Предмет курса.
Зачем нам вообще параллельная работа программ ?

Что является предметом данного курса.

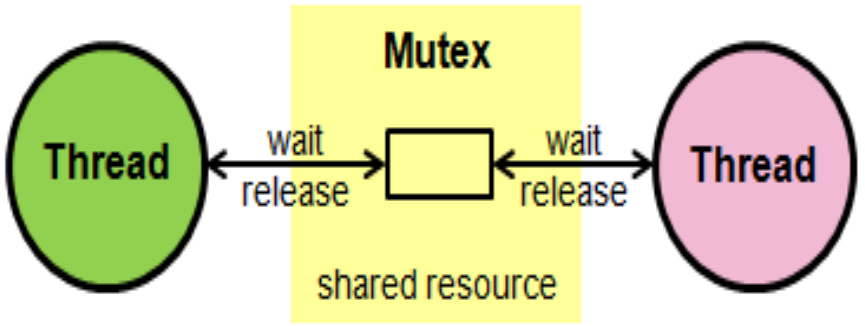
- Подходы к решению проблем в современных высоконагруженных системах
- Внутренее устройство и архитектура ряда решений

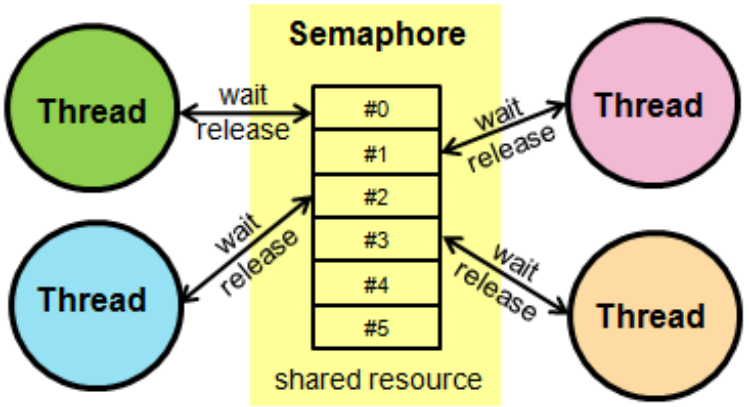
Что не является предметом данного курса

- Математическая база разработки параллельных программ
- Администрирование и развертывание рассматриваемых систем и решений

Классическое программирование параллельных программ

- Семафоры
- Мьютексы
- Атомарные операции типа get and replace
- Транзакции





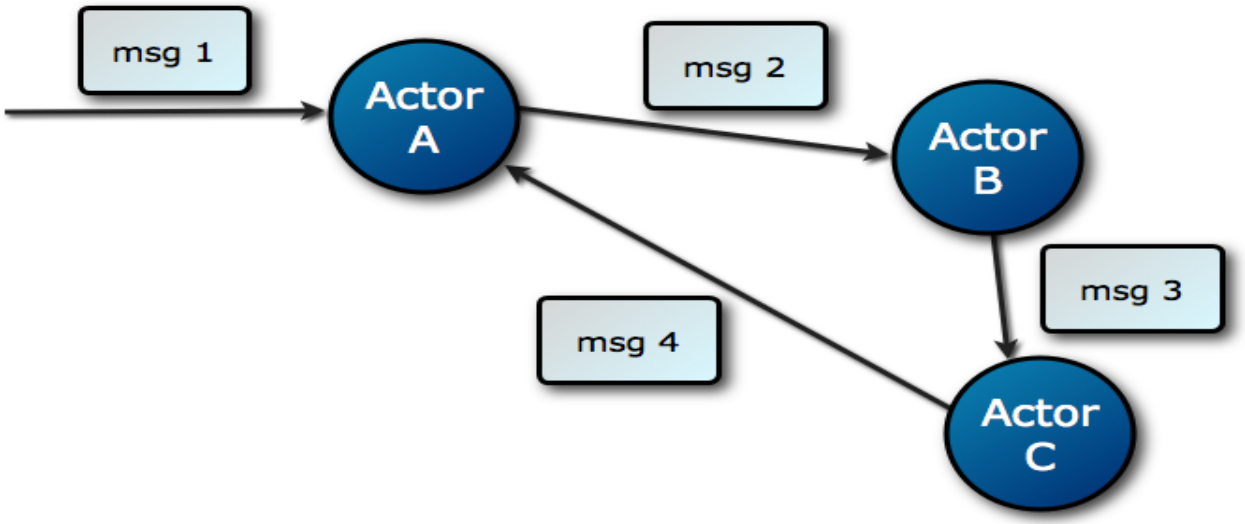
Плюсы и минусы

- Плюсы
- Высокая эффективность (api достаточно близко к физическому уровню)
- Хорошо укладывается в привычные програмные архитектуры и решения
- Легкость при переходе от однопоточной модели к многопоточной

- Минусы
- Сложность разработки (надо предусмотреть все возможные варианты)
- Легко поймать Dead lock или Race conditions
- Сложность масштабирования

“Actor based” подход

- Программа состоит из атомарных единиц – actors
- Каждый actor может посылать другим actor-ам сообщения и принимать их
- Каждый actor может создавать и удалять других actor-ов



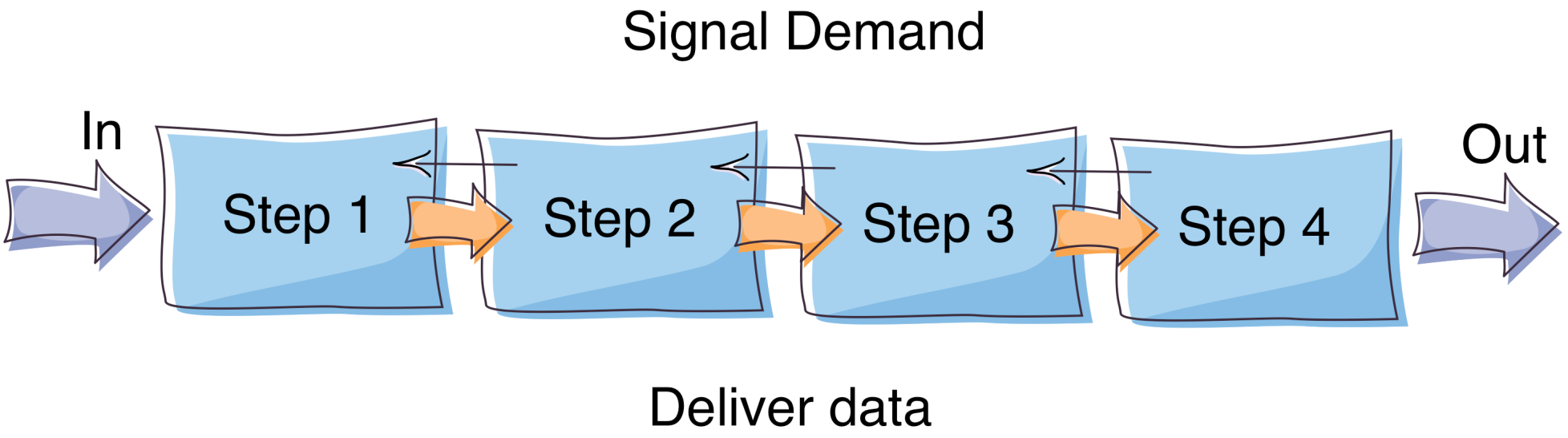
Плюсы и минусы

- Плюсы
- Приложение мало подвержено проблемам с dead locks и race conditions
- Легко масштабируется

- Минусы
- Требует специфических framework которых нет в классических языках и средах программирования
- Требует изменения привычной архитектуры приложения

Потоки данных (Streams)

- Программа состоит из графа по которому двигаются сообщения.
- В узлах графа происходит обработка.
- Backpressure обеспечивает автобалансировку нагрузки и обработки



Плюсы и минусы

- Плюсы
  - Backpressure позволяет защититься от ситуации когда на вход поступает больше данных чем может обработать приложение
  - Устойчивость к race conditions.
  - Легкость масштабирования
- Минусы
  - Необходимость использования во всем приложении одного специфического framework-а
  - Ограничения на масштабирование (неготовность к масштабированию на 1000 узлов)
  - Сложность развертывания

Coroutines, Fibers, Channels

- Программа состоит из нескольких зацикленных функций (corouines, fibers) обменивающихся данными через очереди (channels).
- В момент обращения к очереди на чтение - “паркуется” контекст функции и если данных нет, то поток переключается.
- Переключение потоков осуществляется с помощью “зеленых” потоков, реализованых поверх системных.

Плюсы и минусы

- Плюсы
  - Легкая и понятная модель разработки
  - Достаточная для многих задач степен масштабируемости
  - Высокая производительность
- Минусы
  - Сложность динамической балансировки нагрузки
  - Сложность масштабируемости в распределенном кластере

Классические распределенные системы

- Distributed.net
- seti@home
- TERRA ONE
- ИТ.Д

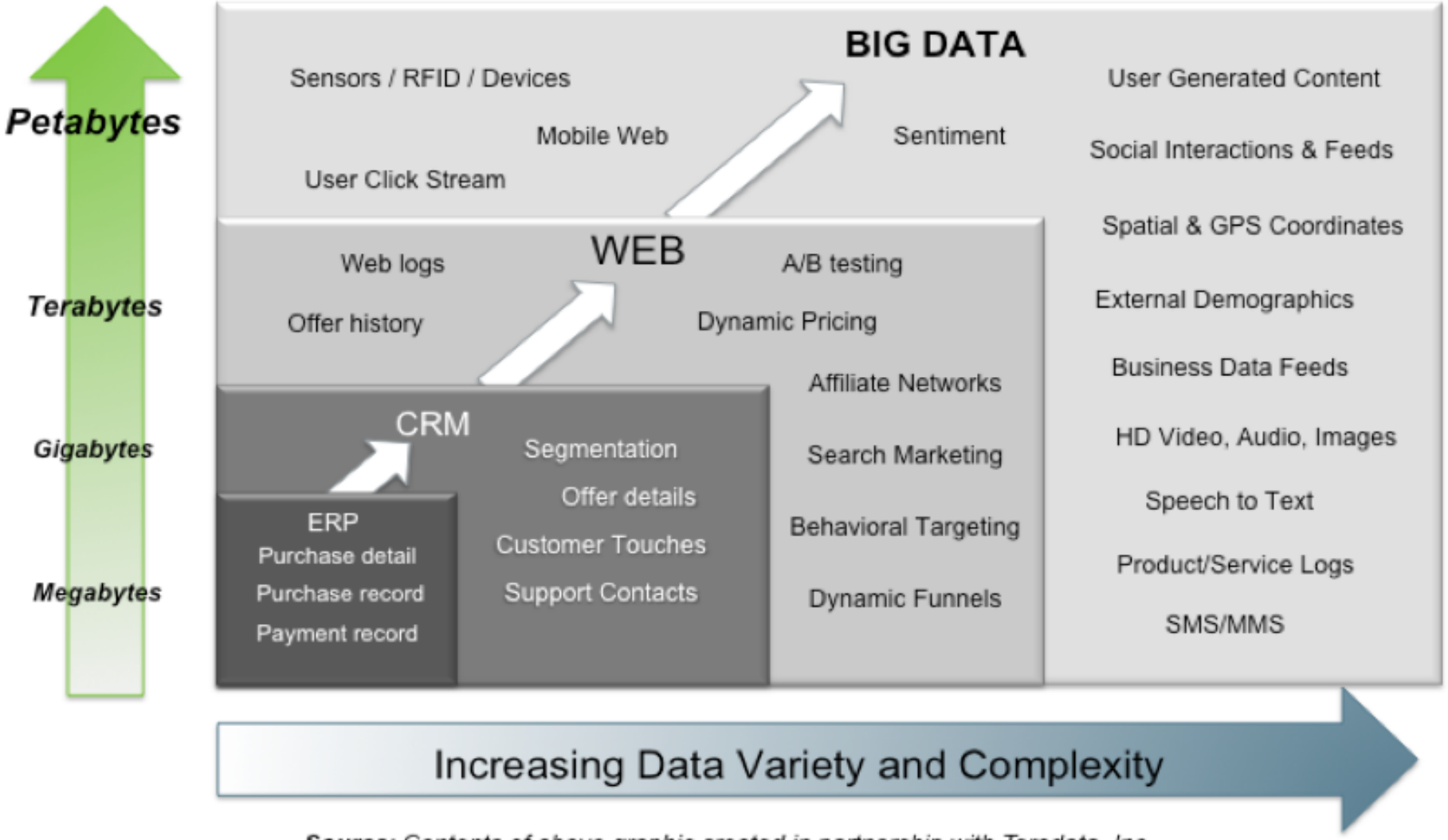
Особенности классического подхода

- Используются низкоуровневые протоколы (например MPI)
- Изначально ориентированы на задачи с небольшим объемом данных (физические и математические расчеты).
- Трудоемкая разработка

BIG Data !

- Концепция была представлена Google в 2006
- Позволяет обрабатывать и хранить петабайты данных
- Основана на нескольких ключевых технологиях:
- Распределенное хранилище файлов
- Big table
- Map/reduce

Big Data = Transactions + Interactions + Observations



***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

Распределенное хранилище файлов в big data

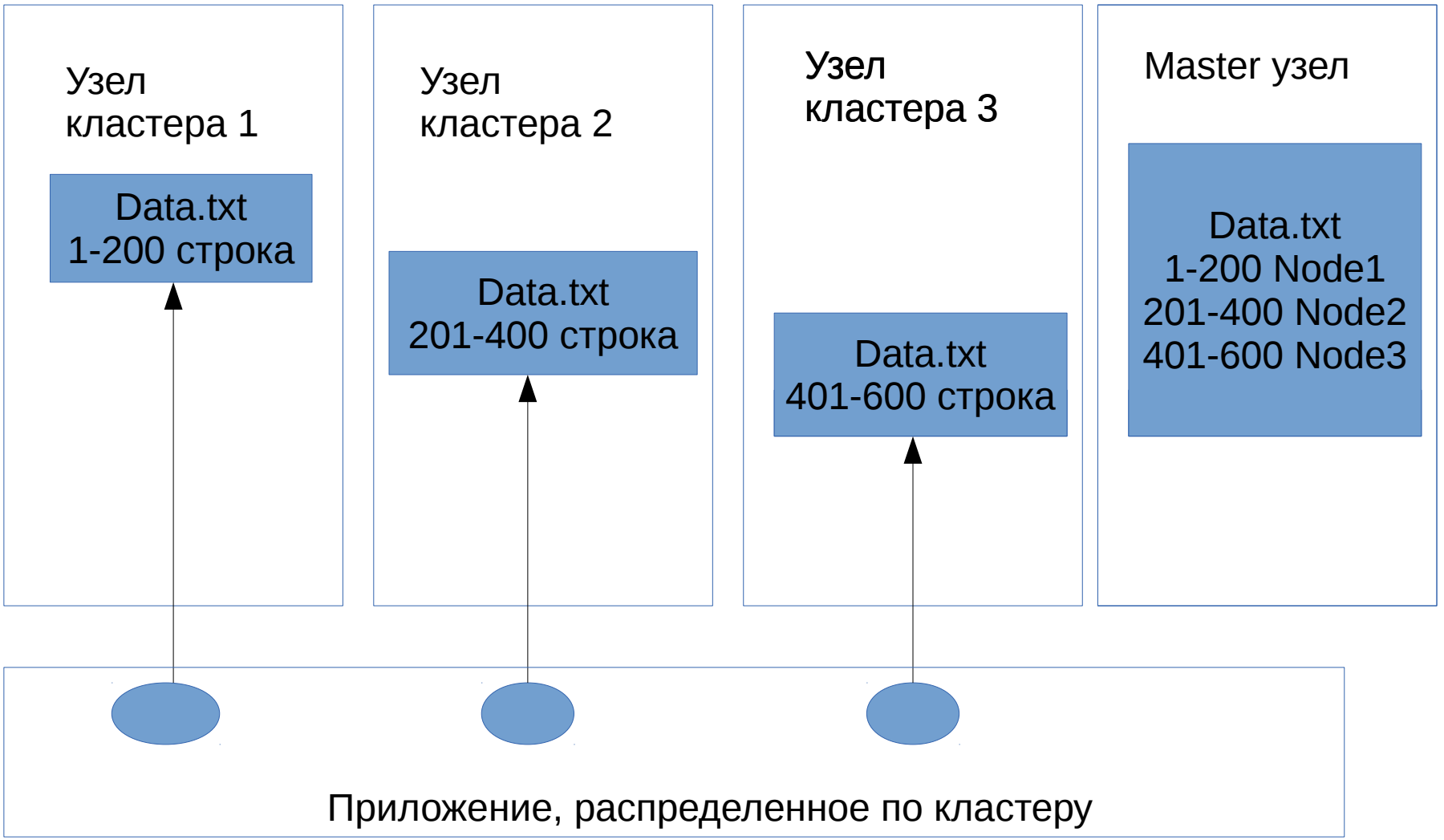
- Какие задачи оно решает ?

- Предназначено для хренения ОЧЕНЬ больших файлов (терабайты и петабайты)
- Параллельная поточная обработка файла
- Должно работать на недорогом “железе”

Ключевые идеи хранилища файлов в big data решении

- Файл должен создаваться один раз, дописывать можно только в конец файла.
- write-once, read-many-times
- Файл бьется на блоки.
- Блоки “размазываются” по кластеру
- Для достижения отказоустойчивости каждый блок имеет резервные копии

На что это похоже



Big Table

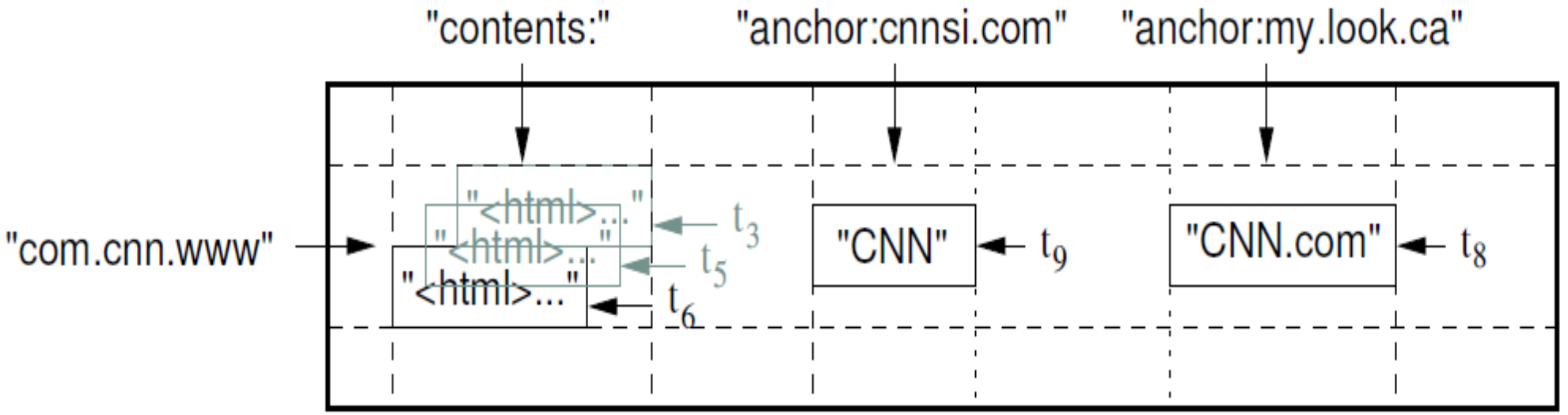
- Какие задачи решает ?

- Хранение ОЧЕНЬ больших объемов структурированной информации
- Миллионы колонок, миллиарды строк

Ключевые идеи Big Table

- NO SQL Database.
- Множество записей таблицы бьется на регионы.
- Регионы распределяются по кластеру
- Запись состоит из ключа и колонок
- Колонки группируются по группам колонок (Column families)
- Каждая ячейка (row key+column) имеет версии

На что это похоже



Map/reduce

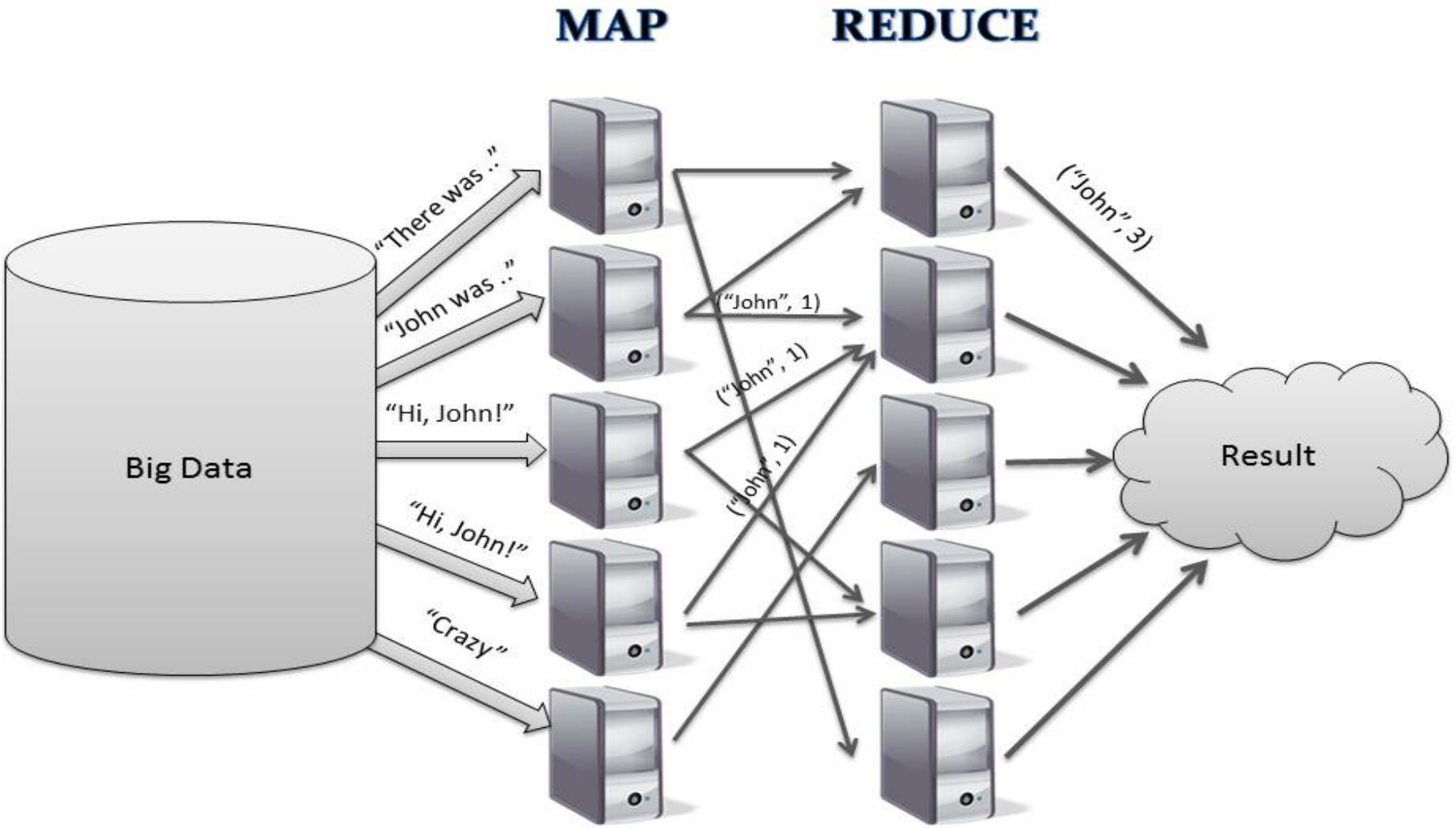
- Какие задачи решает ?

- Параллельная обработка ОЧЕНЬ больших объемов информации
- Параллельность “Из коробки”
- Простой инструментарий программиста

Ключевые идеи Map/reduce

- Программист пишет две функции – map и reduce
- Map применяется к каждой строке распределенного файла (или записи big table) и преобразует ее в пару key:value
- Reduce применяется к каждому набору данных key:[value1, value2] и преобразует ее в значение value3

На что это похоже



Realtime Big Data

- Совмещение концепции actor based параллелизма с динамическим развертыванием и управлением задачами BigData
- Программист разрабатывает топологию состоящую из actor-ов обменивающихся сообщениями в рамках потока данных
- Топология динамически запускается на кластере Realtime Big Data на произвольном количестве равноправных узлов
- Один из самых известных проектов – Storm (разработан для twitter)

На что это похоже