HDFS
(Hadoop Distributed File System)

Архитектура HDFS

Google's Filesystem GFS

research.google.com/archive/gfs-sosp2003.pdf

- Работает на кластере серверов
- Для пользователя как “*один большой диск*”
- Работает поверх обычных файловых систем
  (Ext3, Ext4, XFS)

Fault Tolerant

Данные не теряются, если выходят из строя диски или сервера

Используется обычное “железо”

“Дешевое”серверное оборудование

- **Нет** суперкомпьютерам!
- **Нет** десктопам!
- **Да** обычным (~~ненадежным~~) серверам!

HDFS хорошо подходит для…

Хранения больших файлов

- Терабайты, петабайты…
- Миллионы (но не миллиарды) файлов
- Файлы размером от 100 Мб

HDFS хорошо подходит для…

Стриминг данных

- Паттерн “write once / read-many times”
- Оптимизация под последовательное чтение
  - Нет операциям произвольного чтения
- Операция *append* появилась в Hadoop 0.21

HDFS хорошо подходит для…

Обычные сервера

- Менее надежные, чем суперкомпьютеры

HDFS не подходит для…

Low-latency reads

- Высокая пропускная способность вместо быстрого доступа к данным
- HBase помогает решать эту задачу

HDFS не подходит для…

Большое количество небольших файлов

– Лучше миллион больших файлов, чем миллиард маленьких

лучше 1000 по 1 Гб, чем 100 000 по 10 Мб

HDFS не подходит для…

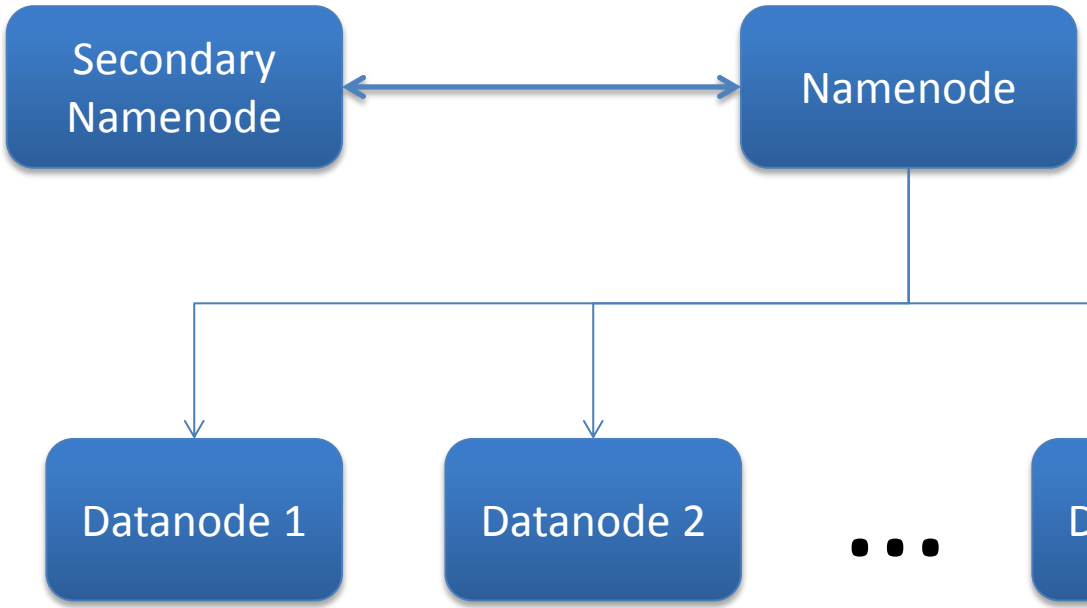
Многопоточная запись

- Один процесс записи на файл
- Данные дописываются в конец файла

Нет поддержки записи по смещению

Демоны HDFS

Демоны HDFS



- Namenode
- Datanode
- Secondary node

Namenode

- Отвечает за:
  - файловое пространство (namespace)
  - мета-иформацию
  - расположение блоков файлов
- Запускается на 1й (выделенной) машине

Datanode

- Хранит и отдает блоки данных
- Отправляет ответы о состоянии на Namenode
- Запускается на каждой машине кластера

Secondary Namenode

- Периодически обновляет fsimage
- Требует то же железо, что и Namenode
- (!) Не используется для high-availability, т.е. это не backup для Namenode

Файлы и блоки

Файлы и блоки

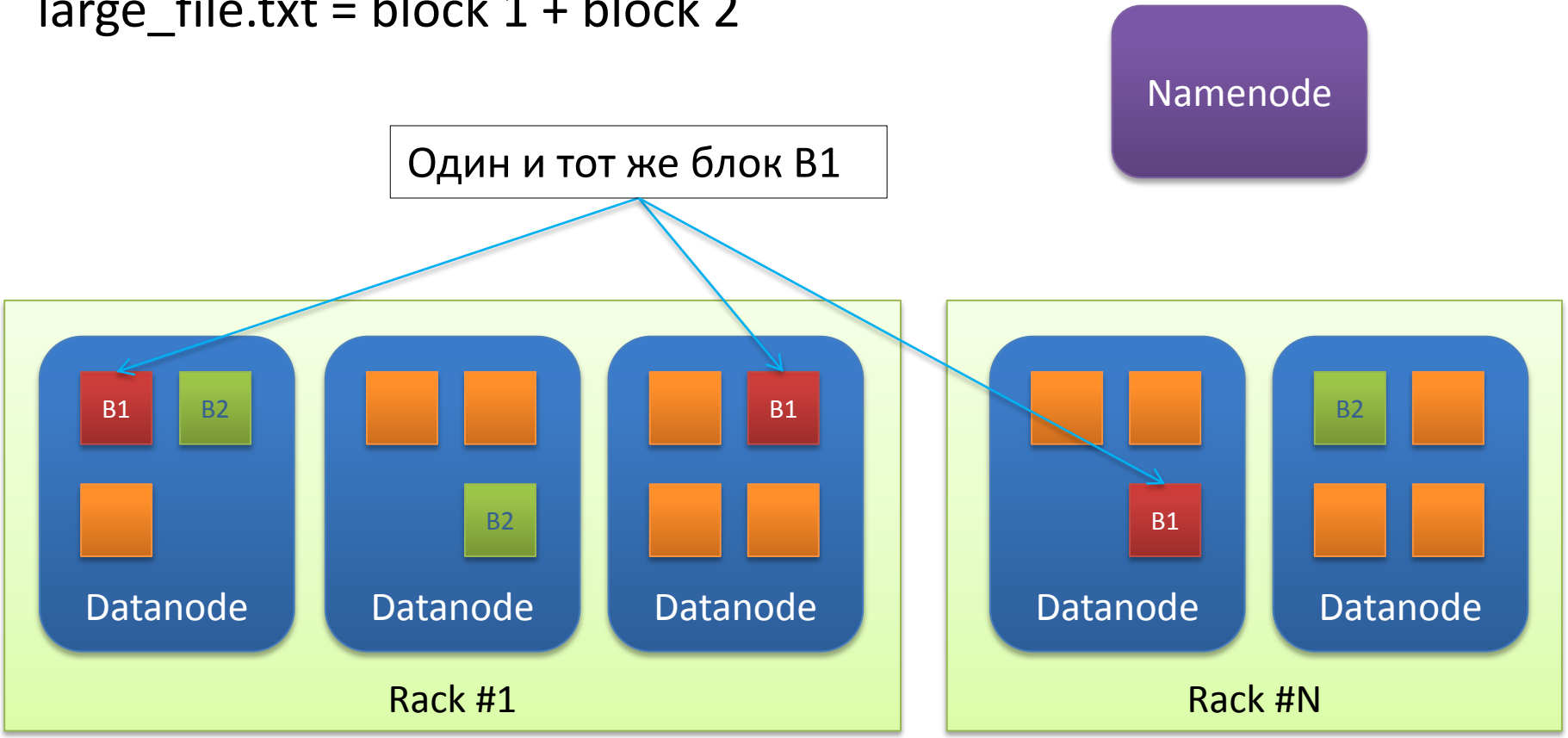
- Файлы в HDFS состоят из блоков
  - Блок – единица хранения данных
- Управляется через Namenode
- Хранится на Datanode

Файлы и блоки

Реплицируются по машинам в процессе записи

- Один и тот же блок хранится на нескольких Datanode
- Фактор репликации по умолчанию равен **3**
- Это нужно для fault-tolerance и упрощения доступа

large_file.txt = block 1 + block 2



Блоки в HDFS

- Стандартный размер блоков 64Мб или 128Мб
- Основной мотив этого – снизить стоимость *seek time* по сравнению со скоростью передачи данных (*transfer rate*)
  - 'Time to transfer' > 'Time to seek'

Пусть

- seek time = 10ms
- transfer rate = 100 MB/s

Каким должен быть размер блока, чтобы *seek time* достигал всего 1% от *transfer rate*?

Репликация блоков

- Namenode определяет, где располагать блоки
- Баланс между надежностью и производительностью
  - Попытка снизить нагрузку на сеть (*bandwidth*)
  - Попытка улучшить надежность путем размещения реплик в разных стойках

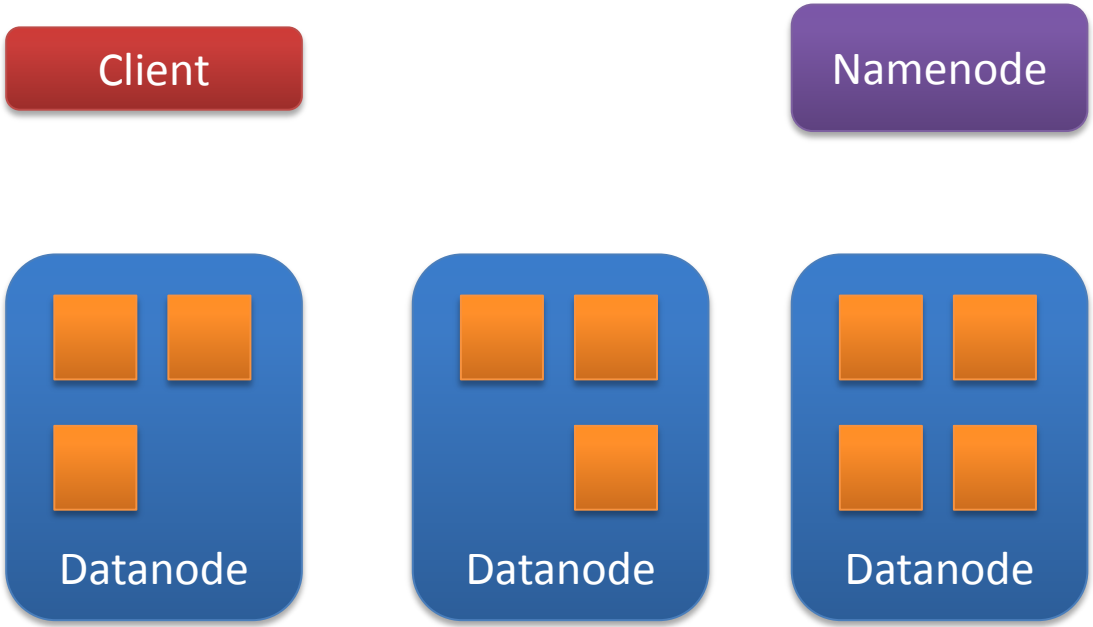
Репликация блоков

Фактор репликации по умолчанию равен **3**

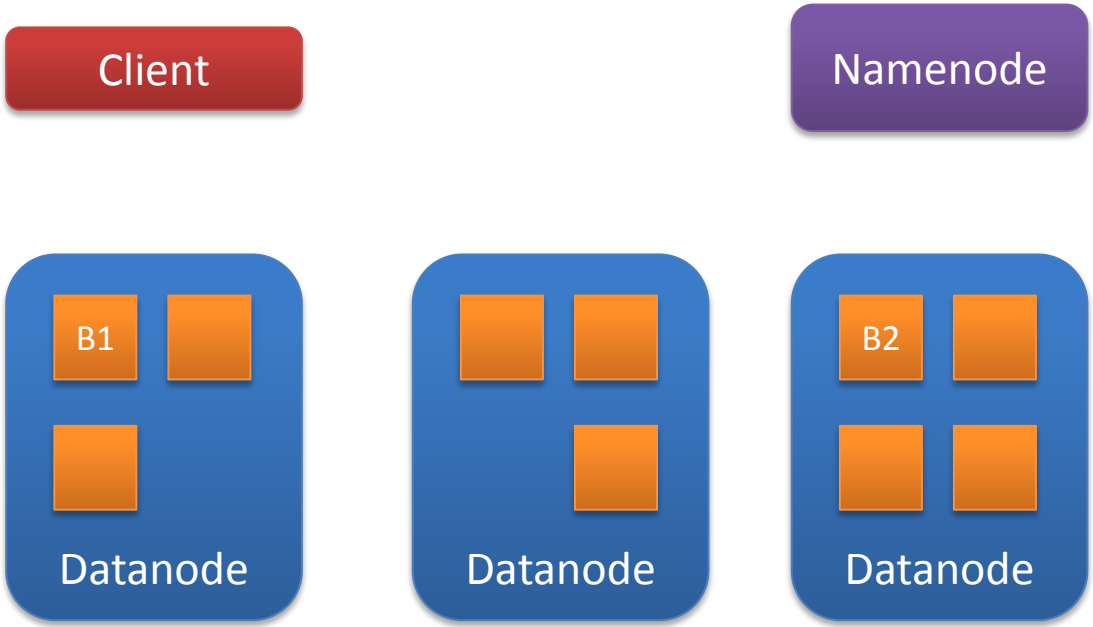
- 1-я реплика на локальную машину
- 2-я реплика на другую машину из той же стойки
- 3-я реплика на машину из другой стойки

Взаимодействие клиента и демонов

Взаимодействие клиента с демонами

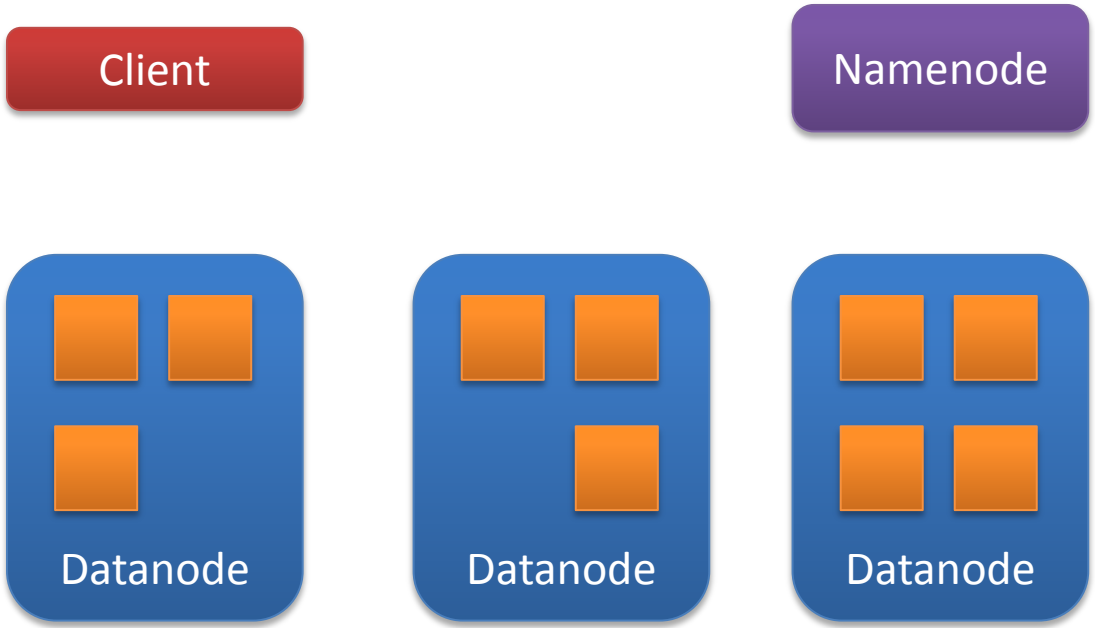


HDFS: чтение файла



HDFS: запись файла

1. Создать новый файл в namespace на NN и определить топологию блоков
2. Отправить данные на 1-ю DN
3. Отправить данные на 2-ю DN
4. Отправить данные на 3-ю DN
5. Подтверждение Success/Failure
6. Подтверждение Success/Failure
7. Подтверждение Success/Failure



Namenode:
Использование памяти
и
Fault-tolerance

Namenode: использование памяти

Для быстрого доступа вся мета-информация о блоках хранится в ОЗУ Namenode

- Чем больше кластер, тем больше ОЗУ требуется

- Лучше миллионы больших файлов, чем миллиарды маленьких
- Работает на кластерах из сотен машин

Namenode: использование памяти

Hadoop 2+

- Namenode Federation
  - Каждая Namenode управляет частью блоков
  - Горизонтальное масштабирование Namenode
- Поддержка кластеров из тысячи машин
- Детали: http://hadoop.apache.org/docs/r2.0.2-alpha/hadoop-yarn/hadoop-yarn-site/Federation.html

Namenode: использование памяти

Как влияет размер блока на максимальный размер FS?

- Больше размер блока –> меньше блоков
- Меньше блоков –> больше файлов в FS

- Пусть у нас есть 200Тб = 209,715,200 Мб
- При размере блока 64Мб:

  209,715,200Мб / 64Мб = 3,276,800 блоков
- При размере блока 128Мб:

  209,715,200Мб / 128Мб = 1,638,400 блоков

Fault-tolerance в Namenode

- Если Namenode падает, то HDFS не работает
- Namenode – это единая точка отказа (*single point of failure*)
  - Должна работать на отдельной надежной машине
  - Обычно, это не бывает проблемой

Fault-tolerance в Namenode

Hadoop 2+

- High Availability Namenode
  - Процесс Active Standby всегда запущен и берет на себя управления в случае падения Namenode
  - Все еще в процессе тестирования
- Более подробно тут:
  - http://hadoop.apache.org/docs/r2.0.2-alpha/hadoop-yarn/hadoop-yarn-site/HDFSHighAvailability.html

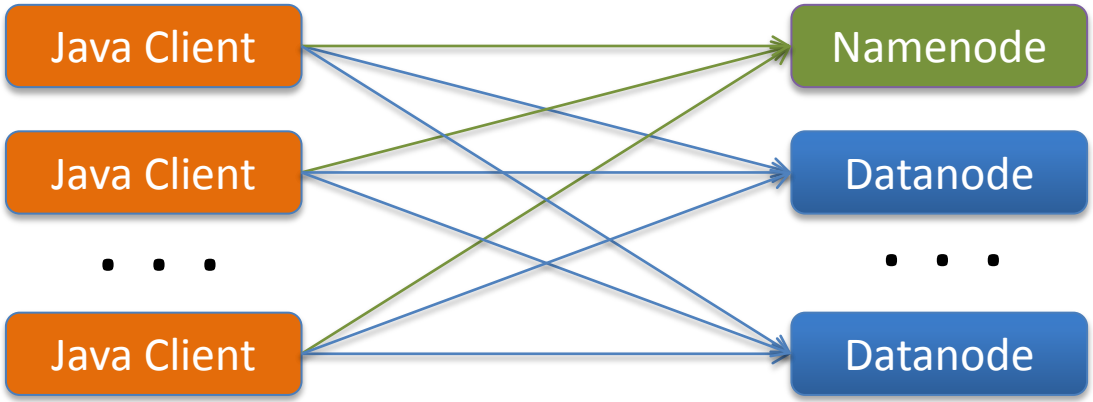
Доступ к HDFS

Доступ к HDFS

- Способы доступа
  - Direct Access
    - Взаимодействует с HDFS с помощью нативного клиента
    - Java, C++
  - Через Proxy Server
    - Досутп к HDFS через Proxy Server – middle man
    - Серверы REST, Thrift и Avro

Direct Access

- API для Java и C++
- Клиент запрашивает метаданные от NN
- Клиент напрямую запрашивает данные от DN
- Используется для MapReduce



Доступ через Proxy Server

- Клиент работает через внешний Proxy Server
- Существует несколько серверов в поставке с Hadoop
  - Thrift – язык определения интерфейса
  - WebHDFS REST – ответы в формате JSON, XML или ProtoBuf
  - Avro – механизм сериализации