Разработка масштабируемой программной инфраструктуры для хранения и обработки данных в задачах вычислительной биологии

О. Д. Борисенко <al@somestuff.ru>
А. В. Лагута <laguta@ispras.ru>
Д. Ю. Турдаков <turdakov@ispras.ru>
С. Д. Кузнецов <kuzloc@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва,
ул. А. Солженицына, дом 25

Аннотация. В работе кратко описывается масштабируемая программная инфраструктура для хранения и обработки данных в задачах вычислительной биологии. Обсуждаются использованные технологии, собственное программное решение для предсказания сайтов связывания транскрипционных факторов в геномах, реализация предоставления решения как части веб-лаборатории с REST API и веб-интерфейсом для исследователей.

Ключевые слова: ISPRAS API, OpenZFS, Swift, виртуальная лаборатория, облачные вычисления, Big Data.

1. Введение

Задачи молекулярной биологии и генетики являются очень востребованными и требуют все больших мощностей для обработки данных. Данные, необходимые для решения этих задач, собираются все быстрее с каждым годом, при этом мощностей индивидуальных компьютеров исследователей уже недостаточно для проведения расчетов. К примеру, референсный геном мыши (mm9) занимает на диске больше трех с половиной гигабайт, при этом даже открытые базы данных известных геномов насчитывают сотни живых организмов, и их число растет очень быстро [1], [2].

Также стоит учитывать процесс получения референсных геномов (это также является одной из возможных задач). Дело в том, что устройства для «оцифровки» геномов (секвенаторы) не получают геномы в собранном виде.

Все существующие на текущий момент секвенаторы выдают так называемые «сырые» данные, которые впоследствии необходимо обрабатывать для получения цифрового представления данных генома. Сырые данные представляют собой множество строк фиксированной длины (длина зависит от модели секвенатора), которые в дальнейшем обрабатываются специальными программами-сборщиками для получения генома. Объем сырых данных может в десятки раз превосходить объем собранного генома.

При этом публично доступных сервисов, предоставляющих открытое API для обработки биологических данных, фактически нет. На момент написания статьи из известных ресурсов подобного толка представлен лишь проект Google Genomics [3] в формате закрытого тестирования. На текущий момент сервис Google Genomics предоставляет REST API для получения данных геномов, обработка этих данных пока не представлена. Также стоит отметить, что это API уже год предоставляется лишь в формате закрытого бета-теста, так что для его использования необходимо предварительно подать заявку с описанием целей проекта в Google и дождаться подтверждения от представителей компании.

Таким образом, исследователи в областях молекулярной биологии и генетики вынуждены самостоятельно настраивать все необходимые им утилиты и рабочие инструменты для каждой задачи, причем большая часть этих инструментов работает только в ОС Linux и предоставляет только консольный способ управления.

В данной работе описывается расширяемая инфраструктура для предоставления публичного API и веб-сервиса для исследователей в областях молекулярной биологии и генетики. В рамках этой инфраструктуры предоставляется доступ к инструменту предсказания сайтов связывания транскрипционных факторов с использованием коллекции HOCOMOCO мотивов ДНК [4].

2. Архитектура решения

В этом разделе описывается устройство программной инфраструктуры для обработки и хранения данных для биологических задач в том виде, в котором решение работает сейчас.

2.1 Общая схема

На рис. 1 иллюстрируется архитектура описываемой системы:

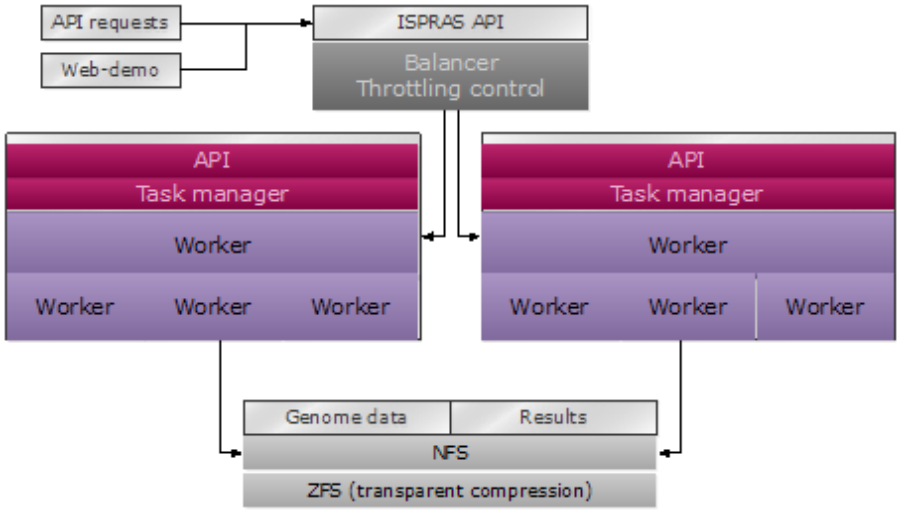

Рис. 1. Архитектура виртуальной лаборатории.

В следующих подразделах описываются составляющие этой схемы. Тем не менее, следует отметить, что такая архитектура системы используется сейчас в связи с малой нагрузкой на реализованный сервис.

Решение разработано таким образом, что в случае увеличения нагрузки мы сможем прозрачно и безболезненно перейти на более гибкую схему с точки зрения хранения и управления ресурсами, так как эта поддержка уже реализована в представленном проекте. На рис. 2 изображена архитектура системы, которую мы будем использовать в случае увеличения нагрузки на сервис.

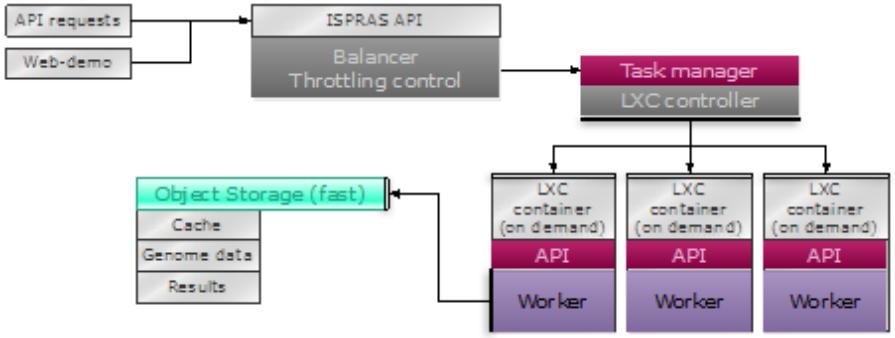

Рис. 2. Альтернативная архитектура виртуальной лаборатории.

Способ миграции на указанную схему работы будет также указан в ходе описания компонентов системы.

2.2 Уровень хранения

Хранение геномов, необходимых для обработки данных и результатов осуществляется на выделенном виртуальном сервере. В качестве файловой системы используется ZFS on Linux (подпроект OpenZFS [5]), в качестве механизма предоставления доступа к данным используется NFS [6]. ZFS on Linux предоставляет те же возможности, что и оригинальная файловая система из ОС Solaris, с том числе возможности прозрачного сжатия, и это очень актуально для задач генетики. Каждый считающий задания узел монтирует систему хранения в свою локальную систему и использует это пространство в качестве своей локальной папки. У этого подхода есть очевидные недостатки: узел хранения становится единой точкой возможного отказа, и потенциально могут возникнуть проблемы с доступом к хранилища из-за особенностей устройства NFS.

Поскольку эти недостатки было несложно предвидеть ещё на этапе проектирования, система была реализована с поддержкой хранения данных в объектном хранилище Openstack Swift [7]. Оно лишено описанных недостатков, предоставляет надежное хранение данных, распределение нагрузки на узлы хранения и позволяет получать доступ к ним при помощи REST API.

2.3 Уровень вычислений

В основе виртуальной лаборатории лежит идея разделения программной части, предоставляющей REST API, и программной части вычислительных сервисов. Вычислительные сервисы могут быть написаны на любом языке программирования и должны отвечать нескольким требованиям для того, чтобы можно было использовать их в рамках виртуальной лаборатории:

1. Программная часть вычислительных сервисов должна гарантировать возможность явного ограничения используемой памяти. Ограничение на память передается из уровня API.
2. Программная часть вычислительных сервисов должна предоставлять данные о статусе выполнения задания в ходе выполнения.
3. Программная часть вычислительных сервисов должна гарантировать возможность ограничения числа подпроцессов, которые могут быть запущены в ходе выполнения.

В качестве примера: введенный в эксплуатацию сервис реализован на языке C++, позволяет в явном виде задавать ограничение по использованию оперативной памяти и позволяет задавать степень многопоточности выполнения.

2.4 Уровень API

В ходе проекта была реализована служебная программа, которая отвечает за обработку и передачу запросов на вычисления нижележащим сервисам. Также

этот программный слой занимается поддержанием пула рабочих процессов. В нынешней схеме размещения ресурсов это означает то, что при инициализации программа оценивает количество ресурсов на локальном узле и выделяет пространство памяти и доступное количество процессоров под выполнение задач. При поступлении запроса на вычисления запрос преобразуется в тот вид, который может принимать на вход нижележащий вычислительный сервис, заданию присваивается идентификатор, и он выдается в качестве ответа на запрос. В дальнейшем при помощи этого идентификатора можно получить статус выполнения задания. Такая схема была реализована прежде всего по причине «тяжеловесности» вычислительных заданий. Для того, чтобы клиенту не нужно было поддерживать соединение, задания ставятся в очередь выполнения, и выполняются в соответствии с количеством свободных рабочих процессов в пуле. В момент, когда статус задания приобретает значение «выполнено», в ответе также указывается URL, по которому располагается файл с результатами вычсислений. Благодаря такой схеме работы возможно использовать как локальные, так и облачные системы хранения. Кроме того, это позволяет реализовать выполнение задач по цепочке (т.е следущий сервис принимает в качестве входных данных результат работы другого сервиса) в случае, если появится потребность в решении таких задач.

Кроме того, поддерживается режим работы, в котором служебная программа следит не за ресурсами локальной машины, а за предоставленными ограничениями на создание экземпляров контейнеров или виртуальных машин в облачной платформе Openstack. При такой схеме работы вычислительные сервисы запускаются не в локальных процессах, а в контейнерах или виртуальных машинах Openstack, и служебная программа следит за созданием контейнеров или виртуальных машин при помощи API Openstack Nova [8].

2.5 Уровни распределения ресурсов

В данной системе используется два уровня распределения ресурсов. Первый уровень реализован на уровне служебной программы, принимающий запросы (см. 2.4). Вторым уровнем распределения ресурсов является технология ISPRAS API, разработанная в Институте системного программирования РАН. ISPRAS API является round-robin балансировщиком нагрузки с единой точкой входа. Со стороны пользователя это выглядит так, что все запросы поступают на один URL (http://api.ispras.ru) со специальными параметрами, и ISPRAS API перенаправляет его на нужный вычислительный сервер. Таким образом обеспечивается прозрачное для пользователя масштабирование сервисов. Кроме того, при помощи ISPRAS API обеспечивается защита от злоупотреблений: каждый пользователь системы (авторизованный или анонимный) обладает заданными для него ограничениями на количество запросов в единицу времени. В случае превышения пользователем этих

ограничений, система сообщает ему об этом, и задание не передается на вычислительные узлы.

2.6 Описание веб-сервиса

Веб-сервис работает по адресу https://api.ispras.ru/demo/gen в демонстрационном режиме и будет дорабатываться в дальнейшем. Сервис реализован с использованием предоставляемого виртуальной лабораторией API и по сути является обыкновенным клиентским приложением к этому API.

Веб-сервис демонстрирует возможности только одного биологического сервиса и позволяет строить предсказания о сайтах связывания транскрипционных факторов на интересующих исследователя наборах участков генома с заданной точностью предсказания. Подробнее задача описывается в следующем разделе.

3. Предсказание сайтов связывания транскрипционных факторов

Поскольку тематика задачи пересекается с этой работой лишь частично, в данной работе приводится постановка задачи и описание возможностей виртуальной лаборатории для решения задач этого класса. Более подробно эта проблема описывается в работах [9], [10], [11]. Описанию решения этой задачи планируется посвятить отдельную статью.

Введем необходимые определения. Геномом называют цифровое представление ДНК организма. Существует множество форматов для хранения геномов для использования в компьютерных программах. В рамках этой задачи традиционно используют формат FASTA [12].

Фактором транскрипции (или фактором инициации транскрипции) называют белок, контроллирующий процесс синтеза матричной РНК при помощи связывания со специфичными для этого фактора участками ДНК. Фактор транскрипции связывается с одним и тем же участком ДНК лишь с какой-то вероятностью.

Участок ДНК (и генома), с которым связывается фактор транскрипции, называется сайтом связывания.

Существует специальный формат моделирования вероятности связывания с заданным участком генома для любого фактора транскрипции (PWM [13]).

Данные для описания модели получаются исследователями экспериментальным путем. Существует набор алгоритмов для вычисления вероятности связывания фактора транскрипции для любого заданного участка генома с использованием PWM модели фактора транскрипции.

Задача предсказания сайтов связывания транскрипционных факторов сводится к нахождению в геноме участков, для которых вероятность связывания с заданным транскрипционным фактором выше заданного исследователем порога.

В качестве входных данных для решения этой задачи пользователь лаборатории может указать следующие параметры:

1. интересующий пользователя геном;
2. транскрипционный фактор, для которого нужно строить предсказания; в системе используется коллекция PWM-моделей транскрипционных факторов HOCOMOCO [4], разработанная исследователями из Института Общей Генетики РАН;
3. степень достоверности предсказания, выраженная через p-value;
4. список отрезков генома, для которых необходимо строить предсказания; исследователей редко интересует весь геном целиком, обычно такой анализ проводится в областях, соответствующих известным генам.

Система обладает самодокументирующим API: при помощи HTTP GET-запросов можно получить оглавление доступных ресурсов (в данном случае списки доступных геномов и транскрипционных факторов с их описанием).

4. Результаты

Построена инфраструктура для решения задач молекулярной биологии и генетики с возможностью эффективного роста и работы в облачных средах.

Разработана система быстрого доступа к результатам обработки данных секвенирования нового поколения с возможностью их обработки на стороне виртуальной лаборатории.

Реализован первый инструмент для решения задач, связанных с предсказанием сайтов связывания транскрипционных факторов с геномом на основе коллекции мотивов ДНК HOCOMOCO, работающий полностью на стороне виртуальной лаборатории по запросу пользователя.

Реализованы REST API для использования лаборатории в собственных программах исследователей и веб-сервис, основанный на реализованном API.

Список литературы

[1]. Страница проекта «1000 Genomes project» — http://www.1000genomes.org/about
[2]. Страница проекта по предоставлению данных геномов и разметки известных генов от университета Санта-Круз — http://genome.ucsc.edu/
[3]. Страница проекта Google Genomics - https://cloud.google.com/genomics/
[4]. Ivan V. Kulakovskiy, Yulia A. Medvedeva, Ulf Schaefer, Artem S. Kasianov, Ilya E. Vorontsov, Vladimir B. Bajic and Vsevolod J. Makeev, HOCOMOCO: a comprehensive collection of human transcription factor binding sites models, Nucleic Acids Research 2012. doi: 10.1093/nar/gks1089
[5]. M. Ahrens, OpenZFS: a Community of Open Source ZFS Developers."AsiaBSDCon 2014, pp. 27-32.
[6]. Страница проекта NFS - http://nfs.sourceforge.net/
[7]. Duan, Zhi Ying, and Yi Zhen Cao, The Implementation of Cloud Storage System Based on OpenStack Swift, Applied Mechanics and Materials. Vol. 644. 2014, pp. 2981-2984.

[8]. Beernaert, L., Matos, M., Vilaça, R., & Oliveira, R., Automatic elasticity in Openstack, In Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, ACM, p. 2.
[9]. Zhou, Qing, and Jun S. Liu., Modeling within-motif dependence for transcription factor binding site predictions., Bioinformatics 20.6, 2004, pp. 909-916.
[10]. Bintu, Lacramioara, et al., Transcriptional regulation by the numbers: applications. Current opinion in genetics & development 15.2, 2005: pp. 125-135.
[11]. Описание механики процесса связывания факторов транскрипции с геномом с точки зрения статистики - http://www.bio-physics.at/wiki/index.php?title=Statistical_Mechanics_of_Binding
[12]. Описание формата FASTA - http://genetics.bwh.harvard.edu/pph/FASTA.html
[13]. Zhang, Xiujun, Position Weight Matrices., Encyclopedia of Systems Biology. Springer New York, 2013, 1721-1722.

Developing scalable software infrastructure for data storage and processing for computational biology problems

O. Borisenko <al@somestuff.ru>
A. Laguta <laguta@ispras.ru>
D. Turdakov <turdakov@ispras.ru>
S. Kuznetsov <kuzloc@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. This article is an overview of scalable infrastructure for storage and processing of genome data in genetics problems. The overview covers used technologies descriptions, the organization of unified access to genome processing API of different underlying services. The article also covers methods for scalable and cloud computing technologies support. The first service in virtual genome processing laboratory is provided and presented. The service solves transcription factors bindning sites prediction problem. The main principles of service construction are provided. Basic requirements for underlying comptutaion software in virtual laboratory environments are provided. Overview describes the implemented web-service (**https://api.ispras.ru/demo/gen**) for transcription factors binding site prediction. Provided solution is based on ISPRAS API project as an API gateway and load-balancer; the middle-ware task-manager software for pool of workers support and for communications with Openstack infrastructure; OpenZFS as an intermediate storage with transparent compression support. The described solution is easy to extend with new services fitting the basic requirements.

Keywords: ISPRAS API, OpenZFS, Swift, virtual laboratory, cloud computing, Big Data.

References

[1]. 1000 Genomes project web page — http://www.1000genomes.org/about
[2]. University of California, Santa Cruz genome project — http://genome.ucsc.edu/
[3]. Google Genomics web page — https://cloud.google.com/genomics/
[4]. Ivan V. Kulakovskiy, Yulia A. Medvedeva, Ulf Schaefer, Artem S. Kasianov, Ilya E. Vorontsov, Vladimir B. Bajic and Vsevolod J. Makeev, HOCOMOCO: a comprehensive collection of human transcription factor binding sites models, Nucleic Acids Research 2012. doi: 10.1093/nar/gks1089
[5]. M. Ahrens, OpenZFS: a Community of Open Source ZFS Developers."AsiaBSDCon 2014, pp. 27-32.
[6]. NFS project web page — http://nfs.sourceforge.net/
[7]. Duan, Zhi Ying, and Yi Zhen Cao, The Implementation of Cloud Storage System Based on OpenStack Swift, Applied Mechanics and Materials. Vol. 644. 2014, pp. 2981-2984.

[8]. Beernaert, L., Matos, M., Vilaça, R., & Oliveira, R., Automatic elasticity in Openstack, In Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, ACM, p. 2.
[9]. Zhou, Qing, and Jun S. Liu., Modeling within-motif dependence for transcription factor binding site predictions., Bioinformatics 20.6, 2004, pp. 909-916.
[10]. Bintu, Lacramioara, et al., Transcriptional regulation by the numbers: applications. Current opinion in genetics & development 15.2, 2005: pp. 125-135.
[11]. Statistical mechanics of transctiption factors binding sites — http://www.bio-physics.at/wiki/index.php?title=Statistical_Mechanics_of_Binding
[12]. FASTA format description — http://genetics.bwh.harvard.edu/pph/FASTA.html
[13]. Zhang, Xiujun, Position Weight Matrices., Encyclopedia of Systems Biology. Springer New York, 2013, 1721-1722.