УДК 004:681.3

Анализ проблем безопасности архитектуры распределённых NoSQL приложений на примере программного каркаса Hadoop

Чуприн В.И., Чернышова А.В., Губенко Н.Е.
Донецкий национальный технический университет
alla@pmi.dgtu.donetsk.ua, gubenko@cs.dgtu.edu.ua, chuprin.vladislav@gmail.com

Чуприн В.И., Чернышова А.В., Губенко Н.Е. "Анализ проблем безопасности архитектуры распределённых NoSQL приложений на примере программного каркаса* Hadoop*". *В статье выделены основные характеристики хранилищ для обработки больших массивов данных. Проанализированы особенности архитектуры распределенных приложений на примере программного каркаса* Hadoop*. Изучены существующие компоненты приведенного каркаса. Предложена альтернатива стеку технологий, поставляемому по умолчанию. Выделены основные проблемы архитектурной и операционной составляющих. Предложены рекомендации по оптимизации подсистемы безопасности на основе приведенных проблем.*

Ключевые слова: Kerberos, SSL, TLS, Hadoop, NoSQL, Большие данные, HDFS, MapReduce

Введение

В статье рассмотрены основные принципы безопасности систем в области обработки «Больших данных», проанализированы встроенные компоненты и слабые стороны этих систем.

Цель работы - выделить проблемы сегмента приложений, который использует решения для обработки больших объемов информации с точки зрения безопасности, а также предложить варианты решения приведенных проблем.

Рассматриваемая в статье совокупность вопросов является актуальной по двум причинам. Во-первых, проекты с «Большими данными» часто встречаются в корпоративной среде. Во-вторых, большинство из систем не используют ничего, кроме паролей пользователей.

1 Характеристики рассматриваемых хранилищ

Существует множество систем, которые позволяют хранить очень большие объемы данных, а также предоставляют возможности по формированию запросов и поддержанию целостности. Рассмотрим основные характеристики репозиториев данных:

- позволяет содержать большое количество (более петабайта) данных;
- поддерживает распределенную избыточность хранения данных;
- содержит механизмы параллельной обработки задач;
- предоставляет механизм обработки данных (MapReduce или ему эквивалентный);
- поддерживает быстрое внесение новых данных;
- имеет централизованное управление машинами в кластере;
- не требует особенного аппаратного обеспечения;
- является гибким в настройке (основные возможности могут быть дополнены или изменены).

2 Структура программного каркаса

Hadoop — проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и программный каркас для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов. Разработан в рамках вычислительной парадигмы MapReduce, согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполняемых на узлах кластера и естественным образом сводимых в конечный результат.

Hadoop состоит из трёх подпроектов:

a) Hadoop Common — набор инфраструктурных программных библиотек и утилит, используемых для других подпроектов и родственных проектов;

b) HDFS — распределённая файловая система;

c) Hadoop MapReduce — каркас для реализации MapReduce-вычислений;

Стандартная установка Hadoop состоит из набора демонов или резидентных программ на различных сетевых серверах. Некоторые из демонов могут запускаться на одном сервере, другие – на нескольких. Существуют следующие демоны:

– NameNode. В Hadoop применяется архитектура главный/подчиненный, как для распределенного хранения (HDFS), так и для распределенных вычислений (MapReduce). NameNode представляет собой главный демон HDFS, который распределяет низкоуровневые задачи ввода/вывода между подчиненными демонами DataNode. Он ведет учет разбиения файлов на блоки, хранит информацию о том, на каких узлах эти блоки находятся, и следит за общим состоянием распределенной файловой системы. Name Node не дублируется как DataNode или TaskTracker.

– DataNode. Выполняет работу распределенной файловой системы – считывание и запись блоков HDFS в физические файлы, находящиеся в локальной файловой системе.

– Secondary NameNode (SNN) – вспомогательный демон, который занимается мониторингом состояния кластера HDFS. Выше отмечалось, что NameNode не дублируется и таким образом является точкой общего отказа Hadoop. Снимки создаваемые SNN, позволяют уменьшить время простоя и свести к минимуму потерю данных.

– JobTracker – посредник между Hadoop и клиентским приложением. В момент передачи кода кластеру JobTracker строит план выполнения, то есть определяет какие файлы обрабатывать, назначает узлы различным задачам и следит за ходом выполнения этих задач. В кластере Hadoop может быть только один демон JobTracker.

– TaskTracker. Демоны вычислений, как и демоны хранения, также построены на базе архитектуры главный/подчиненный. JobTracker – это главный демон, организующий выполнение задач MapReduce, а демоны TaskTracker управляют исполнением отдельных заданий на подчиненных узлах.

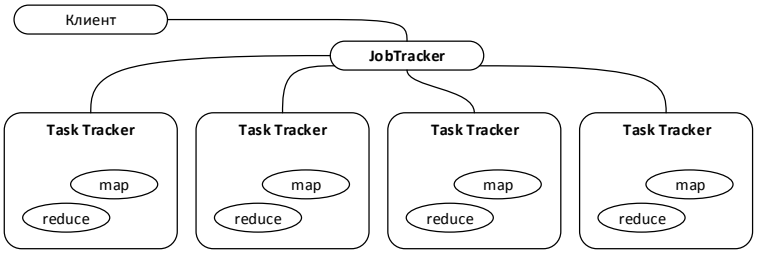

Рисунок 1 – Обработка MapReduce задачи

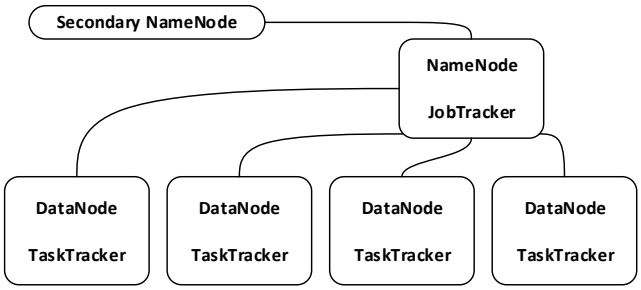

Рисунок 2 – Топология типичного кластера Hadoop

3 Проблемы инфраструктуры распределённого приложения

Для приложений обработки больших массивов данных характерны следующие черты:

a) Множество распределенных узлов: "Перемещение вычислений дешевле, чем перемещение данных" является ключевой парадигмой обработки «Больших данных». Данные обрабатываются везде, где доступны ресурсы, что позволяет распараллеливать вычисления. Однако, такой подход создает сложную среду с большим количеством мест для возможных атак и усложняет проверку безопасности внутри распределенного кластера с широким набором разнородных платформ.

b) Шардируемые данные: Данные в больших кластерах избыточны, с несколькими копиями, перемещаемыми между различными узлами для увеличения производительности и обеспечения отказоустойчивости. «Шард» - срез горизонтально сегментированных данных - общий для нескольких серверов. Это автоматизированное перемещение данных по множеству машин делает очень трудным процесс получения информации, где расположены данные и сколько копий имеется в заданный момент времени. Это идет в разрез с традиционной централизованной моделью защиты данных, в которой одна копия данных ограждена различными системами безопасности, пока она не используется для обработки. «Большие данные» реплицируются на множество узлов и перемещаются по мере необходимости. «Контейнерная» модель безопасности не применима.

c) Доступ к данным / аутентификация и авторизация: Доступ на основе ролей занимает центральное место в большинстве схем безопасности баз данных. Реляционные и псевдореляционные платформы включают в себя роли, группы, схемы, метки безопасности, а также различные другие объекты для ограничения доступа пользователей к подмножествам доступных данных. Большинство окружений «Больших данных» предоставляют ограничения доступа на уровне схемы, однако отсутствует тонкое дробление. Можно логически имитировать метки безопасности и другие расширенные возможности в NoSQL окружениях, но это требует дополнительного проектирования пользовательского приложения для создания этих функций.

d) Взаимодействие между узлами: Hadoop и подавляющее большинство решений (Cassandra, MongoDB, Couchbase, и т.д.) не обеспечивают должного уровня безопасности при взаимодействии - они используют RPC по TCP / IP. TLS и SSL редко встраиваются в сам дистрибутив. Однако, когда они все же поставляются (как это происходит с прокси HDFS), они охватывают лишь взаимодействие клиент-прокси, но не связь прокси-узел.

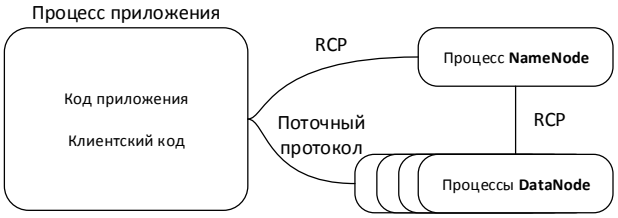

Рисунок 3 – Взаимодействие между узлами

e) Отсутствие встроенных средств обеспечения безопасности: У стеков обработки «Больших данных» почти нет встроенных средств обеспечения безопасности. Помимо авторизации на уровне служб и поддержки веб-прокси для узлов JobTracker, отсутствуют какие-либо средства, предназначенные для защиты хранилищ данных, приложений или основных возможностей Hadoop. Все NoSQL дистрибутивы построены в соответствии с моделью веб-сервисов, с малым количеством или полным отсутствием средств борьбы с десятком основных угроз выделенных Open Web Application Security Project, поэтому большинство API доступа к хранилищам уязвимы к хорошо известным атакам.

Приведенные выше проблемы не являются специфическими для «Больших данных», но NoSQL решения подвержены им по своей природе из-за распределенной архитектуры, использующей простую модель программирования на основе каркаса открытых сервисов. Чтобы добавить средства обеспечения безопасности в среду «Больших данных», они должны также легко масштабироваться, как и сам распределенный каркас. Большинство сторонних средств не масштабируются таким образом. Поскольку управление безопасностью не встроено в поставляемые дистрибутивы, сторонние решения создают несоответствие между Hadoop/NoSQL окружением и сопутствующими средствами безопасности. Поставщики безопасности приспосабливают свои существующие решения, предусматривающие применение в контрольной точке, где данные и команды попадают в кластер, а не перемещаются внутри кластера. Лишь немногие традиционные продукты безопасности могут полностью интегрироваться и динамически масштабироваться с кластерами Hadoop.

4 Операционная безопасность

Ниже приведен обзор наиболее распространенных угроз для систем управления данными. Хранение данных в NoSQL кластерах обычно регулируется законом, и должно быть защищено соответствующим образом. Нападения на корпоративные системы и кражи данных широко распространены и каркасы управления данными должны предотвращать их. «Большие данные» предполагают все те же слабые места, характерные для традиционных информационных систем. Проблема заключается в выборе вариантов, которые работают в распределенной среде. Администраторы кластеров должны учитывать управление безопасностью для следующих областей:

a) Защита данных в режиме покоя: Стандартным способом защиты, в данном случае, является шифрование. Это позволяет защититься от попытки доступа к данным за пределами установленных интерфейсов приложений. При использовании традиционных систем управления данными существует возможность кражи архива или непосредственно чтения файлов с диска. Зашифрованные файлы будут защищены от доступа пользователей, не обладающих ключами шифрования. Большинство NoSQL решений не обеспечивают шифрование данных в покое.

b) Административный доступ к данным: Каждый узел имеет по крайней мере одну административную учетную запись с полным доступом к его данным. В данном случае следует предусмотреть границы или объекты, для которых будет реализовано

разделение обязанностей между различными администраторами. Нежелательный прямой доступ к файлам данных или процессам узла может быть предотвращен путем сочетания управления доступом, разделения ролей и криптографии. Этот аспект зависит выбора от проектировщиком системы управления нужных утилит.

5 Технические рекомендации

Приведенные выше проблемы являются ключевыми при построении безопасной архитектуры распределенного приложения. Далее приведены возможные варианты их решения:

a) Применение Kerberos встроенного в Hadoop, для проверки узлов и клиентских приложений до получения доступа к кластеру и для валидации приложений, формирующих запросы к MapReduce (MR) и аналогичным функциям.

b) Использование слоя шифрования на уровне операционной / файловой систем - для защиты данных в состоянии бездействия.

c) Настройка управления доступом на уровне ключей / сертификатов доступа. Следует использовать центральный сервер управления ключами для защиты ключей шифрования и различными ключами для различных файлов.

d) Проверка узлов во время развертывания – для предоставления доступа новым узлам кластера следует использовать менеджеры управления гипервизорами провайдеров облачных услуг или сторонние продукты, такие как «Chef» или «Puppet».

e) Журналирование операций, аномальных ситуаций и действий с административными привилегиями.

f) Использование протоколов SSL или TLS сетевой безопасности.

Выводы

Результатом анализа можно считать следующие рекомендации:

- применение Kerberos, встроенного в Hadoop;
- использование слоя шифрования на уровне операционной / файловой систем;
- настройка управления доступом на уровне ключей / сертификатов доступа;
- проверка узлов во время развертывания;
- журналирование операций, аномальных ситуаций, и действий с административными привилегиями;
- использование протоколов SSL или TLS сетевой безопасности;

Приведенные методы позволяют защититься от базовых атак с минимальными изменениями архитектуры приложений и бизнес процессов.

Литература

[1] Hadoop. Электронный ресурс. Режим доступа: http://ru.wikipedia.org/wiki/Hadoop

[2] Securing Big Data: Security Recommendations for Hadoop and NoSQL Environments. Электронный ресурс. Режим доступа: https://securosis.com/assets/library/reports/SecuringBigData_FINAL.pdf

[3] Чак Л., Hadoop в действии. М.: ДМК Пресс, 2012. – 424 с.

[4] Tom White: The Definitive Guide. M.: O'Reilly Media / Yahoo Press, 2012. – 628 с.

[5] Alex Holmes, Hadoop in Practice. M.: Manning Publications, 2012. – 536 с.