Системы хранения и обработки
потоков больших данных

Методы оптимизации

Самарев Роман Станиславович, 2022 (черновые наброски)

МГТУ им. Н.Э. Баумана
Кафедра Компьютерные системы и сети

TABLE OF CONTENTS

Оптимизация плана выполнения

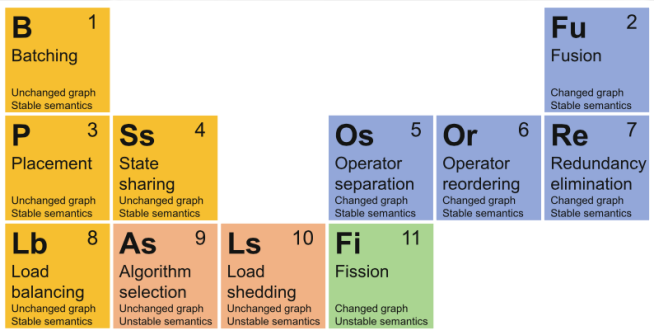
M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm. A catalog of stream processing optimizations.

***ACM Comput. Surv.*, 46(4):46:1–46:34, Mar. 2014** M. Hirzel, R. Soulé, B. Gedik, and S. Schneider.

Stream query optimization.

In S. Sakr and A. Y. Zomaya, editors, *Encyclopedia of Big Data Technologies*. Springer, 2019

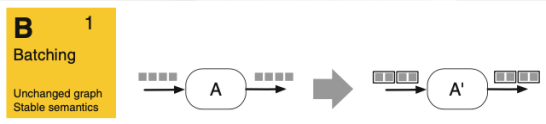
Задача - для заданного логического плана выполнения выполнить его оптимизацию так, чтобы эффективно использовать доступные ресурсы кластера. Данный вид оптимизации, в первую очередь, основывается на возможностях языка программирования, используемого для формирования логического плана. Однако некоторые виды оптимизации могут быть проведены на более низком уровне.



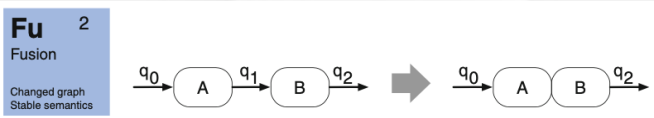
M. Hirzel, R. Soulé, B. Gedik, and S. Schneider. Stream query optimization.

In S. Sakr and A. Y. Zomaya, editors, *Encyclopedia of Big Data Technologies*. Springer, 2019

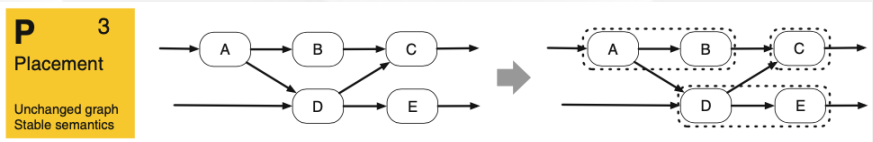
Batching



"Пакетирование"снижает накладные расходы обслуживания за счёт групповой обработки данных. Повышает производительность за счёт уменьшения количества служебных операций подтверждения. Однако повышается задержка отклика.

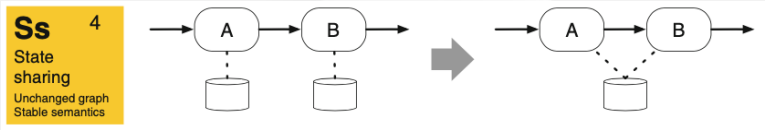


"Слияние"объединяет операторы в один для устранения накладных расходов сериализации и передачи данных. Операторы могут объединяться различными способами, включая их выполнение в рамках одного потока или использование разных потоков, но в одном адресном пространстве. Слияние операторов может понизить степень параллельного выполнения.



"Размещение"переносит операторы плана выполенения на конкретные вычислительные узлы и ядра процессоров для снижения накладных расходов коммункации и лучшего использования вычислительных ресурсов. При этом необходимо соблюдать баланс между снижением нагрузки на сеть и увеличением нагрузки на диски, процессоры и оперативную память.

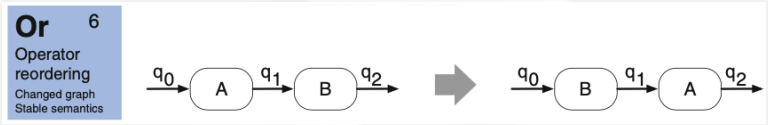
State sharing



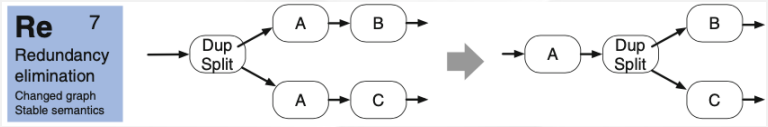
"Общее состояние"позволяет избежать лишних копий данных, снижая нагрузку на подсистему защиты от сбоев. С уменьшением размера "состояния снижается нагрузка на диски.

Operator separation

"Разделение оператора"позволяет разделить долгие и объемные вычисления на последовательность коротких шагов. В ряде случаев, такая оптимизация может снизить уровень использования доступных ресурсов. Часто использовуется для реализации других оптимизаций, например для переупорядочивания операторов.

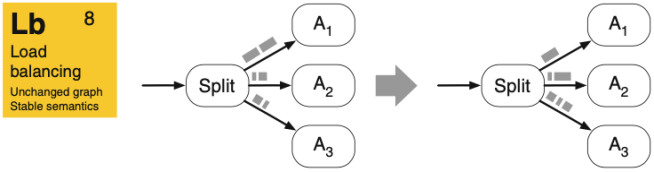


"Переупорядочивание операторов"перемещает в начало операторы, наиболее существенно фильтрующие или уменьшающие объем данных. Снижает нагрузку на весь конвейер за счёт исключения ненужных сообщений.



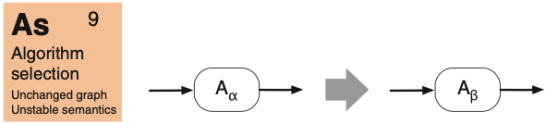
"Удаление избыточности"исключает дублирующие и ненужные вычисления. При этом гарантируется, что удалённые операторы не повлияют на результат вычислений.

Load balancing



"Балансирование нагрузки"позволяет распределить использование узлов кластера за счёт перераспределения потоков сообщений.

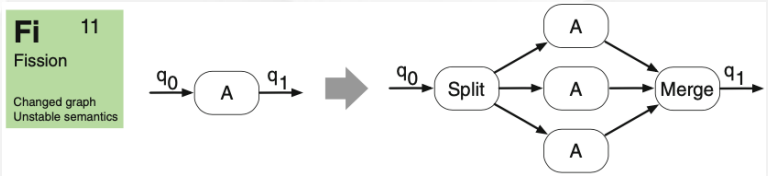
Algorithm selection



"Выбор алгоритма"заменяет одну реализацию оператора другой, если известен более эффективный алгоритм обработки.

Load shedding

"Сброс нагрузки"позволяет снизить вычислительную нагрузку на узлы кластера и предотвратить аварийную ситуацию. Данная оптимизация может изменить результат, поскольку может измениться состав данных при сохранении предыдущих правил обработки в окне.



"Расщепление"позволяет осуществить параллельную обработку данных за счёт кратного дублирования оператора. При этом предполагается, что порядок данных будет сохранён.

H. Röger and R. Mayer. A comprehensive survey on parallelization and elasticity in stream processing.

***ACM Comput. Surv.*, 52(2), Apr. 2019**

Свойства системы потоковой обработки

- **Тип**: общего назначения, CEP
- **Модель программирования**: декларативная, императивная
- **Выделенные потоки (Sub-stream)**: разделение по ключам или окнами
- **Модель инфраструктуры**: тип - single-node, cluster, cloud, or fog solutions, и модель памяти - сообщения, разделяемая память и пр.
- **Модель операторов**: без состояния, с хранением состояния. Управление состоянием - внешнее или внутреннее.
- **Методы распараллеливания операторов**: на уровне задач или на уровне данных

См. [4]

Распараллеливание задачи (оператора) [4]

Параллельная обработка потока несколькими задачами. Например, конвертер видеопотока в несколько форматов.

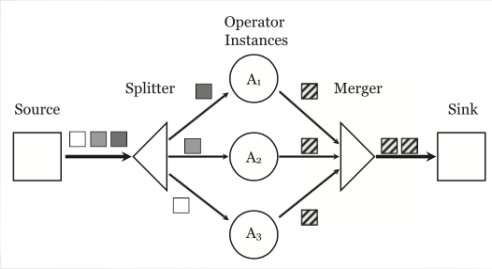
Ограничения:

- сетевой трафик может кратно увеличиться;
- возможно возникновение дизбаланса между операторами если скорость обработки окажется разной;
- масштабирование может быть ограниченным, если распараллеленные задачи имеют общие ресурсы.

Распараллеливание данных [4]

Позволяет достичь лучшего балансирования нагрузки и сохранить порядок обработки.

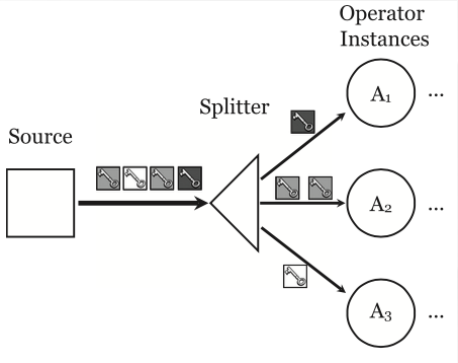
- key-based - может возникнуть дисбаланс из-за изменения состава данных;
- Shuffle Grouping - имеются ограничения на операции аггрегации.



- **Shuffle Grouping** - случайный порядок
- **Key-based Splitting** - на основе некоего ключа
- **Window-based Splitting** - окна с перекрытием
- **Pane-based Splitting** - последовательные интервалы без перекрытия

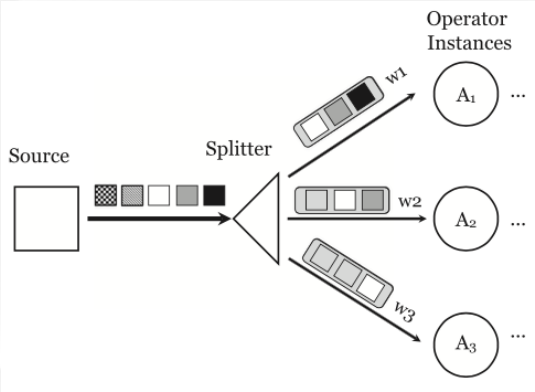
См. [4]

Key-based Splitting



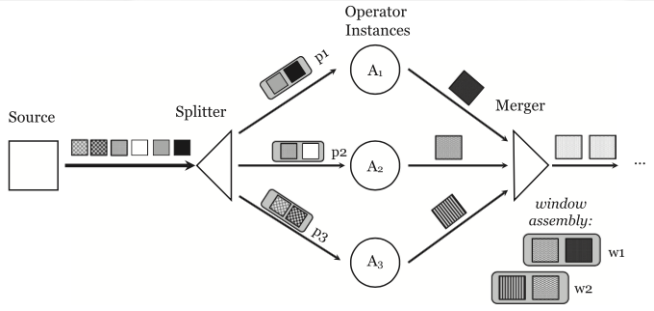
Key-based Splitting - на основе некоего ключа

См. [4]



Window-based Splitting - окна с перекрытием

См. [4]



Pane-based Splitting - последовательные интервалы без перекрытия

См. [4]

Аспекты обеспечения эластичности выполнения операторов [4]:

- **Входные данные (Input Data)** – подбор данных и алгоритмов под решаемую задачу
- **Синхронизация (Timing)** – реактивный, проактивный подход к адаптации
- **Цель эластичности** – QoS, производительсность, минимизация задержек...
- **Гарантия обработки** – процент сообщений, обработанных за регламентное время/уровень задержки
- **Методология** – threshold-policy driven, model driven, and learning-based
- **Централизация или распределение** – касается режима управления
- **Миграция состояний** – определяет режим восстановления после сбоев
- **Режим выполнения** – управление оптимизациями на уровне операторов

- **Централизованная эластичность**:
  - на основе порогов - при достижении определённых значений загрузки, активировать новые узлы;
  - реактивные модели - в зависимости от загрузки графа выполнения задач, запускать новые узлы и перераспределять медленные цепочки;
  - проактивные модели - на основе известной информации о параметрах выполнения, планировать узлы и распределение операторов.
- **Распределённая эластичность**: многооператорные и одноператорные методы.

См. [4]

Automatic Parameter Tuning for Big Data Processing Systems H. Herodotou, Y. Chen, and J. Lu. A survey on automatic parameter tuning for big data processing systems.

***ACM Comput. Surv.*, 53(2), Apr. 2020**

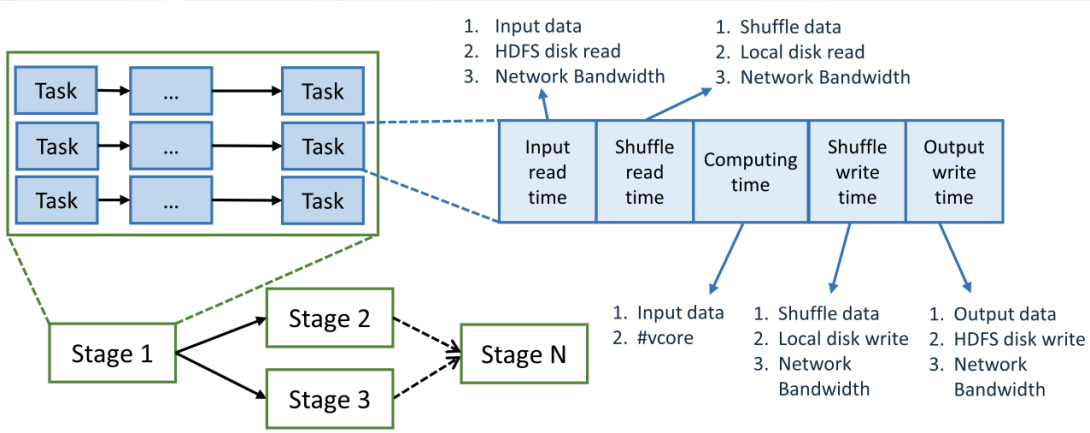
Основные проблемы автоматической подстройки параметров

- Большое количество и нечёткое влияние параметров на эластичность. Hadoop, Spark, Storm – более двухсот параметров конфигурации, изменение одних из которых может влиять на других.
- Размер и сложность обслуживаемой бизнес-системы.
- Недостаточность статистики функционирования систем.

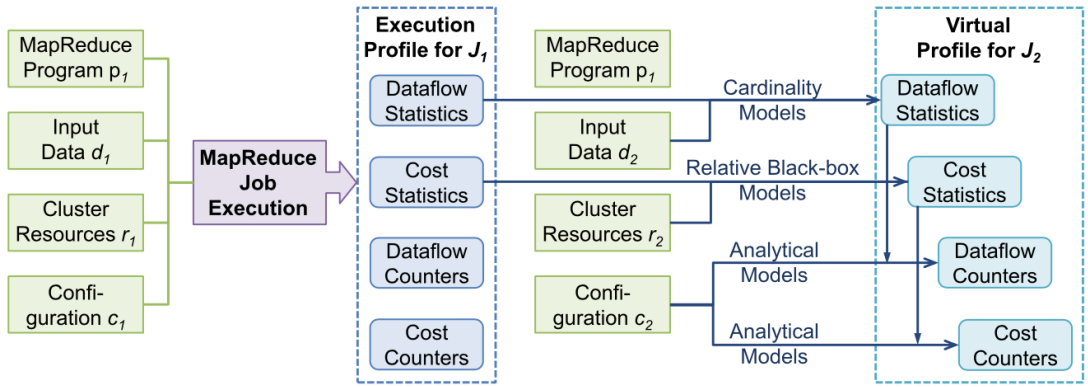
Методы подстройки параметров:

- **Основанные на правилах (Rule-based)** – используются как вспомогательные стредства в помощь администраторам. Правила формируются экспертами и могут быть сформулированы в инструкциях, учебных пособиях и пр. По шаблону можно быстро развернуть типовую бизнес-систему.
- **Стоимостной оценки (Cost modeling)** – используется модель предсказания с использованием аналитических функций стоимости, синтезированных на основе информации о системе (white-box). Обычно требуются журналы операций и статистика для начального конфигурирования.
- **Иммитационные (Simulation-based)** – основаны на моделях, описывающих поведение отдельных частей или всей системы в зависимости от различных параметров.
- **Подборные – экспериментальные с обратной связью (Experiment)** – подстройка параметров на основе собранных журналов
- **Основанные на машинном обучении (Machine learning)** – не предполагают использование знаний о структуре системы, но выстраивают закономерности между изменением параметров и результатом.
- **Адаптивные(Adaptive)** – предполагают подстройку значений параметров в процессе выполнения программ.

Сложность подстройки параметров



Создание профиля системы



Automatic Parameter Tuning

Feature	Rule-based	Cost modeling	Simulation	Experiment-driven	Machine learning	Adaptive
Key modeling technique	rules	cost functions	simulation	search algorithms	ML models	mixed
# of parameters modeled	few	some	some	many	many	some
System understanding	strong	strong	strong	light	no	strong
Need for history logs	no	light	light	strong	strong	light
Need for data input stats	no	light	light	no	strong	light
Real tests to run	no	some	no	yes	yes	yes
Time to build model	efficient	efficient	medium	slow	slow	medium
# of metrics predicted	few	few	some	few	many	some
Prediction accuracy	low	medium	medium	medium	high	medium
Adapt to workload	adaptive	light	light	no	no	adaptive
Adapt to system changes	no	no	light	no	adaptive	light

Слабые и сильные стороны подстройки параметров

Достоинства и недостатки различных методов автоматической подстройки параметров

Подход	Достоинства	Недостатки
На правилах	• Не требует узко специализированного ПО • Некоторые параметры легко регулируются	• Время затратный процесс подбора в процессе экспериментов • Необходимо в деталях знать особенности работы • Высокий риск деградации прозводительности
Оценка стоимости	• Высокая эффективность в оценке производительности • Хорошая точность в большинстве (типовых) сценариев	• Сложно учесть внутреннюю сложность системы и подсоединяемые компоненты (например, планировщики) • Модель основана на упрощенных оценках • Не эффективны в гетерогенных кластерах
Имитационные	• Высокая точность имитации динамических систем • Эффективны для гранулярной оценки производительности	• Сложно полноценно описать внутреннюю структуру системы • Нет возможности описать динамическую загрузку кластера • Не эффективны в поиске оптимального значения
Подборные	• Находят параметры на основе реальных тестов реальных систем • Работают на разных версиях ПО и оборудования	• Требуют много времени для запуска в различных режимах • Для однократно запускаемых приложений не эффективны
Машинное обучение	• Способность уловить сложное поведение системы • Не зависят от внутреннего устройство ПО и ооборудования • Основанные на реальных измерениях	• Требуют большого объема данных для обучения • Обучение на журналах операций приводит к подгонке данных • Низкая точность для впервые запускаемых приложений • Сложно выбрать подходящую модель обучения
Адаптивные	• Подбирают параметры на реальных тестах реальных систем • Могут подстраиваться под динамические режимы • Сносно работают с однократно запускаемыми приложениями	• Применимы только для долго работающих приложений • Некорректная настройка может привести к деградации (например запаздыванием) • Игнорируют доступные ресурсы системы в целом

[1] H. Herodotou, Y. Chen, and J. Lu. A survey on automatic parameter tuning for big data processing systems. *ACM Comput. Surv.*, 53(2), Apr. 2020.

[2] M. Hirzel, R. Soulé, B. Gedik, and S. Schneider. Stream query optimization. In S. Sakr and A. Y. Zomaya, editors, *Encyclopedia of Big Data Technologies*. Springer, 2019.

[3] M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm. A catalog of stream processing optimizations. *ACM Comput. Surv.*, 46(4):46:1–46:34, Mar. 2014.

[4] H. Röger and R. Mayer. A comprehensive survey on parallelization and elasticity in stream processing. *ACM Comput. Surv.*, 52(2), Apr. 2019.