłem zostaje ono utworzone. Węzły worker powiadamiają głównego mastera o gotowości do kolejnej fazy. Przejście do kolejnej fazy następuje, gdy wszystkie węzły skończą obliczenia.

Faza V – reduce

Węzły worker pobierają funkcję Reduce i uruchamiają ją dla otrzymanych danych. Wynik funkcji reduce wysyłany jest do głównego węzła master.

Rozwiązania techniczne

- Węzeł master napisany w C# .Net Core
- Węzły worker napisane w C#, Java, Python
- Kod funkcji Map/Reduce uruchamiany w interpreterze PyPy w środowisku sandboxowanym
- W przypadku awarii węzła master, nowy węzeł master(kolejny według adresów ip) rozgłasza się działającym workerom „przejmując" ich
- Główny węzeł master cyklicznie wywołuje funkcję ping na workerach w celu wykrycia ewentualnego zawieszenia się.
- Główny master wystawia api rest umożliwiające odpytywanie z zewnątrz o stan systemu.

Szczegóły dotyczące przepływu danych między węzłami

W trakcie działania funkcja Map „emituje" pary klucz-wartość. Nie są one jednak wysyłane pojedynczo do węzłów współpracujących, lecz trafiają do buforów. Buforów jest (maksymalnie) tyle ile węzłów współpracujących (czyli n-1), alokowane są one leniwie. Dopiero po uzbieraniu się w buforze odpowiedniej ilości wpisów (parametr możliwy do skonfigurowania) pary te są przesyłane w tablicy. Na potrzeby przesyłu stworzona jest dodatkowa struktura danych, dalej nazwana **KeyValueEntity**.
Składa się ona z trzech pól:

- ilość
- klucz
- wartość

W przypadku bardzo dużej ilości par z jednakowymi wartościami klucz-wartość zapewni to zauważalną redukcję ilości przesyłanych informacji. Jest to analogia do kompresji RLE. Korzystamy z faktu, że kolejność par jest bez znaczenia dla przetwarzania. Na koniec procesu Map pary pozostałe w buforach są dosyłane.

W trakcie etapu Reduce każdy węzeł worker może być zaangażowany do przetworzenia danych dla więcej niż jednego klucza (np. sytuacja kiedy kluczy jest k, węzłów n i k>n). Funkcja Reduce jest uruchamiana dla kolejnych kluczy, po zakończeniu przetwarzania każdego klucza wynik działania jest przesyłany do węzła Master.

Wstępny zarys protokołu komunikacyjnego

Komunikacja przez Thrift:

Master	Worker
bool RegisterWorker(int32 ip, int32 port) – zgłoszenie gotowości do pracy przez workera **void Reconnect()** – zgłoszenie się do węzła Master po utracie połączenia, lub po przełączeniu się na węzeł zapasowy **void FinishedMap()** – zgłoszenie ukończenia procesu Map na danym węźle **void RegisterResult(string key, string value**) – zgłoszenie wyniku cząstkowego z operacji Reduce(w przypadku gdy węzeł przetwarza dane dla wielu różnych kluczy to zarejestruje wiele wyników) **void FinishedReduce()** – zgłoszenie ukończenia procesu Reduce dla wszystkich przydzielonych kluczy	**bool AssignWork(string dataFileName, string mapFileName, reduceFileName, list<~~string~~int32> workersList)** – przydzielenie danych dla funkcji map, przekazanie potrzebnych funkcji oraz listy innych workerów do współpracy **void StartMap()** **void RegisterMapPair(~~Array~~list<KeyValueEntity> pair) –** przekazanie par klucz wartość węzłowi na potrzeby funkcji reduce **void StartReduce()** **int Ping() –** sprawdzenie czy węzeł odpowiada

Komunikacja multicast:

Treść nieszyfrowana:
HELLOIMMASTER

- komunikat rozgłoszeniowy węzłów master dla węzłów master

Treść zaszyfrowana kluczem prywatnym węzła:
GOTOSTEP2

- komunikat nakazujący zapasowym masterom zaprzestanie rozgłaszania się i przejście do następnej fazy

Treść nieszyfrowana:
HELLOWORKERS

- komunikat wysyłany przez główny węzeł master na grupę multicastową workerów

Treść zaszyfrowana kluczem prywatnym węzła:
IAMALIVE

- komunikat wysyłany przez ~~główny~~ węzeł Master na grupę multicastową masterów, sygnalizuje działania węzła – w przypadku braku pakietu przez 30sekund węzłem master staje się następny w kolejności węzeł master, który czeka na połączenia od workerów, które połączą się do niego po utracie komunikacji z dotychczasowym głównym masterem. Nowy węzeł natychmiast zaczyna nadawać ten komunikat.

Treść nieszyfrowana:
IAMNEWMASTER

- komunikat wysyłany przez nowy węzeł master w przypadku awarii poprzedniego, wysyłany jest na grupę multicastową workerów. Workerzy po otrzymaniu go zamykają połączenie z dotychczasowym węzłem(o ile jeszcze jest) i łączą się do nowego.

Treść zaszyfrowana kluczem prywatnym węzła:
STATE:[CONTENT]

- komunikat przesyłany przez główny węzeł master w celu synchronizacji stanu systemu między węzłami master, zawiera wszystkie informacje niezbędne do kontynuacji działania systemu.
[CONTENT] – stan w formacie json, skompresowany gzipem.

Ogólny szkic działania systemu