Санкт-Петербургский государственный университет

Информационно-аналитические системы

Группа 22.Б10-мм

Связь моделей на языке Python с моделирующим ядром на C++

Баранов Артём Артурович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
Старший преподаватель кафедры ИАС, К. К. Смирнов

Консультант:
Разработчик ПО ООО «Софтком» И. А. Андреев

Санкт-Петербург
2024

Оглавление

Введение **3**

1. Постановка задачи **5**

2. Обзор **6**

2.1. Технологии связи между процессами 6

2.2. Работа с плагинами в моделирующей системе реального времени smm-2ng 9

3. Реализация **11**

3.1. Многопоточность и метаданные 11

4. Заключение **14**

Список литературы **15**

Введение

Моделирующая система реального времени — это программное обеспечение, которое используется для создания моделей и симуляции процессов, происходящих в реальном времени. Она позволяет анализировать и предсказывать поведение системы на основе входных данных и параметров. Моделирующие системы реального времени являются важной частью разработки систем [6], применяемых в различных областях, включая управление различного транспорта, медицина [7], космическая отрасль и так далее.

В учебной практике работа связана с проектом smm-2ng от компании “ООО Софтком”, в основе которого лежит моделирующая система реального времени, способная работать в математическом и полунатурном режиме моделирования. Математический режим — режим, в котором моделируется и тестируется алгоритм, но не учитываются временные требования. Полунатурный режим — режим, который используется для тестирования аппаратного обеспечения с помощью использования различных алгоритмов. Моделирующая система реального времени написана на языке C++. В свою очередь, создание алгоритмов на C++ требует значительных усилий и времени [1].

Высокоуровневые языки, такие как Python, имеющие большую экосистему библиотек [3], способны ускорить прототипирование различного рода алгоритмов [4]. Однако у Python присутствует существенный недостаток — он медленный [8]. Однако несмотря на недостатки языка Python, заказчик требует возможности написания алгоритмов на нём. Кроме того, в ходе учебной практики рассматривается работа только с математическим режимом моделирования, который акцентирует внимание на алгоритмах, а не на скорости выполнения моделирования, что позволяет использовать Python.

За предыдущий семестр была добавлена возможность работы с плагинами, написанными на Python. Однако есть ряд мест, которые возможно доработать или улучшить. Основные из которых — это многопоточность и метаданные. Их доработка и улучшение позволяют исполь-

зовать Python алгоритмы наравне с алгоритмами на C++.

1. Постановка задачи

Целью работы является доработка уровня работы Python алгоритмов с ядром smm-2ng до уровня работы C++ алгоритмов с ядром smm-2ng. Для её выполнения были поставлены следующие задачи:

1. сделать обзор необходимых для реализации технологий связи;
2. разработать и реализовать возможность работы Python алгоритмов в многопоточном режиме в ядре;
3. добавить получение метаданных Python алгоритмов для ядра.

2. Обзор

2.1. Технологии связи между процессами

Существует большое количество различных технологий и протоколов, позволяющих передавать и обрабатывать сообщения. Для выбора чего-то конкретного следует выделить связь процессов между друг другом в smm-2ng.

На данном рисунке1 показана связь между родительском процессом и дочерним. Основная идея заключается в том, что ядро вызывает методы init, run, getMetaData в классе PythonAlgorithm в родительском процессе. PythonAlgorithm же, не зная их реализации, делает запрос в дочерний процесс у которого есть соответствующая реализация. После исполнения необходимого метода в дочернем процессе результат передаётся в родительский процесс, в результате чего становится известен ядру. Исходя из данного замечания, можно выделить основную концепцию — использование удаленных вызовов процедур, что более известно, как технология связи RPC. Работу RPC можно увидеть на рисунке2.

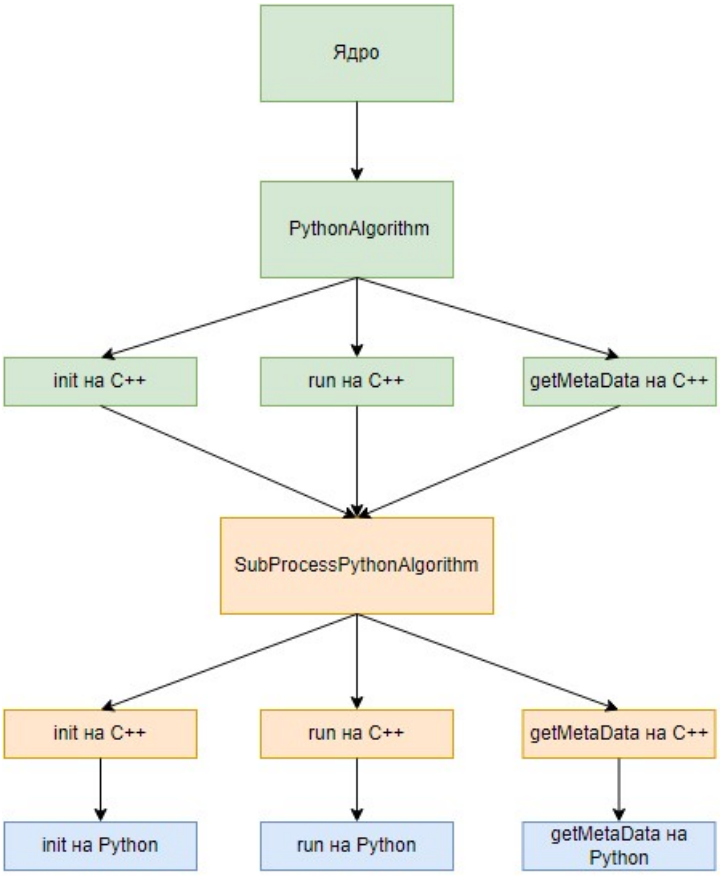

Рис. 1: Связь родительского процесса и дочернего

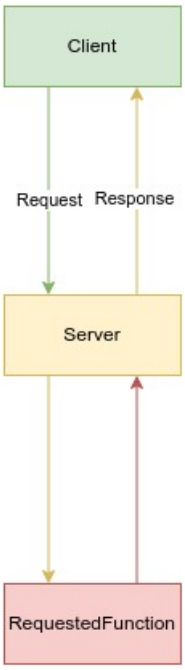

Рис. 2: Работа RPC

Для выбора какого-то определённого решения следует также рас-

смотреть библиотеки, которые реализуют RPC и написаны на языках C или C++. Таких существует довольно много, рассмотрим некоторые наиболее известные из них.

2.1.1. Protobuf

Protobuf — это библиотека, реализующая протокол gRPC, разработанная компанией Google, является open source проектом. Она использует собственный язык Protobuf для определения типов передаваемых сообщений. Ключевыми преимуществами использования этой библиотеки являются: механизм, который обеспечивает эффективную и автоматизированную сериализацию структурированных данных [5], а также простота обновления схемы. Недостатками выбора этой библиотеки являются следующие: при возникновении ошибок с сериализацией или RPC возникает усложнение процесса отладки из-за дополнительного уровня абстракции, предоставляемого Protobuf. Кроме того, если рассматривать минусы в контексте текущего решения, то также возникает проблема введения нового языка в использование в smm-2ng там, где это можно избежать, что влечет за собой уменьшение однородности.

2.1.2. ice

ice — это библиотека, реализующая RPC, являющаяся open source проектом. Она использует язык IDL или же “Interface description language” для определения типов передаваемых сообщений. Ключевым преимуществом использования этой библиотеки является: простота вызова методов удалённых объектов [2]. Её минусы схожи с минусами библиотеки Protobuf, также стоить отметить, что ice использует GNU General Public License version 2 [1].

2.1.3. rpclib

rpclib — это библиотека, реализующая RPC, являющаяся open source проектом. Она не использует языки наподобие Protobuf или же IDL,

[1] Ссылка на лицензию ice https://github.com/zeroc-ice/ice/blob/main/ICE_LICENSE

которые необходимо изучать, прежде чем её будет возможно использовать. Ключевым преимуществом использования этой библиотеки является: отсутствие этапа генерации кода для интеграции в сборку, только C++. Из минусов можно выделить различные зависимости, среди которых есть Boost.

2.1.4. libjson-rpc-cpp

libjson-rpc-cpp — это библиотека, реализующая JSON-RPC, являющаяся open source проектом. Преимуществами этой библиотеки являются: возможность работать как с JSON-RPC 2.0, так и с JSON-RPC 1.0, наличие jsonrpcstub — инструмента, который генерирует классы-заглушки для клиентских и серверных приложений. Недостатком можно отметить лишь недостаточную документацию.

2.1.5. Вывод

Помимо недостатков библиотек, выделенных в обзоре, стоит также упомянуть о недостатках, которые касаются выбора любой библиотеки: появление зависимости по отношению к стороннему проекту, а следовательно и сопровождению выбранной библиотеки. Стоит отметить, что последние обновления большинства библиотек были 3-4 года назад. Кроме того, каждая из библиотек имеет функциональность, превышающая необходимую. Из всего выше перечисленного следует, что наиболее выгодной является собственная реализация RPC через протокол JSON-RPC, в силу того, что json уже активно используется в smm-2ng. Сам протокол основан на технологии RPC и использует сериализацию данных в формате json.

2.2. Работа с плагинами в моделирующей системе реального времени smm-2ng

В основе моделирующей системы лежит ядро, написанное на C++, оно способно работать в нескольких режимах. Однако научная практи-

ка строится только на одном — математическом. В основе этого режима лежит тестирование алгоритма, но не его скорости исполнения. Для того, чтобы работать с алгоритмами в ядре реализован базовый класс — AbstractAlgorithm. От него пользователь реализует свой собственный алгоритм, который представляет собой плагин. Кроме этого, пользователь пишет конфигурацию алгоритма и его параметров, а именно название аргументов и их конкретные значение, всё это передаётся ядру в виде json файла. Со стороны ядра также реализован специальный макрос, который позволяет получить тип передаваемых аргументов в реализованный алгоритм.

Далее на рисунке 3 представлено устройство smm-2ng в контексте работы с алгоритмами

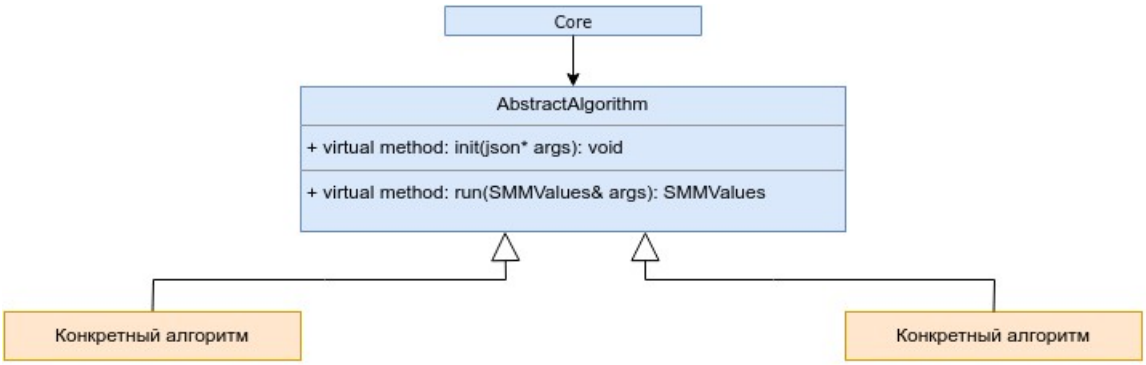

Рис. 3: Работа с плагинами в smm-2ng

3. Реализация

3.1. Многопоточность и метаданные

За предыдущий семестр была добавлена функциональность, позволяющая работать ядру smm-2ng с Python алгоритмами в однопоточном режиме. В ходе текущего же семестра появилась возможность работы не только в многопоточном режиме, а также был переработан однопоточный.

На рисунке 4 можно увидеть архитектуру проекта до изменений, а на рисунке 5 диаграмму последовательностей, которая раскрывает архитектуру проекта после изменений. Ключевые изменения коснулись класса PythonAlgorithm. Если в предыдущем семестре он являлся некоторым связующим звеном между Python алгоритмами и ядром, то в текущем семестре его роль изменилась, он стал прослойкой между ядром и процессами. Методы PythonAlgorithm init и run были также модифицированы. Теперь при вызове метода init класса PythonAlgorithm происходит создание нового дочернего процесса с использованием системного вызова fork и pipe для обмена данными между процессами, в силу того, что работа процессов идёт на одной машине. После этого дочерний процесс запускает исполняемый файл в котором создаётся экземпляр Python алгоритма с помощью использования нового класса: PythonAlgorithmSubProcess. В этом классе находится функциональность PythonAlgorithm с предыдущего семестра. Что касается метода run, благодаря установленному каналу на этапе инициализации, родительский процесс может передать данные в дочерний процесс, а именно в исполняемый файл, где будет вызван аналогичный метод run в PythonAlgorithmSubProcess. В PythonAlgorithm и PythonAlgorithmSubProcess был добавлен метод getMetaData. Этот метод необходим для извлечения названий и типов аргументов у методов init и run различных алгоритмов на Python. Стоит также добавить, что все методы дочернего процесса способны сигнализировать об ошибках в родительский процесс.

В заключение необходимо пояснить, что в текущем семестре работа изначально была написана на более новой версии ядра. Однако на моменте тестирования и отладки кода стало понятно, что запустить программу невозможно. Это было связано с тем, что ядро в новой версии использовало пользовательские типы данных, преобразование которых в C++ под Python и последующее использование их было затрудненно.

Более подробно ознакомиться с кодом можно в репозитории[2].

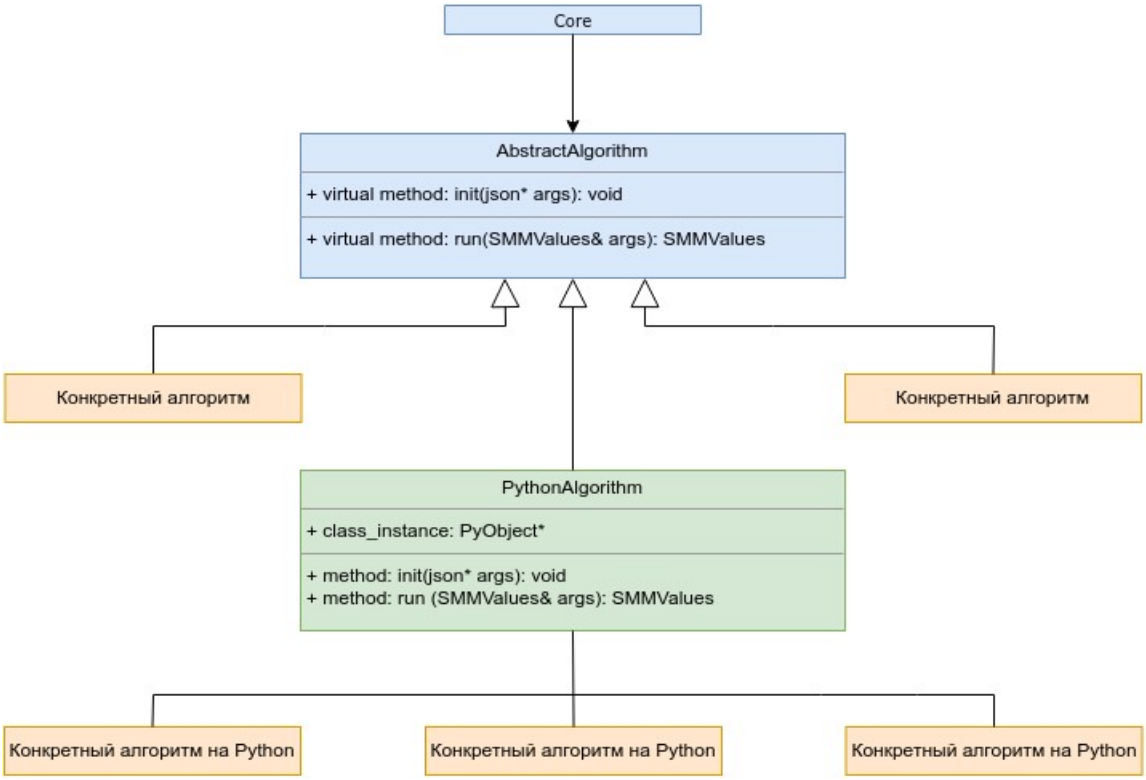

Рис. 4: Архитектура проекта в предыдущем семестре

[2]Ссылка на gitlab с кодом: https://gitlab.softcom.su/artem.baranov/smm-2ng-py/-/tree/AnotherFourthSemester?ref$_type = heads$

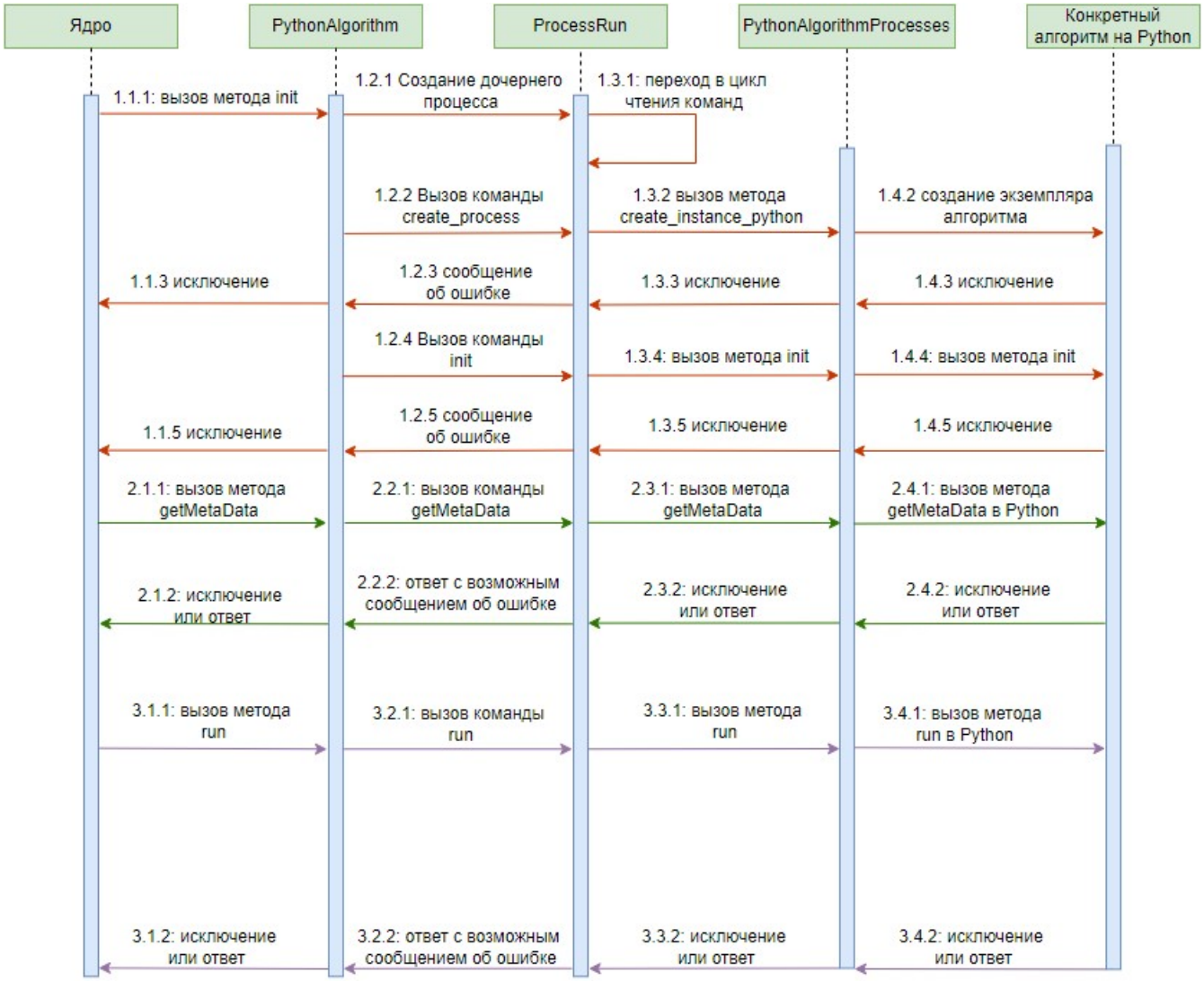

Рис. 5: Диаграмма последовательностей

4. Заключение

По итогам учебной практики были выполнены следующие задачи:

- проведён обзор библиотек, реализующих RPC;
- добавлена возможность работы с Python алгоритмами в многопоточном режиме;
- добавлены методы для получения методанных Python алгоритмов.

Более подробно с кодом можно ознакомиться в репозитории[3].

[3]Ссылка на gitlab с кодом: https://gitlab.softcom.su/artem.baranov/smm-2ng-py

Список литературы

[1] Grosse-Kunstleve Ralf W. why choose python. — URL: https://scfbm.biomedcentral.com/articles/10.1186/1751-0473-7-5 (дата обращения: 12 2023 г.).

[2] Henning Michi. Distributed Programming with Ice. — URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ca00c98b58e32f6edf15c2e31a088b8145ae2875 (дата обращения: 27 апреля 2024 г.).

[3] Pike Thomas. Python libraries. — URL: https://arxiv.org/ftp/arxiv/papers/2201/2201.06040.pdf (дата обращения: 1 января 2024 г.).

[4] Rubinsteyn Alex. Python prototype. — URL: https://www.usenix.org/system/files/conference/hotpar12/hotpar12-final37.pdf (дата обращения: 1 января 2024 г.).

[5] Sanfilippo Filippo. Integrated Flexible Maritime Crane Architecture for the Offshore Simulation Centre AS (OSC): A Flexible Framework for Alternative Maritime Crane Control Algorithms. — URL: https://ieeexplore.ieee.org/abstract/document/7156157 (дата обращения: 21 апреля 2024 г.).

[6] Sifakis Joseph. Modeling system in real time. — URL: https://www.researchgate.net/publication/4117414_Modeling_Real-Time_Systems (дата обращения: 11 2023 г.).

[7] Simonov Michael. Real-time modeling system in medicine. — URL: https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002861 (дата обращения: 12 2023 г.).

[8] Srinath K.R. Slow Python. — URL: https://www.irjet.net/archives/V4/i12/IRJET-V4I1266.pdf (дата обращения: 12 2023 г.).