POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INŻYNIERIA INTERNETOWA
KURS: APLIKACJE INTERNETOWE I ROZPROSZONE

GRAVISIM

Dokumentacja projektowa

AUTORZY:
Adam Dłubak
Jakub Błaszczyk

PROWADZĄCY:
Dr inż. Marek Woda

WROCŁAW, 2017

Spis treści

Rozdział 1

Wprowadzenie

GRAVIsim *to narzędzie umożliwiające przeprowadzanie astrofizycznej symulacji grawitacyjnej N-ciał zaimplementowane w systemie rozproszonym. Interfejsem symulacji jest aplikacja internetowa pozwalająca autoryzowanym członkom instytutu badawczego na zlecanie oraz analizowanie (interpretowanie) zadań badawczych.*

1.1. Pomysł, motywacja, stan rynku

Pomysł stworzenia GRAVIsim zrodził się w głowach autorów poprzez ich zamiłowanie do astrofizyki. Operowanie na systemach rozproszonych doskonale wpasowuje się w tematykę przetwarzania dużej ilości informacji, potrzebnej do przeprowadzenia symulacji zderzeń kosmicznych.

Z racji, iż na ogólnodostępnym rynku brak jest aplikacji wykonującej podobne zadania, autorzy GRAVIsim mieli silną motywację do stworzenia narzędzia, które mogłoby zobrazować zjawisko zderzeń astrofizycznych występujących w kosmosie, jednak niedostrzegalnych dla człowieka.

Dzięki stworzeniu środowiska pozwalającego zarządzać symulacjami można lepiej poznać oraz zobrazować siły oddziaływania grawitacyjnego w makroskali. Wyniki takich badań mogą być wykorzystywane zarówno przez naukowców jak i w celach popularnonaukowych. Dzięki możliwości zobrazowania, w formie graficznej animacji, wyników przeprowadzonych badań narzędzie to może promować takie dziedziny nauki jak fizyka i astronomia wśród dzieci, młodzieży, ale również i osób dorosłych.

1.2. Zarys projektu

GRAVIsim to z jednej strony narzędzie pozwalające zarządzać klastrem obliczeniowym złożonym z dowolnej liczby komputerów klasy PC w celu wykonywania konkretnych obliczeń na sieci rozproszonej, z drugiej - jest to prosta i przyjazna w użytkowaniu aplikacja internetowa.

Zalogowany użytkownik może dzięki niej wykorzystać potencjał obliczeniowy klastra do symulowania zderzeń grawitacyjnych dowolnej liczby obiektów kosmicznych (np. planet, gwiazd, gromad, galaktyk). Aplikacja obsługuje cały proces takiego zadania zaczynając od utworzenia danych wejściowych, poprzez zlecenie pomiarów, jego monitorowanie, kończąc aż na analizie danych końcowych i jej graficznej wizualizacji.

1.3. Przeznaczenie projektu

Celem projektu jest stworzenie symulacji grawitacyjnej N-ciał wraz z interfejsem webowym pozwalającym na zlecanie badań dla różnych scen/obiektów i graficzna interpretacja otrzymanych wyników.
Projekt przeznaczony jest do celów naukowo-badawczych, obrazujący podstawową mechanikę oddziaływać grawitacyjnych. Skala i wartości obiektów symulacji (odległość, masa, siły) są rzeczywistym przekształceniem odpowiadających wartości świata rzeczywistego, co gwarantuje abstrakcyjnym obiektom symulacji faktyczną nośność informacji względem odpowiadających obiektom rzeczywistym.

Rozdział 2

Zakres projektu

W pierwszej fazie projektu (jeszcze przed oficjalnym Kick-Off) został ustalony cel projektu, który w ramach kursu chcieliśmy osiągnąć, a także funkcjonalności, które aplikacja GRAVIsim powinna posiadać. Funkcjonalności te zostały podzielone na podstawowe oraz rozszerzone, czyli takie które nie są krytyczne dla założonego funkcjonowania systemu jednak ich zaimplementowanie jest dodatkową wartością dla systemu i będzie dużym udogodnieniem (lub rozszerzeniem możliwości) dla osób z niego korzystających.

2.1. Cel projektu

Zbudowanie aplikacji internetowej, która poprzez interfejs użytkownika umożliwi obsługę zadań wykonywanych na systemie rozproszonym. Użytkowanie aplikacji możliwe jest poprzez dostęp do hermetycznego systemu, do którego użytkownik może dostać się jedynie poprzez poprawne jego zarejestrowanie w bazie przez administratora, a następnie zalogowanie do serwisu. Użytkownik systemu GRAVIsim po poprawnym zalogowaniu może wykonać następujące funkcjonalności:

- Generować dane wejściowe symulacji
- Zlecać zadania z wcześniej wygenerowanego pliku
- Zarządzać zadaniami – parametry takie jak: tytuł, opis, ilość iteracji, priorytet, status zadania
- Obserwować postęp prowadzonej symulacji
- Analizować wynik symulacji

2.2. Funkcjonalności podstawowe

- **Symulacja grawitacyjna N-ciał oparta na systemie rozproszonym**

  Zaimplementowanie algorytmu dokładnego (zupełnego) operującego na systemie rozproszonym, który wykonuje symulacje grawitacyjne N-ciał.

- **Interfejs webowy pozwalający na zlecenie oraz zarządzanie badaniami**

  Stworzenie prostej, nowoczesnej i przyjaznej dla użytkownika aplikacji internetowej opartej o freamworki AngularJS oraz front-endowy Bootstrap. Serwis ma być intuicyjny oraz zapewniać dostęp do wszystkich możliwości jakie oferuje GRAVIsim. Użytkownik koniecznie musi mieć możliwość:

  - Zlecać badania.
  - Konfigurować parametry oraz opis zadania.
  - Zarządzać statusem zadania (anulować, wstrzymywać, ustawiać w kolejce zadań).
  - Jako rezultat pomiarów otrzymać wizualizację wyników w formie poklatkowej animacji.

- **System uwierzytelniania użytkownika**

  Powinien zostać zaprojektowany system uwierzytelniania użytkowników w aplikacji wraz z zaimplementowane następujące funkcjonalności:

  - Rejestracja nowego użytkownika.
  - Logowanie w systemie.
  - Zarządzanie innymi użytkownikami poprzez konto administratora.
  - Automatyczne wygaszanie sesji w przypadku określonego czasu braku działań na koncie użytkownika.

- **Możliwość wprowadzenia danych wejściowych z poziomu aplikacji webowej**

  Aplikacja powinna być wyposażona w formularz umożliwiający generowanie zadania z poziomu interfejsu oraz jego parametryzowanie poprzez:

  - Wgrywanie pliku wejściowego z danymi.
  - Nazywanie i opisywanie zadania.
  - Ustawianie ilości iteracji.
  - Wybieranie priorytetu wykonywalności.

- **Graficzna interpretacja wyników**

  Możliwość wizualnej interpretacji wyników wykonanego badania z poziomu szczegółowego widoku zadania w aplikacji. Okno animacji ma mieć możliwość:

  - Wstrzymywania oraz ponowego uruchamiania animacji.
  - Ustawiania szybkości wyświetlanej animacji (*FPS - Frame per second*).
  - Uruchamiania animacji w trybie pełnoekranowym.
  - Pobierania danych generujących animację (wynikowych badania).

2.3. Funkcjonalności rozszerzone

- **Historia badań**

  Możliwość podglądu archiwalnych i obecnych zadań z pełną dostępnością informacji (parametry, logi, rezultaty) dla danego użytkownika oraz analogicznie dla wszystkich użytkowników jedynie z konta administratora.

- **Monitorowanie postępu prowadzonego badania**

  Użytkownik ma mieć możliwość monitorowania postępu wykonywanego zadania w czasie rzeczywistym z poziomu aplikacji. Interfejs ma posiadać dane dotyczące stopnia wykonanego zadania (pasek postępu) wraz ze wszystkimi dodatkowymi informacjami w postaci logów systemowych.

- **Generator danych wejściowych**

  Generator w formie graficznego narzędzia w przyjazny i intuicyjny dla użytkownika sposób ma generować dane wejściowe dla systemu badań grawitacyjnych. Generator ma umożliwiać:
  - Tworzenie dwóch rodzajów próbek: pojedynczych punktów lub chmury punktów.
  - Dla pojedynczej próbki wybór koloru oraz masy.
  - Dla chmury próbek dodatkowo: rozmiar chmury oraz ilość pojedynczych próbek w chmurze.

2.4. Wymagania technologiczne

Aplikacja kliencka powinna zostać zbudowana w oparciu o technologię **AngularJS**, w formie aplikacji typu *One-Site-Page* w języku *Javascript*. Obsługa danych w aplikacji ma być zapewniona przez framework **Django**, napisany w *Python*. W celu zapewnienia aplikacji estetycznego wyglądu zastosowany zostanie framework **Boostrap** wraz z *CSS3* oraz *HTML5*. Do wspomagania autoryzacji będą używane **tokeny JWT**. Moduł aplikacji wykonujący obliczenia rozproszone będzie wykonany przy użyciu **Apache Spark**, natomiast komunikacja pomiędzy aplikacją, a dokładniej **Django Web API** oraz modułem obliczającym będzie opierać się o dedykowany moduł **Job Scheduler'a** oraz **Loggera** (do przesyłania informacji o stanie badań) napisanych w języku *Python* oraz *Javascript*.

Rozdział 3

Charakterystyka projektu

Rozdział 4

Implementacja

Rozdział 5

Instalacja i uruchomienie aplikacji

5.1. Instalacja Apache Spark na systemie Windows

W celu zainstalowania Apache Spark na systemie Windows (konieczny do uruchomienia Job Schedulera dla aplikacji GRAVIsim) należy wykonać następujące kroki:

- Pobrać i zainstalować oprogramowanie:
  - **Java SE Development Kit (SDK) w wersji 8 lub nowszej** - Link do pobrania
    Aby zweryfikować poprawność instalacji, należy otworzyć konsolę (`cmd`), wpisać `java` i kliknąć *Enter*. Jeśli pojawi się wiadomość:
    `Java is not recognized as an internal or external command.`
    należy skonfigurować zmienne środowiskowe: `JAVA_HOME` (podając ścieżkę do miejsca zainstalowania *Java JDK*) oraz `PATH` jako `%JAVA_HOME%\bin` (instrukcja poniżej).

  - **Scala** - Link do pobrania
    Gdy Scala zostanie zainstalowana, należy ustawić zmienne środowiskowe `SCALA_HOME` do folderu, gdzie zainstalowano *Scalę* i `PATH` jako `%SCALA_HOME%\bin`.

  - **Python w wersji 3.5** (jeśli jeszcze nie został zainstalowany) - Link do pobrania

  - **SBT** - Link do pobrania
    Gdy *SBT* zostanie zainstalowane, należy ustawić zmienną środowiskową `SBT_HOME` jako ścieżkę do miejsca, gdzie zainstalowano *SBT*.

  - **WinUtils** - Link do pobrania
    Dopóki nie ma zainstalowanego lokalnego *Hadoop'a* na Windows konieczne jest pobranie `winutils.exe` i umieszczenia go w folderze `Hadoop/bin/`, który należy uprzednio utworzyć w dowolnej lokalizacji na dysku, a następnie ustawić zmienną środowiskową :
    `HADOOP_HOME = Lokalizacja_utworzonego_folderu_Hadoop`

– **Apache Spark z Hadoop 2.7** - Link do pobrania
Wybierz najnowszą wersję oraz *package type* jako *Pre-built for Apache Hadoop 2.7 and later*. Gdy *Spark* zostanie zainstalowany, należy ustawić zmienne środowiskowe: `SPARK_HOME` jako ścieżkę do miejsca, gdzie zainstalowano *Spark'a* oraz dodać ścieżkę `%SPARK_HOME%\bin` do `PATH`.

- Otworzyć konsolę (`cmd`), i wpisać komendę:

`Spark-shell`

Apache Spark powinien przejść proces instalacji i wyświetlić charakterystyczny napis Spark. Wówczas instalacja została zakończona pomyślnie.

Rys. 5.1: Komunikat informujący o poprawnie zakończonej instalacji Apache Spark

Jak ustawić zmienne środowiskowe:

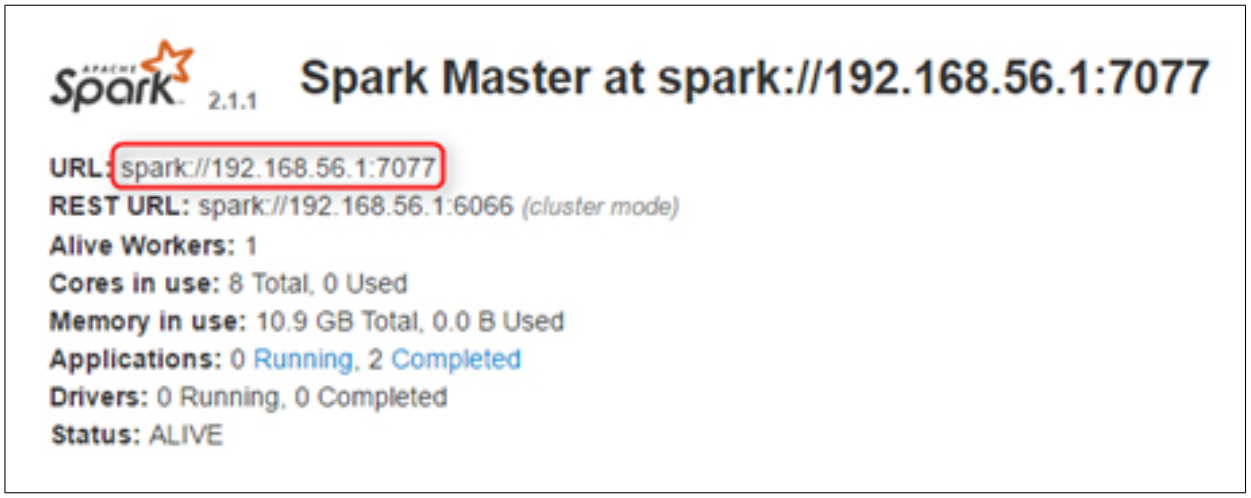
1. Przejdź do Control `Panel\System and Security\System`
2. Wybierz `Advance System Settings`
3. Wybierz `Environment Variables`
4. W polu `user variables` dodaj nową zmienną środowiskową np. `SCALA_HOME` podając jako wartość ścieżkę do miejsca zainstalowania *Scali*.
5. W polu `user variables` edytuj zmienną `PATH` dodając nowy klucz np. `%SCALA_HOME%\bin`

5.2. Uruchomienie Apache Spark

1. Poprzez konsolę uruchomić węzeł zarządzający - Master:
   `Spark-class org.apache.spark.deploy.master.Master`

2. Przejść pod adres:
   `localhost:8080`

3. Znaleźć URL Spark Master i skopiować, np.
   `spark://192.168.56.1:7077`



Rys. 5.2: Spark Master - Adres URL

4. Poprzez konsolę uruchomić węzeł wykonawczy - Worker:

`Spark-class org.apache.spark.deploy.worker.Worker spark://MASTER-ADRES`

W tym momencie Spark zostanie uruchomiony z jednym węzłem zarządzającym oraz jednym węzłem obliczeniowym. W celu uruchomienia wielu węzłów obliczeniowych należy powtórzyć wykonywanie ostatniej komendy.

Rozdział 6

Instrukcja użytkowa

Rozdział 7

Badanie złożoności obliczeniowej

Badania przeprowadzono w politechnicznym laboratorium komputerowym przy pomocy 9 identycznych komputerów, z których jeden służył za główny serwer aplikacji oraz zarządcę klastra, a pozostałe 8 za węzły obliczeniowe (równolegle od 1 do 8 - zależnie od specyfikacji testu).

7.1. Zestaw konfiguracyjny

Testy przeprowadzono dla wybranych zestawów konfiguracyjnych, gdzie parametrami były: • Ilość węzłów obliczeniowych • Wielkość pliku wejściowego (ilość punktów masy) • Ilość iteracji symulacji Każdy z węzłów obliczeniowych otrzymywał na wyłączność badań po 2 rdzenie (procesor i7 3.4MHz) oraz 2.0 GB pamięci RAM (DDR3). W trakcie badania parametry oraz zasoby poszczególnych węzłów nie były zmieniane. Symulacja ma charakter liniowy (liniowa charakterystyka iteracji) dlatego dla większych wartości istnieje możliwość dokładnej estymacji czasu obliczania w oparciu o empiryczne wyniki pomiarów dla wartości mniejszych.

7.2. Wyniki badań

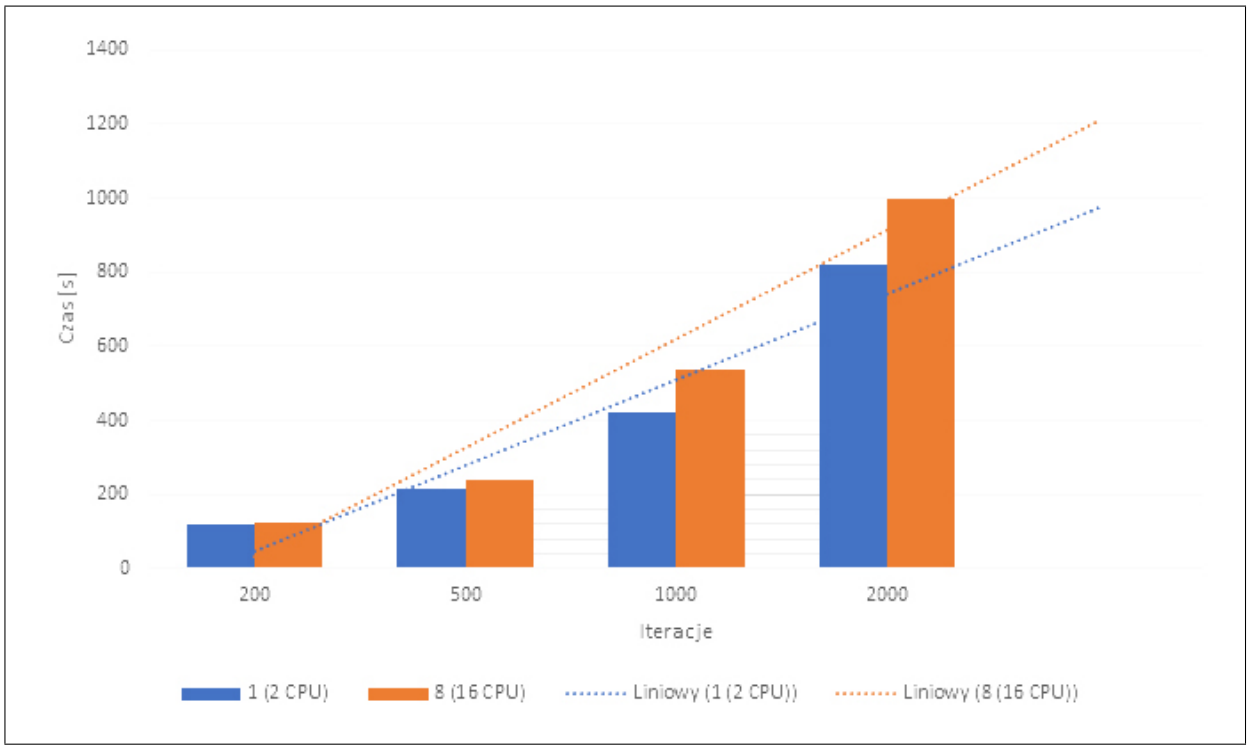
Wyniki badań oraz analizy przedstawia poniższe zestawienie oraz wykresy:

7.3. Analiza wyników

Na podstawie wyników przeprowadzonych badań można zaobserwować, iż wszystkie 3 badane parametry mają znaczący wpływ na czas wykonywania zadania. Klaster obliczeniowy opiera się na komunikacji poszczególnych węzłów, która odbywając się poprzez sieć skutkuje znaczącym opóźnieniem w wykonywaniu zadania. Ma to ogromny wpływ dla zadań liczonych stosunkowo szybko (mała ilość punktów masy w symulacji), stąd też zaobserwowano, iż klaster o większej ilości węzłów obliczył to samo zadanie w czasie o 10-15

Id	Węzły	CPU	RAM [GB]	Ilość punktów	Ilość iteracji	Wynik [s]
1	1	2	2.0	100	500	215.67
2	1	2	2.0	100	1000	421.42
3	1	2	2.0	100	2000	821.24
4	2	4	4.0	5000	10	256.46
5	2	4	4.0	5000	20	528.44
6	4	8	8.0	5000	10	94.43
7	4	8	8.0	5000	20	201.78
8	8	16	16.0	100	200	121.90
9	8	16	16.0	100	500	238.60
10	8	16	16.0	100	1000	537.77
11	8	16	16.0	5000	10	58.99
12	8	16	16.0	5000	20	116.29
13	8	16	16.0	5000	500	2704.34

Tab. 7.1: Wyniki czasowe pomiarów wydajności klastra obliczeniowego



Rys. 7.1: Porównanie złożoności czasowych symulacji małej ilości punktów dla skrajnych konfiguracji klastra

Rozdział 8

Podsumowanie i wnioski

Spis rysunków

Spis tabel