Наконец отдельным классом стоят NoSQL – к ним относят графовые базы данных. Они располагают функционалом для естественного предоставление информации. Прямая альтернатива SQL – документарные базы данных, их главная особенность –это отсутствие схемы, присущие реляционным базам данных.

Преимущества графовых баз данных:

- База может сохранять любой объект, например объект с большим коллиствам полей.
- Гибкая система расширения и модернизация в любой момент времени.

Недостатки графовых баз данных:

- Проблемы с работой в параллельных архитектурах.
- Низкая производительность при большом колличестве связей и больших объемах.

Графовые базы данных оказываются универсальном вариантом, которые помогают подстраховаться на случай изменения требований и увеличением либо уменьшением функционала. Добавление новых связей делает непригодной для использования документарную базу данных, а рост количества сырых объектов с несвзяной структурой снижает производительность реляционной БД. Сегодня ведется активная разработка и доработка RDF – основного стандарта, согласно которому работают графовые базы данных. Именно стандартизация SQL делала популярным реляционные БД. При этом ряд проектов демонстрируют поддержку OData, для создания классических веб-запросов через HTTP. Язык SPARKQL, обладающий возможностями для работы с вариативными видами запросов и данных. За счет развития архитектуры производительность графовых баз данных тоже растет, возможно в будущем за счет развития окажется выше реляционной даже при небольшом количестве связей.

Список использованных источников:

1. Ян Робинсон, Джим Вебер. Грфовые базы данных. Новые возможности для работы со связными данными. 2016. – 256с.

КЛАСТЕРИЗАЦИЯ В HADOOP

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Владыко В.Д., Чурин А.П.

Теслюк В.Н. – к.ф.-м.н., доцент

Непрерывный рост данных и увеличение скорости их генерации порождают проблему их обработки и хранения. Неудивительно, чтотема «больших данных» (Big Data) является одной из самых обсуждаемых. Одним из самых известных проектов в области распределенных вычислений является Hadoop — набор из утилит, библиотек и фреймворк для разработки и выполнения программ распределенных вычислений. Кластеризация один из важных моментов работы с большими данными.

Hadoop – это проект с открытым исходным кодом, находящийся под управлением Apache Software Foundation. Hadoop используется для надежных, масштабируемых и распределенных вычислений, но может также применяться и как хранилище файлов общего назначения, способное вместить петабайты данных.

В состав проекта Hadoop входят следующие подпроекты:

Common — набор компонентов и интерфейсов для распределенных файловых систем и общего ввода-вывода;

MapReduce — модель распределённых вычислений, предназначенная для параллельных вычислений над очень большими (до нескольких петабайт) объемами данных;

HDFS — распределенная файловая система, работающая на больших кластерах типовых машин.

Классическая конфигурация кластера Hadoop состоит из одного сервера имён (NameNode), одного мастера MapReduce (т.н. JobTracker) и набора рабочих машин, на каждой из которых одновременно крутится сервер данных (DataNode) и обработчик (TaskTracker).

Namenode — сервер имён. Как правило, один узел на кластер. Хранит в себе все метаданные системы, сами файлы на этом узле не хранятся.

JobTracker — узел, который координирует параллельную обработку данных используя MapReduce.

TaskTracker — узел, который принимает задачи по обработке данных от JobTracker.

DataNode — таких узлов в кластере очень много. Они хранят непосредственно блоки файлов. Узел регулярно отправляет NameNode свой статус и ежечасно — информацию обо всех хранимых на этом узле блоках. Это необходимо для поддержания нужного уровня репликации.

При правильной архитектуре приложения, с помощью информации о том, на каких машинах (узлах) расположены блоки данных, позволяет запустить на них же вычислительные процессы и выполнить большую часть вычислений локально, т.е. не передавая данные по сети.

Кластеры в Hadoop позволяют ускорить анализ данных для приложений и улучшить их масштабируемость. Если при растущих объемах данных, кластер начинает не справляться, то можно добавить дополнительные узлы для увеличения пропускной способности. Также кластер обладает высокой отказоустойчивостью, поскольку каждый блок данных копируется на другие узлы, гарантируя, что данные не будут потеряны, даже если один из узлов выдаст ошибку.

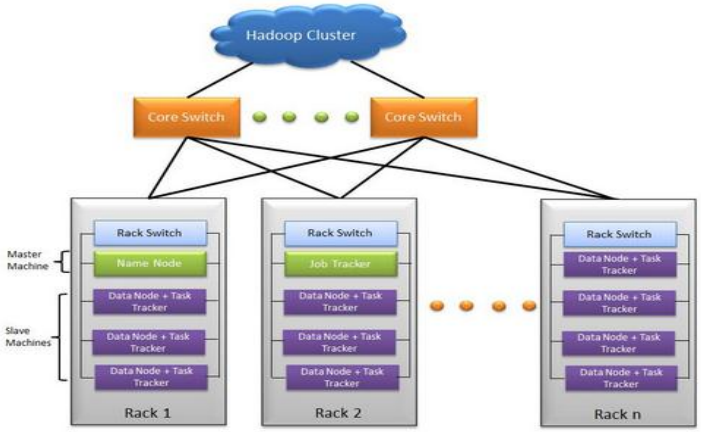

Рис. 1 - Структура кластера Hadoop

Список использованных источников:
1. KevinT. Smith, BorisLublinsky. Professional Hadoop Solutions. 2013.
2. Sammer E. Hadoop Operations. 2012 - 41c.
3. Tom White. Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale. 2015 - 283c.

МЕТОДЫ УЛУЧШЕНИЯ АЛГОРИТМА Q-ОБУЧЕНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Волчек А.Ю., Соболь А.В.

Волорова Н.А. – к.т.н., доцент

В настоящее время применение машинного обучения помогает людям решать задачи в разных сферах жизни: медицина, физика, химия, системы безопасности, игровая индустрия и др. Уже сейчас есть прототипы самодвижущихся автомобилей, которые сами контролируют движение транспорта. Одним из главных методов для обучения такого рода искусственного интеллекта является метод обучения с подкреплением – самостоятельное и уже вполне сформировавшееся направление кибернетических исследований, одна из разновидностей которого – Q-обучение.

Q-обучение — метод, применяемый в искусственном интеллекте при агентном подходе. Относится к экспериментам вида обучение с подкреплением. На основе получаемого от среды вознаграждения агент формирует функцию полезности Q, что впоследствии дает ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Одно из преимуществ Q-обучения — то, что оно в состоянии сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде марковского процесса принятия решений.

Q-обучение является важной вехой, однако известные некоторые ограничения этого алгоритма, для которых предложены несколько улучшений. В работе предложено несколько улучшений, в каждом из которых поясняется причина ограничений. Мы постарались выделить разноплановые улучшения, относящиеся к различным этапам алгоритма.

Двойное Q-обучение. Параметры нейросети в глубоком Q-обучении оптимизируются стохастическим градиентным спуском, минимизируя функцию потерь:

$$(R_{t+1} + \gamma_{t+1} \max_{a'} q_{\bar{\theta}}(S_{t+1}, a') - q_{\theta}(S_t, A_t))^2 \quad (1)$$

В работе (vanHasselt 2010) показано, что из-за применения операции максимизации в уравнении (1) оценка Q-функции почти всегда является смещённой. Предложенный в данной работе подход позволяет частично избавиться от этой проблемы. Предлагается поддерживать рядом ещё одну нейросеть (targetnetwork), параметры в которую копируются из основной раз в несколько итераций. Вместо максимизации в уравнении (1) мы сначала выбираем действие с максимальным значением Q-функции, основываясь на данных из основной сети, но дальше используем значения Q-функции для этого действия из дополнительной сети.

Приоретизированный буфер опыта. Буфер опыта (experiencereplay) значительно ускоряет обучение и улучшает его стабильность, позволяя обучаться на ранее виденных ситуациях. В обычном его варианте мы выбираем обучающее множество равновероятно среди всех его элементов. Очевидно, что на практике существует много “простых” ситуаций, для которых сеть выучила хорошую аппроксимацию Q-функции и некоторое количество “сложных”. В приоритезированном буфере опыта мы выбираем каждый элемент в