Computational Cluster

Katarzyna Węgiełek
Paweł Własiuk
Kamil Sienkiewicz
Marcin Wardziński

11 stycznia 2015

Spis treści

Rozdział 1

Wstęp

Tematem projektu jest stworzenie dokumentacji dla klastra obliczeniowego, który będzie umożliwiał rozwiązywanie problemów o dużej złożoności. System składa się z *serwerów komunikacyjnych*, *menadżerów zadań*, *klientów* i *węzłów obliczeniowych*.
Rolą *serwera* jest zapewnienie komunikacji pomiędzy wszystkimi komponentami. Elementy systemu wymieniają informacje wykorzystując protokół XML. W klastrze może być kilka serwerów. Jeden z nich jest serwerem głównym - zarządza przekazywaniem komunikatów i nadzoruje stan wszystkich elementów systemu. Pozostałe pełnią rolę pomocniczą - na nich znajdują się kopie zapasowe częściowych i ostatecznych rozwiązań problemów, zadań oczekujących na rozwiązanie i informacji o trwających obliczeniach. W przypadku awarii serwera głównego, jeden z serwerów pomocniczych przejmuje jego zadania. Wykonywaniem kopii danych zajmuje się serwer główny. On też informuje menadżerów zadań, klientów oraz węzły obliczeniowe o istnieniu serwerów pomocniczych, aby w przypadku awarii możliwa była ich zamiana.
Zadaniem *aplikacji klienckiej* jest pobranie danych wejściowych od użytkownika i wysłanie do serwera komunikacyjnego zlecenia rozwiązania podanego problemu. Gdy obliczenia zostaną zakończone, klient odbiera od serwera rozwiązanie zadania i prezentuje otrzymane wyniki użytkownikowi.
Menadżer zadań zajmuje się organizacją obliczeń i uzyskaniem ostatecznego wyniku z rozwiązań cząstkowych. Po otrzymaniu problemu od serwera komunikacyjnego, dzieli go na mniejsze podproblemy, przeznaczone do rozwiązania przez pojedyncze węzły obliczeniowe. Następnie wysyła je do serwera i oczekuje na wyniki częściowe. Na ich podstawie oblicza rozwiązanie końcowe i

przesyła je na serwer. Każdy menadżer zadań zawiera jeden lub kilka modułów, odpowiedzialnych za rozwiązanie konkretnego typu problemu. Wysyła on do serwera nazwy typów problemów, jakie potrafi obsłużyć, dzięki czemu serwer wie, które zadania może do niego kierować.
Zadaniem *węzła obliczeniowego* jest rozwiązywanie podproblemów otrzymanych od serwera komunikacyjnego. Węzeł nie ma dostępu do informacji o całym zadaniu i nie zna wyników uzyskanych przez inne węzły. Otrzymuje jedynie problem częściowy, wyznaczony przez menadżera zadań, rozwiązuje go i wysyła do serwera wyniki swoich obliczeń. Węzeł składa się z jednego lub więcej modułów, zawierających algorytmy do rozwiązywania konkretnych typów podproblemów. Zanim otrzyma zadanie, wysyła do serwera informację dotyczącą rodzajów problemów, które potrafi rozwiązać. Podczas wykonywania algorytmu węzły na bieżąco informują serwer, że obliczenia trwają. W przypadku awarii węzła lub przerwania połączenia, do serwera przestaną dochodzić komunikaty i będzie on mógł skierować dany problem do innego węzła, aby uzyskać brakujący wynik częściowy.
Przykładowym zadaniem, które można rozwiązać korzystając z opisanego klastra obliczeniowego jest problem marszrutyzacji (Dynamic Vehicle Routing Problem).

Rozdział 2

Diagramy Przypadków Użycia

2.1 Konfiguracja

2.1.1 Serwer Komunikacyjny

2.1.2 Menadżer Zadań

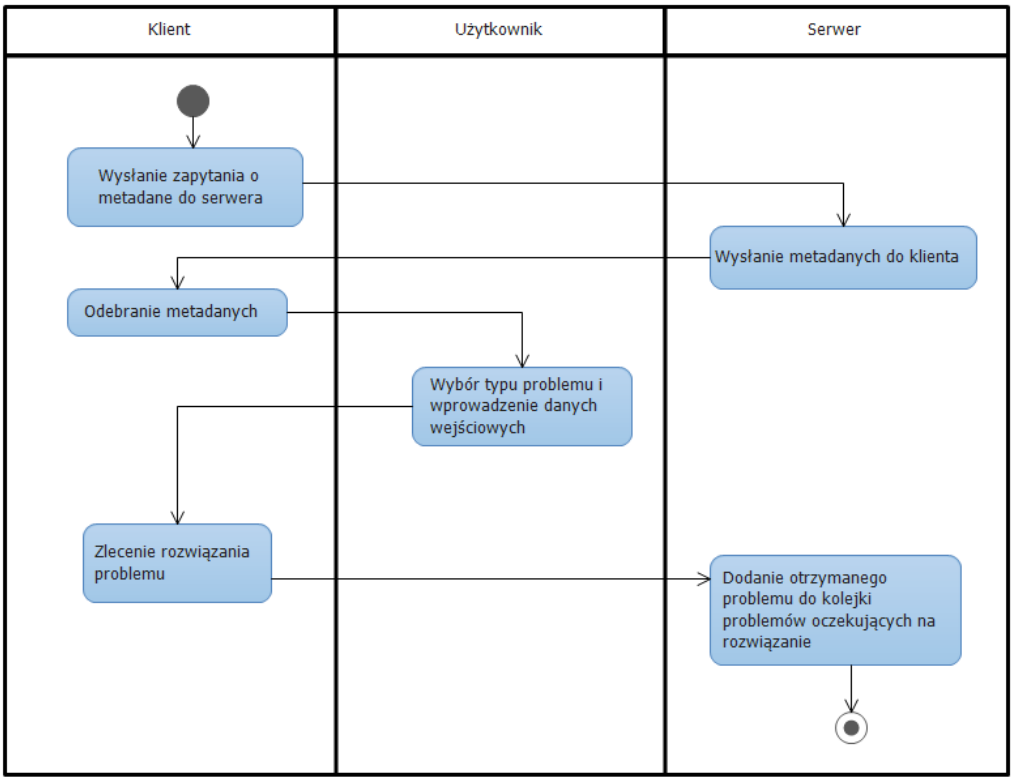
2.1.3 Węzeł obliczeniowy

2.1.4 Klient

Rozdział 3

Diagramy Aktywności

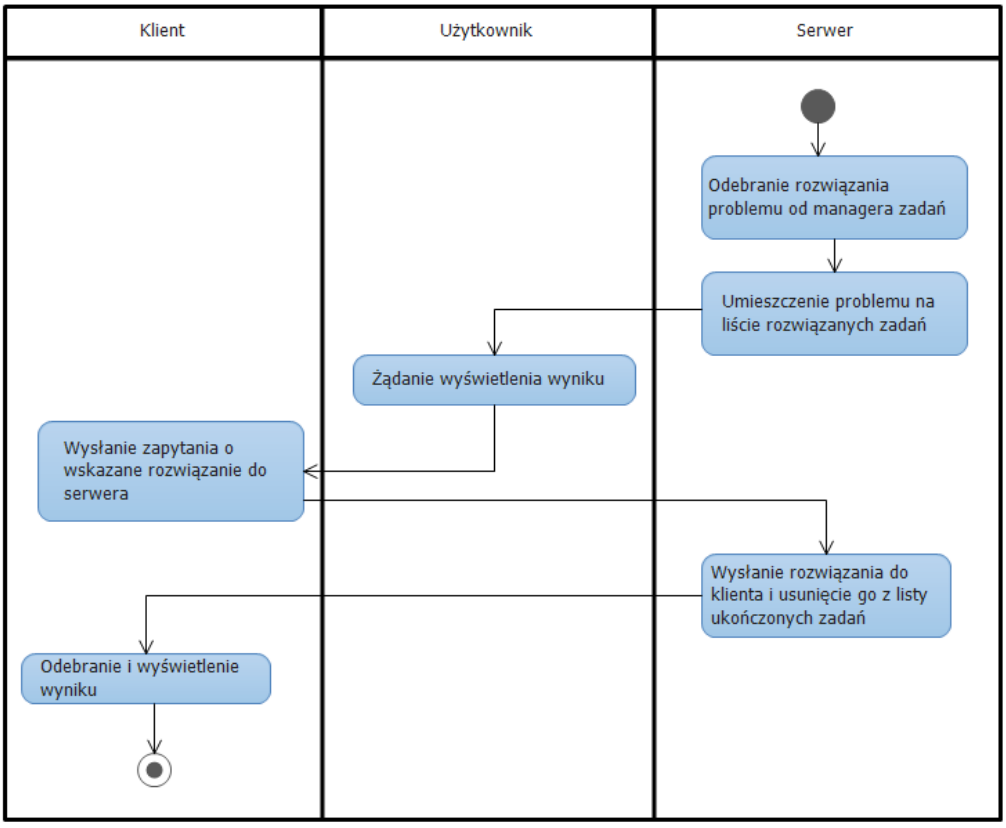
3.1 Zlecenie rozwiązania problemu



Aplikacja kliencka nawiązuje połączenie z serwerem głównym, używając adresu IP serwera zapisanego w pliku konfiguracyjnym. Wysyła zapytanie o metadane - adresy IP serwerów backup'owych oraz nazwy klas problemów,

które mogą być rozwiązane przez ten klaster obliczeniowy (takie, że istnieje przynajmniej jeden *menadżer zadań* potrafiący obsłużyć dany typ problemu). Serwer przesyła metadane do klienta. Następnie użytkownik wybiera spośród dostępnych nazw klas problemów typ zadania jakie ma zostać rozwiązane i wprowadza do programu wszystkie potrzebne dane wejściowe. Aplikacja kliencka wysyła do serwera zlecenie rozwiązania problemu i podane przez użytkownika dane w formacie XML. Serwer odbiera zlecenie przysłane przez klienta i umieszcza problem w kolejce problemów danego typu oczekujących na rozwiązanie. Zadanie znajduje się w kolejce, dopóki któryś z *menadżerów zadań*, potrafiących rozwiązać problem tej klasy, nie zakończy obliczeń i nie będzie mógł się nim zająć.

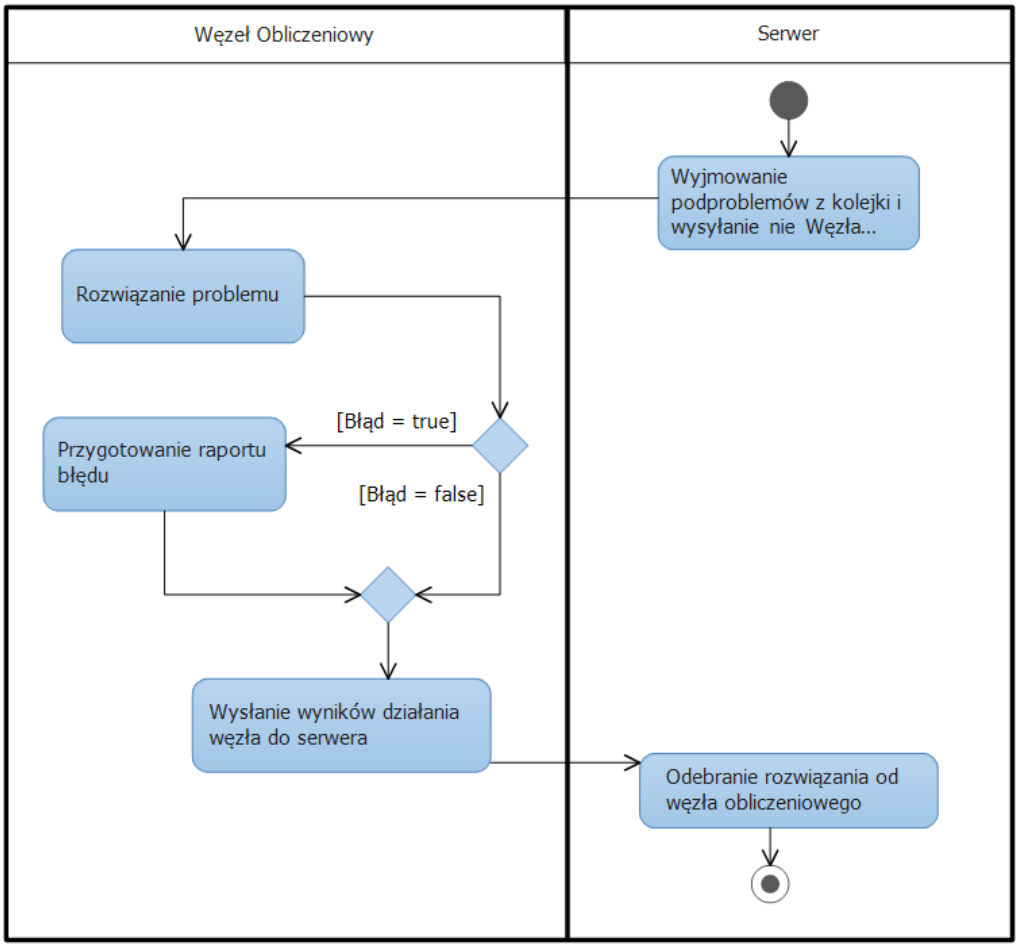
3.2 Odczytanie wyniku



Po otrzymaniu ostatecznego rozwiązania od *menadżera zadań*, serwer umieszcza problem na liście ukończonych, ale jeszcze nieodczytanych rozwiązań.

Następnie użytkownik wybiera z listy problem, którego rozwiązanie chce zobaczyć. Aplikacja kliencka wysyła do serwera żądanie pobrania wskazanego wyniku. Serwer wysyła rozwiązanie odpowiedniego problemu do klienta i usuwa je z listy ukończonych zadań. Klient odbiera wyniki i wyświetla je użytkownikowi.

3.3 Węzeł obliczeniowy



Węzeł obliczeniowy przy uruchomieniu zgłasza swoją obecność serwerowi komunikacyjnemu. Informacje temat serwera znajdują się w pliku konfiguracyjnym węzła. Węzeł wysyła do serwera informacje na temat typów problemów które jest w stanie rozwiązać, dzięki temu serwer komunikacyjny może uwzględniać go przy przesyłaniu do **Menadżera Zadań** informacji na temat

ilości węzłów potrafiących rozwiązać dany typ problemu. Węzeł obliczeniowy nie wykonujący w danym momencie obliczeń otrzymuje od serwera skolejkowany podproblem, podzielony na części przez **Menadżer Zadań**. Zadaniem węzła obliczeniowego jest rozwiązanie otrzymanego zadania oraz przesłanie rozwiązania do serwera. Jeżeli podczas wykonywania obliczeń wystąpi błąd, węzeł ma za zadanie przygotować raport o błędach a następnie wysłać go do serwera.

Rozdział 4

Komunikacja

4.1 Przykłady wiadomości

- prośba klienta o listę możliwych klas problemów do rozwiązania

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns
      :xsd="http://www.w3.org/2001/XMLSchema" messageGuid="70c61dd5-
      afcb-452f-9ad4-621a0ef78e6c">
3   <MessageType>ProblemsListRequest</MessageType>
4 </Message>
```

examples.xml

- odpowiedz serwera z listą dostępnych klas problemów do rozwiązania

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns
      :xsd="http://www.w3.org/2001/XMLSchema" messageGuid="70c61dd5-
      afcb-452f-9ad4-621a0ef78e6c">
3   <MessageType>ProblemsList</MessageType>
4   <ProblemClassList>
5     <ProblemClass problemClassName="name1" problemClassId="ee28fec2
          -6361-4a58-aaf1-b9ff0f509743" />
6     <ProblemClass problemClassName="name2" problemClassId="f9ed0a8f-
          a9f1-494a-aea6-68ffc533934e" />
7   </ProblemClassList>
8 </Message>
```

examples.xml

- zlecenie zadania przez klienta

```
<?xml version="1.0" encoding="UTF-8"?>
<Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns
    :xsd="http://www.w3.org/2001/XMLSchema" messageGuid="70c61dd5-
    afcb-452f-9ad4-621a0ef78e6c">
  <MessageType>TaskOrder</MessageType>
  <problemClassId>4fbcbba1-014e-4643-b60f-f7888a95bb54</
      problemClassId>
  <Data>dane w postaci XML</Data>
  <TaskName>CustomTaskName</TaskName>
</Message>
```

examples.xml

- lista rozwiązanych zadań przesyłana do klienta

```
<?xml version="1.0" encoding="UTF-8"?>
<Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns
    :xsd="http://www.w3.org/2001/XMLSchema" messageGuid="70c61dd5-
    afcb-452f-9ad4-621a0ef78e6c">
  <MessageType>TaskSolvedList</MessageType>
  <TasksList>
    <Task name="CustomName1" id="8844d38c-ac3a-4774-9e81-957a708c6f0b
        " problemClassName="className1" />
    <Task name="CistomName2" id="d104742a-76bb-4942-92bb-9f92149f4863
        " problemClassName="className2" />
  </TasksList>
</Message>
```

examples.xml

- rozwiązanie przesłane do klienta

```
<?xml version="1.0" encoding="UTF-8"?>
<Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns
    :xsd="http://www.w3.org/2001/XMLSchema" messageGuid="70c61dd5-
    afcb-452f-9ad4-621a0ef78e6c">
  <MessageType>TaskSolved</MessageType>
  <problemClassId>4fbcbba1-014e-4643-b60f-f7888a95bb54</
      problemClassId>
  <Data>rozwiazaniew postacie XML</Data>
  <TaskName>CustomTaskName</TaskName>
</Message>
```

examples.xml

4.2 Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="
    qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F
          ]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12})|(\{[0-9a-fA-F]{8}-[0-9a-fA-F
          ]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}\})"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="messageTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ProblemsListRequest"/>
      <xs:enumeration value="ProblemsList"/>
      <xs:enumeration value="TaskOrder"/>
      <xs:enumeration value="TaskSolvedListRequest"/>
      <xs:enumeration value="TaskSolvedList"/>
      <xs:enumeration value="TaskSolved"/>
      <xs:enumeration value="SubtaskOrder"/>
      <xs:enumeration value="SubtaskSolved"/>
      <xs:enumeration value="NodeError"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="Message">
    <xs:complexType>
      <xs:sequence >
        <xs:element type="messageTypes" name="MessageType"/>
        <xs:element type="guid" name="problemClassId" minOccurs="0"/>
        <xs:element type="xs:string" name="Data" minOccurs="0"/>
        <xs:element name="ProblemClassList" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ProblemClass" maxOccurs="unbounded"
                  minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:string" name="problemClassName"
                          use="required"/>
```

```
37                    <xs:attribute type="guid" name="problemClassId" use="
                          required"/>
38                  </xs:extension>
39                </xs:simpleContent>
40              </xs:complexType>
41            </xs:element>
42          </xs:sequence>
43        </xs:complexType>
44      </xs:element>
45      <xs:element type="xs:string" name="TaskName" minOccurs="0"/>
46      <xs:element name="TasksList" minOccurs="0">
47        <xs:complexType>
48          <xs:sequence>
49            <xs:element name="Task" maxOccurs="unbounded" minOccurs="0">
50              <xs:complexType>
51                <xs:simpleContent>
52                  <xs:extension base="xs:string">
53                    <xs:attribute type="xs:string" name="name" use="
                          required"/>
54                    <xs:attribute type="guid" name="id" use="required"/>
55                    <xs:attribute type="xs:string" name="problemClassName"
                           use="required"/>
56                  </xs:extension>
57                </xs:simpleContent>
58              </xs:complexType>
59            </xs:element>
60          </xs:sequence>
61        </xs:complexType>
62      </xs:element>
63    </xs:sequence >
64    <xs:attribute type="guid" name="messageGuid" use="required"/>
65  </xs:complexType>
66 </xs:element>
67 </xs:schema>
```

messageSchema.xsd