



Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация» (РК)

КАФЕДРА «Системы автоматизированного проектирования» (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ по дисциплине: *«Методы оптимизации»*

на тему: *«Удаленный запуск GBSE-решателей»*

Студент РК6-73
(Группа)

(Подпись, дата)

В.О. Голубев
(И.О.Фамилия)

Руководитель курсовой работы

(Подпись, дата)

А.П. Соколов
(И.О.Фамилия)

Консультант

(Подпись, дата)

А. Ю. Першин
(И.О.Фамилия)

К защите! Рекомендованная оценка отлично!

2020 г.

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ Карпенко А.П.
(И.О.Фамилия)
« ____ » _____ 20__ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Методы оптимизации

Студент группы РК6-73

Голубев Владислав Олегович
(Фамилия, имя, отчество)

Тема курсовой работы «Удаленный запуск GBSE-решателей»

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
практическая

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения КР: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 16 нед.

Техническое задание Программная реализация удалённого запуска графоориентированных решателей систем инженерного анализа

Оформление курсового проекта:

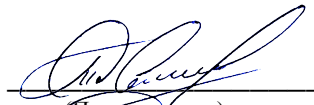
Расчетно-пояснительная записка на 20 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
работа включает 9 рисунков

Дата выдачи задания « 9 » сентября 2019 г.

Руководитель курсовой работы

Студент


(Подпись, дата)

А.П. Соколов
(И.О.Фамилия)

(Подпись, дата)

В.О. Голубев
(И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

АННОТАЦИЯ

Работа посвящена разработке программной реализации удаленного запуска графоориентированных решателей для РВС GCD. В рамках данной работы проводился обзор существующих средств удаленного запуска процедур и исполнительных процессов, протоколов удаленного доступа и других средств, использующихся для решения задачи удаленного запуска. Разработана программная архитектура, позволяющая запустить произвольный графоориентированный решатель.

Тип работы:	Курсовая работа
Тема работы (проект темы):	Удаленный запуск GBSE-решателей
Объект разработки:	Удаленный запуск исполнительных процессов

СОКРАЩЕНИЯ

RPC – Удалённый запуск процедур (англ. *Remote Procedure Call*);

SDK – Набор средств разработки (англ. *Software Development Kit*);

PBC GCD – Распределенная Вычислительная Система GCD;

GUI – графический пользовательский интерфейс (англ. *Graphical User Interface*);

GBSE – графоориентированная программная инженерия (англ. *Graph Based Software Engineering*);

aDOT – формат описания графовых моделей сложных вычислительных методов (англ. *Advanced DOT*);

aINI – формат представления исходных данных (англ. *Advanced INI*);

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	8
1. ПОСТАНОВКА ЗАДАЧИ	11
1.1. Концептуальная постановка задачи	11
2. АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ	12
2.1. Схема архитектуры	12
2.2. Описание алгоритма удаленного запуска решателя	13
2.2. Протокол коммуникации между клиентом и сервером	15
2.3. О технологии ZeroMQ	16
3. ТЕСТИРОВАНИЕ И ОТЛАДКА	18
3.1. Общий принцип тестирования	18
3.2. Результаты тестирования	19
4. АНАЛИЗ РЕЗУЛЬТАТОВ	20
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	21

ВВЕДЕНИЕ

При решении сложных ресурсоёмких задач зачастую вычислительных мощностей локального персонального компьютера уже не хватает и возникает потребность в использовании многопроцессорных удалённых машин. Для успешного задействования мощностей удаленного вычислительного кластера необходимо решить задачи получения соединения, передачи данных и команд по определенному протоколу, сериализации и десериализации данных, распределения нагрузки на процессоры и др [1], [2]. Над решением подобных задач работало много исследователей, а также производителей программного и аппаратного обеспечения.

В 70-х - 80-х годах появился ряд реализаций подхода удаленного запуска процедур (RPC) [3], [4]. Подобные технологии предлагали подход запуска удаленных вычислений таким образом, чтобы это было похоже на вызов обычной процедуры в языках программирования. Среди основных можно выделить DCOM [5], CORBA [6], SOAP(XML-RPC) [7] и др. На сегодняшний день существуют более современные аналоги, такие как Thrift [8], Finagle [9] gRPC [10].

Нельзя не упомянуть архитектурный подход к взаимодействию компонентов распределённого приложения в сети — REST [11], который очень хорошо себя зарекомендовал как способ обмена данными и командами в клиент-серверной архитектуре. Поскольку REST использует такие стандарты, как HTTP, URL, JSON и XML, послать запрос на удаленный запуск процедуры на сервере не представляет большого труда.

Реализации RPC предлагали не только способы удаленного вызова, но и технологии сериализации-десериализации данных и даже шифрования, однако для передачи больших данных, которые использовали ученые и исследователи, производительность была недостаточной. Для научных наборов данных, имеющих большой объём, таких как данные, получаемые от спутников, или численные модели климата, погоды и океанов, были

разработаны специальные бинарные стандарты сериализации, например HDF [12] и netCDF [13].

Для передачи разнородных данных по сети особую популярность приобрел язык разметки XML, однако, программная обработка XML может оказаться неоправданно затратной, по сравнению с работой с данными более простой структуры. В таком случае разработчики рассматривают средства, изначально ориентированные на данные, такие как INI, YAML, JSON.

Несмотря на большое количество технологий удаленного запуска процедур, одним из самых практичных до сих пор считается удаленный доступ к вычислительному кластеру и запуску на его мощностях необходимых вычислений. Существуют специальные программные системы для запуска вычислений на удаленных кластерах, которые используют протоколы SSH, FTP и SFTP. Например, программа COMSOL Desktop [14] позволяет с помощью графического интерфейса запустить задачу на расчет, просматривать текущий статус вычислений и получить результирующий файл, в котором содержатся результаты вычислений.

Запуском вычислительного процесса на кластере занимается специальная программа — система управления кластером, главными задачами которой является запуск вычислительных процессов, балансировка нагрузки на CPU и GPU, защита от сбоев, максимально быстрый возврат результата вычислений. Примерами таких систем могут служить Microsoft Cluster Server [15], Beowulf, Condor, MOSIX, EnFuzion, PBS [16] и др. Многие системы представлены в виде набора дополнений к ядру операционной системы, позволяющее распределять процессы между узлами в составе кластера.

Одной из альтернатив запуску больших вычислений на кластере являются GRID-сети, которые применяются учеными для вычислительных задач, когда производительности кластеров из мощных серверов недостаточно [17].

Алекперов Р.К. [18] в своем исследовании рассмотрел организацию распределенных вычислительных сред на основе GRID-технологии и проанализировал специфические особенности этих задач. В своей работе он

привел исторические примеры, когда распределенные вычисления помогли решать задачи, требующие десятки лет вычислений, за несколько недель.

Одной из самых популярных программных систем для запуска вычисления и развертывания GRID-сетей является BOINC, которое используется в десятках научных проектов.

Несмотря на большую производительность, развертка GRID-сети не является самым лучшим решением для проектов с ограниченными ресурсами и задачами не требующих множества часов вычислений. Более рациональным вариантом может оказаться задействование облачных платформ для запуска вычислительных задач. На сегодняшний день существуют платформы для запуска решателей от всемирно известных производителей программного обеспечения, таких как Siemens, ANSYS, MSC [19], [20] и др. Подобное решение позволит исследователю сосредоточиться на решении прикладных задач, а представитель сервиса берет на себя обязательства по предоставлению необходимых вычислительных мощностей с предустановленным программным обеспечением.

Изучив различные подходы к удаленному запуску вычислений, можно сделать вывод, что одним из самых популярных способов на сегодняшний день является удаленный доступ к вычислительной машине. Создание приложения для программного запуска вычислений подразумевает решение множества задач, для чего существуют различные средства в виде сетевых протоколов, форматов данных, готовых библиотек для языков программирования и т.д.

В данной работе ставится цель разработать программную реализацию удалённого запуска графоориентированных решателей систем инженерного анализа.

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Концептуальная постановка задачи

Объект разработки: Удаленный запуск исполнительных процессов.

Цель разработки: Создание программного обеспечения удалённого запуска произвольных графоориентированных решателей PBC GCD.

Задачи курсовой работы.

- 1) Провести обзор литературы по теме «Технологии и методы удалённого запуска процедур и функций на высокопроизводительных вычислительных системах».
- 2) Доработать функционал веб-клиента в части автоматической генерации GUI.
- 3) Разработать плагин к серверу приложений PBC GCD на языке Python, позволяющий запустить произвольный решатель системы PBC GCD удалённо.

Требования.

- 1) Тестовая функция GRPH_SOLVER_WEB должна запускаться, пользователь должен иметь возможность выбрать решатель (среди доступных, определённых в виде aDOT файла в репозитории gcddot), выбрать файл постановки задачи (среди доступных, определённых на основе aINI файла в репозитории gcdini) и запустить обработку.
- 2) В результате обработки на основе тестового решателя на стороне web-клиента должно выдаваться сообщение о завершении обработки.

2. АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ

2.1. Схема архитектуры

Решение реализовано в рамках клиент-серверной архитектуры PBC GCD.

Клиентская часть (comwpc) реализована на языке Python с использованием web-фреймворка Django [21]. Сервер приложений реализован на языке Python. Для работы с сокетами используется модуль ruymq — реализация технологии ZeroMQ [22] для языка Python. В качестве сервера баз данных используется PostgreSQL. Для решения задачи были задействованы модули comsdk и pycomsdk - SDK для программных реализаций сложных вычислительных методов в рамках графоориентированной технологии GBSE на языках C++ и Python. Диаграмма компонентов системы удаленного запуска представлена на рис.1.

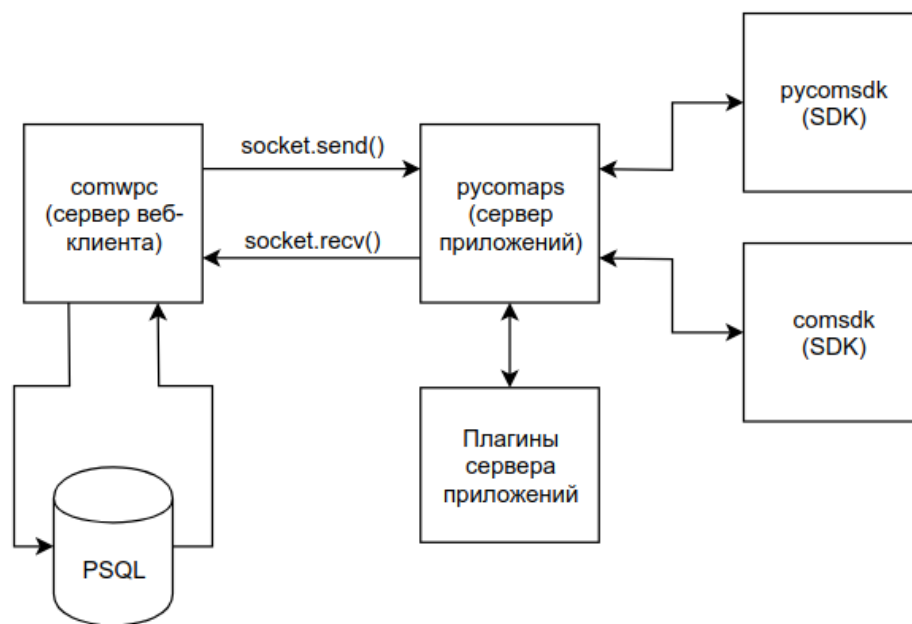


Рисунок 1. Диаграмма компонентов системы удаленного запуска

2.2. Описание алгоритма удаленного запуска решателя

Веб-клиент использует собственный генератор GUI, который на основе файла исходных данных в формате .slv создаёт GUI пользователя для ввода входных данных, таких как имя решателя и имя файла входных данных для него в формате .tsk. Генератор, также, создает кнопку «Обработать» и

привязывает к ней необходимый субплагин на сервере веб-клиента - GRPH_SOLVER_WEB. В субплагине формируется запрос на сервер приложений. Более подробное описание протокола коммуникации между клиентом и сервером представлено в разделе 2.2.

Сервер приложений принимает запрос и создаёт отдельный процесс (worker) для обработки запроса. Worker запускает плагин к серверу приложений для запуска решателя. На основании интерпретации тела запроса плагин получает название aDOT-файла с представлением графовой модели решателя и название файла с исходными данными в формате .tsk. Далее управление передается парсеру файлов формата aDOT из модуля `pycomsdk`. Парсер представляет из себя экземпляр класса `Parser`, который имеет несколько методов для обработки файла графовой модели (см. рис. 2) Метод `Parser.parse_file()` формирует Python-объект `Graph`, в котором содержится информация о рёбрах и вершинах графа. Метод `Parser.generate_cpp()` генерирует исходный код решателя на языке C++.

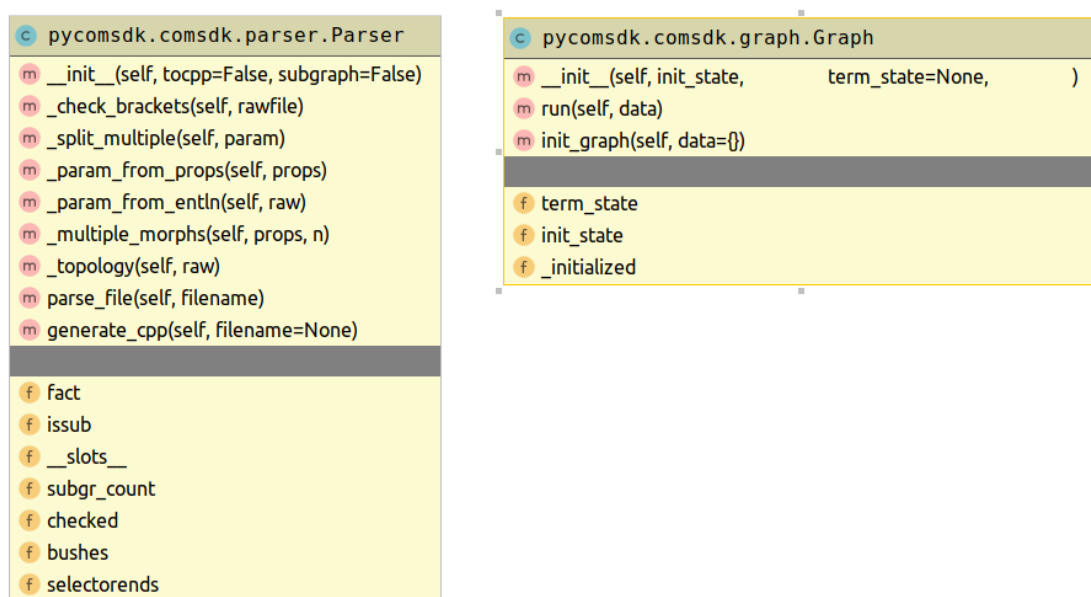


Рисунок 2. UML-диаграмма классов `Parser` и `Graph`.

Непосредственный запуск решателя может осуществляться одним из двух способов. Если функции-обработчики и функции-предикаты, привязанные к рёбрам могут быть импортированы как Python-модули, то производится обход графа напрямую из Python с помощью метода `Graph.run()`. Если функции не

могут быть найдены, то генерируется исходный код решателя на языке C++, в котором подключаются необходимые динамические библиотеки из модуля comsdk. Далее происходит компиляция и запуск решателя. По завершении работы плагина на клиент возвращается ответ об успешном завершении запуска, либо сообщение с кодом ошибки и её описанием. Блок-схема алгоритма представлена на рис. 3.

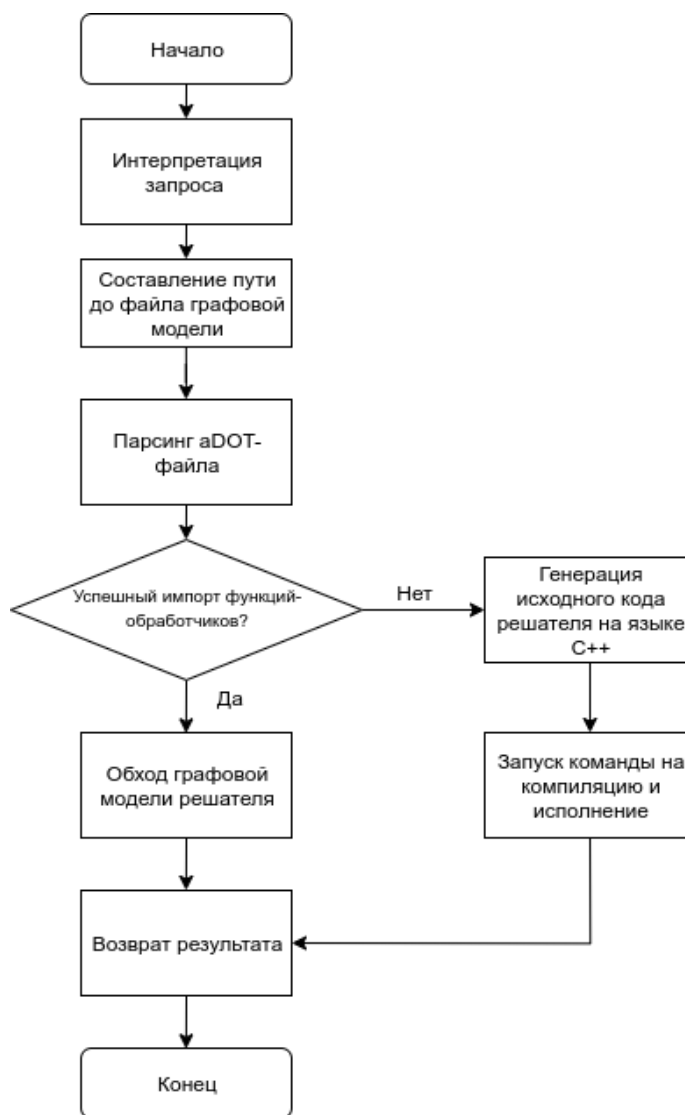


Рисунок 3. Блок-схема алгоритма работы плагина сервера приложений для удаленного запуска

2.2. Протокол коммуникации между клиентом и сервером

Текстовый протокол основывается на формате JSON. Пример сообщения на запуск обработчика на сервере приложений:

```
{
  'command': 'create_worker',
```

```
'worker_name': data['processing'],  
'input': {'data': data['data']}  
}
```

Здесь в значении ключа `command` находится название команды, которую сервер приложений должен исполнить, `worker_name` — название обработчика запроса, `input` — поле, в котором хранится информация с передаваемыми данными.

Сериализация JSON в поток байт производится функцией стандартного модуля `json` языка Python. Десериализация на стороне сервера приложений производится, также, модулем `json`.

Кроме команды на создание обработчика, сервер приложений поддерживает команду запроса текущего статуса работы обработчика — `get_worker_result`. При этом необходимо обязательно указывать идентификатор обработчика.

2.3. О технологии ZeroMQ

Сервер приложений принимает сообщения из TCP-сокета, работа с которым осуществляется с помощью библиотеки `ruzmq`, представляющую реализацию технологии ZeroMQ на языке Python.

Технология ZMQ представляет API для удобной работы с сокетами, который реализует наиболее востребованные паттерны работы в сетевом программировании. Например, самый распространенный паттерн Request-Reply представлен специальной парой сокетов REQ и REP. Функции библиотеки берут на себя сложную работу с очередями сообщений, осуществляя обработку запросов без блокировок. Для этого в ZMQ очереди строятся на неблокируемых структурах данных [23]. Благодаря такому подходу работа с сокетами упрощается до вызова нескольких методов для передачи и принятия сообщений.

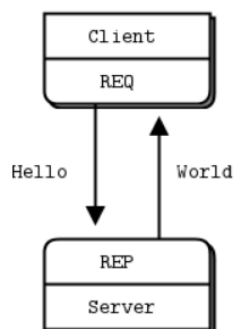


Рисунок 4. Схема паттерна Request-Reply реализованного на zmq-сокетах

На сервере приложений русомарс используется сочетание сокетов Router-Dealer. Этот паттерн позволяет разработать асинхронный сервер, который может принимать множество запросов и распределять нагрузку на параллельные процессы-обработчики. Схематичное изображение работы сокетов Router-Dealer представлено на рисунке 5.

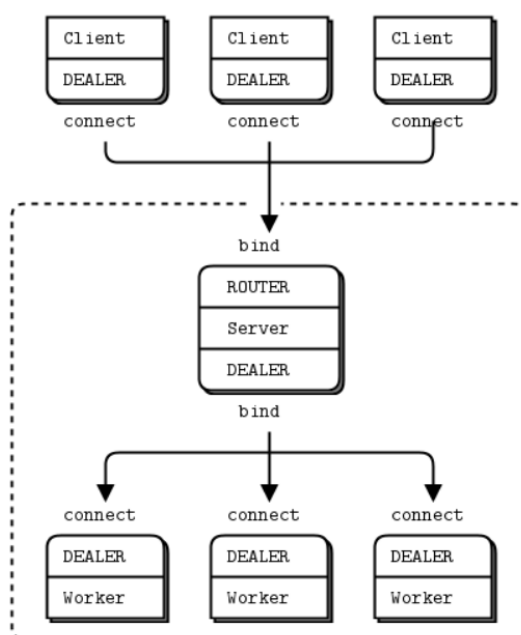


Рисунок 5. Схема паттерна Extended Request-Reply, реализованного на zmq-сокетах

3. ТЕСТИРОВАНИЕ И ОТЛАДКА

3.1. Общий принцип тестирования

В рамках тестирования реализованного функционала был локально развернут сервер приложений русомарс и веб-клиент comwps. Из клиента была вызвана функция GRPH_SOLVER_WEB. Функция WPC_GUI_INI_BUILDER

сгенерировала графический пользовательский интерфейс ввода данных для запуска решателя (см. рис. 6). С помощью нажатия кнопки «Обработать» был отправлен запрос на сервер приложений и вызвана функция GRPH_SOLVER_WEB_HANDLER для обработки тестового решателя. Текст aDOT-файла тестового решателя представлен на рисунке 7.

Рисунок 6. Форма ввода сгенерированная из файла в формате .slv.

Данный файл описывает простой граф из 3 узлов. К ребрам графа привязаны тестовые функции-обработчики и функции-предикаты из динамической библиотеки libcomsdk.

```
digraph gcdfem_gbse_model
{
// Определение функций-обработчиков
PASS_PROCESSOR [module=libcomsdk, entry_func=pass_processor]

// Определение функций-предикатов
PASS_PREDICATE [module=libcomsdk, entry_func=pass_predicate]

// Определение морфизмов
PASS_MORPHISM [predicate=PASS_PREDICATE, function=PASS_PROCESSOR]

// Определение топологии графовой модели метода конечных элементов
BEGIN --> S_1
S_1 --> S_2 [morphism=PASS_MORPHISM, comment = "Препроцессинг завершён, осуществляем подготовку к расчету..."]
S_2 --> S_3 [morphism=PASS_MORPHISM, comment = "Расчет завершён, осуществляем подготовку к постпроцессингу..."]
S_3 --> __END__ [comment = "Постпроцессинг завершён. Расчет завершён."]
}
```

Рисунок 7. Графовая модель тестового решателя в формате aDOT.

3.2. Результаты тестирования

Ожидаемый результат после запуска обработки на удаленный запуск — успешная компиляция сгенерированного srr-файла исходного кода решателя и успешный запуск программы.

Лог сервера приложений, представленный на рисунке 8, показывает, что во время парсинга файла не удалось найти модуль libcomsdk, поэтому был сгенерирован исходный код решателя на языке C++.

```
BEGIN -> S_1
LOADING function pass predicate from libcomsdk module
Module was not found in python modules. Generating .cpp
BEGIN -> S_1
S_1 -> S_2
S_2 -> S_3
S_3 -> END
BUILDING gcdfem_gbse_model
States:
    BEGIN
    S_1
    S_2
    S_3
    END
Generating done!
```

Рисунок 8. Лог сервера приложений, показывающий процедуру парсинга и генерации исходного кода решателя

Компиляция и запуск решателя произошли без ошибок. Необходимые функции-обработчики были найдены в динамической библиотеке libcomsdk и запущены. Результат представлен на рисунке 9.

```
nakmak98@XPS-15:~/Source/com/comsdk$ ../pycomsdk/cpp/run.sh ../../pycomaps/gcdfem_gbse_model.cpp
Successful Compilation!
Running!
Termination!
```

Рисунок 9. Лог bash-скрипта запуска компиляции и исполнения программы.

4. АНАЛИЗ РЕЗУЛЬТАТОВ

Реализация возможности запускать графоориентированные решатели удалённо позволяет существенно упростить решение сложных вычислительных задач с использованием удалённых многопроцессорных вычислительных машин, имеющих большие мощности, чем домашние компьютеры. Это в свою очередь позволит существенно ускорить и упростить проведение вычислительных экспериментов и научных исследований. Разработанный подход поддается легкой модификации, что позволит в будущем усовершенствовать его или адаптировать под новый функционал PBC GCD. На основе разработанного подхода можно спроектировать архитектуру удаленного

запуска решателе на любой вычислительной машине или кластере вне серверной среды PBC GCD. Предпосылками к этому являются разработанный парсер aDOT-файлов с возможностью генерации исходного кода решателя и большое количество библиотек функций-обработчиков.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был проведен анализ подходов к реализации технологии удалённого запуска процедур в различных системах, обзор современных и перспективных технологий. Выявлены наиболее рациональные способы удаленного запуска вычислений в зависимости вычислительной сложности и трудозатрат на развертку необходимого программного обеспечения. Разработана программная архитектура удаленного запуска графориентированных решателей с использованием современных технологий. Как результат была разработана программная реализация плагина к серверу приложений на языке Python, позволяющая запустить произвольный решатель. Был сделан вывод, что подобный подход можно развить и получить решение, которое позволит запускать графориентированные решатели вне серверной среды PBC GCD.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Tanenbaum, Andrew S. Distributed systems: principles and paradigms. Upper Saddle River, NJ: Pearson Prentice Hall, 2006, 702 p.
2. Magnoni, L. Modern Messaging for Distributed Sytems (sic). Journal of Physics: Conference Series, 2015, p. 10-15.
3. RFC-1050: RPC: Remote Procedure Call [Электронный ресурс]. – Режим доступа: <https://tools.ietf.org/html/rfc1050>
4. RFC-1057: RPC: Remote Procedure Call Protocol Specification Version 2 [Электронный ресурс]. – Режим доступа: <https://tools.ietf.org/html/rfc1057>

5. Distributed Component Object Model [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/4a893f3d-bd29-48cd-9f43-d9777a4415b0
6. CORBA to WSDL/SOAP Interworking Specification [Электронный ресурс]. – Режим доступа: <https://www.omg.org/spec/C2WSDL/About-C2WSDL/>
7. SOAP Version 1.2 [Электронный ресурс]. – Режим доступа: <https://www.w3.org/TR/soap12/>
8. Apache Thrift Documentation [Электронный ресурс]. – Режим доступа: <https://thrift.apache.org/docs/>
9. Finagle Official Page [Электронный ресурс]. – Режим доступа: <https://twitter.github.io/finagle/>
10. gRPC Documentation [Электронный ресурс]. – Режим доступа: <https://www.grpc.io/docs/>
11. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. // Springer Science & Business Media, 2011, 528 p.
12. hHDF5 Software Documentation [Электронный ресурс]. – Режим доступа: <https://support.hdfgroup.org/HDF5/doc/>
13. Белоушко, К. Е. Формат NetCDF как стандарт для обмена данными в атмосферных исследованиях // Тезисы II конференции «Базы данных, инструменты и информационные основы полярных геофизических исследований» (POLAR-2012), Троицк, ИЗМИРАН, 2012.
14. Как работать с кластерами через графический интерфейс COMSOL Desktop [Электронный ресурс]. – Режим доступа: <https://www.comsol.ru/blogs/how-to-run-on-clusters-from-the-comsol-desktop-environment/>
15. Raj Rajagopal. Introduction to Microsoft Windows NT Cluster Server: Programming and Administration, 2000, 320 p.
16. Аветисян А.И., Грушин Д.А., Рыжов А.Г. Системы управления кластерами // Труды Института Системного Программирования РАН, Москва, 2002.

17. Zurek R.W, Martin L.J. GridPP: development of the UK computing grid for particle physics // New York: Journal of Physics G: Nuclear and Particle Physics, 2006, p. 1–20.
18. Алекперов А. Организация распределенных вычислений на базе GRID-технологии // Баку: Искусственный интеллект. - 2011. - С. 6-14.
19. Gompute High Performance Computing [Электронный ресурс]. – Режим доступа: <https://www.gompute.com/>
20. TotalCAE Supported Engineering Applications [Электронный ресурс]. – Режим доступа: <https://www.totalcae.com/engineering-applications.php>
21. Django Web Framework [Электронный ресурс]. – Режим доступа: <https://www.djangoproject.com/>
22. ØMQ - The Guide [Электронный ресурс]. – Режим доступа: <http://zguide.zeromq.org/py:all>
23. M. Herlih. A Methodology for Implementing Highly Concurrent Data Object // ACM Transactions on Programming Languages and Systems, 1993, p.746-770.