МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Ижевский государственный технический университет им. М. Т. Калашникова»

К защите

Руководитель направления
д.т.н., профессор Лялин В.Е.
25 мая 2016 г.

Филиппов Алексей Николаевич

Модели RPC в проектировании олимпиадного сервера

09.04.04 – Программная инженерия
231000.68-1 – Разработка программно-информационных систем

Диссертация на соискание академической степени магистра

Магистрант
Филиппов А.Н.

Научный руководитель
к.т.н., профессор
Тарасов В.Г.

Руководитель программы
д.т.н., профессор Лялин В.Е.

Ижевск 2016

РЕФЕРАТ

Объем и структура диссертационной работы. Диссертация содержит введение, три? главы, заключение и три? приложения, изложенные на TODO страницах машинописного текста. В работу включены TODO рисунков, TODO таблиц, список литературы из TODO наименований. В приложениях представлены основные фрагменты исходного кода разработанных инструментов, структура базы данных.

TODO

Ключевые слова: распределённые системы, удалённый вызов процедур, модели RPC, система проведения олимпиад.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработка систем автоматизации образовательных процессов является важным этапом информатизации общества. Это позволяет повысить эффективность образовательного процесса путём снижения нагрузки на сотрудников, исключения ошибок и неточностей, допускаемых человеком. Также это позволяет создавать программы удалённого обучения без непосредственного участия преподавателя.

Автоматизация в сфере образования затрагивает множество аспектов, от электронных табелей успеваемости и расписаний занятий до автоматизированных тестирований учащихся.

Неотъемлемой частью образовательного процесса является практическая работа. Она характеризуется решением определённого класса задач в пределах изучаемого курса. Существуют группы задач, решения для которых могут быть проверены в автоматическом режиме, к примеру из курсов программной инженерии. Существуют системы, автоматизирующие проверку решений для таких задач, а также проведения лабораторных и практических работ, соревнований и олимпиад.

С ростом количества пользователей таких систем возрастают и системные требования. Одним из способов решения проблем производительности является распределение нагрузки между отдельными серверами проверяющей системы. Это позволяет добиться горизонтальной масштабируемости, то есть возможности увеличения мощности системы путём увеличения количества узлов сети. Такая система уже будет являться распределённой [1]. Для них характерно распределение функций и ресурсов между множеством узлов. Такие системы часто реализуют избыточность ресурсов, что позволяет им оставаться работоспособными даже при выходе части узлов из строя.

Одной из проблем, которые необходимо решить при создании такой системы, это координация работы узлов системы и передача результатов вычислений между ними. Для решения этой проблемы может применяться концепция RPC – remote procedure call. Эта концепция позволяет организовать процесс передачи данных в виде запрос-ответ. Различные типы запросов и передаваемых данных создают необходимость использования различных технологий RPC.

Целью работы является анализ различных моделей использования RPC и разработка технологии RPC, которая требуется олимпиадной системе, но не имеет аналогов, удовлетворяющих требованиям.

Для достижения цели необходимо решить следующие **задачи**:

- исследование моделей использования RPC в олимпиадной системе BACS, а также требований, предъявляемых к RPC в рамках каждой модели;
- исследование существующих технологий RPC и анализ их соответствия полученным моделям;
- разработка технологии RPC для проверяющей системы, для которой не существует аналогов, удовлетворяющих требованиям моделей.

– разработка протокола broker

– реализация протокола для C++, Go, Python и C#

Объектом исследования являются распределённые системы.

Предметом исследования являются межпроцессные взаимодействия в распределённых системах посредством RPC.

На защиту выносятся результаты разработки и исследования моделей RPC, а также результаты практической реализации технологии RPC.

Научная новизна работы состоит в разработке моделей RPC.

Представленные в диссертации модели описывают качественные особенности и требования, предъявляемые к RPC. Это вносит ясность в работу инженера при выборе определённой технологии, предоставляя понятный для него алгоритм действий.

Практическая ценность работы. На базе полученных моделей разработана технология RPC, которая позволяет организовать асинхронную и надёжную передачу данных с учётом распределения нагрузки и горизонтальной масштабируемости. Данная технология может быть использована для организации вычислений ряда распределённых систем, в том числе рассмотренной в диссертации.

1. АНАЛИТИЧЕСКИЙ ОБЗОР МЕЖПРОЦЕССНЫХ ВЗАИМОДЕЙСТВИЙ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К НИМ

Целью данной главы является анализ межпроцессных взаимодействий в олимпиадной системе BACS, обзор существующих реализаций, анализ требований и постановка цели и задач работы.

1.1. Общая характеристика

Межпроцессное взаимодействие IPC (inter-process communication) – это обмен данными между отдельными процессами. Процессы могут быть запущены как в одном адресном пространстве, так и на удалённых компьютерах. Обеспечиваются такие взаимодействия как посредством ядра операционной системы, так и при помощи механизмов пользовательского пространства (к примеру внешних программных модулей).

Среди механизмов IPC выделяют механизмы обмена сообщениями, синхронизации, разделения памяти и удалённых вызовов (RPC – remote procedure call). Для распределённых систем актуальны механизмы, независимые от ядра операционной системы и позволяющие реализовывать связь между узлами, запущенными под управлением различных операционных систем. К таким относятся механизмы обмена сообщениями и удалённых вызовов.

1.1.1. Обмен сообщениями

Обмен сообщениями – это форма связи, используемая в параллельных вычислениях, реализуемая путём посылки пакетов информации получателям, которые могут содержать команды и уведомления, а также данные для обработки.

Обычно обмен сообщениями реализуется асинхронно, сообщение может быть доставлено неопределённому кругу получателей, в том числе независимо от отправителя, если используется программа-посредник – брокер. Наличие брокера с одной стороны приводит к централизации системы, с другой позволяет упростить архитектуру. Брокер хранит сообщения до момента доставки, уведомляет об ошибках, реализует маршрутизацию сообщений.

1.1.2. RPC

Удалённый вызов процедур – это класс технологий, позволяющих производить вызов процедур одной компьютерной программы из адресного пространства другой. Как правило реализации RPC позволяют абстрагироваться от деталей сетевого взаимодействия, фактически стирая разницу между вызовом удалённой и локальной процедуры.

Технологии RPC удобно применять для организации вычислений в распределённых системах.

В языках программирования высокого уровня команда есть вызов процедуры. В распределённых системах таким образом команда есть удалённый вызов процедуры. RPC позволяет передавать структурированные данные небольшого объёма (десятки мегабайт). Так как используются типы данных языков программирования высокого уровня, происходит прозрачная интеграция различных частей системы.

Любая технология RPC состоит из двух частей: протокола RPC и реализации протокола для конкретных языков программирования. Следует обратить внимание, что для большинства языков программирования следует создавать отдельную реализацию (в некоторых случаях обёртку к существующей), так как в каждом языке есть свои характерные особенности стиля работы, что позволяет сделать реализацию максимально приближенной к языку и понятной пользователям.

Протокол RPC с другой стороны является общим, он един для всех реализаций. Это позволяет объединять различные узлы распределённой системы, реализованные на различных языках программирования.

Протокол RPC обычно является двухуровневым. Уровень представления данных определяет методы кодирования информации, такие как байтовое представление чисел и других простых типов данных, массивов, строк, структур. Транспортный уровень определяет механизмы передачи уже закодированных байт по сети. Часто используется какой-либо существующий протокол в качестве транспортного.

1.2. Обзор существующих технологий RPC

1.2.1. XML-RPC

Использует XML для представления данных, HTTP в качестве транспортного протокола.
Пример запроса:

```
<?xml version=”1.0”?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
     <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>
```

Пример ответа:

```
<?xml version=”1.0”?>
<methodResponse>
  <params>
```

```
    <param>
      <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

Сильными сторонами технологии являются исключительная простота и распространённость среди языков программирования. Недостатками являются низкая производительность и избыточность представления данных (недостаток XML). Последнее влечёт за собой чрезмерную нагрузку на сеть при значительных объёмах передаваемых данных. В случае же если преобладают бинарные данные, они кодируются в base64, специальной кодировке для представления бинарных данных в текстовом виде. Особенностью данной кодировки является увеличение размера передаваемой информации на 33%.

1.2.2. JSON-RPC

Технология аналогична XML-RPC, за исключением использования JSON вместо XML. Удобна для реализации в Web ввиду распространённости поддержки JSON в браузерах. Сильные и слабые стороны те же.

1.2.3. D-BUS

Система для организации RPC в пределах одной операционной системы. Оперирует концепцией сервиса: каждое серверное приложение при запуске регистрирует сервис с заранее известным именем. Клиенты будут обращаться к сервису при помощи данного имени.

Сообщения в D-Bus бывают четырёх видов: вызовы методов, результаты вызовов методов, сигналы (широковещательные сообщения) и ошибки.

Используется собственный бинарный протокол представления данных. В качестве транспортного протокола используется как правило доменный сокет UNIX, но может быть применён и TCP.

Сильными сторонами являются высокая скорость работы и широкая поддержка в языках программирования. Так как технология ориентирована на работу в пределах одного компьютера, для распределённых систем не подходит.

1.2.4. Java RMI

Java remote method invocation – программный интерфейс вызова удалённых методов в языке Java.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
```

```
public interface RmiServerIntf extends Remote {
    public String getMessage() throws RemoteException;
}
```

Для кодирования данных используется встроенная в язык технология сериализации. В качестве транспортного протокола используется TCP с реализованным поверх него собственным бинарным протоколом.

Сильными сторонами являются высокая скорость работы и бесшовная интеграция с кодом на языке Java. Недостатком – поддержка только платформы Java. Последнее не позволяет создавать по-настоящему кросс-платформенные интерфейсы в распределённых приложениях, если только для разработки не используется исключительно Java.

1.2.5. SOAP

Simple Object Access Protocol – простой протокол доступа к объектам. Является расширением XML-RPC. Он является более гибким, но также добавляет и свои недостатки к XML-RPC: существуют несовместимые реализации протокола. Технология, как и XML-RPC, снижает скорость работы за счёт передачи данных в форме XML.

1.2.6. Apache Thrift

Это язык описания интерфейсов для определения и создания служб на различных языках программирования. Является фреймворком RPC. Содержит генератор кода для таких языков как C#, C++, Cappuccino, Cocoa, Delphi, Erlang, Go, Haskell, Java, OCaml, Perl, PHP, Python, Ruby и Smalltalk.

Поддерживает множество форматов передачи и кодирования данных, среди них бинарные и текстовые, в том числе отладочный (легко читаемый человеком), JSON.

Поддерживает различные транспортные протоколы, в том числе сетевые.

Сильными сторонами являются гибкость, распространённость реализаций для различных языков программирования и совместимость между ними, высокая скорость работы. Основным недостатком выделяют низкое качество документации. Кроме этого технология сериализации данных неотделима от фреймворка, потому её использование для сохранения данных например на жёсткий диск может быть затруднительно.

Технология широко применяется компанией Facebook для построения распределённых систем.

1.2.7. gRPC

gRPC – высокопроизводительный фреймворк для построения сервисов и клиентов. Разрабатывается компанией Google, находится в открытом доступе.

Технология использует Google Protocol Buffers в качестве технологии представления данных. Эта технология широко распространена среди различных языков программирования.

В качестве протокола передачи данных используется HTTP/2 – современная версия протокола HTTP. Она отличается поддержкой параллельной передачи данных в рамках одного соединения.

gRPC имеет реализации для C, C++, Java, Go, Node.js, Python, Ruby, Objective-C, PHP и C#. Отличается высокой скоростью работы и совместимостью между реализациями. К недостаткам относят сложность анализа закодированных данных. Также следует иметь ввиду что в настоящее время многие реализации являются новыми и не получили достаточной стабильности работы.

1.3. Требования к технологии RPC

Среди требований, предъявляемых к RPC, выделяют:

- переносимость данных;
- надёжность;
- производительность;
- диагностика неисправностей.

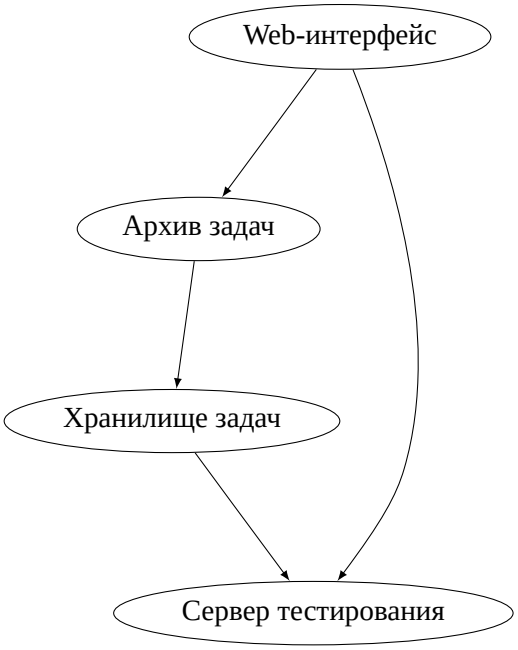

Рис. 1.1: RPC в BACS

В зависимости от

1.3.1. Переносимость данных

Разные части системы могут быть запущенны на различных платформах. Это включает в себя как различные архитектуры процессора, операционные системы, так и языки программирования и интерпретаторы. Представление данных на различных платформах может существенно отличаться, к примеру, различным порядком байт, выравниванием, размером указателя, представлением строк и массивов. Простое копирование представления данных из оперативной памяти не позволит взаимодействовать различающимся частям системы. Необходим механизм сериализации в платформо-независимое представление для последующей передачи в другую часть системы.

1.3.2. Надёжность и производительность

Отдельные компоненты могут быть запущены на различных серверах. Сбои при работе сети, технические работы на сервере (даже кратковременные), моменты пиковой нагрузки с временным отказом от обработки новых запросов, – всё это может приводить к недоступности или медленной работе отдельных узлов системы. При запросах к этим узлам остальные узлы должны адекватно реагировать на их недоступность, корректно определяя это в приемлемое время, повторяя запрос при необходимости.

Существует два типа запросов:

– Запрос существующих данных, пример запросов между web-интерфейсом и архивом задач приведён на рисунке 1. При штатной работе системы обработка запроса занимает строго ограниченное время. В случае сбоя запрос может зависнуть. Необходимо прекращать обработку данного запроса, информируя вызывающую сторону об ошибке.
– Запрос на совершение ресурсоёмкой операции. Такие запросы происходят между web интерфейсом и сервером тестирования решений. Длительность тестирования решения заранее неизвестна, потому следует обеспечить надёжность асинхронной обработки запроса. При этом ответ web-интерфейсу отправляется сервером тестирования самостоятельно.

1.3.3. Диагностика неисправностей

При обработке запросов могут возникать ошибки. Важно как можно более полно передавать информацию об этих ошибках для последующей диагностики системными администраторами или разработчиками системы. В информацию об ошибке может входить стек вызовов, время вызова, данные запроса, идентификаторы вызывающей и принимающей сторон.

1.4. Выводы, постановка цели и задач работы

2. ИССЛЕДОВАНИЕ МОДЕЛЕЙ ИСПОЛЬЗОВАНИЯ RPC

3. РАЗРАБОТКА RPC НА ОСНОВЕ RABBITMQ

4. ЗАКЛЮЧЕНИЕ

Список литературы

[1] Таненбаум Э. С. Распределенные системы. Принципы и парадигмы / Эндрю Таненбаум, М. ван Стеен − СПб. : Питер, 2003. − 880 с.

[2] RabbitMQ - Messaging that just works [Online] // RabbitMQ - Messaging that just works. URL: https://www.rabbitmq.com/

[3] grpc / grpc.io [Online] // grpc / grpc.io. URL: http://www.grpc.io/

[4] Таненбаум Э. С. Компьютерные сети / Таненбаум Э. С., Уэзеролл Д. − СПб. : Питер, 2016. − 960 с.

[5] Ejudge [Сайт]. 2006. URL: http://ejudge.ru/ (Дата обращения: 16.06.2014).

[6] ACM Server [Сайт].2008. URL: http://acm-server.ru/ (Дата обращения: 16.04.2016).

[7] Bacs 2.0 2006. [Сайт] URL: http://bacs.cs.istu.ru/ (Дата обращения: 16.04.2016).

[8] XMLRPC 2011, [Сайт] URL: http://xmlrpc.scripting.com/default.html (Дата обращения: 16.04.2016).

[9] ACM ICPC 2014, [Сайт] URL: http://icpc.baylor.edu/ (Дата обращения: 16.04.2016).

[10] Мирзаянов М. Р. Интерактивные задачи: алгоритм тестирования, [Сайт] URL: http://codeforces.ru/blog/entry/5152 (Дата обращения: 16.06.2014).

[11] Michael Kerrisk. Namespaces in operation. 2014 [Сайт] URL: http://lwn.net/Articles/531114/ (Дата обращения: 16.04.2016).

[12] Paul Menage. CGROUPS 2004. [Сайт] URL: https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt (Дата обращения: 16.04.2016).

ПРИЛОЖЕНИЕ 1

ИСХОДНЫЙ КОД