YT — новая платформа распределенных вычислений

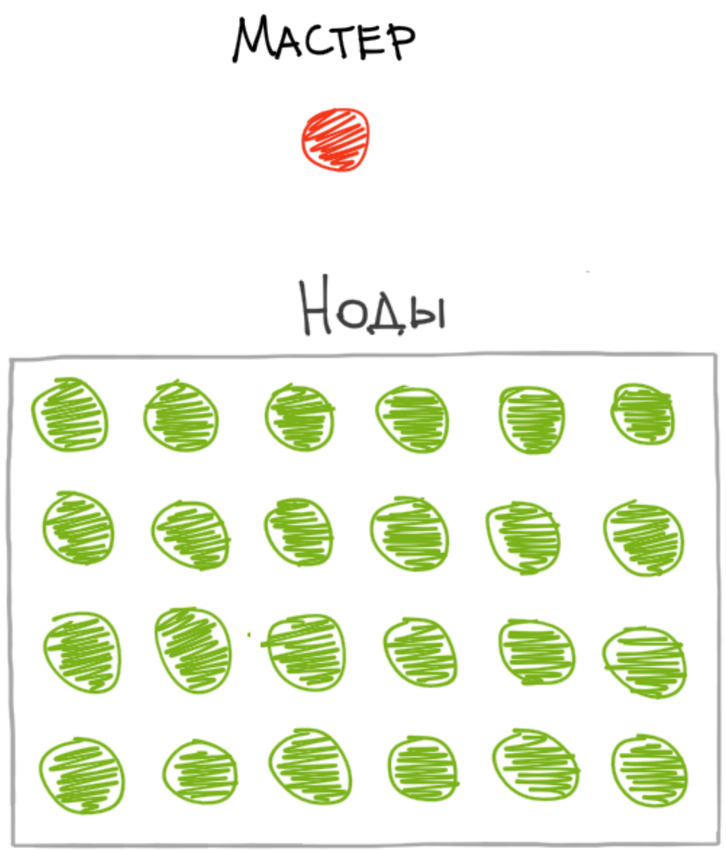
Пузыревский Иван

История вопроса

Хронология событий

- 2004 — Статья Google про модель MapReduce и ее реализацию
  - Mapreduce: Simplified data processing on large clusters (Jeffrey Dean, Sanjay Ghemawat)
- 2006 — Прототип модели MapReduce в Яндексе (**YAMR**)
- 2007 — Публичный релиз **Apache Hadoop**
  - 4 September, 2007: release 0.14.1 available
- 2011 — Начало работ над **YT**

Структура кластера YAMR



Ограничения YAMR

- Мастер-сервер (хранящий метаданные) является единственной точкой отказа
- Шедулер совмещен с мастером, ограниченные возможности по масштабированию
- Нет разделения между слоями хранения данных и их обработки
- Слабая поддержка метаданных и возможностей интроспекции состояния системы

Ограничения YAMR

- Негибкая система распределения ресурсов в шедулере (слоты с фиксированным объемом памяти)
- Неясные перспективы масштабирования на большие кластера и множество ДЦ
- Большой объем унаследованного кода, масса неудачных архитектурных решений
  - однако и большое количество клиентов
- ...и многое другое

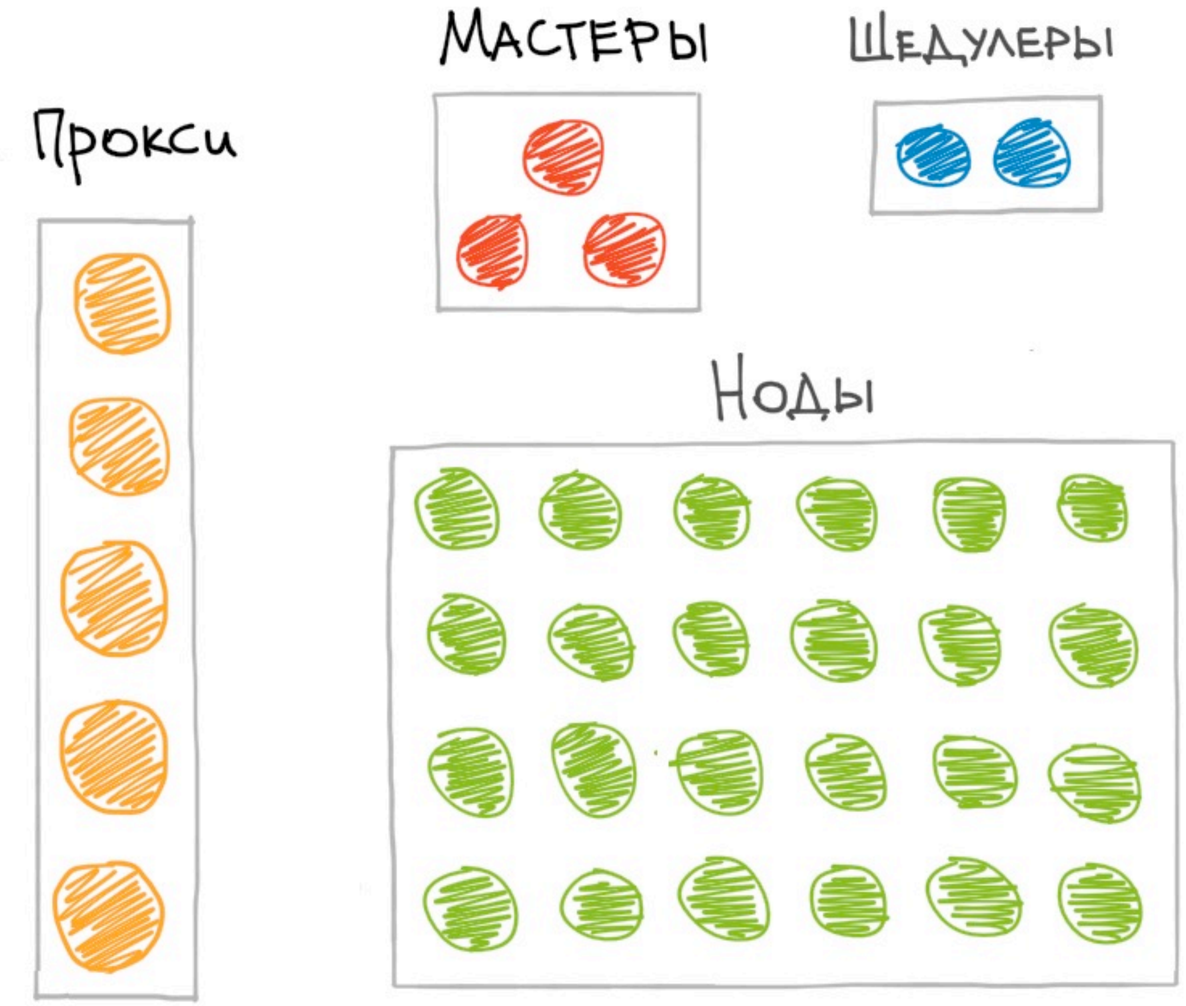
Дизайн и архитектура

Взгляд сверху

Что такое YT?

- **Распределенное хранилище**
  - Метаданные
  - “Большие данные” (хранилище чанков)
- **Среда вычислений**
  - Базовые примитивы MapReduce
  - Дополнительные табличные операции (sort, join и др.)
- **Примитивы для построения светлого будущего**
  - Гидра

Структура кластера YT



Структура кластера YT

Мастеры

- Поддерживают в памяти метаинформацию о системе (чанки, транзакции, Кипарис и пр.)
- Синхронно реплицированы, способны работать пока есть кворум
- Автоматический горячий failover (~10 сек)
- Координируют процессы репликации, балансировки и восстановления данных на нодах

Структура кластера YT

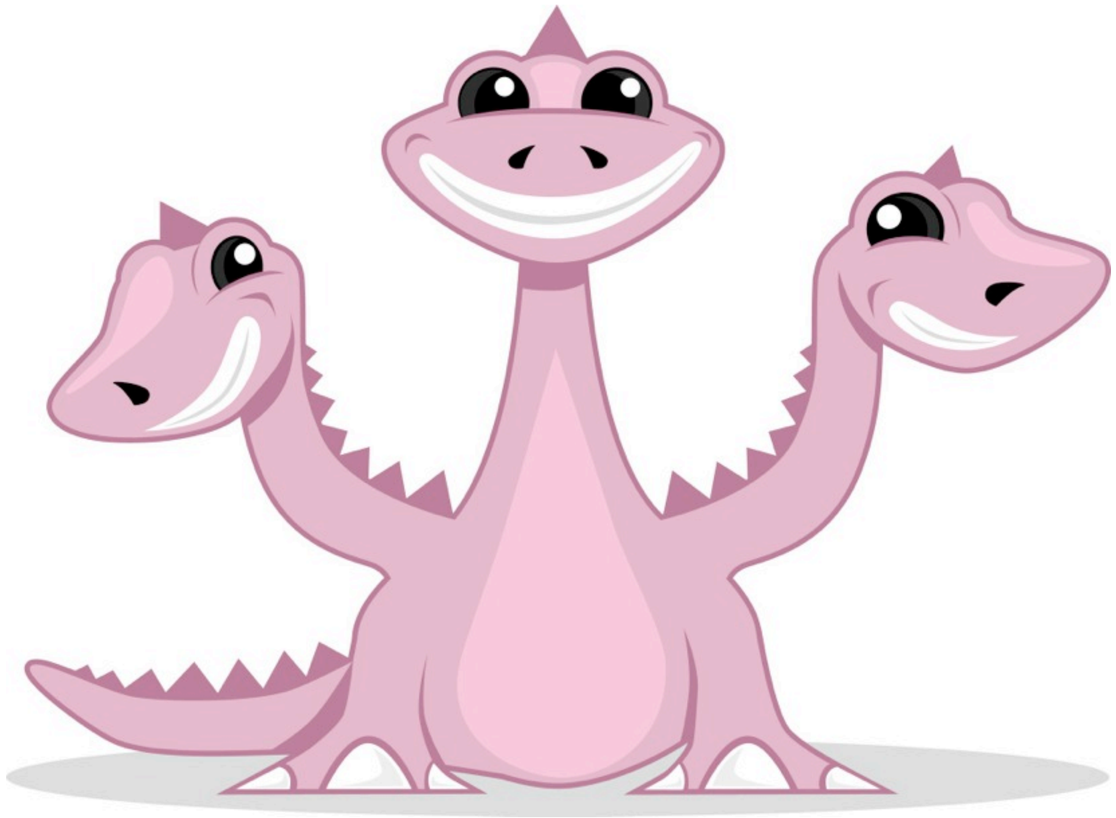
- **Шедулеры**
  - Планируют и запускают операции на данных (map, reduce, sort, join etc)
  - Реплицированы, периодически сохраняют снепшоты состояния
  - Автоматический failover (~1 минуты)
- **Ноды**
  - Хранят **чанки** с данными
  - Выполняют элементарные шаги вычисления

Дизайн и архитектура

Метаданные

Репликация состояния

- Replicated state machine via Atomic broadcast
  - “**Гидра**”
  - Более простой подход по сравнению с Paxos
  - Используется в Zookeeper
  - Формальное обоснование: Raft



Гидра

- **Эффективное решение**
  - ~100 000 мутаций в секунду
  - Объем состояния ограничен RAM (~100 GB)
- **Автоматическое восстановление при сбоях (диски, сеть, машины целиком)**
  - Горячий failover (~10 сек)
- **Периодическая кросс-валидация состояний мастеров**
  - Логи и снепшоты снабжены контрольными суммами
- **Минорные апдейты серверов без даунтайма**

Гидра

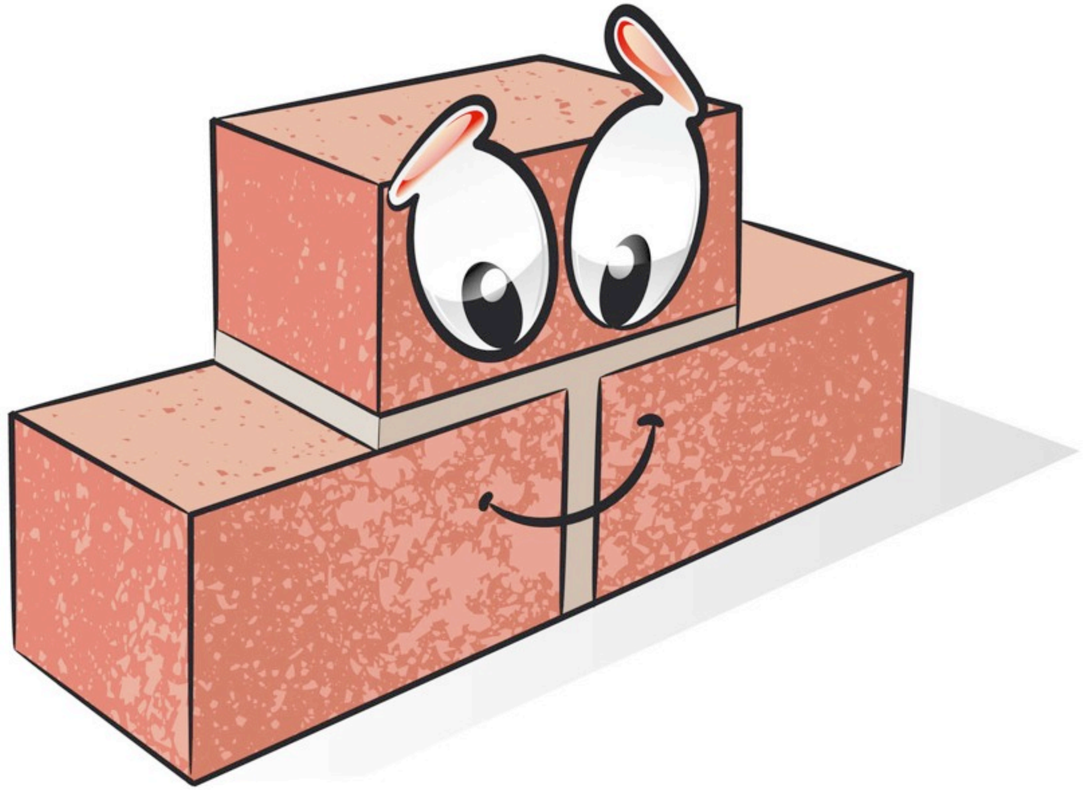
- **Универсальный** и **переиспользуемый** компонент
- Подходит для репликации любого конечного автомата
- Полностью автоматические репликация и восстановление

Дизайн и архитектура

“Большие данные”

Чанки

- Иммутабельные блобы
  - Желаемый размер 100Mb-1Gb
- Хранятся на нодах
  - Сотни тысяч чанков на одной ноде
- Базовый строительный блок для файлов и таблиц



Репликация и erasure coding

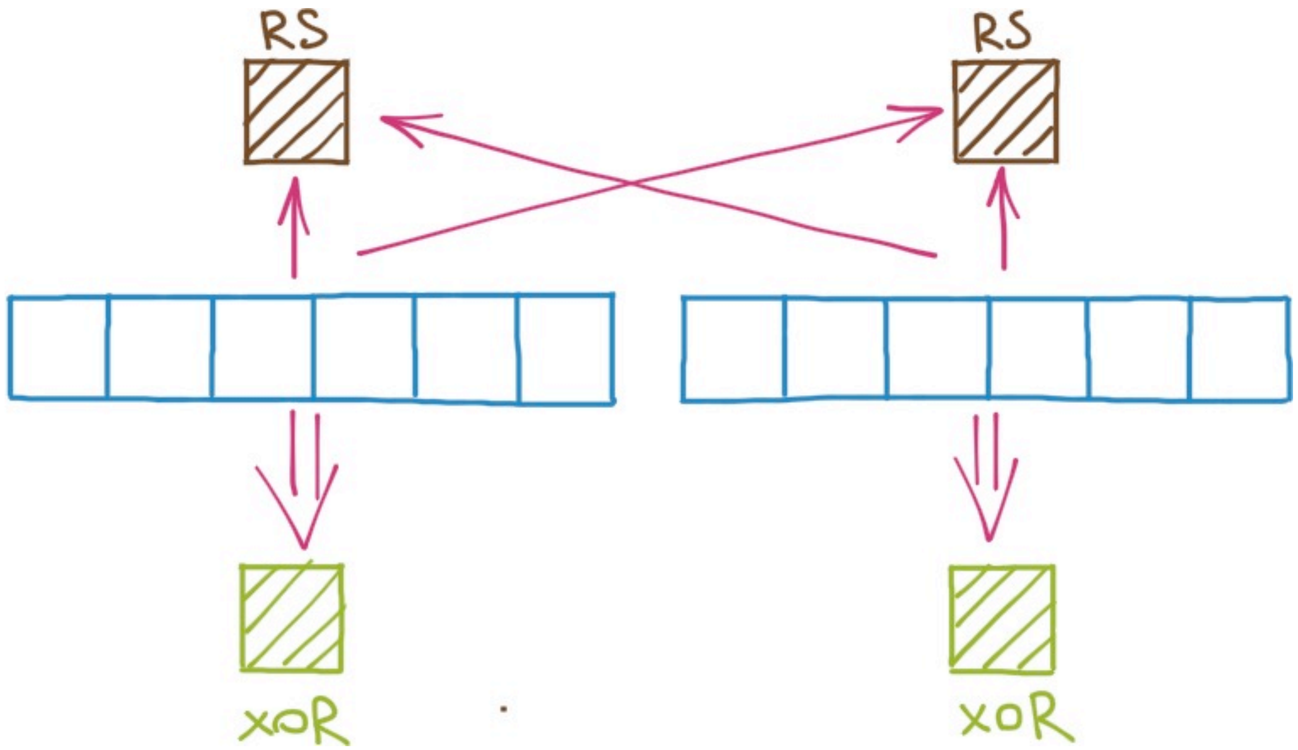
- Ноды кластера выпадают
  - ~10 замен дисков на кластере размера ~1000 в неделю
- **Репликация**: *k* реплик, оверхед *k*-1, выдерживает выпадение *k*-1 нод
- **Коды Коши-Рида-Соломона** RS(*n*, *k*): оверхед *k*/*n*, выдерживает выпадение *k* нод
- Сравнительный пример
  - 3-кратная репликация: оверхед 200%
  - RS(6, 3): оверхед 50%
  - Доступность RS(6,3) не хуже тройной репликации

Репликация и erasure coding

- Почему бы всегда не использовать RS вместо репликации?
- Восстановление может быть дорогим
  - Репликация требует одного чтения при восстановлении
  - RS(*n*, *k*) требует *n* чтений
- Пример
  - 3-кратная репликация: оверхед 200%, 1 чтение
  - RS(6,3): оверхед 50%, 6 чтений

Репликация и erasure coding

- Нужен компромисс
  - **Коды с локальным восстановлением** (LRC)
- LRC(12,2,2)
  - оверхед 33%
  - выдерживает выпадение 3 нод
  - <7 чтений в среднем



Планы на будущее

Масштабнее

- Шардирование метаинформации
- Шардирование шедулера
- Междатацентровые инсталляции
  - Дополнительный координирующий слой (поверх Гидры)
  - Синхронная и асинхронная репликация данных между ДЦ
- Слой совместимости с Hadoop

Интерактивнее

- Таблицы с быстрым доступом по ключу
  - Статические данные в виде табличных чанков
  - In-memory дельты с репликацией состояния
- Табличные транзакции на основе timestamps
  - MVCC
- SQL-подобный язык запросов
- Потоковые модели вычислений (Storm, MillWheel)

Спасибо за внимание!

Пузыревский Иван
старший разработчик YT

+7(926)2475146

sandello@yandex-team.ru