РАСПРЕДЕЛЯЙ И ВЛАСТВУЙ -2: ПОТОКИ ДАННЫХ НАНОСЯТ ОТВЕТНЫЙ УДАР

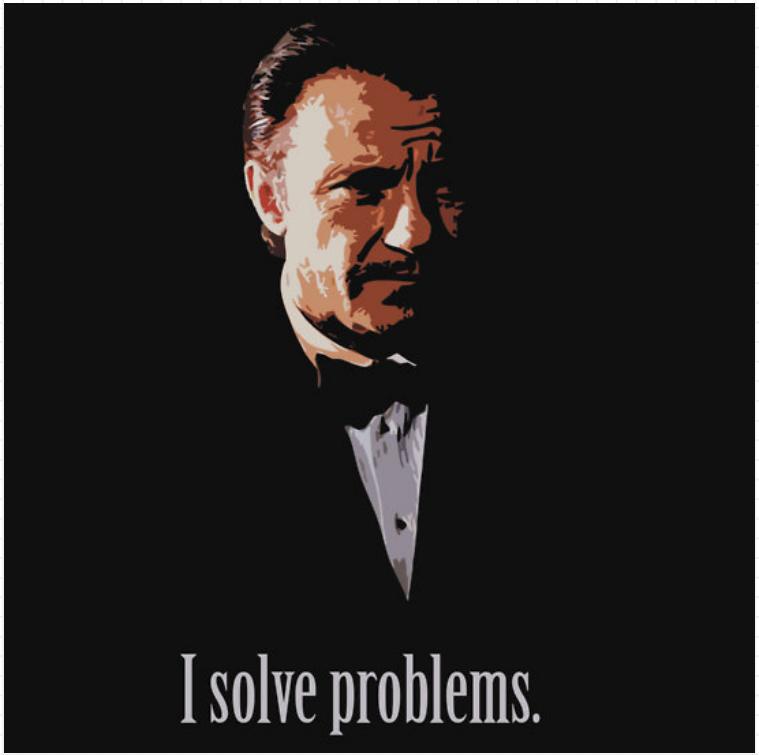
@gamussa @hazelcast #jpoint


Stream
v.
Batch

КТО?

КТО?

Solutions Architect



КТО?

Solutions Architect
Developer Advocate

КТО?

Solutions Architect
Developer Advocate

@gamussa везде в интернете

КТО?

Solutions Architect
Developer Advocate

@gamussa везде в интернете

А ты зафоловил меня, $userName ©


JUST DEUX IT.
HOT SHOTS!
PART DEUX
TWENTIETH CENTURY FOX Presents A JIM ABRAHAMS Film CHARLIE SHEEN HOT SHOTS! PART DEUX LLOYD BRIDGES
VALERIA GOLINO BRENDA BAKKE and RICHARD CRENNA Music by BASIL POLEDOURIS Costume Designer MARY MALIN Edited By MALCOLM CAMPBELL
Production Designer WILLIAM A. ELLIOTT Director of Photography JOHN R. LEONETTI Executive Producer PAT PROFT Written By JIM ABRAHAMS & PAT PROFT
PG-13 PARENTS STRONGLY CAUTIONED Some Material May Be Inappropriate for Children Under 13
Produced By BILL BADALATO Directed By JIM ABRAHAMS
DOLBY STEREO
©1993 TWENTIETH CENTURY FOX

Распределяй и властвуй: введение в распределенные системы

Виктор Гамов
Hazelcast

Jpoint Student Day

Disclaimer: Нам пишут

* авторская орфография сохранена

Disclaimer: Нам пишут

? Из названия не было понятно, что это пропоганда **хазлкаста**, что не правильно.

* авторская орфография сохранена

Disclaimer: Нам пишут

Disclaimer: Нам пишут

? Из названия не было понятно, что это пропоганда **хазлкаста**, что не правильно.

* авторская орфография сохранена

Disclaimer: Нам пишут

? Из названия не было понятно, что это пропоганда **хазлкаста**, что не правильно.

✓ Все так 😏

* авторская орфография сохранена

Disclaimer: Нам пишут

? Из названия не было понятно, что это пропоганда **хазлкаста**, что не правильно.

✓ Все так 😏
✓ Читайте абстракт
✓ Как и Вы, я здесь в коммандировке

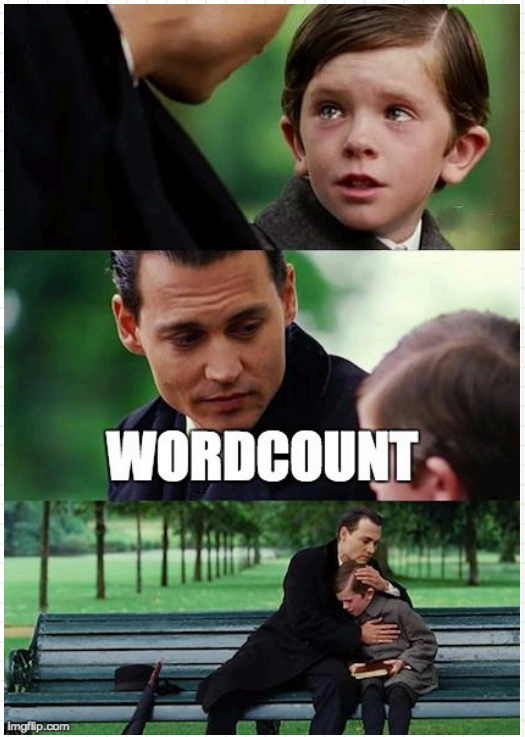
* авторская орфография сохранена

Disclaimer: Нам пишут

* авторская орфография сохранена

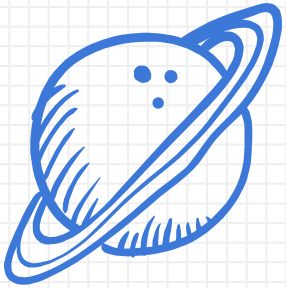
Disclaimer: Нам пишут

? Спикер … уделяет время лишь одному **банальному примеру подсчёта слов в файле** с использованием фреймворка их компании.



* авторская орфография сохранена


MAKE IN MEMORY
GREAT AGAIN



Пакетная Обработка

Данные в состоянии покоя

@gamussa @hazelcast #jpoint

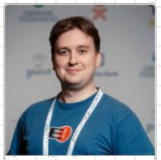
Данные и запросы

Происхождение и обработка

Sir William Davenant @SirWilliamD · 8h

Big data in the past. Staff hand sorting 4 million used #London Underground tickets to analyse line use in 1939. Photograph by Gerry Cranham

@gamussa @hazelcast #jpoint

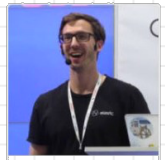


Алексей Шипилёв
Red Hat

Charles Nutter
Red Hat

Евгений Борисов
Naya Technologies

Николай Алименков
EPAM



Philipp Krenn
Elastic

Arun Gupta
Couchbase

Кирилл Толкачев
Альфа-Лаборатория

Алексей Зиновьев
EPAM

Андрей Бреслав
JetBrains

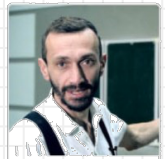
Тагир Валеев
JetBrains

Барух Садогурский
JFrog

Сергей Куксенко
Oracle

Андрей Паньгин
Одноклассники

Виктор Гамов
Hazelcast



Алексей Савватеев

Владимир Долженко
IHS Markit

Егор Бугаенко
Teamed.io

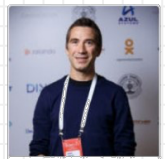
Sven Ruppert
Macros Reply GmbH

Владимир Иванов
Oracle

Sasha Goldshtein
Sela Group

Никита Липский
Excelsior

Антон Кекс
Codeborne



Volker Simonis
SAP

Milen Dyankov
Liferay

@gamussa @hazelcast #jpoint

Данные...

Данные...

Данные...

✓ ... привязаны ко времени

✓ ... immutable по своей сути

CRUD -> CR

МАРИВАННА, В УГЛУ СКР..©

Обработка – это запрос

Обработка – это запрос

Функция по полному набору данных

Обработка – это запрос

Функция по полному набору данных
Проекции

Обработка – это запрос

Функция по полному набору данных
Проекции
Агрегации

Обработка – это запрос

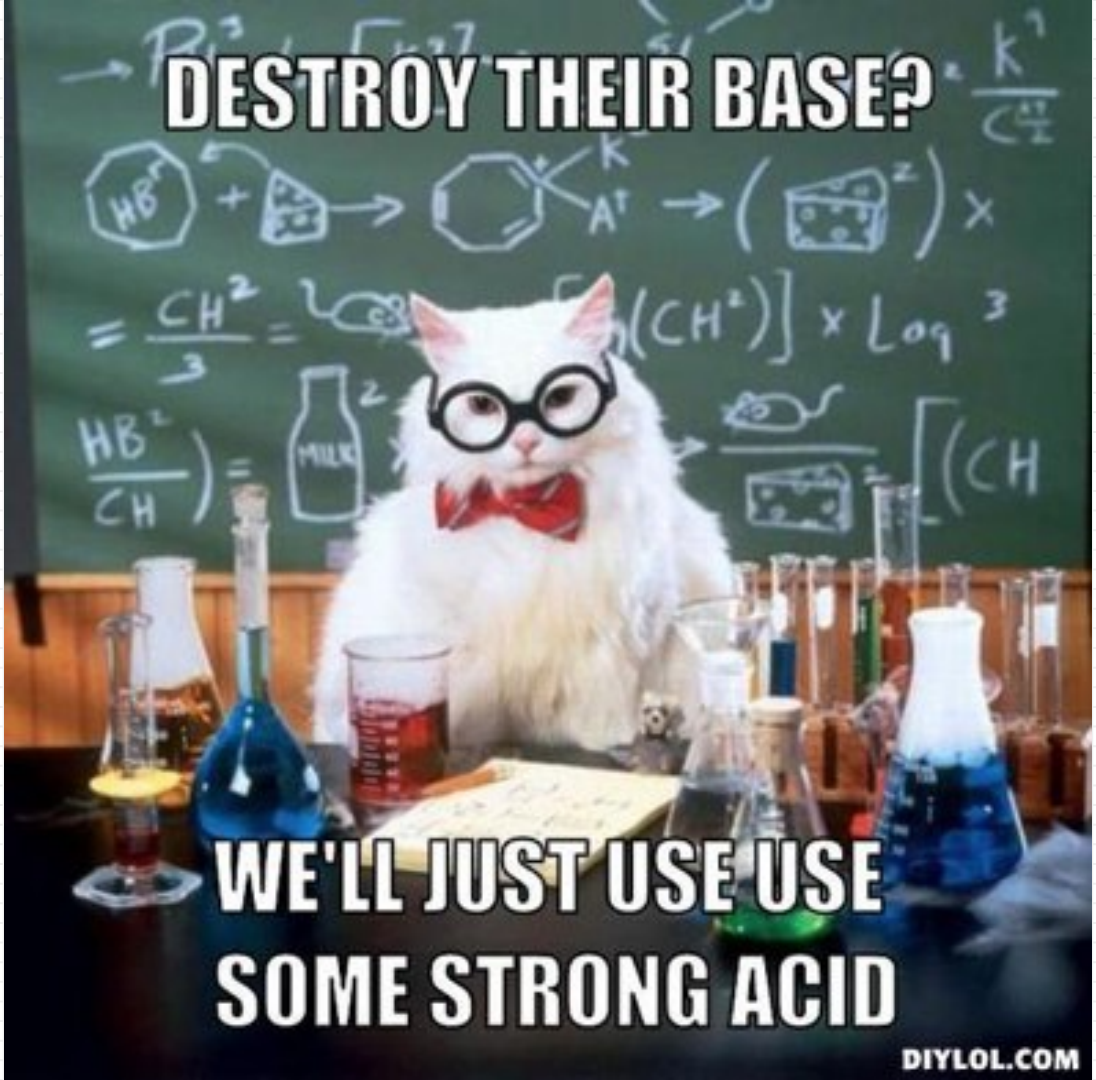
Функция по полному набору данных
Проекции
Агрегации
Joins

```
SELECT
  user_vote, count(*)
FROM AccessLog
WHERE event_date
BETWEEN"04/07/2017" AND "04/07/2017"
GROUP BY user_vote;
```

```
SELECT
   user_vote, count(*)
FROM AccessLog
WHERE event_date
BETWEEN "04/7/2017" AND "04/08/2017"
GROUP BY user_vote;
```

```
SELECT
  user_vote, count(*)
FROM AccessLog
WHERE event_date
BETWEEN"04/07/2017" AND "04/08/2007"
GROUP BY user_vote;
```

```
private static void countVotes(IMap<String, Vote> userVotes) {
    // execute the aggregation and print the result
    long countVotes = userVotes
            .aggregate(Aggregators.<String, Vote>count());
}
```


DESTROY THEIR BASE?
WE'LL JUST USE USE
SOME STRONG ACID
DIYLOL.COM

Lambda architecture origins

How to beat the CAP theorem

THURSDAY, OCTOBER 13, 2011

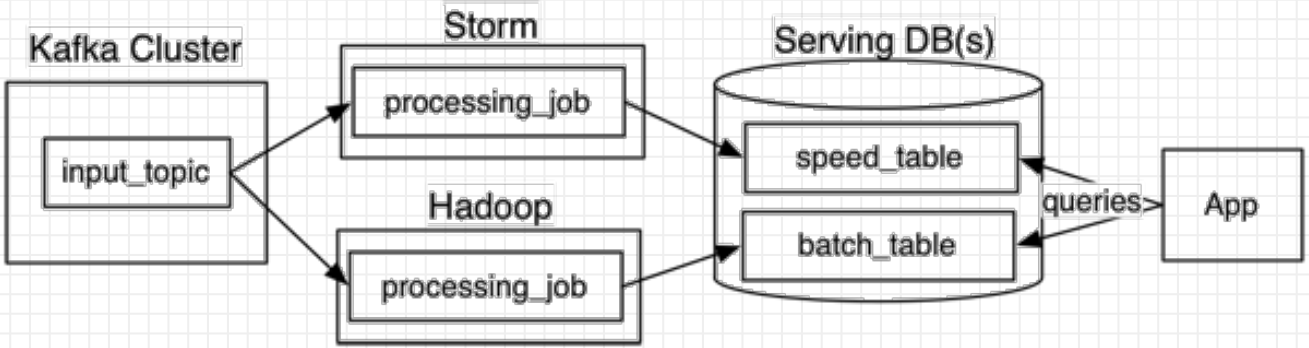
The CAP theorem states a database cannot guarantee consistency, availability, and partition-tolerance at the same time. But you can't sacrifice partition-tolerance (see here and here), so you must make a tradeoff between availability and consistency. Managing this tradeoff is a central focus of the NoSQL movement.

Consistency means that after you do a successful write, future reads will always take that write into account. Availability means that you can always read and write to the system. During a partition, you can only have one of these properties.

Systems that choose consistency over availability have to deal with some awkward issues. What do you do when the database isn't available? You can try buffering writes for later, but you risk losing those writes if you lose the machine with the buffer. Also, buffering writes can be a form of inconsistency because a client thinks a write has succeeded but the write isn't in the database yet. Alternatively, you can return errors back to the client when the database is unavailable. But if you've ever used a product that told you to "try again later", you know how aggravating this can be.

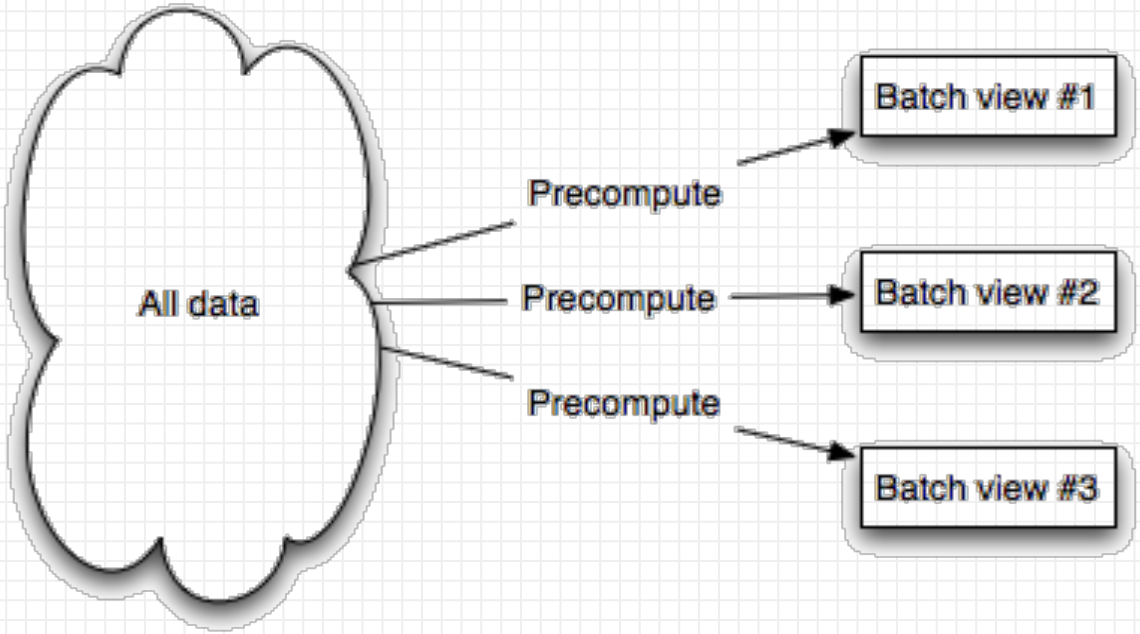
http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html

Lambda Architecture



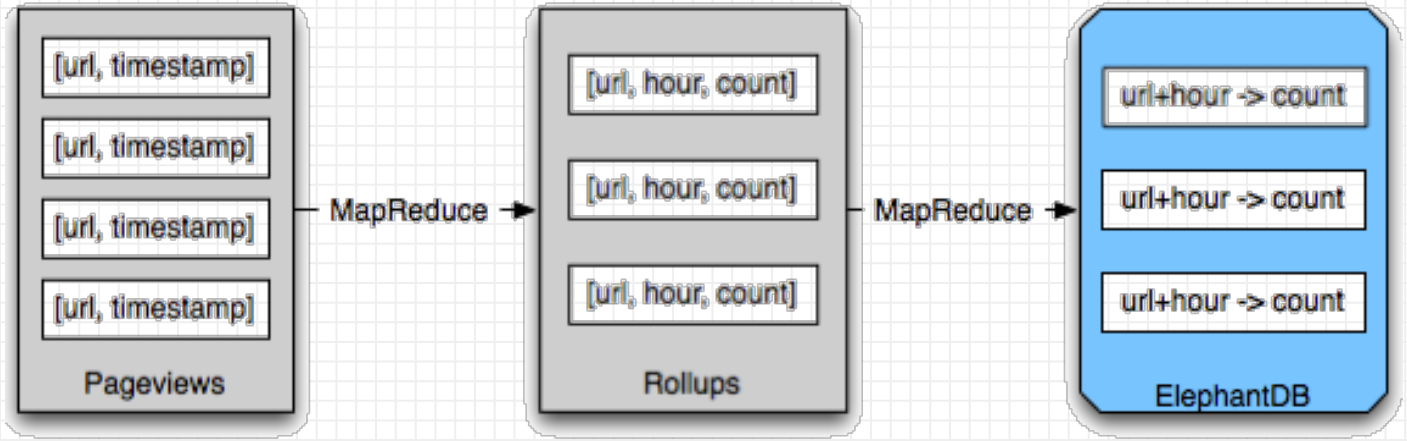

hadoop
Map Reduce

Предподсчитанный результат



http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html

Пакетный процесс



http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html

Варианты ответа - Ответы -

- Spring - Глубоко и не очень – Евгений Борисов / Naya Technologies 71,63% 101
- Shenandoah: сборщик мусора, который смог – Алексей Шипилёв / Red Hat 51,06% 72
- Java Puzzlers NG S02: Всё чудесатее и чудесатее – Тагир Валеев / JetBrains; Барух Садогурский / JFrog 50,35% 71
- Сделаем Hibernate снова быстрым – Николай Алименков / EPAM 49,65% 70
- Аннотации в Java — это ошибка – Егор Бугаенко / Teamed.io 46,10% 65
- Проклятие Spring Test – Кирилл Толкачев / Альфа-Лаборатория; Евгений Борисов / Naya Technologies 44,68% 63
- Распределяй и властвуй — 2: Потоки данных наносят ответный удар – Виктор Гамов / Hazelcast 44,68% 63
- Динамический поиск потенциальных дедлоков в многопоточных приложениях на Java – Никита Коваль / Devexperts 43,97%6

Хранилище выдачи

Хранилище выдачи

Очень легко читать

Хранилище выдачи

Очень легко читать

K,V – в идеале

Хранилище выдачи

Очень легко читать

K,V – в идеале

Очень быстро читать

Хранилище выдачи

Очень легко читать

K,V – в идеале

Очень быстро читать

хранить в памяти

Хранилище выдачи

Очень легко читать

K,V – в идеале

Очень быстро читать

хранить в памяти

хранить предподсчитанный результат

Данные слишком важны, чтобы хранить их на одной машине

ORACLE COHERENCE

ORACLE®

Coherence

@gamussa @hazelcast #jpoint

ORACLE COHERENCE

In-memory data grid

ORACLE®
Coherence

ORACLE COHERENCE

In-memory data grid

Распределенные кэши

ORACLE COHERENCE

In-memory data grid

Распределенные кэши

Коммерческий продукт

INFINISPAN

Infinispan

@gamussa @hazelcast #jpoint

INFINISPAN

In-memory data grid

Infinispan

@gamussa @hazelcast #jpoint

INFINISPAN

In-memory data grid

распределённые кэши

Infinispan

@gamussa @hazelcast #jpoint

INFINISPAN

In-memory data grid
распределённые кэши
Лицензия Apache v2

Infinispan

@gamussa @hazelcast #jpoint

Еще хотелки...

Еще хотелки...

Простота

Еще хотелки...

Простота

знакомый API

Еще хотелки...

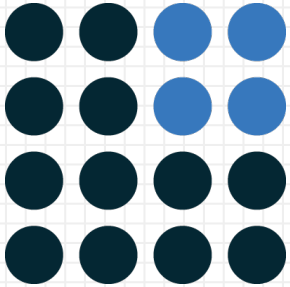
Простота
знакомый API
встраиваемость

Еще хотелки...

Простота
знакомый API
встраиваемость
Cloud Native

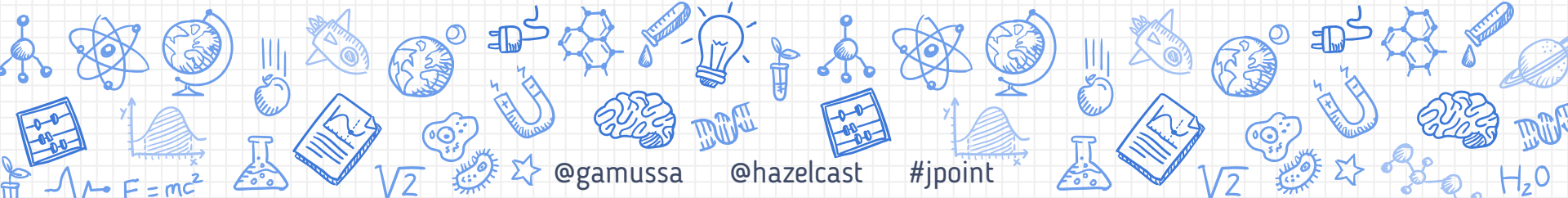
Еще хотелки...

Простота
знакомый API
встраиваемость
Cloud Native



ПСС, ПАРЕНЬ, HAZELCAST IMDG, НЕ ХОЧЕШЬ?

Ну очень быстрый грид ©



Hazelcast IMDG, берем?

Hazelcast IMDG, берем?

In-memory Data Grid

Hazelcast IMDG, берем?

In-memory Data Grid

Распределенные

Кэши (IMap, JCache)

Проекции, Агрегации

Java колекции (IList, ISet, IQueue)

Система обмена сообщений (Topic, RingBuffer)

Вычисления (ExecutorService, M-R)

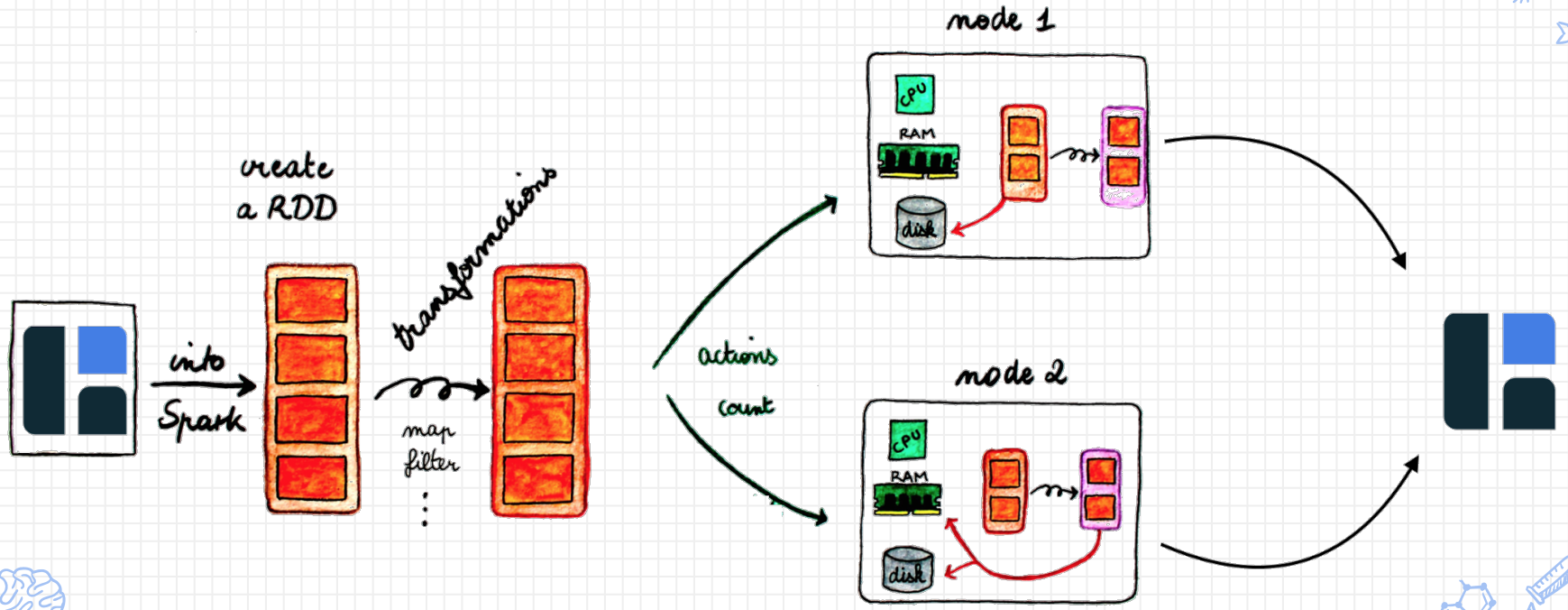

So, how do I query the database?
It's not a database. It's a key-value store!
Ok, it's not a database. How do I query it?
You write a distributed map reduce function in Erlang!
Did you just tell me to go fuck myself?
I believe I did, Bob.

Распределяй и властвуй: введение в распределенные системы

Виктор Гамов
Hazelcast

Jpoint Student Day

Пример: ПАКЕТНАЯ ОБРАБОТКА НА HAZELCAST и SPARK



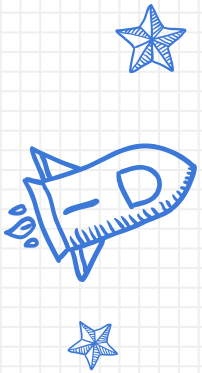
ДАННЫЕ НЕ ДОЛЖНЫ ОБНОВЛЯТЬСЯ ВО ВРЕМЯ ЧТЕНИЯ

@gamussa @hazelcast #jpoint

ПОЧЕМУ?

ПРИ РАСШИРЕНИИ, МАР **ПЕРЕРАСПРЕДЕЛЯЕТ** ДАННЫЕ ВНУТРИ КОНТЕЙНЕРА

КУРСОР НЕ УКАЗЫВАЕТ НА КОРРЕКТНУЮ ЗАПИСЬ. МОГУТ ВОЗНИКАТЬ **ДУБЛИКАТЫ** ИЛИ ДАННЫЕ ПРОПАДАТЬ



ПОТОКОВАЯ ОБРАБОТКА

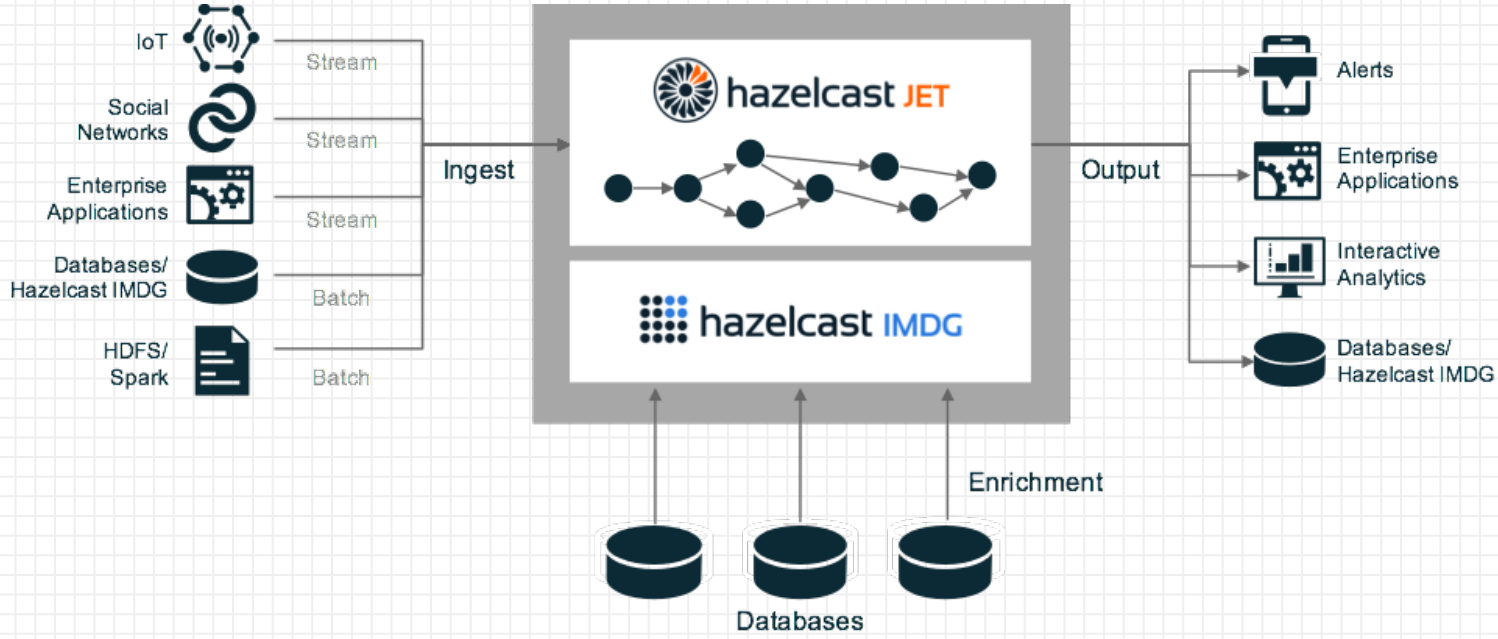
Данные в движении

@gamussa @hazelcast #jpoint

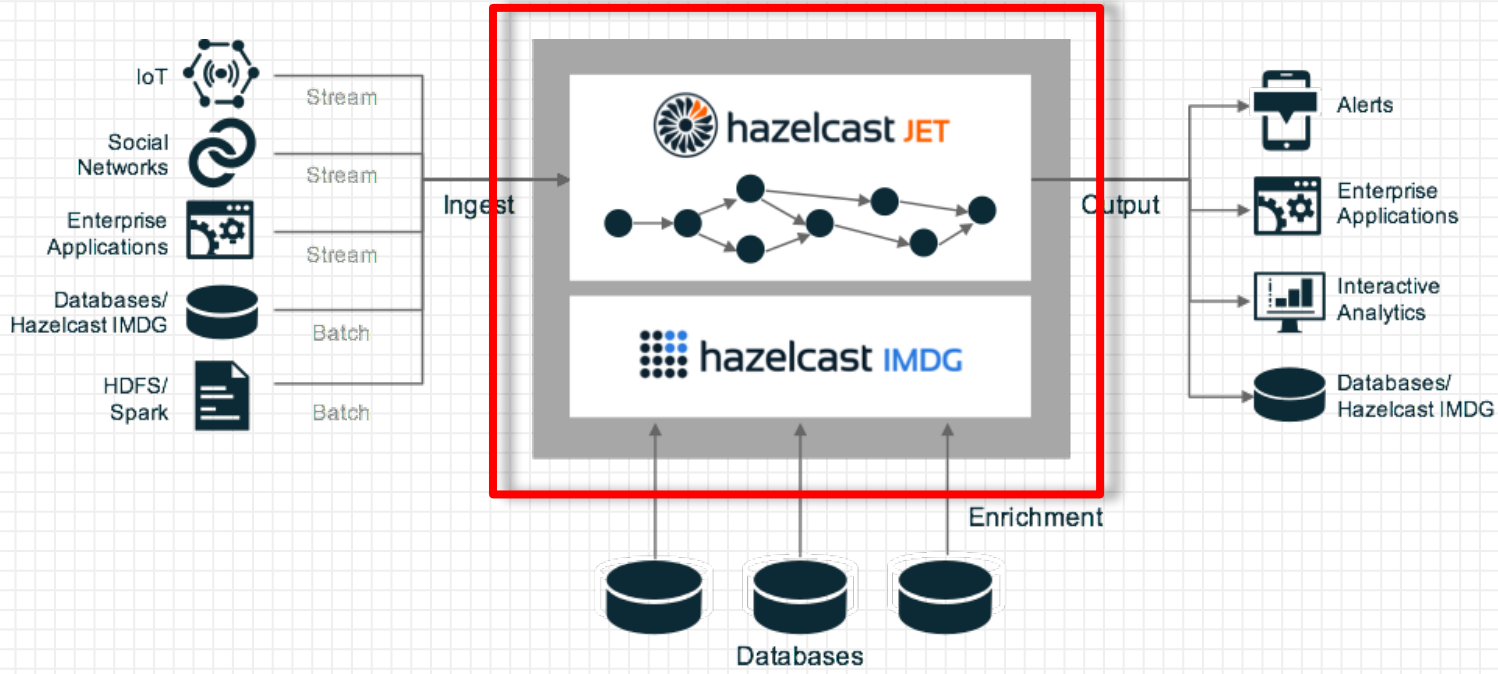
HAZELCAST JET

Считаем ваши слова. Быстро. В памяти

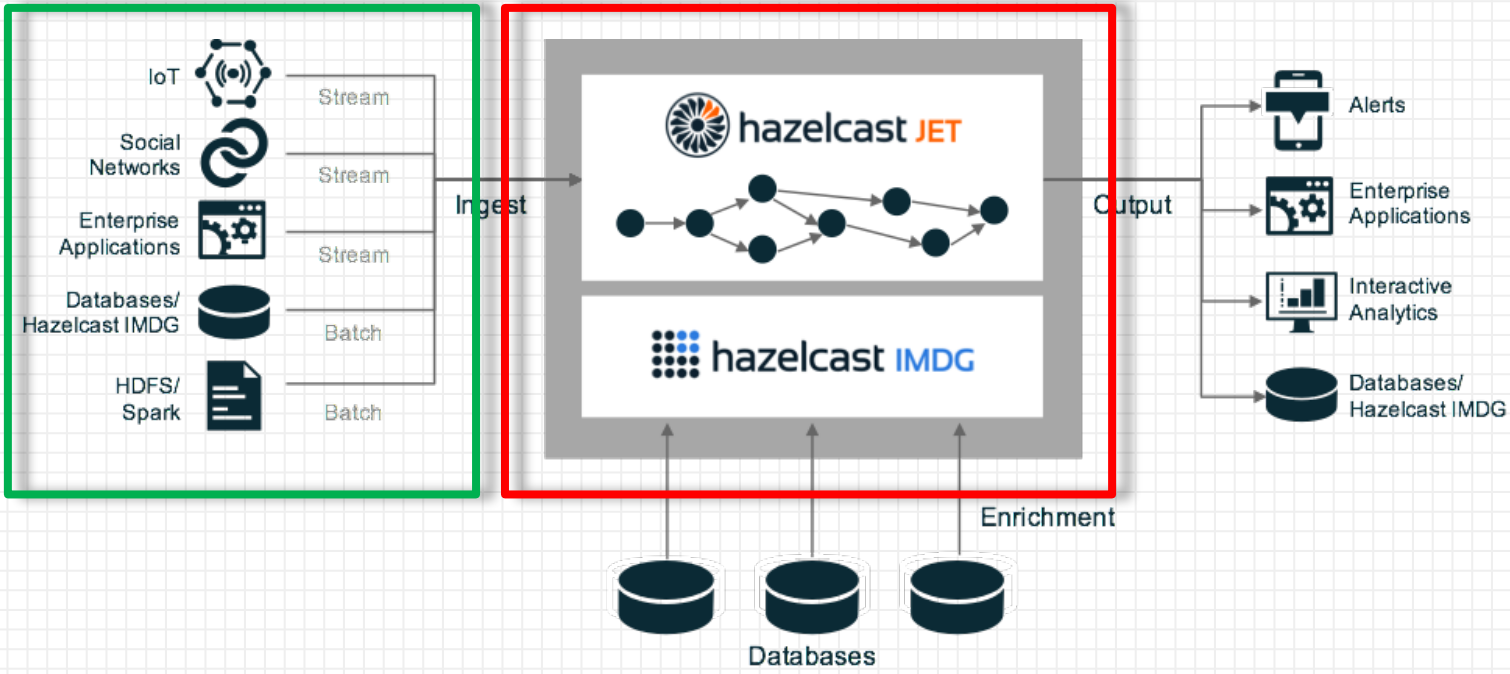
Потоковая и пакетная обработка в памяти



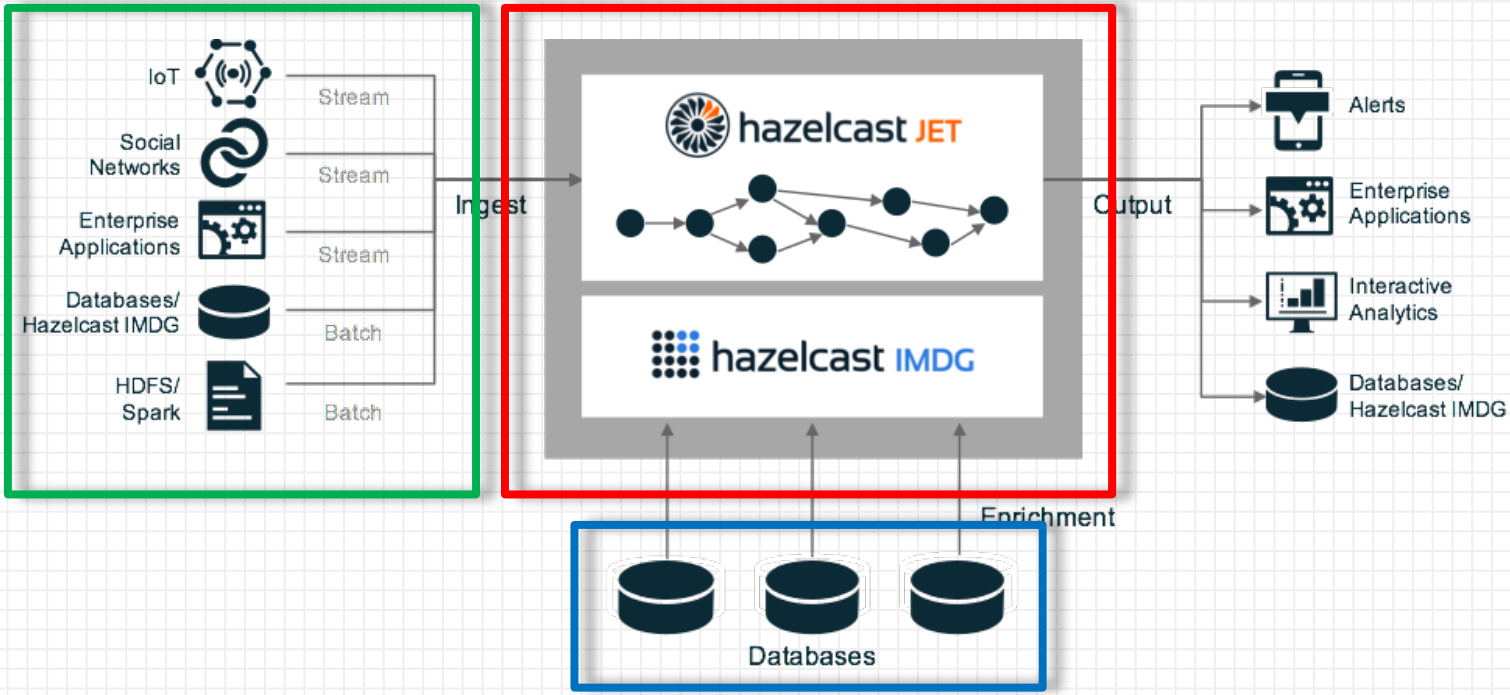
Потоковая и пакетная обработка в памяти



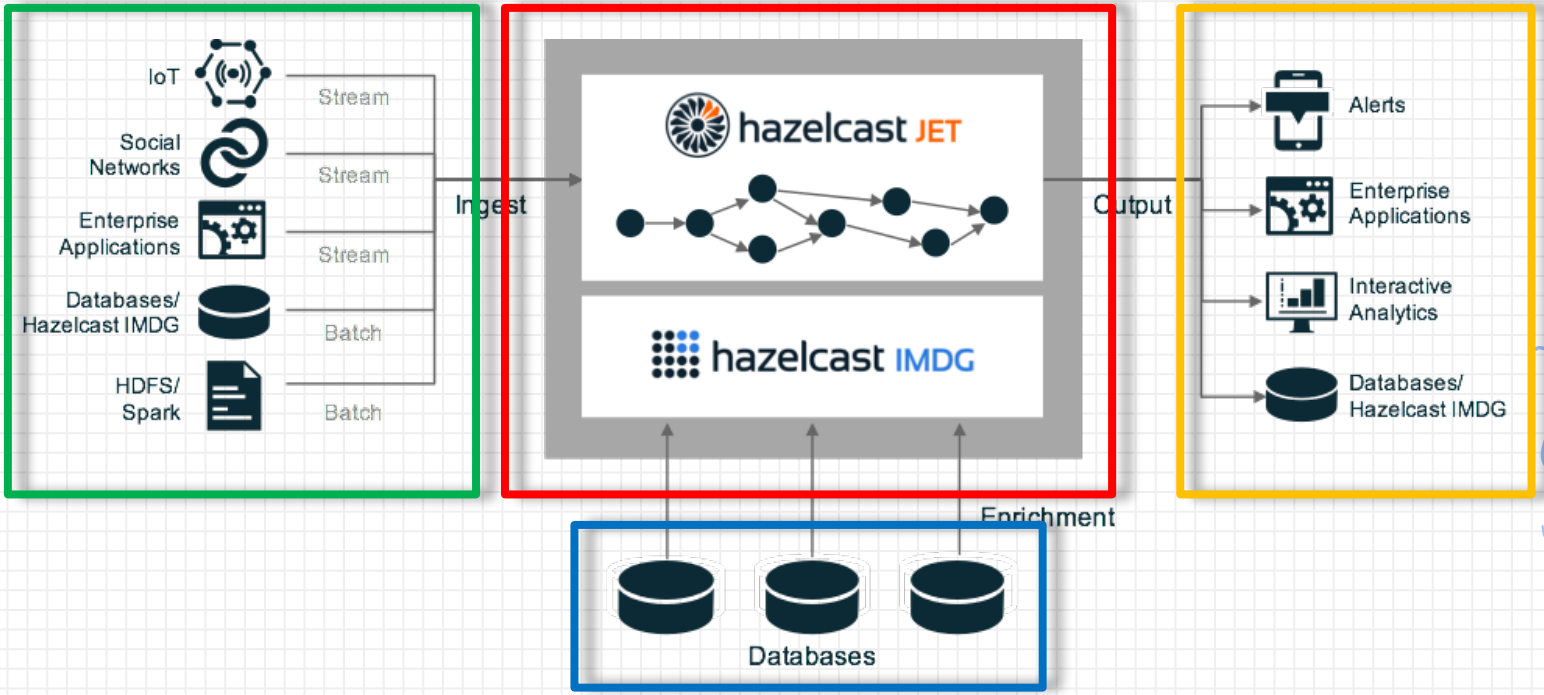
Потоковая и пакетная обработка в памяти



Потоковая и пакетная обработка в памяти



Потоковая и пакетная обработка в памяти



Jet против мира Big Data

Jet против мира Big Data

Простота

Jet против мира Big Data

Простота
в разработке

Jet против мира Big Data

Простота

в разработке

в развертывании (даже в облаках)

Jet против мира Big Data

Простота
- в разработке
- в развертывании (даже в облаках)

Скорость

Jet против мира Big Data

Простота
- в разработке
- в развертывании (даже в облаках)

Скорость
- data affinity

Jet против мира Big Data

Простота
- в разработке
- в развертывании (даже в облаках)

Скорость
- data affinity
- cooperative multitasking

Jet против мира Big Data

Простота
- в разработке
- в развертывании (даже в облаках)

Скорость
- data affinity
- cooperative multitasking

Hazelcast IMDG

Jet против мира Big Data

Простота
- в разработке
- в развертывании (даже в облаках)

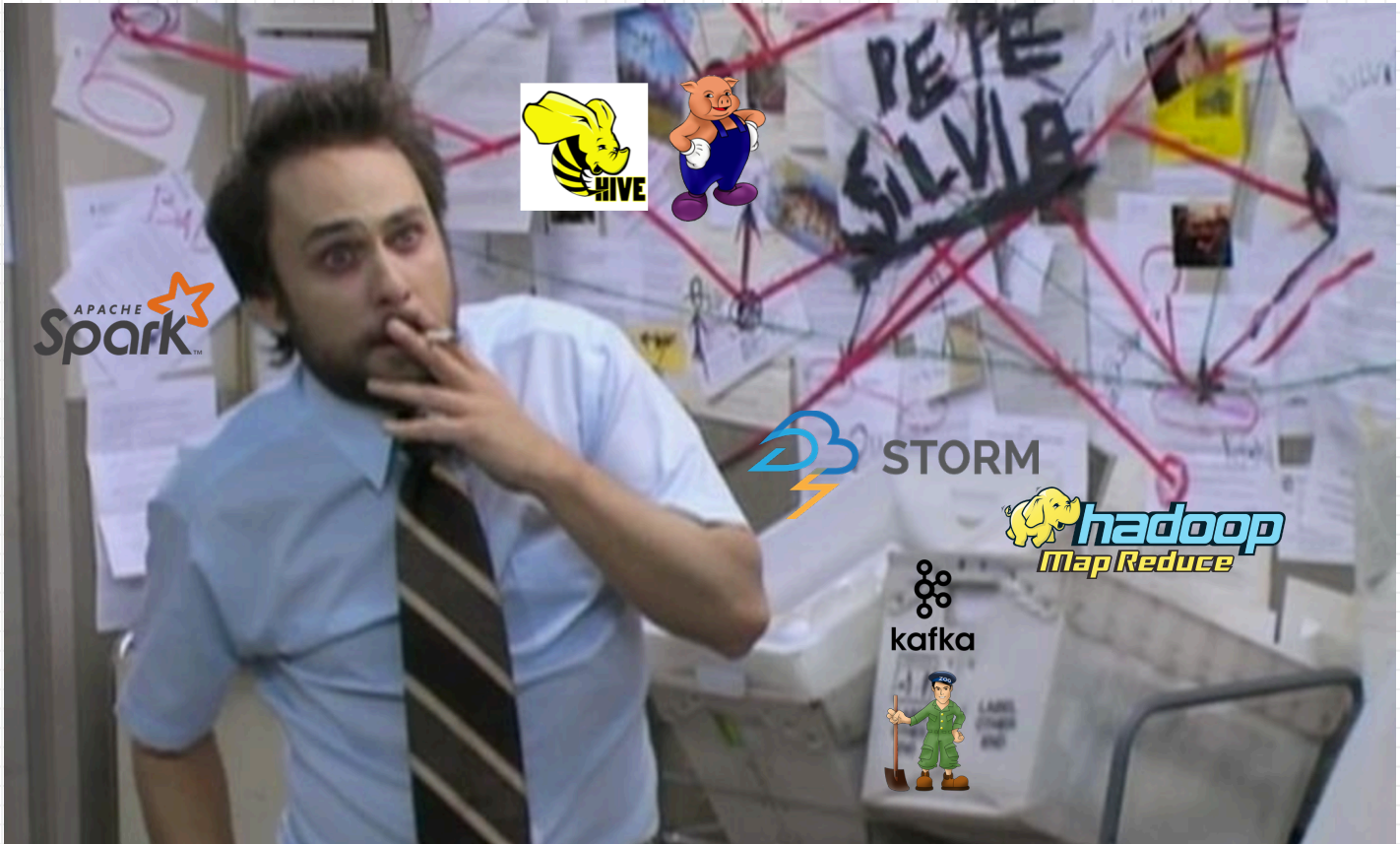
Скорость
- data affinity
- cooperative multitasking

Hazelcast IMDG
- распределенные данные

Jet против мира Big Data

- Простота
  - в разработке
  - в развертывании (даже в облаках)
- Скорость
  - data affinity
  - cooperative multitasking
- Hazelcast IMDG
  - распределенные данные
  - discovery

Когда пытаешься объяснить современный мир Big Data



Локальность и привязка данных

Локальность и привязка данных

Скорость и низкие задержки

Локальность и привязка данных

Скорость и низкие задержки

данные и вычисления расположены на одной ноде

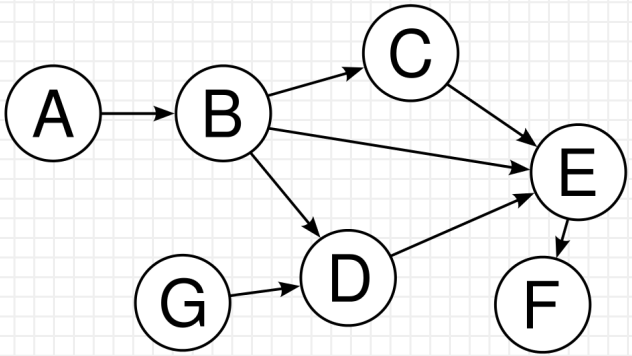
Локальность и привязка данных

Скорость и низкие задержки

данные и вычисления расположены на одной ноде

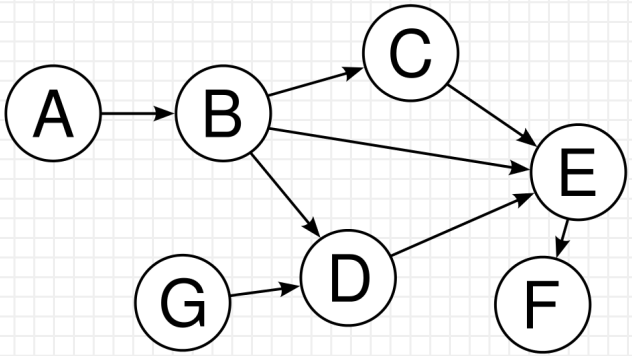
Привязка к структуре партиций

Направленный ациклический граф



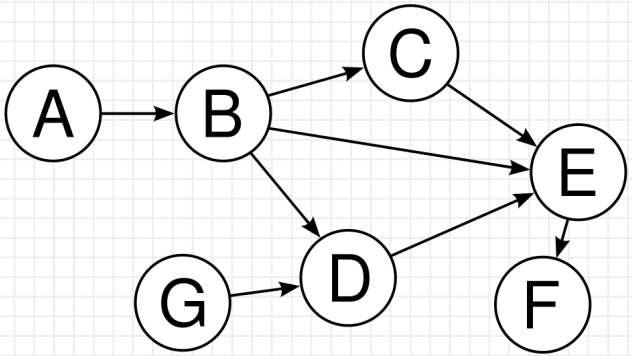
Направленный ациклический граф

Модель описания выполнения



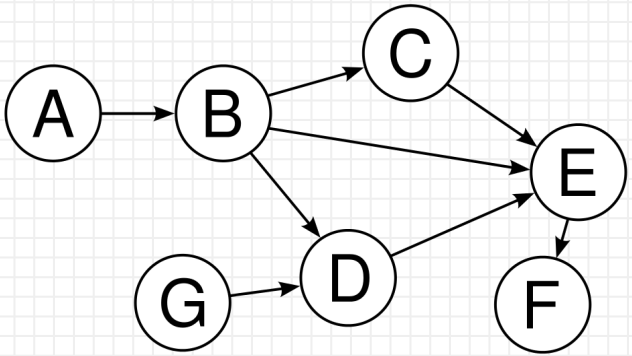
Направленный ациклический граф

Модель описания выполнения
Вершина - шаг выполнения



Направленный ациклический граф

Модель описания выполнения
Вершина - шаг выполнения
Работает как для пакетной и потоковой



Исполнение графа

Исполнение графа

Каждая нода кластера исполняет граф целиком

Исполнение графа

Каждая нода кластера исполняет граф целиком
Каждая вершина графа исполняется набором **tasklet-ов**

Исполнение графа

Каждая нода кластера исполняет граф целиком
Каждая вершина графа исполняется набором **tasklet-ов**
Ограниченное число «настоящих» потоков

Исполнение графа

Каждая нода кластера исполняет граф целиком
Каждая вершина графа исполняется набором **tasklet-ов**
Ограниченное число «настоящих» потоков
~ кол-во процессоров

Исполнение графа

Каждая нода кластера исполняет граф целиком
Каждая вершина графа исполняется набором **tasklet-ов**
Ограниченное число «настоящих» потоков
~ кол-во процессоров
Work-stealing между потоками

Исполнение графа

Каждая нода кластера исполняет граф целиком
Каждая вершина графа исполняется набором **tasklet-ов**
Ограниченное число «настоящих» потоков
~ кол-во процессоров
Work-stealing между потоками
Back pressure между вершинами

Cooperative Multithreading

Cooperative Multithreading

Cooperative Processors выполняются в цикле, который выполняется в **native треде**

Cooperative Multithreading

Cooperative Processors выполняются в цикле, который выполняется в **native треде**

нет переключения контекста

привязка к ядру процессора

Cooperative Multithreading

Cooperative Processors выполняются в цикле, который выполняется в **native треде**

нет переключения контекста

привязка к ядру процессора

Каждый tasklet выполняет небольшой небольшую часть работы (<1ms)

Cooperative Multithreading

Cooperative Processors выполняются в цикле, который выполняется в **native треде**

нет переключения контекста

привязка к ядру процессора

Каждый tasklet выполняет небольшой небольшую часть работы (<1ms)

Cooperative Multithreading

Cooperative Multithreading

1 поток может выполнять ~1000 tasklet

Cooperative Multithreading

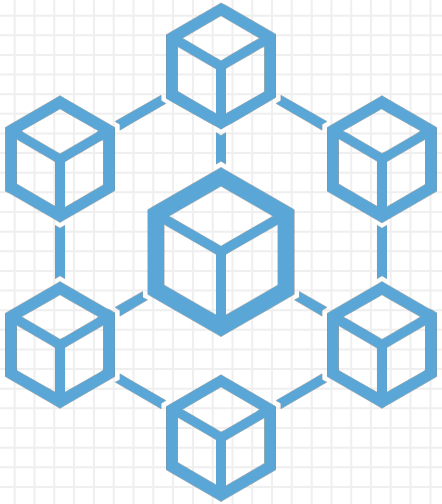
1 поток может выполнять ~1000 tasklet
Если нечего делать, тред

Cooperative Multithreading

1 поток может выполнять ~1000 tasklet
Если нечего делать, тред
Ребра графа имплементированы с помощью RingBuffer - single producer / single consumer

Cooperative Multithreading

1 поток может выполнять ~1000 tasklet
Если нечего делать, тред
Ребра графа имплементированы с помощью RingBuffer - single producer / single consumer

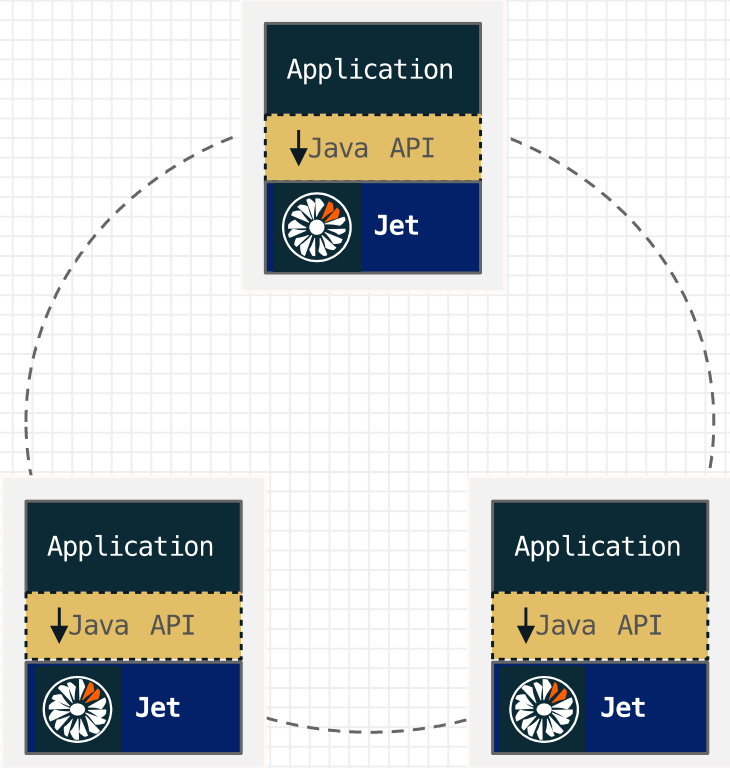


Топологии

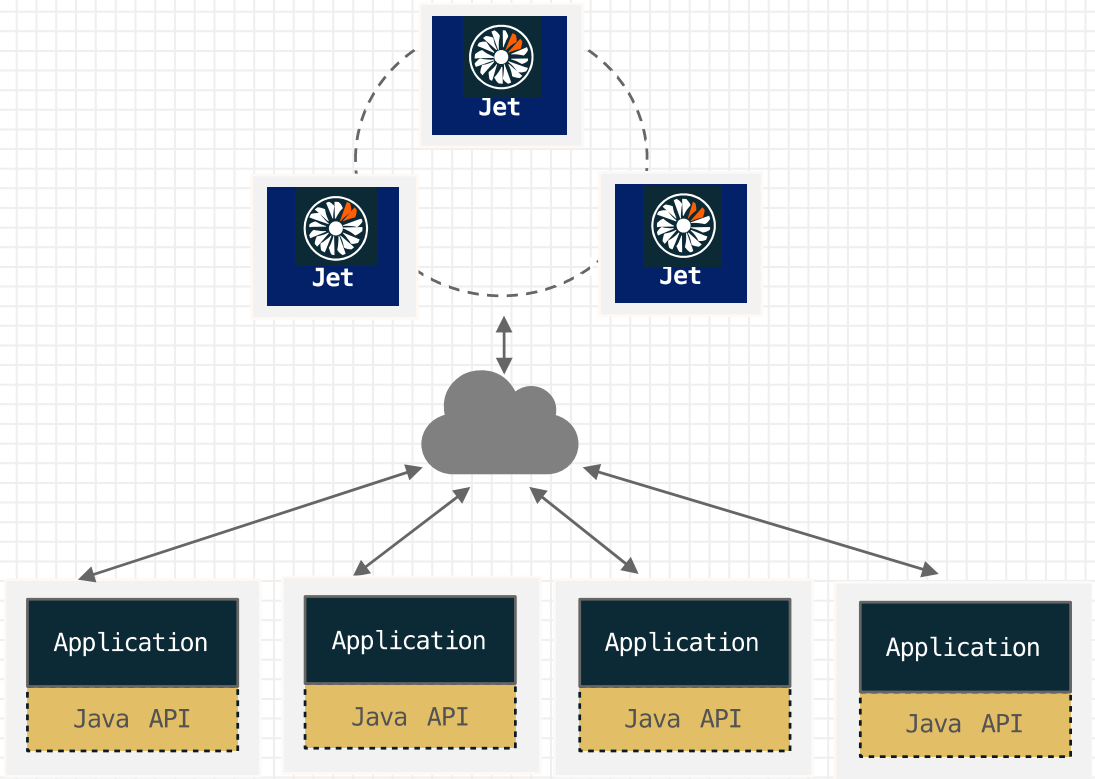
Что нам стоит кластер построить

Топологии

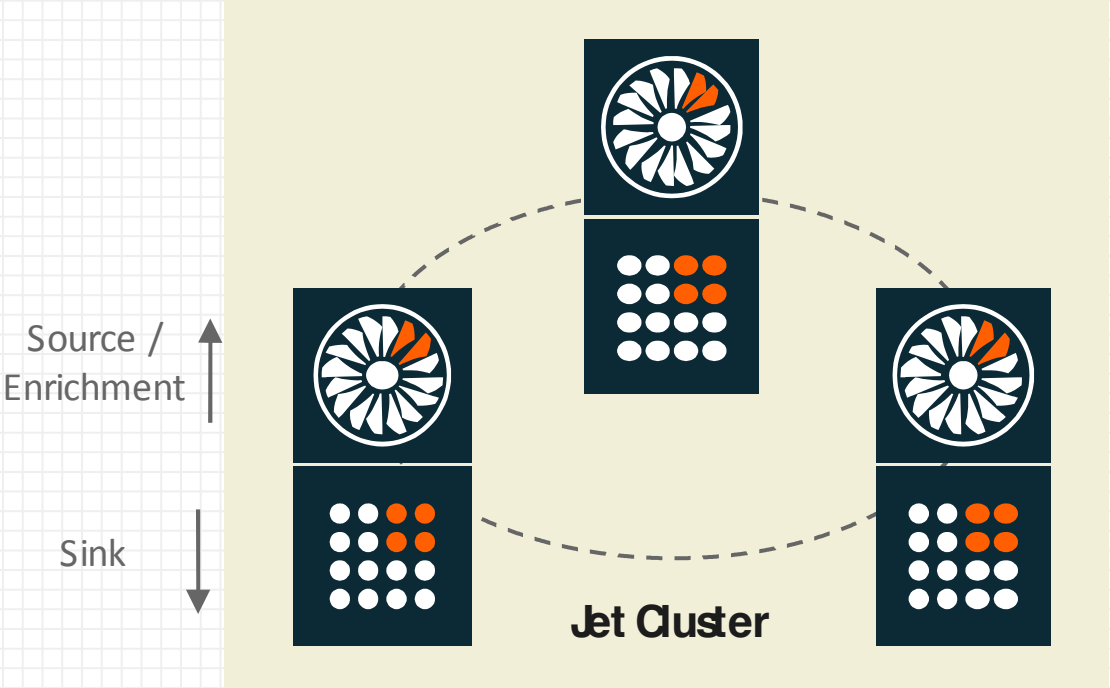
Топологии


Application
Java API
Jet
Application
Java API
Jet
Application
Java API
Jet

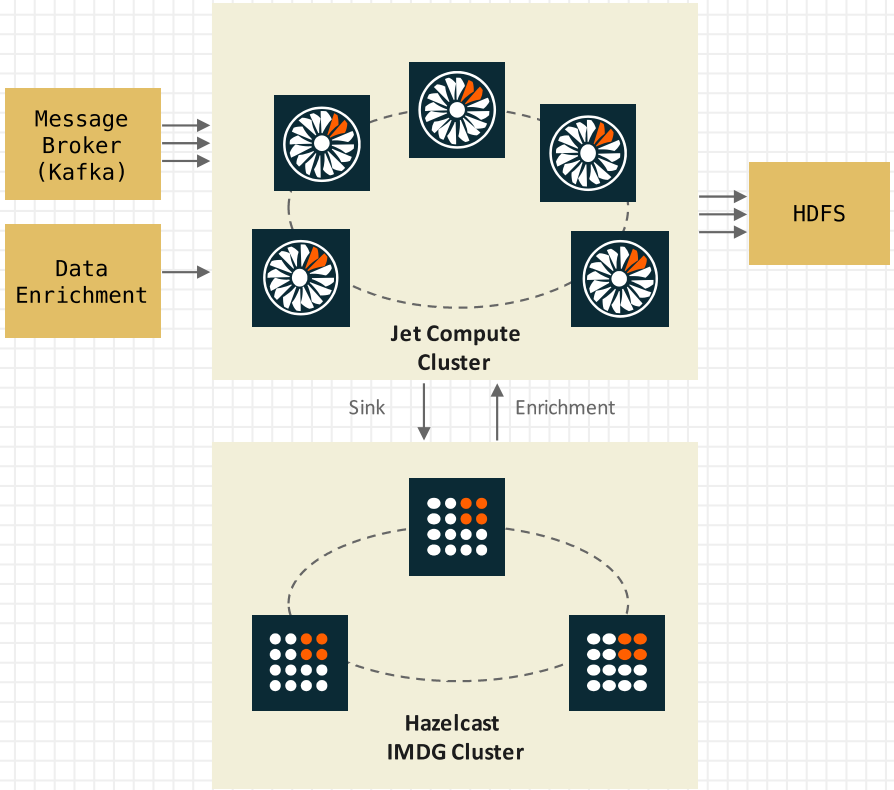
Топологии

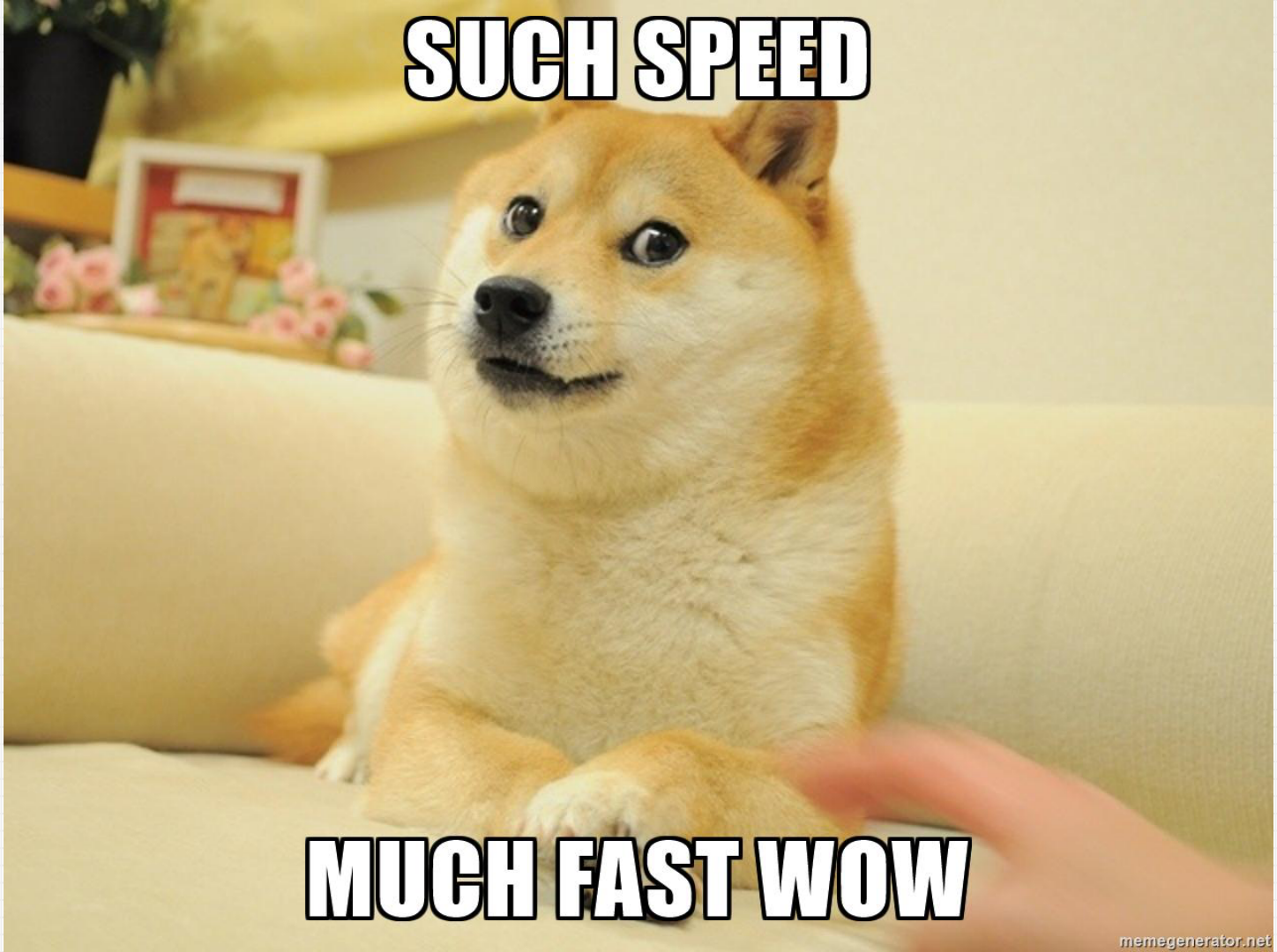

Jet
Jet
Jet
Application
Java API
Application
Java API
Application
Java API
Application
Java API

Топологии

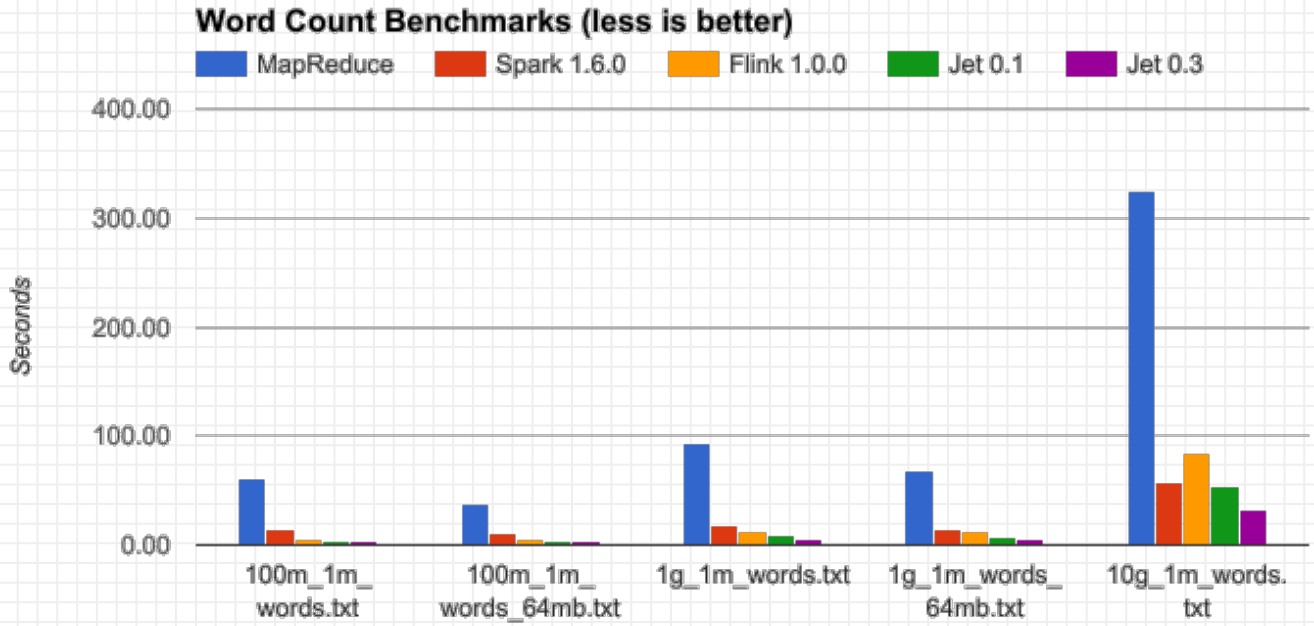


Топологии

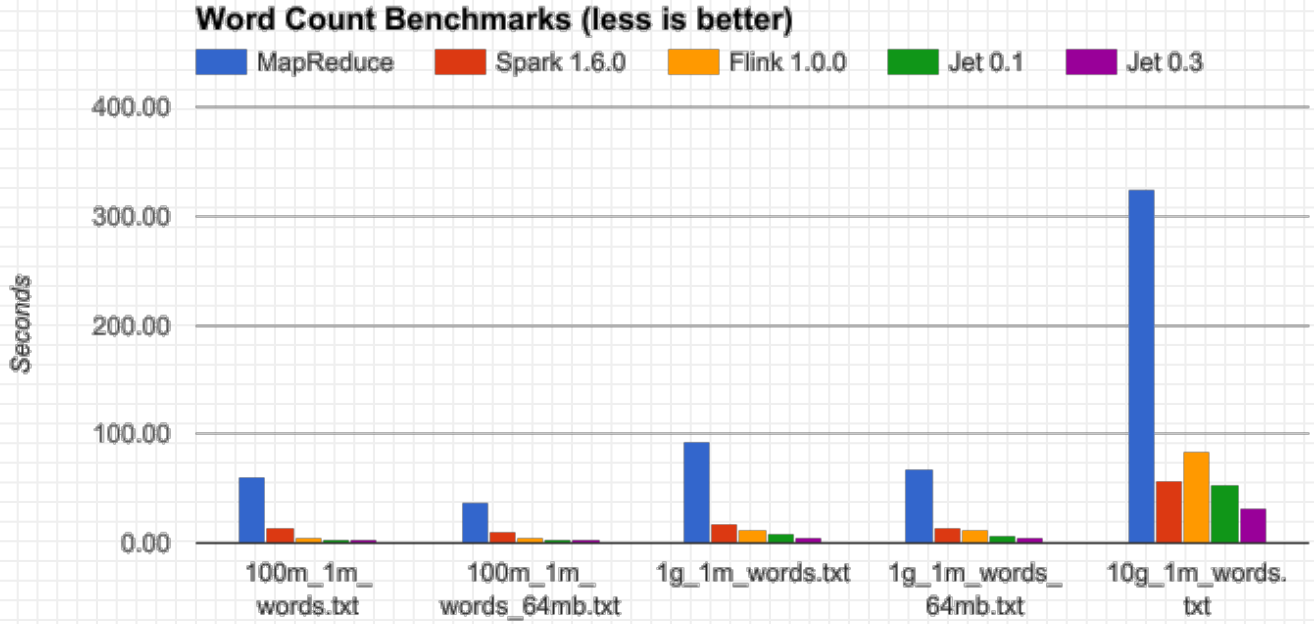

Message Broker (Kafka)
Data Enrichment
HDFS
Jet Compute Cluster
Sink
Enrichment
Hazelcast IMDG Cluster

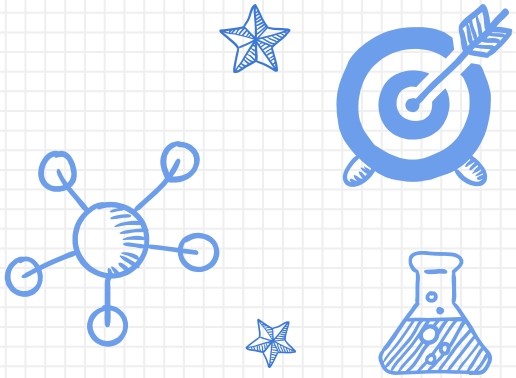

SUCH SPEED
MUCH FAST WOW
memegenerator.net

Benchmarks


Word Count Benchmarks (less is better)
MapReduce
Spark 1.6.0
Flink 1.0.0
Jet 0.1
Jet 0.3
400.00
300.00
200.00
100.00
0.00
Seconds
100m_1m_words.txt
100m_1m_words_64mb.txt
1g_1m_words.txt
1g_1m_words_64mb.txt
10g_1m_words.txt

Benchmarks


Word Count Benchmarks (less is better)
MapReduce
Spark 1.6.0
Flink 1.0.0
Jet 0.1
Jet 0.3
400.00
300.00
200.00
100.00
0.00
Seconds
100m_1m_words.txt
100m_1m_words_64mb.txt
1g_1m_words.txt
1g_1m_words_64mb.txt
10g_1m_words.txt



DEMO

Disclaimer: Нам пишут

* авторская орфография сохранена

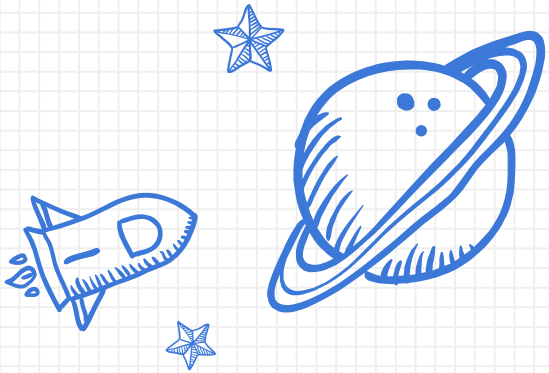
Disclaimer: Нам пишут

? Про фэйлы нод в **хазелкасие** хотелось бы услышать откровений и примеры из жизни.

* авторская орфография сохранена

Disclaimer: Нам пишут

? Про ф
хазел
бы усл
откров
приме



ПРОБЛЕМЫ

Текущее состояние

@gamussa @hazelcast #jpoint

Проблемы

Проблемы

Устойчивость к сбоям

Проблемы

Устойчивость к сбоям
Работа с «бесконечными» данными

Проблемы

Устойчивость к сбоям
Работа с «бесконечными» данными
Неупорядоченные и «опоздавшие» события


I FOUND YOUR LACK OF FAULT TOLERANCE
disturbing

Бэкапы

Консистентный бэкап системы
Обработка «At-least once» vs «Exactly once»
Снэпшот распределенной системы

Бэкапы

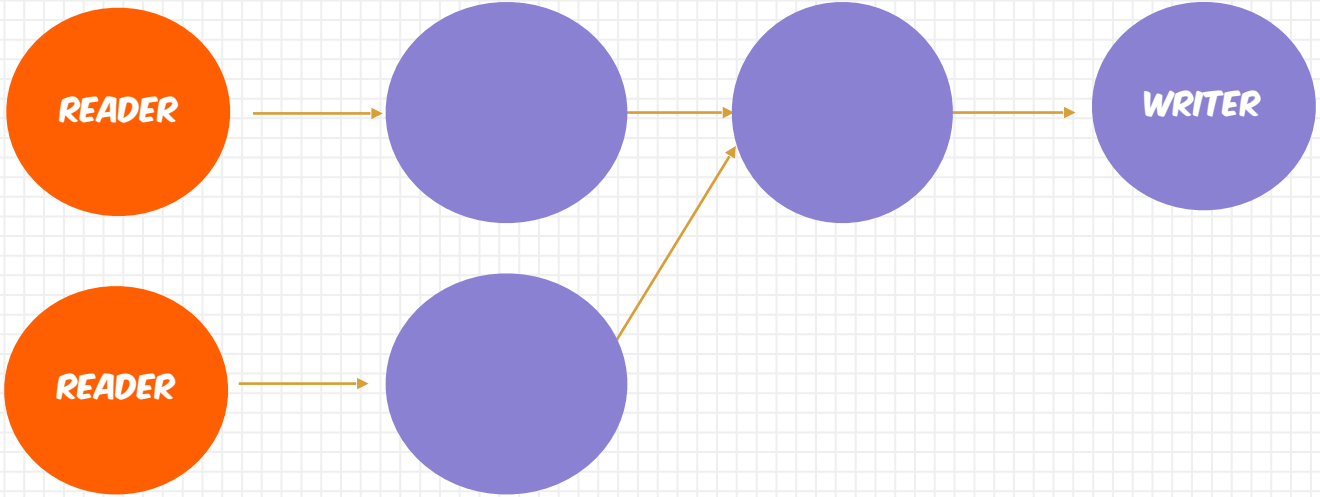
Консистентный бэкап системы
Обработка «At-least once» vs «Exactly once»
Снэпшот распределенной системы

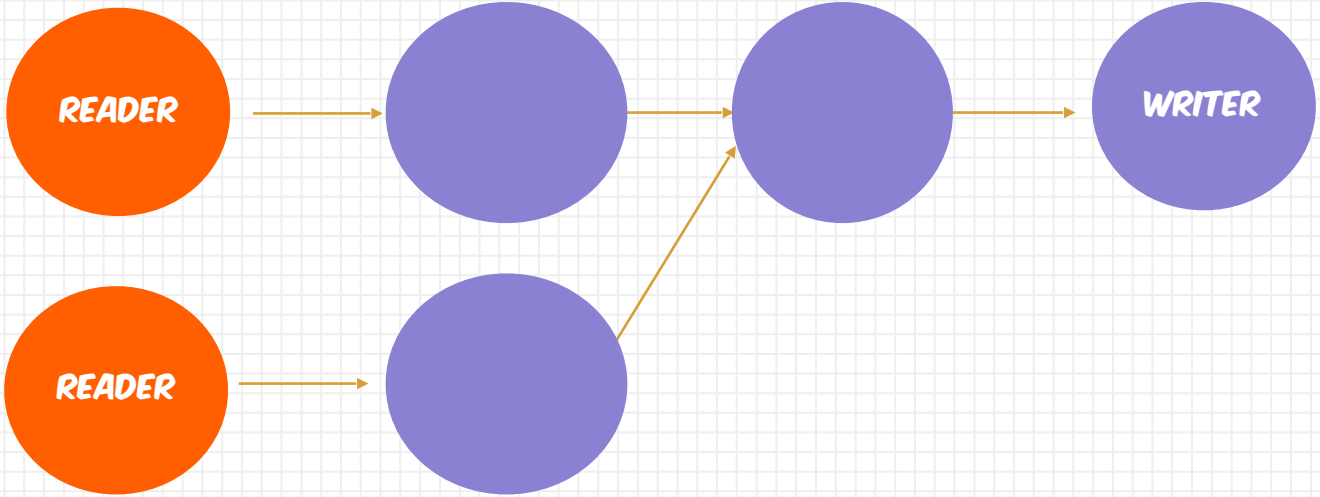
Бэкапы

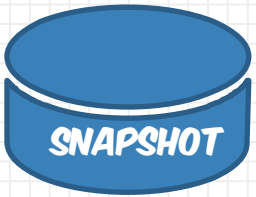
Консистентный бэкап системы
Обработка «At-least once» vs «Exactly once»
Снэпшот распределенной системы

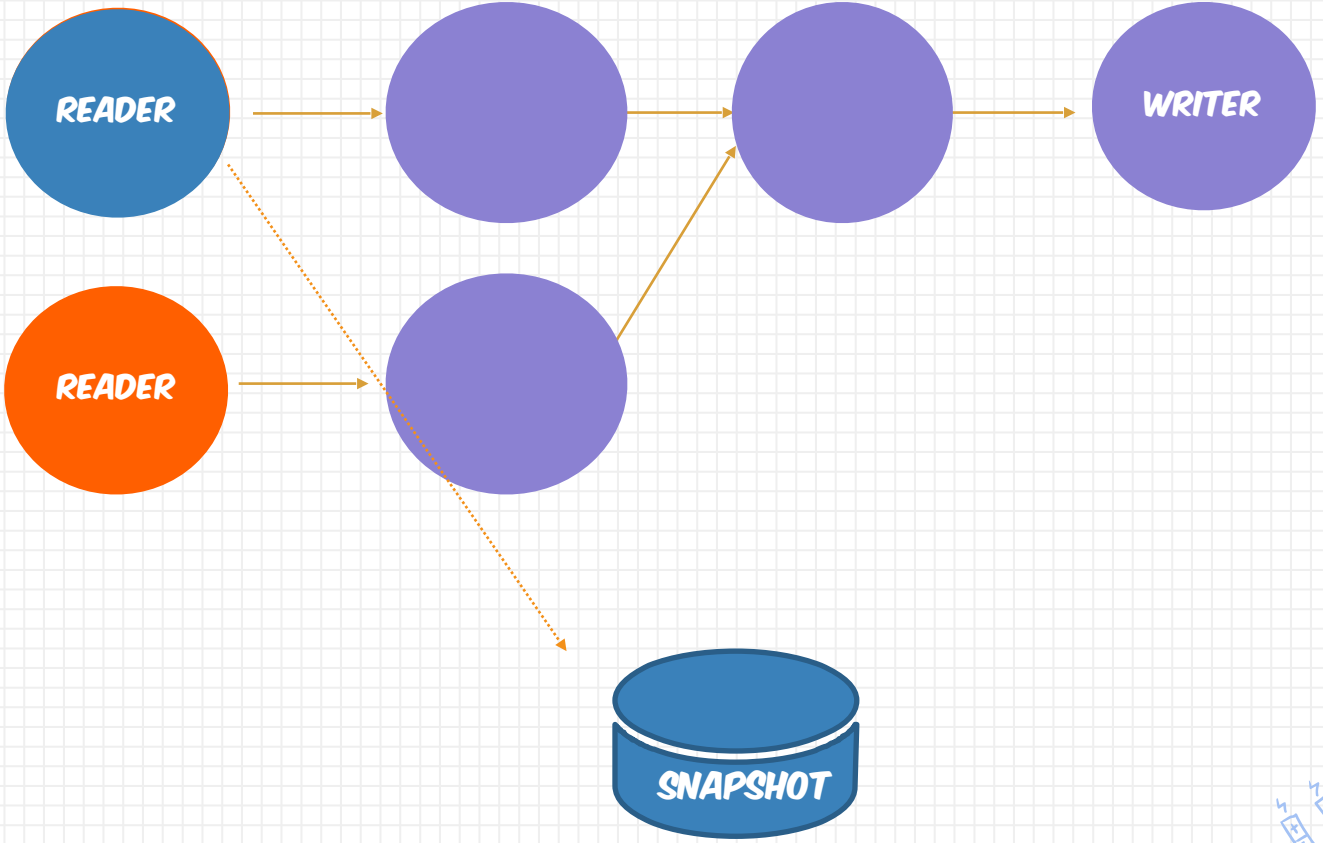
Бэкапы

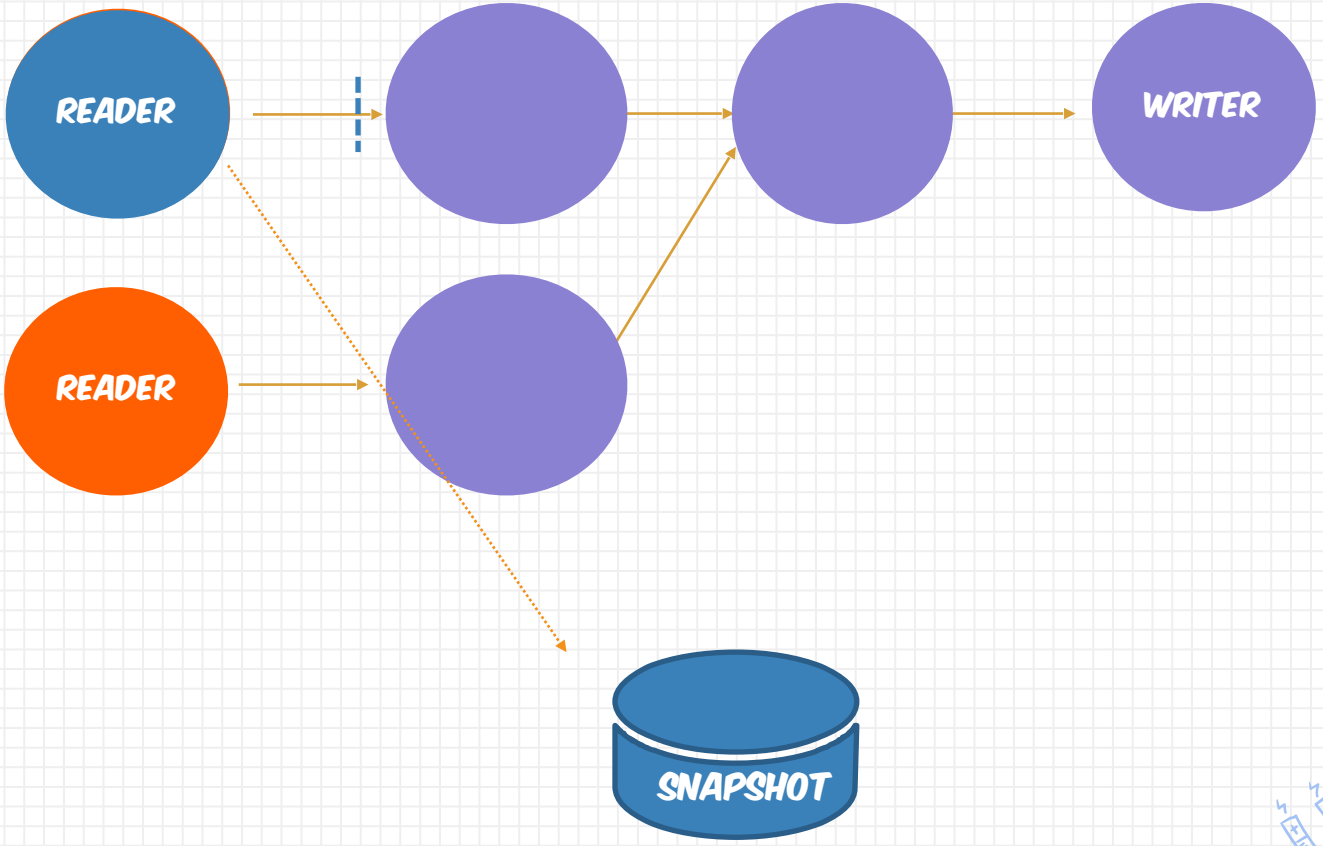
Консистентный бэкап системы
Обработка «At-least once» vs «Exactly once»
Снэпшот распределенной системы

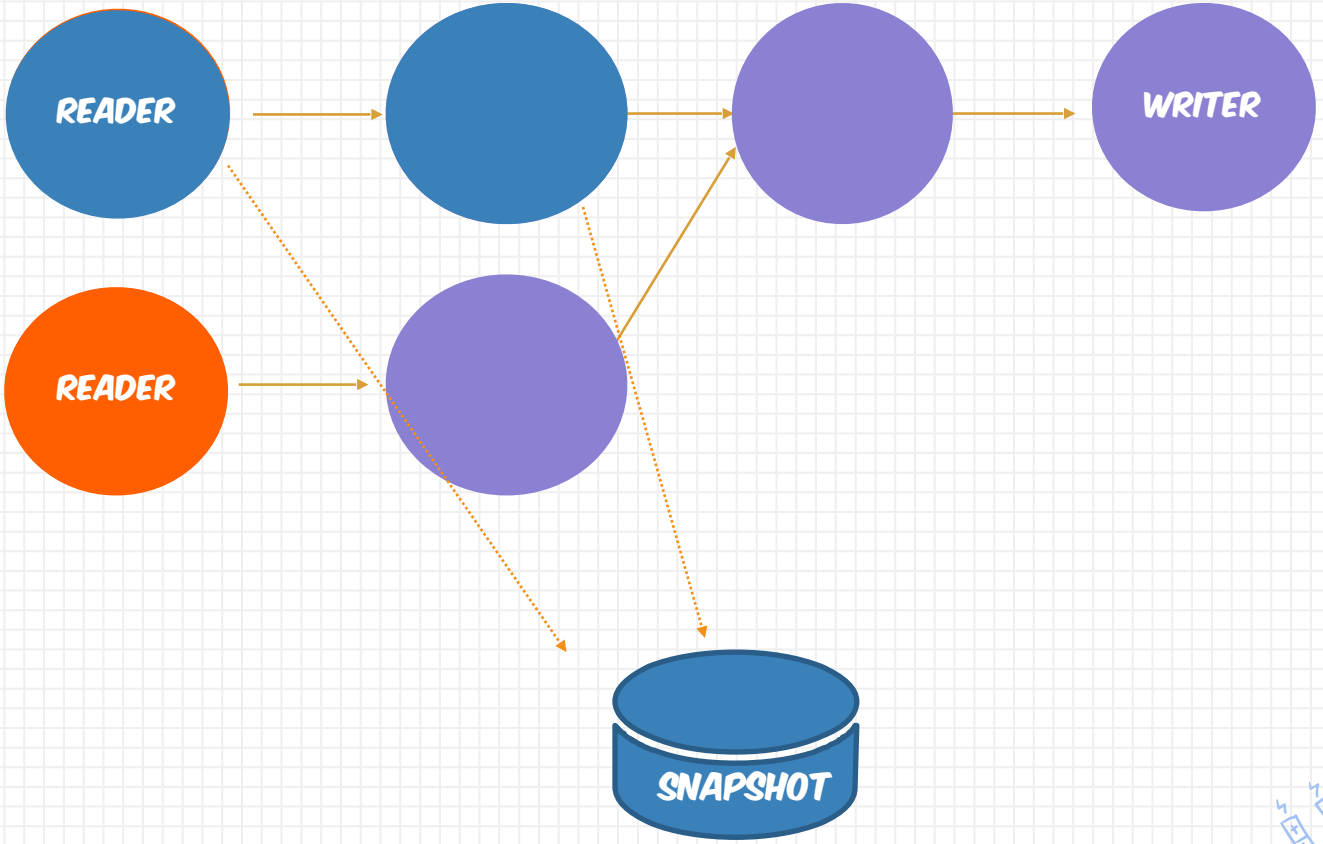

READER
READER
WRITER

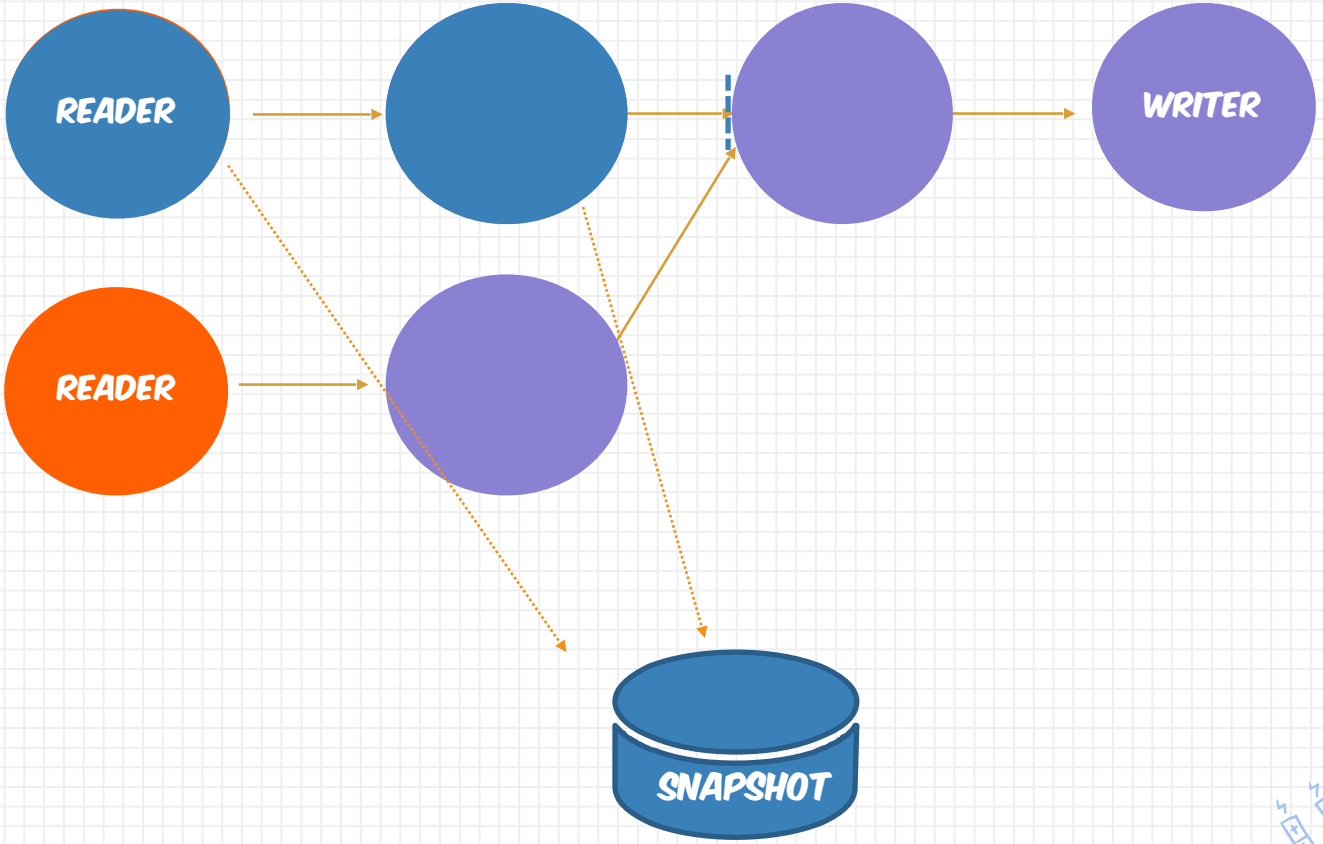

READER
READER
WRITER

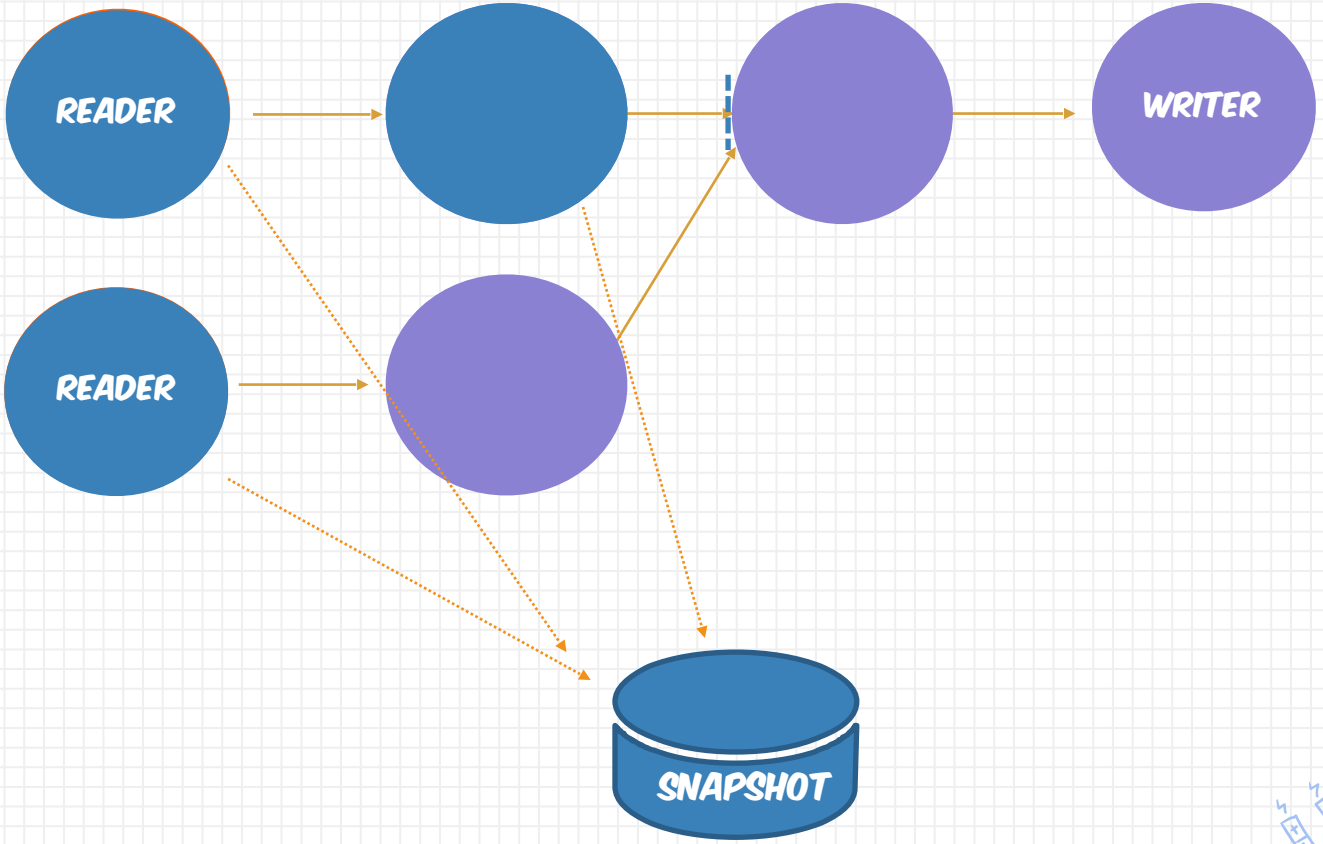

SNAPSHOT

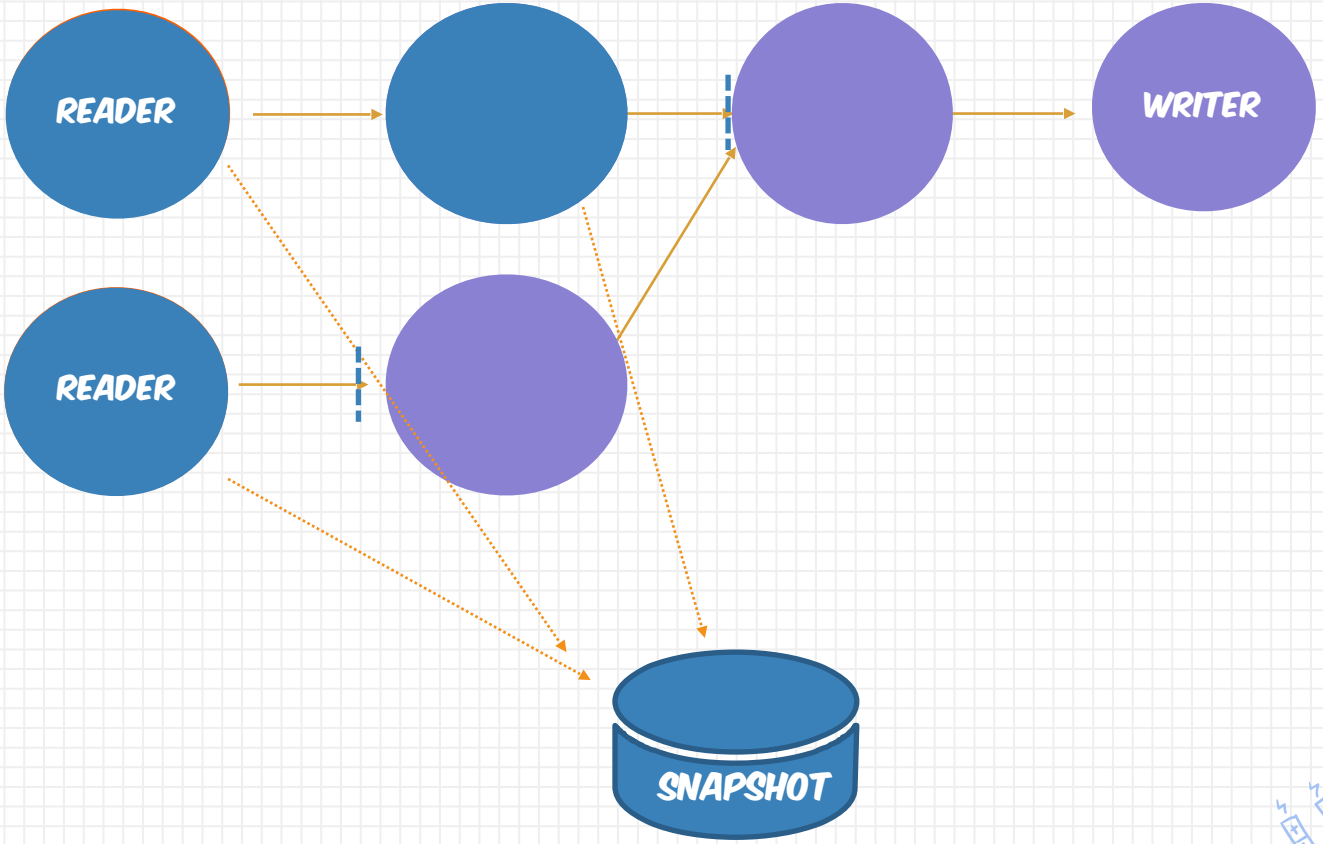

READER
READER
WRITER
SNAPSHOT

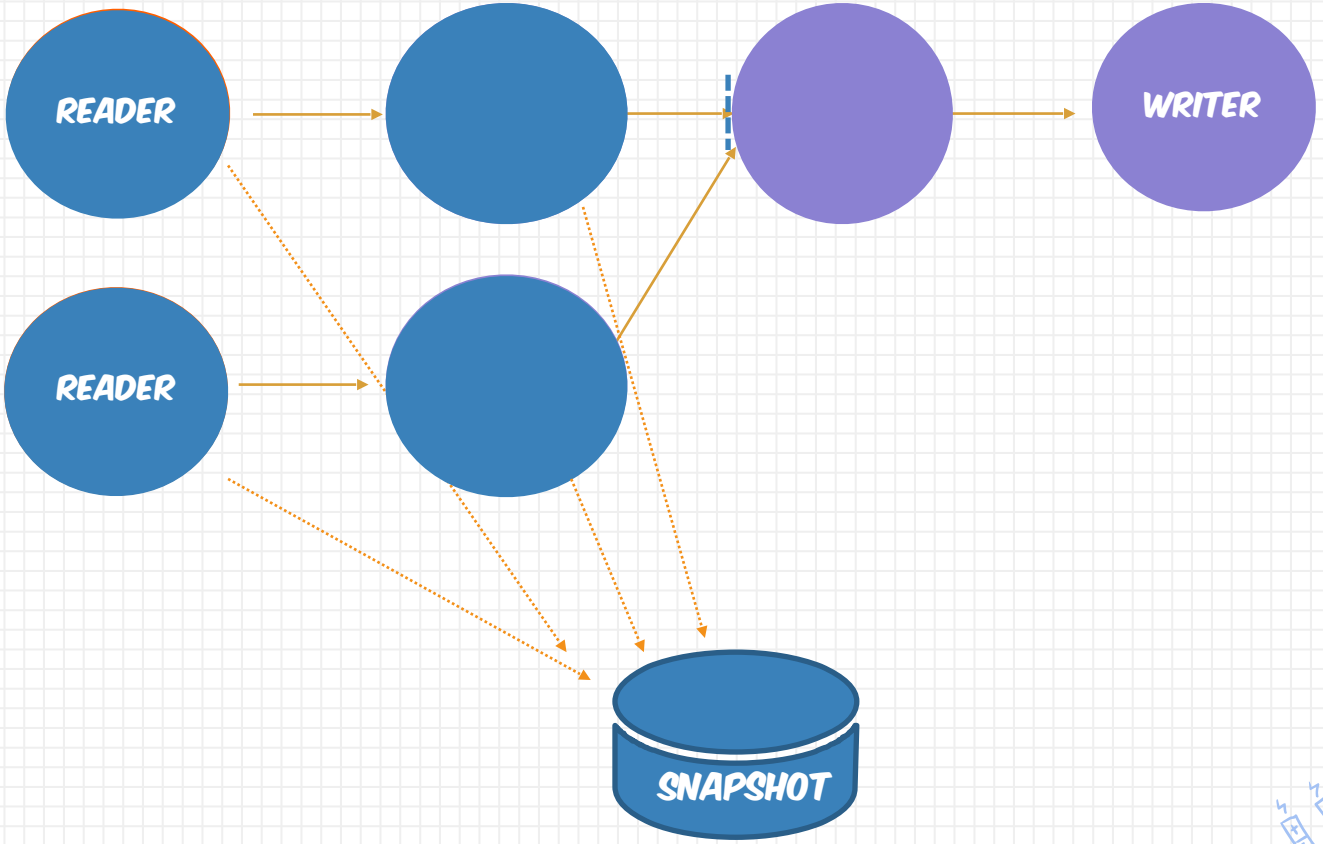

READER
READER
WRITER
SNAPSHOT

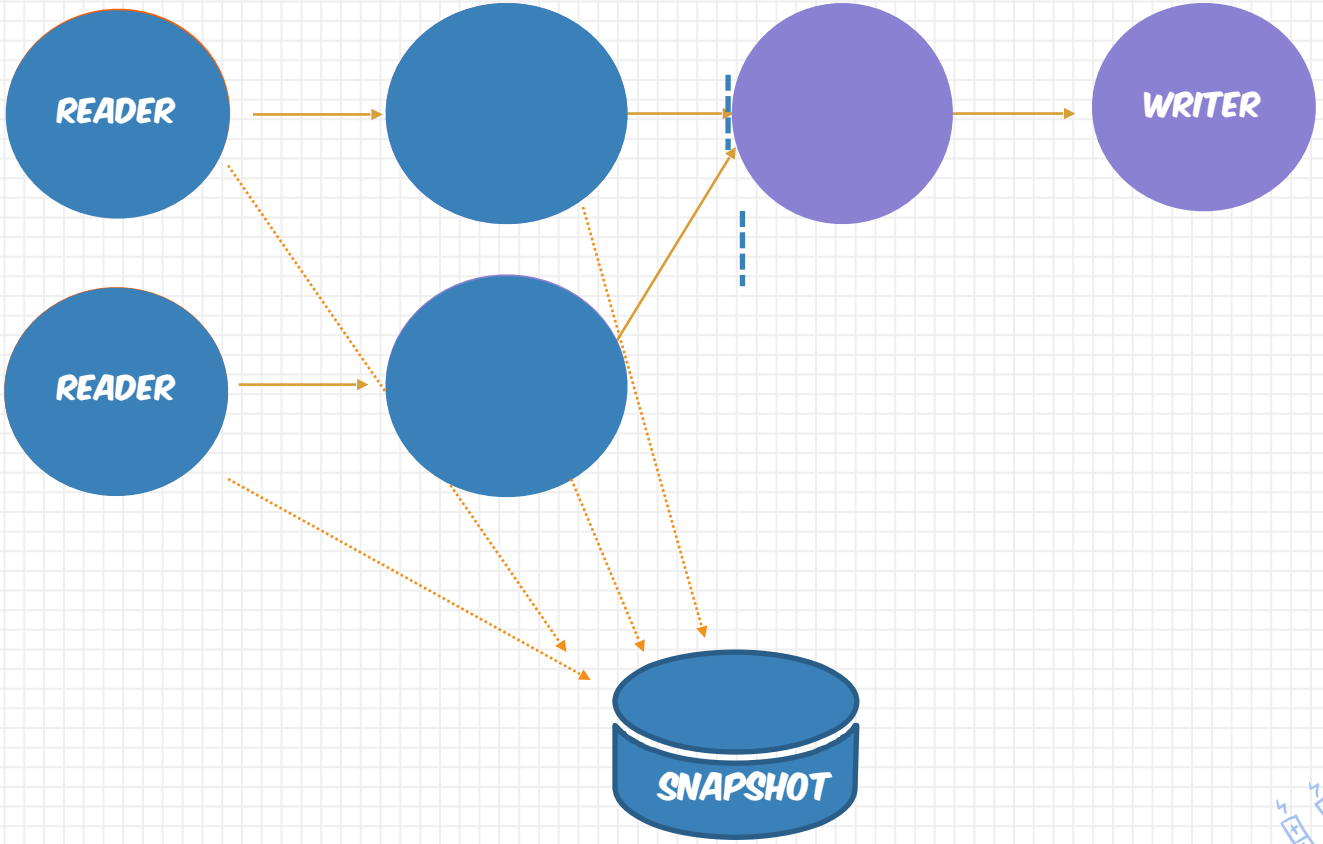

READER
READER
WRITER
SNAPSHOT

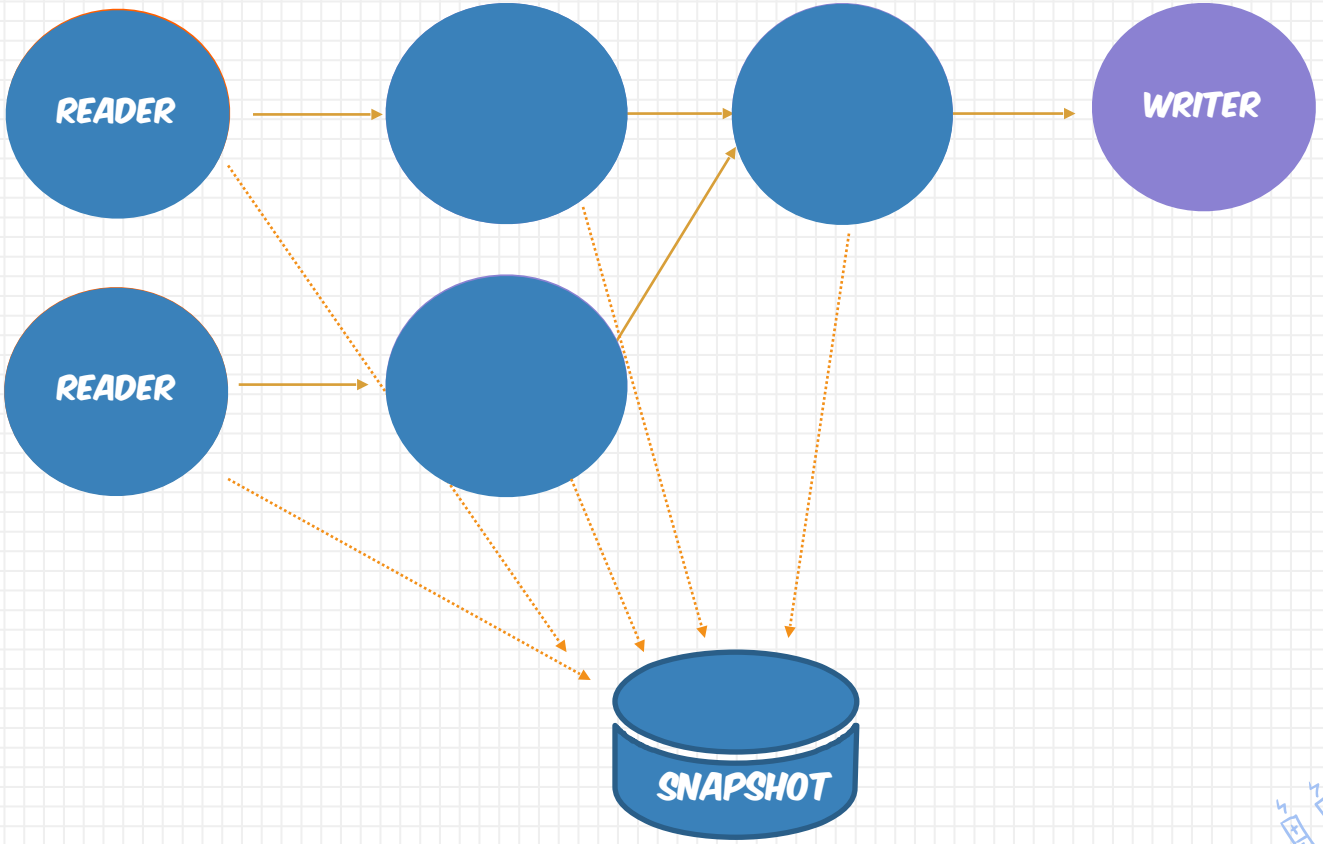

READER
READER
WRITER
SNAPSHOT

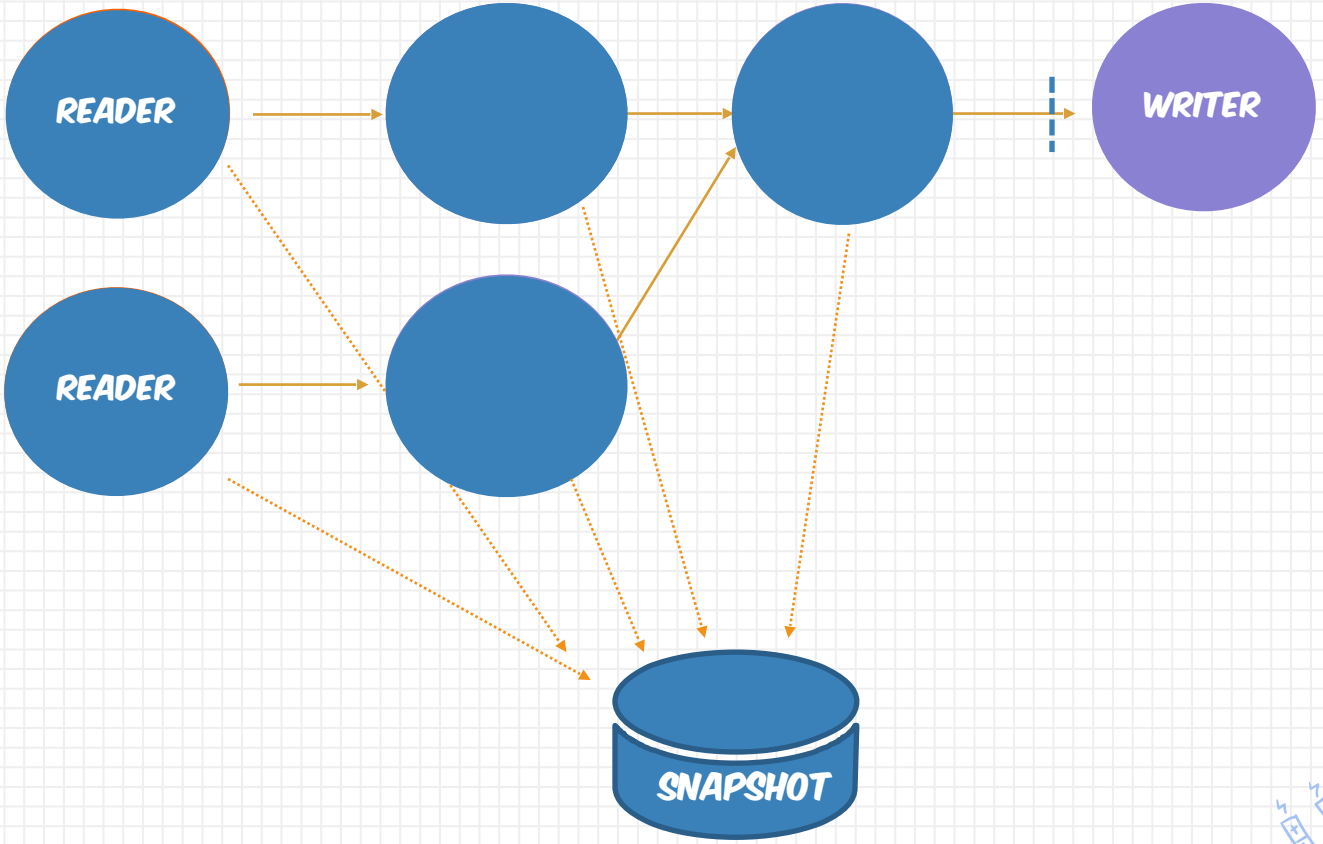

READER
READER
WRITER
SNAPSHOT

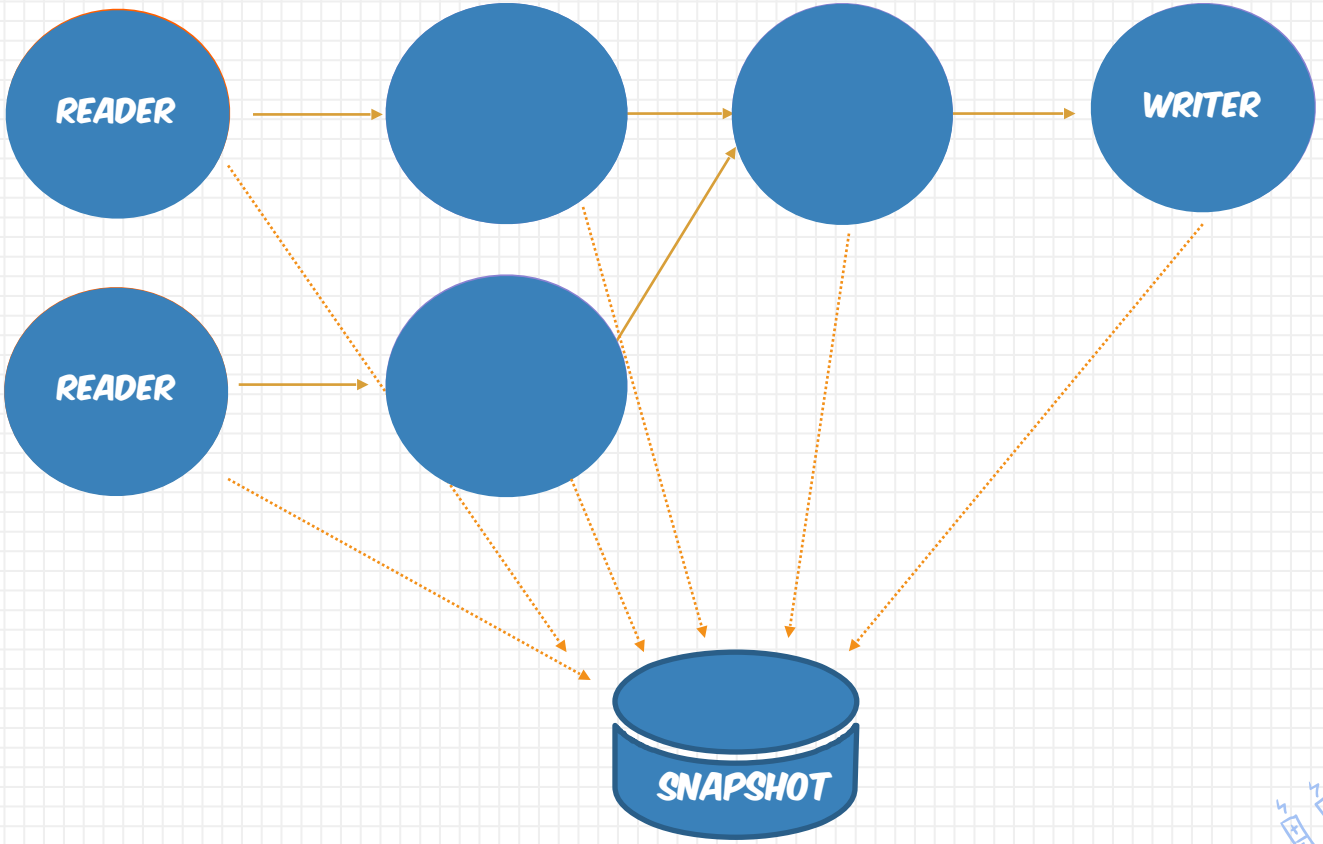

READER
READER
WRITER
SNAPSHOT

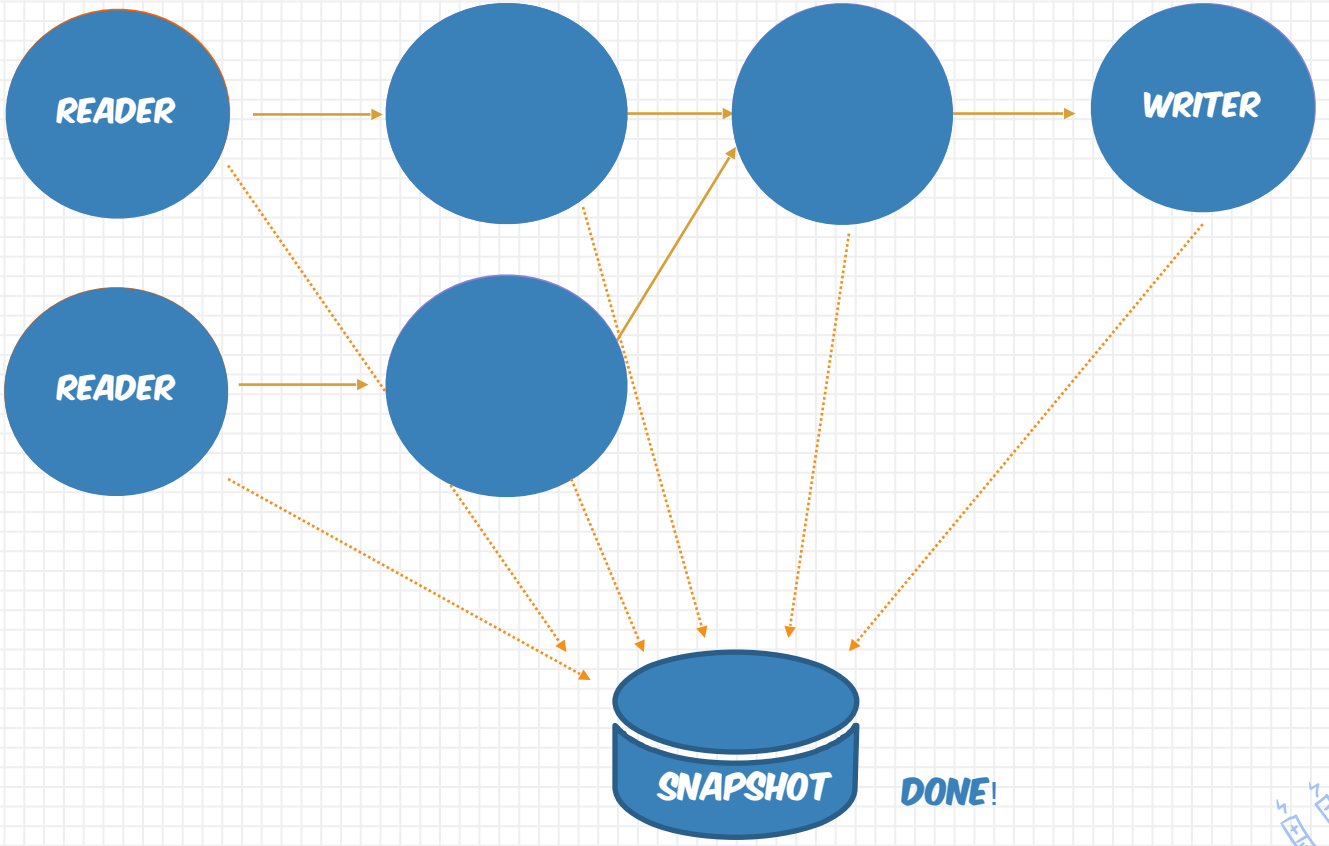

READER
READER
WRITER
SNAPSHOT


READER
READER
WRITER
SNAPSHOT


READER
READER
WRITER
SNAPSHOT


READER
READER
WRITER
SNAPSHOT

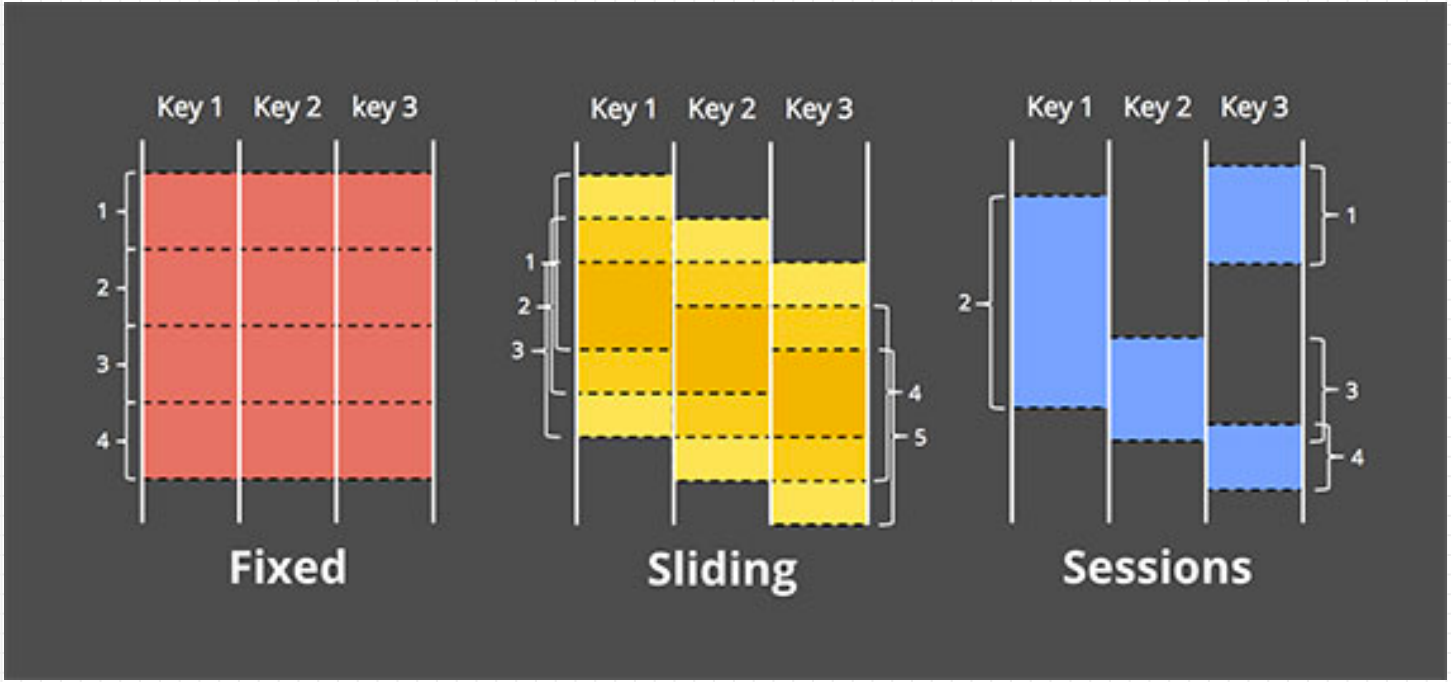

READER
READER
WRITER
SNAPSHOT


READER
READER
WRITER
SNAPSHOT
DONE!

Как считать «бесконечные» данные?

Конечное представление бесконечных данных


ОКНА



https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101

Time-based обработка

Time-based обработка

Привязка записей к окнам на основе

- Времени события
- Времени обработки

Time-based обработка

Привязка записей к окнам на основе

- Времени события
- Времени обработки

Сколько ждать «запоздалых» данных?

Time-based обработка

Привязка записей к окнам на основе

- Времени события
- Времени обработки

Сколько ждать «запоздалых» данных?

Точность vs Память

FATALITY


BATTLE SELECT
gAmUssA85
LVL 51
105
92,581
250
STORE
FIGHT 1/3
gAmUssA85
LVL 51
hugo
LVL 48
II
Ninjutsu
SCORPION
50
11K
20K
I
Cryomancer
SUB-ZERO
25
6,008
8,904
I
Undead Hunter
JOHNNY CAGE
22
6,762
8,791
VS
VI
KENSHI
50
11K
21K
V
SCORPION
49
8,593
14K
Kraken
REPTILE
10
2,451
3,172
FIGHT

Потоковая обработка: итоги

Потоковая обработка: итоги

- Получать результаты вычислений **реальном времени** возможно!

Потоковая обработка: итоги

- Получать результаты вычислений **реальном времени** возможно!
- **Окна** – конечное представление бесконечных данных
- Окна основаны на временнЫх параметрах (время события + время обработки)

Потоковая обработка: итоги

- Получать результаты вычислений **реальном времени** возможно!
- **Окна** – конечное представление бесконечных данных
  - Окна основаны на временнЫх параметрах (время события + время обработки)
- Обработка «запоздалых» событий
  - Вам решать, сколько ждать

hazelcast/hazelcast-jet-code-samples

СПАСИБО!

Вопросы?

@gamussa
viktor@hazelcast.com