точности в некоторые случаях позволяет исключить дорогостоящую операцию join на таблицах. Например, анкета пользователя, включающая сотню полей, но при этом часть из них всегда отображается на клиентских приложениях. В такой ситуации имеет смысл ввести избыточность. А если паттерны работы с данными кардинально различны, то есть смысл задуматься о введении для этих случаев своих хранилищ с разными схемами данных, оптимизированными под конкретный вариант использования. Но это все достаточно специфично для ситуации.

Рассмотренные походы для построения высоконагруженных систем с учетом требований к надежности, могут применяться как по отдельности, так и вместе. Все зависит от принятых архитектурных решений и бизнес-логики системы, а также понимания того, что может делать пользователь в системе и правил обработки данных. При принятии архитектурных решений, нужно определять планируемый порядок обработки объема данных, их скорость прироста, соотношение чтения/записи, решить какова допустимая деградация системы, чем можно пренебречь. Понимание всех этих проблем позволяет выполнить проектирование системы с учетом полученной информации, а также с применением вышеперечисленных принципов масштабирования данных.

Список использованных источников:

1. Decomposing Twitter: Adventures in Service-Oriented Architecture [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.infoq.com/presentations/twitter-soa.
2. Cache coherence [Электронный ресурс]. – Электронные данные. – Режим доступа: https://en.wikipedia.org/wiki/Cache_coherence.
3. Round-robin DNS [Электронный ресурс]. – Электронные данные. – Режим доступа: https://en.wikipedia.org/wiki/Round-robin_DNS.
4. Rüdiger Schollmeier, A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002).
5. Bittorrent Protocol Specification v1.0 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://wiki.theory.org/BitTorrentSpecification.
6. Gossip protocol [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Gossip_protocol.

СПОСОБЫ АВТОМАТИЗИРОВАННОГО АНАЛИЗА РАСХОЖДЕНИЙ ЗНАЧЕНИЙ ПАРАМЕТРОВ НА БОЛЬШИХ СОВОКУПНОСТЯХ ДАННЫХ

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

Мыц С. И.

Волорова Н. А. – канд. техн. наук, доцент

В условиях наличия больших объемов данных возникает необходимость в автоматизированном анализе их параметров. Ни одному человеку или группе людей не будет под силу вручную обработать весь объем информации поступающей от достаточно крупной современной информационной системы. Для решения такой задачи применяются инструменты, созданные с использованием алгоритмов и математических методов.

Взглянем на некоторый частный случай и на его примере рассмотрим возможные подходы к решению проблемы. Допустим, у нас есть некоторый веб-сайт, к которому пользователи задают запросы, а затем как-то взаимодействуют с результатами этих запросов. Чтобы исследовать это взаимодействие в глобальном плане, вводятся параметры, считаемые по некоторому множеству запросов. Эти параметры назовём метриками.

В контексте текущей задачи метрикой будем называть некоторую числовую величину, которая считается на основе данных из множества запросов и является показателем некоторого интересного нам свойства этого множества. В качестве примеров простых метрик можно привести следующие:

- общее количество действий пользователя со всеми результатами запроса
- отношение количества действий пользователя с опрделённым результатом к количеству показов этого результата
- среднее время до первого взаимодействия пользователя с результатами в рамках запроса
- количество пользователей, взаимодействующих с системой

Могут возникать ситуации, когда значения метрик изменяются и нам нужно понять, по каким причинам это произошло и что больше всего повлияло на это изменение. Это можно делать с помощью анализа значений метрик на срезах данных. Срезом данных назовём подмножество исходного множества запросов, взятое по какому-то критерию.

Теперь можно обобщить постановку задачи. Пусть у нас имеется множество объектов (см. множество запросов), имеется числовая функция от произвольного подмножества объектов (см. метрика) и есть

набор предикатов, позволяющих выделять подмножества исходного множества (см. срезы данных).

Поставленную выше задачу можно решать различными способами ввиду того, что нет чёткого определения “наибольшего влияния на изменения”, но мы пока остановимся на трёх подходах:

- Использование анализа чувствительности
- Деревья решений и коэффициент влияния факторов
- Сравнение среза с другой подвыборкой

Воспользоваться анализом чувствительности можно следующим образом:

3. Возьмём в качестве аргументов некоторой функции все имеющиеся срезы (0 или 1, принадлежит запрос этому срезу или нет), а затем с помощью регрессии построим такую искусственную функцию, которая будет приближать значение метрики на совокупности срезов.
4. Подсчитаем коэффициенты Соболя (Sobol Indices) для этой функции и на основе их в качестве результата вернём список, отсортированный по их значениям.

Деревья решений являются одним из подходов в машинном обучении, позволяющим решать, среди прочих, задачу регрессии. Полезным побочным эффектом их применения является возможность подсчёта “полезности” отдельного аргумента в деле предсказания значения целевой функции. Можно воспользоваться этими коэффициентами для выделения менее и более “важных” срезов, по аналогии с предыдущим пунктом.

Ещё одним способом решения задачи является следующий:

5. Рассмотрим каждый из срезов данных.
6. Выделим ему срез для сравнения: случайный срез аналогичного размера, всё непопавшее в срез и т.п.
7. Сравним с использованием статистических критериев значения на исходном и парном срезе.
8. Используя результаты сравнения отранжируем срезы.

На основе перечисленных выше методов строится обобщённый инструмент, предоставляющий возможности конфигурирования, предподсчёта данных, запроса списка наиболее повлиявших срезов и визуализации результатов.

Подход, описанный в этом докладе, позволяет автоматизировать анализ и помогает в исследовании изменений большой системы. Благодаря этому можно оперативнее реагировать на проблемы, а также лучше и точнее анализировать результаты вносимых в систему правок.

Список использованных источников:

1. C.D. Manning, P. Raghavan, H. Schütze. Introduction to Information Retrieval, Cambridge UP, 2008
2. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola Global Sensitivity Analysis: The Primer, John Wiley & Sons, 2008

МОДЕЛЬ АКТОРОВ. ПРИМЕНЕНИЕ ДЛЯ СОЗДАНИЯ МНОГОПОТОЧНЫХ ПРИЛОЖЕНИЙ

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

Селюк М. И., Владыко В. Д.

Сиротко С. И. – канд. физ.-мат. наук, доцент

В настоящее время наращивание вычислительных мощностей и производительности систем достигается в основном благодаря параллельным вычислениям. Это делает особенно актуальным эффективную реализацию параллелизма в программном обеспечении. Современная модель параллелизма, основанная на низкоуровневых потоках, имеет ряд серьёзных проблем. Новая концепция - модель акторов - успешно решает эти проблемы.

Цель данной работы – освоение технологии акторов, её применение на примере создания простейшего многопоточного приложения на языке Scala с использованием библиотеки Akka, а также анализ особенностей и перспектив модели на основе полученного опыта.

Недостатки традиционной модели вычислений, основанной на понятии машины Тьюринга, становятся очевидны при разработке сложных вычислительных систем с использованием параллелизма. Понятие глобального времени, когда в каждый момент выполняется одна неделимая операция, приводит к необходимости использовать низкоуровневые примитивы синхронизации потоков, для разрешения коллизий, связанных с разделяемым состоянием. Высокая сложность создания таких программ, вызванная разрывом между человеческим и машинным представлением, влечёт за собой совершение большого количества ошибок, возникновение гонки за данными, взаимных блокировок, а также плохой масшабируемости. Таким