Принципы реализации вычислительных веб-сервисов для работы с большими данными*

О.В. Сухорослов[1]

Институт проблем передачи информации им. А.А. Харкевича Российской академии наук[1]

В работе рассматриваются вопросы создания предметно-ориентированных веб-сервисов, реализующих обработку больших объемов данных на высокопроизводительных вычислительных ресурсах. Создание подобного рода сервисов сопряжено с рядом проблем, таких как интеграция с внешними хранилищами данных, обеспечение эффективной передачи данных на ресурс, организация хранения данных пользователей сервиса на ресурсе, управление обработкой данных и реализация удаленного доступа к данным пользователей. Предлагается подход к реализации сервисов для работы с большими данными на базе платформы Everest, учитывающий характерные особенности данных сервисов и поддерживающий их быстрое развертывание на основе имеющейся вычислительной инфраструктуры. Описывается пример созданного на базе платформы сервиса, реализующего анализ данных секвенирования нового поколения на Hadoop-кластере.

Ключевые слова: большие данные, веб-сервисы, передача данных, управление данными, распределенная обработка данных, Hadoop, MapReduce

1. Введение

Лавинообразный рост данных, наблюдаемый в самых различных сферах от научных исследований до коммерции, требует использования высокопроизводительных ресурсов и эффективных средств для хранения и обработки больших массивов данных. В течение последнего десятилетия активно развиваются технологии распределенной обработки данных на кластерах, такие как Hadoop и Spark. В то же время, сложность используемой аппаратно-программной инфраструктуры затрудняет ее непосредственное использование неспециалистами и требует создания удобных инструментов для решения отдельных классов задач.

Одним из способов реализации подобных инструментов является создание предметно-ориентированных сервисов на основе модели облачных вычислений «приложение как услуга» (Software as a Service). Данная модель позволяет пользователям подобных сервисов быстро, без установки ПО, воспользоваться готовыми реализациями методов обработки данных для решения стоящих перед ними задач. При этом пользователю не требуется вникать в особенности организации хранения и обработки данных на высокопроизводительных ресурсах, стоящих за данными сервисами.

Сервисы для работы с большими данными (СРБД), в сравнении в обычными вычислительными сервисами с небольшим объемом данных, начали развиваться недавно, поэтому принципы и варианты реализации данных сервисов пока мало изучены. Существует несколько академических разработок, ориентированных на поддержку отдельных областей исследований, например сервис Globus Genomics [1] для анализа данных секвенирования нового поколения и портал PDACS [2] для хранения и анализа данных в области космологии. Первая система использует в качестве вычислительной инфраструктуры ресурсы облака Amazon, в то время как вторая задействует ресурсы вычислительного центра NERSC и научного облака Magellan. Коммерческие облачные решения, такие как, например, Amazon ML, Microsoft Azure ML, Databricks Cloud, представляют собой платформы общего назначения, включающие набор универсальных сервисов, а также собственную инфраструктуру

*Исследование выполнено за счет гранта Российского научного фонда (проект №16-11-10352).

для хранения и обработки данных.

Плохо изучены вопросы реализации СРБД на базе имеющейся инфраструктуры для работы с большими данными, например, кластера под управлением платформ Hadoop и Spark, которые все чаще используются для анализа научных данных [3–5]. Также мало внимания уделено интеграции СРБД с имеющимися репозиториями и хранилищами данных, в том числе облачными, а также с другими сервисами. Большой опыт в области интеграции распределенных ресурсов для хранения и обработки данных был накоплен в рамках грид-инфраструктур [6], однако данные среды сложны в использовании и не поддерживают использование новых моделей вычислений и технологий, таких как Hadoop. Наконец, отсутствуют программные инструментарии и платформы для реализации и развертывания СРБД, которые бы предоставляли готовые решения типовых проблем, возникающих при создании подобного рода сервисов.

Данная работа призвана восполнить указанные пробелы. В главе 2 описываются характерные особенности и требования к СРБД. В главе 3 рассматриваются принципы реализации СРБД на основе платформы Everest, изначально ориентированной на создание сервисов с небольшим объемом данных. Отличительной особенностью предлагаемого подхода является поддержка быстрой реализации СРБД на основе имеющихся вычислительных ресурсов и хранилищ данных. В главе 4 описывается пример созданного на основе данного подхода сервиса для картирования коротких прочтений на Hadoop-кластере.

2. Характерные особенности и требования к СРБД

Под *сервисом для работы с большими данными (СРБД)* будем понимать доступный по сети сервис для решения определенного класса задач с входными данными большого объема. СРБД должен предоставлять удаленные интерфейсы, как правило в виде пользовательского веб-интерфейса и интерфейса прикладного программирования (API). Интерфейс СРБД должен позволять пользователю задать входные наборы данных и параметры решаемой задачи в терминах предметной области.

СРБД должен использовать высокопроизводительные и масштабируемые (как правило, распределенные) реализации алгоритмов анализа данных, требующие соответствующей вычислительной инфраструктуры для обработки и хранения данных. В качестве такой инфраструктуры должны использоваться один или несколько вычислительных кластеров, например, под управлением платформы Hadoop или аналогичной технологии. СРБД должен транслировать запрос пользователя в запуски на кластере одного или нескольких вычислительных заданий, использующих готовые реализации (например, на основе модели MapReduce) для решения указанных задач.

Пользователь СРБД должен иметь возможность передать сервису произвольные входные данные. В случае, если данные изначально находятся на компьютере пользователя или внешнем по отношению к сервису ресурсе (например, репозитории данных), СРБД должен реализовывать передачу данных по сети на используемый кластер. При передаче больших объемов данных важно обеспечить максимальную скорость передачи и автоматическую обработку отказов. Поскольку работа с большими данными часто носит поисковый характер, требующий многократных запусков, СРБД должен поддерживать повторное использование загруженных на кластер данных. В целях оптимизации использования сети, СРБД также должен кэшировать на кластере часто используемые данные. Функции передачи данных также могут быть реализованы в виде отдельных вспомогательных сервисов.

Важно отметить, что СРБД может функционировать обособленно от используемых вычислительных ресурсов. Ресурсов может быть несколько, они могут располагаться в различных точках. Также возможен вариант, когда сервис использует ресурс, предоставленный пользователем (см. главу 3). В данных случаях из соображений эффективности необходимо избегать транзитной передачи входных данных от пользователя на ресурс через сервис и

передавать данные напрямую.

На практике анализ данных часто представляет собой многоэтапный процесс, требующий решения различных задач на разных стадиях анализа. В таких случаях результаты работы одного сервиса могут подаваться на вход другому сервису. В случае, если сервисы используют различные ресурсы, здесь также возникает проблема передачи данных между ресурсами. В общем случае СРБД должен позволять пользователю как загрузить выходные данные на свой компьютер или внешний ресурс, так и передать эти данные напрямую другому сервису. Кроме того, СРБД может предоставлять дополнительные функции для удаленного просмотра и визуализации данных. Данные функции также могут быть реализованы в виде отдельных вспомогательных сервисов.

СРБД должен поддерживать одновременную работу с ним множества пользователей. При этом требуется обеспечить защиту данных пользователей, распределение ресурсов между пользователями и изоляцию вычислительных процессов. В случае использования облачной инфраструктуры, СРБД должен управлять динамическим выделением и освобождением ресурсов в облаке в соответствии с текущей нагрузкой.

3. Принципы реализации СРБД на базе платформы Everest

Everest [7, 8] - облачная программная платформа, поддерживающая публикацию, выполнение и композицию вычислительных приложений в распределенной среде. В отличие от традиционных платформ распределенных вычислений, Everest реализует облачную модель Platform as a Service (PaaS) путем предоставления своей функциональности через удаленные веб- и программный интерфейсы. Один экземпляр платформы может применяться одновременно многими пользователями для создания, запуска и совместного использования приложений[1]. Каждое созданное в рамках платформы приложение автоматически публикуется как веб-форма и веб-сервис. Другой отличительной чертой Everest является поддержка подключения пользователями внешних вычислительных ресурсов и гибкого связывания ресурсов с приложениями. Платформа реализует интеграцию с серверами и вычислительными кластерами при помощи агента, который выполняется на стороне ресурса и играет роль посредника между платформой и ресурсом.

Преимуществом использования платформы Everest для создания СРБД является наличие готовых средств быстрого развертывания вычислительных сервисов и интеграции с вычислительными ресурсами, не требующих отдельной установки платформы. В то же время, поскольку платформа создавалась изначально для поддержки сервисов с небольшим объемом данных, эффективная реализация СРБД на базе Everest требует ряда доработок. В частности, необходимо реализовать поддержку прямой передачи данных из внешних хранилищ на ресурс и в обратном направлении, минуя платформу. Кроме того, требуется реализовать интеграцию агента с компонентами платформы платформы Hadoop или аналогичной технологии, используемой для хранения и обработки данных на кластере.

На рис. 1 изображена предлагаемая схема реализации СРБД на базе платформы Everest и существующего Hadoop-кластера. Рассмотрим сценарий использования сервиса, включающий отмеченные на рисунке шаги.

На шаге 1 пользователь сервиса предварительно загружает интересующие его данные в некоторое доступное по сети хранилище или выбирает данные, уже размещенные в хранилище. В качестве данного хранилища могут выступать облачные сервисы (Dropbox, Google Drive и т.д.), репозитории научных данных (Dataverse, FigShare, Zenodo и т.д.), специализированные базы (например, 1000 Genomes Project), грид-сервисы или отдельные файловые серверы (протоколы HTTP, FTP, GridFTP, rsync). Широкий спектр существующих хранилищ делает более важной задачу интеграции СРБД с ними, в сравнении с дублированием их функциональности в самом сервисе. Отметим, что в качестве хранилища данных также

[1]Публично доступный экземпляр платформы развернут по адресу http://everest.distcomp.org/

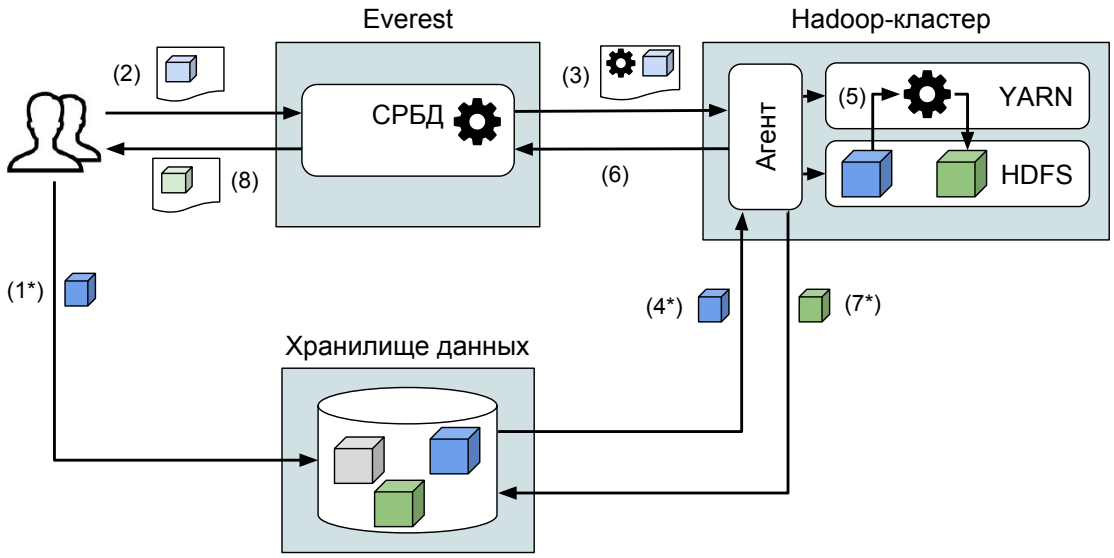

Рис. 1. Схема функционирования СРБД на базе платформы Everest

может выступать компьютер пользователя. В этом случае необходимо развертывание на компьютере ПО, предоставляющего доступ по сети к файлам пользователя. Опыт реализации подобного ПО для обеспечения передачи по сети научных данных уже имеется [9].

На шаге 2 пользователь формирует и отправляет запрос к СРБД, включающий ссылку на входные данные и значения требуемых сервисом параметров. Передаваемая ссылка должна позволять загрузить данные из внешнего хранилища уже без участия пользователя. В некоторых случаях для этого требуется предварительно передать сервису реквизиты доступа к хранилищу, например OAuth-токен или прокси-сертификат.

На шаге 3 сервис на основе запроса пользователя формирует вычислительную задачу и отправляет ее агенту, размещенному на используемом сервисом ресурсе. Вместе с описанием задачи сервис передает агенту ссылку на входные данные. Также, как показано на рисунке, при запуске задачи от сервиса к агенту может передаваться код программной реализации, используемой для обработки данных.

Наиболее широко используемые для распределенной обработки данных платформы Hadoop и Spark используют для реализации программ языки Java, Scala и Python. В отличие от языков C и Fortran, часто используемых в научных параллельных приложениях, программы на данных языках могут быть относительно легко перенесены с одного кластера на другой, включая их зависимости, без необходимости компиляции. Это открывает возможность для реализации на базе уже созданных программ и библиотек для Hadoop и Spark сервисов, которые могут быть использованы в связке с произвольным кластером, указанным пользователем. Данная модель позволяет значительно упростить публикацию и повторное использование наработок в данной области, не требуя при этом от владельца сервиса предоставления собственных ресурсов. Это также позволяет избежать многократной реализации сервисов, использующих одну программу с связке с различными ресурсами.

На шаге 4 агент производит загрузку входных данных из внешнего хранилища на локальный кластер. Для реализации данного шага планируется добавить в агент поддержку загрузки данных из основных репозиториев и типов хранилищ. В настоящий момент реализованы базовая поддержка загрузки данных по ссылкам протоколов HTTP и FTP, а также экспериментальные интеграции с сервисами Dropbox и репозиторием Dataverse. Загруженные на кластер данные размещаются в распределенной файловой системе Hadoop Distributed File System (HDFS), откуда их может прочитать запускаемая на следующем

шаге программа.

На шаге 5 агент производит запуск указанной в описании задачи программы на заданных входных данных. Запуск производится через менеджер ресурсов кластера, в роли которого обычно выступает Yet Another Resource Negotiator (YARN), компонент платформы Hadoop, поддерживающий запуск MapReduce- и Spark-программ. Для взаимодействия агента с YARN был реализован специальный адаптер, аналогичный по своим функциям ранее созданным адаптерам для интеграции агента с системами очередей HPC-кластеров. После запуска агент отслеживает состояние соответствующего задания (приложения в терминах YARN) и транслирует информацию о ходе выполнения запроса сервису (шаг 6), который в свою очередь отображает эту информацию пользователю через веб-интерфейс. По завершению работы программы агент передает сервису выходные файлы (небольшого объема) и окончательный статус задачи.

В случае, если в результате выполнения программы получены выходные данные большого объема, агент должен поддерживать прямую загрузку этих данных по сети в указанное пользователем хранилище (шаг 7). Необходимая для этого информация должна передаваться пользователем при отправке запроса к сервису на шаге 2. В настоящий момент реализована поддержка загрузки выходных данных на указанный FTP-сервер или в папку Dropbox.

На шаге 8, по окончанию обработки запроса, сервис передает пользователю результаты обработки в виде набора выходных параметров и ссылок на выходные файлы. Часть данных файлов могут храниться самим сервисом (например, лог выполнения программы), а часть - оставаться на кластере или располагаться во внешнем хранилище.

Отметим, что отмеченные звездочкой шаги 1, 4 и 7, связанные с передачей данных по сети, не всегда требуются или могут быть пропущены. Например, шаг 1 не требуется, если данные уже находятся во внешнем хранилище или на кластере, что верно для часто используемых наборов данных. Шаг 4 может быть пропущен, если данные уже были загружены ранее на кластер агентом или вручную администратором. Для этого агент должен хранить информацию о загруженных данных и кэшировать их для повторного использования. Шаг 7 не требуется, если полученные данные являются промежуточным результатом и будут поданы на вход другому сервису с использованием этого же кластера. Учет данных особенностей позволяет существенно уменьшить объем передаваемых по сети данных и, тем самым, повысить скорость обработки запросов.

Рассмотрим кратко вопросы обеспечения безопасности. Поскольку пользователи не могут модифицировать код программы, запускаемой сервисом на кластере, то риск несанкционированного доступа к данным других пользователей сведен к минимуму. При реализации кэширования данных на кластере агент также должен ограничивать повторное использование конфиденциальных данных только пользователем, который изначально передал эти данные. Что касается распределения ресурсов кластера между пользователями и изоляции вычислительных процессов, то данные функции уже реализованы в менеджере YARN.

В заключение отметим, что несмотря на то, что описанный в данном разделе подход подразумевает использование платформы Hadoop, он может быть легко адаптирован для интеграции с любой другой платформой хранения и обработки больших данных.

4. Пример реализации СРБД

Для отработки описанного подхода на базе Everest был реализован прототип сервиса для картирования коротких прочтений, одной из базовых биоинформатических задач анализа результатов секвенирования нового поколения (NGS). Данная задача обычно является начальной и самой тяжелой в вычислительном отношении стадией анализа NGS-данных, характеризующейся большими объемами входных и выходных данных.

На вход сервису требуется передать один или два (парных) файла с прочтениями в фор-

мате FASTQ. Размер данных файлов в сжатом виде обычно составляет несколько гигабайт. В качестве основного хранилища входных данных выбран публичный репозиторий проекта 1000 Genomes Project. Данный репозиторий предоставляет возможность загружать данные с FTP-сервера, где доступны как короткие прочтения, так и референсные геномы, необходимые для решения задачи картирования. Соответственно файлы передаются сервису в виде ссылок на данный или любой другой FTP-сервер.

Для решения задачи картирования был использован пакет BigBWA [10], реализующий параллельное выполнение другого известного пакета BWA (Burrows-Wheeler aligner) в парадигме MapReduce на Hadoop-кластере. При обращении к сервису пользователь может выбрать один из реализованных в пакете BWA алгоритмов картирования. Более тонкая настройка параметров в настоящее время не реализована. Также во всех запусках используется один референсный геном человека размером около 5 Гб, предварительно загруженный на кластер. Полный объем входных данных задачи на тестовых запусках составлял около 10-15 Гб.

При вызове сервиса, в соответствии с описанной в главе 3 схемой, происходит прямая загрузка файлов прочтений с FTP-сервера на Hadoop-кластер. После загрузки выполняется распаковка и конвертация файлов в формат, используемый BigBWA. Ранее загруженные на кластер прочтения кэшируются и, в случае совпадения адреса файла в запросе с уже загруженным, шаг загрузки данных пропускается. После загрузки данных происходит запуск на кластере MapReduce-задания с пакетом BigBWA.

По окончании выполнения задания сервис по умолчанию возвращает пользователю путь к файлу с результатами картирования на кластере. Данный подход был выбран поскольку, как отмечалось ранее, картирование является только начальным этапом анализа NGS-данных. Поэтому на практике данные результаты обычно будут сразу поданы на вход другому сервису. По требованию пользователя результаты картирования также могут быть загружены на внешний FTP-сервер. Объем выходных данных на тестовых запусках составлял около 5-10 Гб в формате SAM. В дальнейшем планируется реализовать конвертацию результатов в более компактный формат BAM.

Решение задачи картирования на Hadoop-кластере с использованием созданного сервиса позволило значительно уменьшить время получения результата. Так, например, запуск пакета BWA для картирования по двум чтениям на одиночном сервере в 4 потоках занял более часа, а аналогичный запуск через сервис (28 map-задач) - около 10 минут.

5. Заключение

В работе рассмотрены характерные особенности и требования к реализации предметно-ориентированных сервисов для работы с большими данными. Предложен подход к реализации данных сервисов на основе платформы Everest, изначально ориентированной на создание вычислительных сервисов с небольшим объемом данных. Отличительной особенностью данного подхода, в сравнении с коммерческими облачными решениями, является поддержка быстрой реализации сервисов на основе уже имеющихся вычислительных ресурсов и хранилищ данных. Описан пример созданного сервиса, реализующего анализ данных секвенирования нового поколения на Hadoop-кластере. В настоящее время продолжаются реализация и тестирование отдельных элементов описанного подхода.

Литература

1. Madduri R. et al. Experiences Building Globus Genomics: A Next-generation Sequencing Analysis Service Using Galaxy, Globus, and Amazon Web Services // Concurrency and Computation: Practice and Experience. 2014. Vol. 26, No. 13. P. 2266–2279.

2. Madduri R. et al. PDACS: A Portal for Data Analysis Services for Cosmological

Simulations // Computing in Science & Engineering. 2015. Vol. 17, No 5. P. 18–26.

3. Ekanayake J., Pallickara S., Fox G. MapReduce for Data Intensive Scientific Analyses // 2008 IEEE International Conference on eScience (eScience'08). IEEE, 2008. P. 277–284.
4. Zhang Z. et al. Scientific Computing Meets Big Data Technology: An Astronomy Use Case // 2015 IEEE International Conference on Big Data. IEEE, 2015. P. 918–927.
5. Nothaft F. A. et al. Rethinking Data-intensive Science Using Scalable Analytics Systems // 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015. P. 631–646.
6. Foster I., Kesselman C. (ed.). The Grid 2: Blueprint for a New Computing Infrastructure. Elsevier, 2003.
7. Сухорослов О.В. Интеграция вычислительных приложений и распределенных ресурсов на базе облачной программной платформы // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. 5, № 4(22). С. 171–182.
8. Sukhoroslov O., Volkov S., Afanasiev A. A Web-based Platform for Publication and Distributed Execution of Computing Applications // 14th International Symposium on Parallel and Distributed Computing (ISPDC). 2015. P. 175–184.
9. Chard K., Tuecke S., Foster I. Efficient and Secure Transfer, Synchronization, and Sharing of Big Data // Cloud Computing. IEEE, 2014. Vol. 1, No. 3. P. 46–55.
10. Abuín J. M. et al. BigBWA: Approaching the Burrows–Wheeler Aligner to Big Data Technologies // Bioinformatics. 2015. doi:10.1093/bioinformatics/btv506

On implementation of computational web services for big data processing*

O.V. Sukhoroslov[1]

Institute for Information Transmission Problems
of the Russian Academy of Sciences (Kharkevich Institute)[1]

The paper considers development of domain-specific web services for processing of large volumes of data on high-performance computing resources. The development of these services is associated with a number of challenges, such as integration with external data repositories, implementation of efficient data transfer, management of user data stored on the resource, execution of data processing jobs and provision of remote access to the data. An approach for building big data processing services on the base of Everest platform is presented. The proposed approach takes into account the characteristic features and supports rapid deployment of these services on the base of existing computing infrastructure. An example of Everest-based service for short-read sequence alignment that processes NGS data on a Hadoop cluster is described.

Keywords: big data, web services, data transfer, data management, distributed data processing, Hadoop, MapReduce

References

1. Madduri R. et al. Experiences Building Globus Genomics: A Next-generation Sequencing Analysis Service Using Galaxy, Globus, and Amazon Web Services // Concurrency and Computation: Practice and Experience. 2014. Vol. 26, No. 13. P. 2266–2279.

2. Madduri R. et al. PDACS: A Portal for Data Analysis Services for Cosmological Simulations // Computing in Science & Engineering. 2015. Vol. 17, No 5. P. 18–26.

3. Ekanayake J., Pallickara S., Fox G. MapReduce for Data Intensive Scientific Analyses // 2008 IEEE International Conference on eScience (eScience'08). IEEE, 2008. P. 277–284.

4. Zhang Z. et al. Scientific Computing Meets Big Data Technology: An Astronomy Use Case // 2015 IEEE International Conference on Big Data. IEEE, 2015. P. 918–927.

5. Nothaft F. A. et al. Rethinking Data-intensive Science Using Scalable Analytics Systems // 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015. P. 631–646.

6. Foster I., Kesselman C. (ed.). The Grid 2: Blueprint for a New Computing Infrastructure. Elsevier, 2003.

7. O. Sukhoroslov. Integration of Distributed Computing Applications and Resources on the Base of Cloud Platform // Program Systems: Theory and Applications. 2014. Vol. 5, No. 4(22). P. 171–182. (In Russian)

8. Sukhoroslov O., Volkov S., Afanasiev A. A Web-based Platform for Publication and Distributed Execution of Computing Applications // 14th International Symposium on Parallel and Distributed Computing (ISPDC). 2015. P. 175–184.

9. Chard K., Tuecke S., Foster I. Efficient and Secure Transfer, Synchronization, and Sharing of Big Data // Cloud Computing. IEEE, 2014. Vol. 1, No. 3. P. 46–55.

*This work is supported by the Russian Science Foundation (project No. 16-11-10352).

10. Abuín J. M. et al. BigBWA: Approaching the Burrows–Wheeler Aligner to Big Data Technologies // Bioinformatics. 2015. doi:10.1093/bioinformatics/btv506