Haystack

«Использование баз данных»

Цесько Вадим Александрович
https://incubos.org
@incubos

ТЕХНОПОЛИС

24 мая 2017 г.

Содержание

Материалы

- Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel, Facebook Inc. *Finding a needle in Haystack: Facebook's photo storage*. 2010

Мотивация

Зачем мы это рассматриваем:

- Промышленная внедрённая технология
- Логика проектирования
- Интересные проектные решения
- Просто интересно, как делают взрослые дядьки

Цифры

На 2010 год:

- **65 млрд.** оригинальных фоток
- 4x размера = **260 млрд.** фоток = **20 ПБ**
- +**1 млрд.** фоток (**60 ТБ**) каждую неделю
- **1 Mrps** в пике
- В проде **2 года**

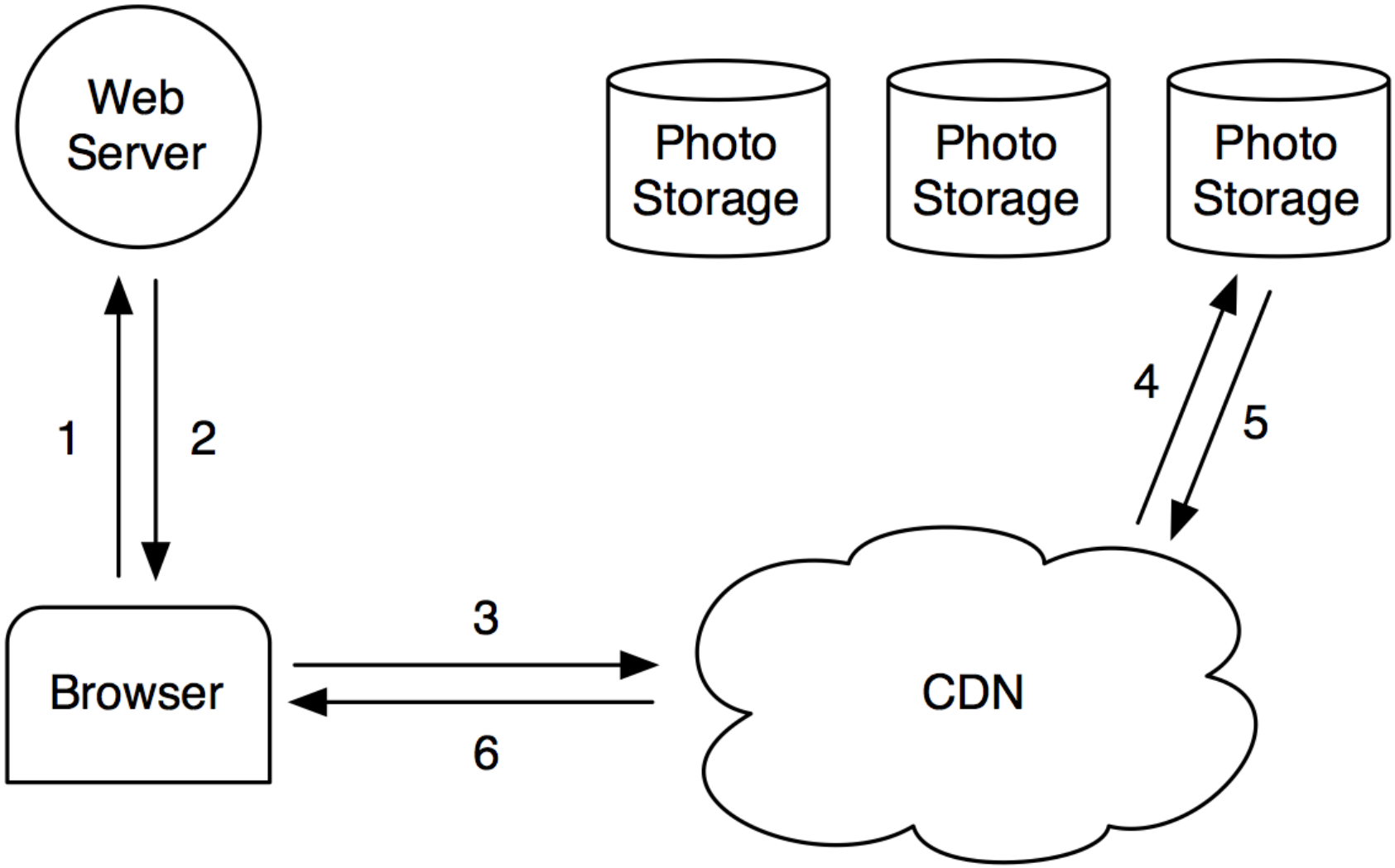
Характер запросов

- Пишем — *один раз*
- Читаем — *часто*
- *Никогда* не перезаписываем
- Удаляем — *редко*

Основные цели

- **High throughput and low latency**
  - UGC, CDN, User Experience
  - 1 поиск по диску максимум
  - Микрометаданные в памяти
- **Fault-tolerant**
  - UGC
  - Георепликация
- **Cost-effective**
  - \$ / TB, read rps / TB
  - \$ / TB — **на 28% меньше**, read rps / TB — **в 4 раза больше** vs NAS
- **Simple**

Путь запроса



CDN

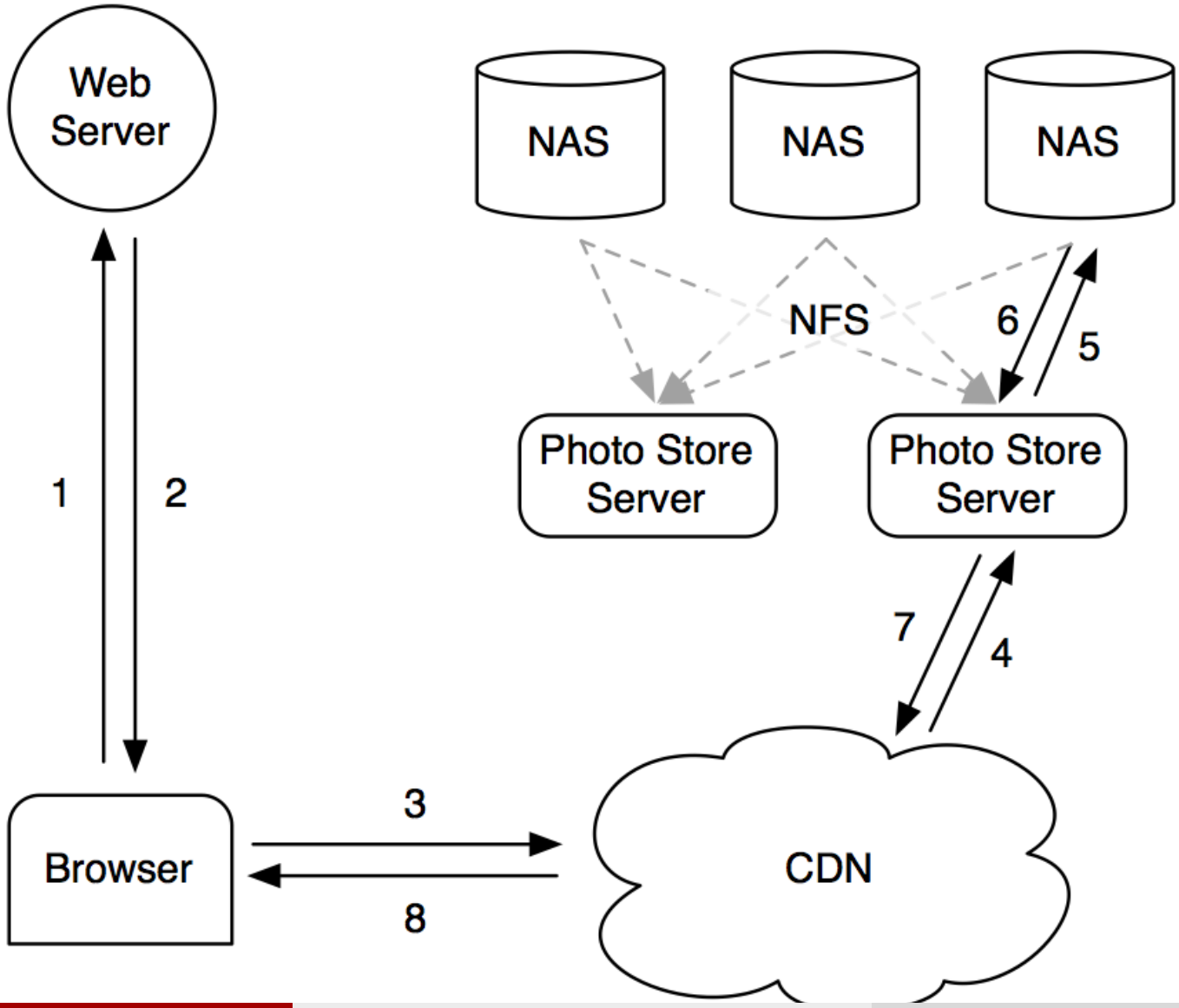
- Клиент получает URL от Web Server
- Идёт с URL'ом в CDN
- Если у CDN есть данные, то отдаёт
- Если нет, то идёт в Photo Storage и кэширует
- URL содержит необходимую CDN информацию

Главный урок

CDN не подходит для социальных фотосервисов

- Эффективно раздаёт «горячие» фотки (профили и свежие)
- **Длинный хвост** (непопулярные и/или старые)
- Длинный хвост невозможно кэшировать

Путь запроса



Хранение фоток

- Каждая фотка в отдельном файле на коммерческом NAS
- Photo Store монтирует все разделы через NFS
- Photo Store из URL вынимает путь, читает файл по NFS и отдаёт CDN

Проблемы

- Тысячи файлов в каждом каталоге
  - Больше 10 дисковых операций на одну фотку
- Сотни файлов в каждом каталоге
  - Больше 3 дисковых операций на одну фотку (метаданные каталога, `inode`, содержимое файла)

Оптимизации

- Кэш из имени файла в файловый дескриптор
- Пропатчили ядро — системный вызов `open_by_filehandle`
- Не сильно помогло на длинном хосте

Вывод

Кэш не всегда спасает

Альтернативы

Из используемого в Facebook:

- Аналог GFS
- MySQL
- Hadoop
- ФС (`inode` — сотни байт на файл)

Вывод

Всё не очень подходит для длинного хвоста

Новая версия

- CDN для популярных фоток (пока что)
- Haystack для длинного хвоста

Задача

- Уменьшить нагрузку на диск
- Только необходимые операции
- Уменьшить расход памяти на метаданные
- Держать все метаданные в памяти

Подход

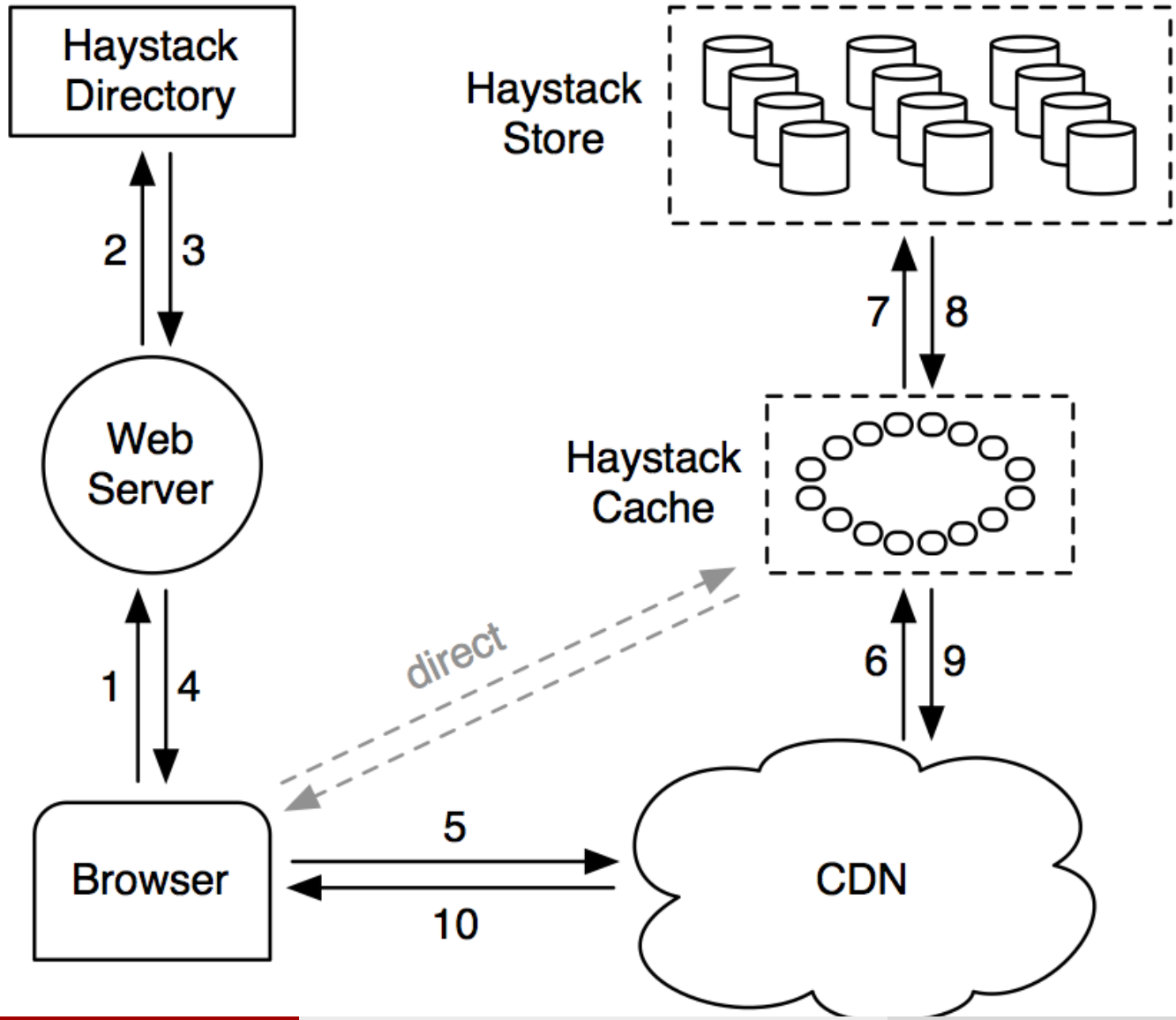
Haystack

Огромные файлы с кучей фоток в каждом

Компоненты

- **Store**
  - Physical volume (на каждой машине 100 x 100 ГБ = 10 ТБ)
  - Logical volume = Replica Set
- **Directory**
  - Logical volume → Physical volume
  - Фотка → Logical volume
  - Logical volumes with free space
- **Cache**
  - Прикрывает Store
  - Защищает при перезапуске CDN
  - Уменьшает зависимость от CDN

Чтение фоток



URL

CDN URL

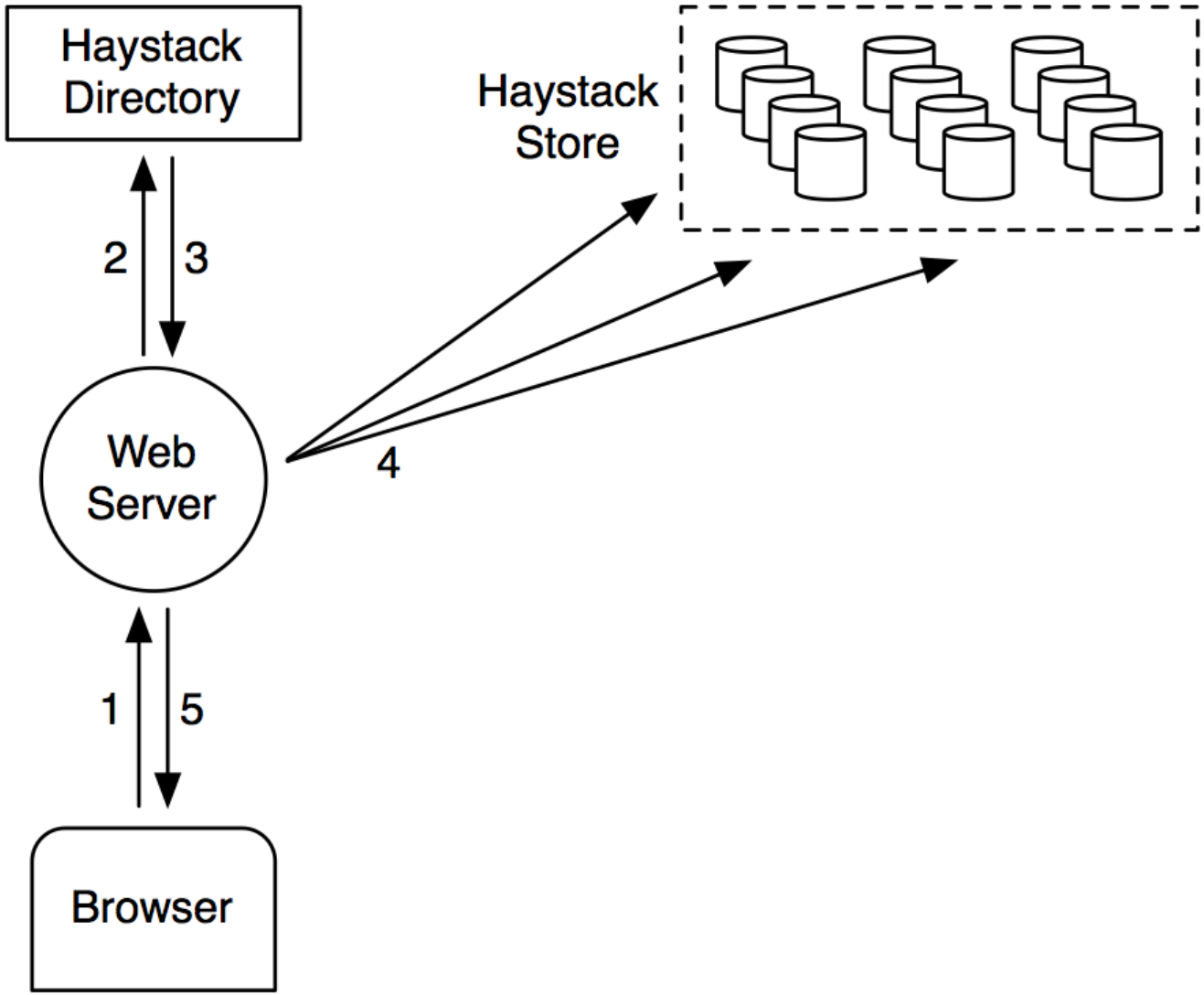
`http://<CDN>/<Cache>/<Store>/<LV, Photo>`

- Браузер идёт на <CDN>
- CDN ищет у себя, используя <LV, Photo>
- Если не нашёл, отрезает <CDN> и идёт в <Cache>
- Cache ищет у себя, используя <LV, Photo>
- Если не нашёл, отрезает <CDN> и идёт в <Store>

Direct URL

`http://<Cache>/<Store>/<LV, Photo>`

Добавление фоток



Функции Directory

- Отображение из логических томов в физические
  - При добавлении фоток
  - При конструировании URL'ов фоток
- Балансирует нагрузку
  - Запись по логическим томам
  - Чтение по физическим томам
- Определяет, кто будет отдавать: CDN или Cache
- Помечает логические тома как read-only
  - Эксплуатационные причины
  - Нет свободного места

Write-enabling

- Новые Store доступны на запись
- Только на такие машины добавляются фотки
- Когда место на машине кончается — помечается как read-only

Внимание!

Это влечёт определённые последствия.

Устройство Directory

- Хранит данные в реплицированной БД
- Простой сервис на PHP
- Использует memcached
- Если теряем Store, то удаляем соответствующее отображение и добавляем после замены машины

Функции Cache

- Получает HTTP-запросы от CDN и браузеров
- DHT: ключ — id фотки
- Если нет в кэше, то идёт на Store
- Кэширует фотку только при выполнении двух условий:
  - Запрос пришёл от пользователя, а не от CDN
    - CDN сам закэширует
  - Фотка загружена с write-enabled Store
    - Свежие фотки более популярны — прикрываем write-enabled Store
    - ФС на Store лучше работает, когда либо чтение, либо запись, но не смесь
- Планируется push свежих фоток в Cache

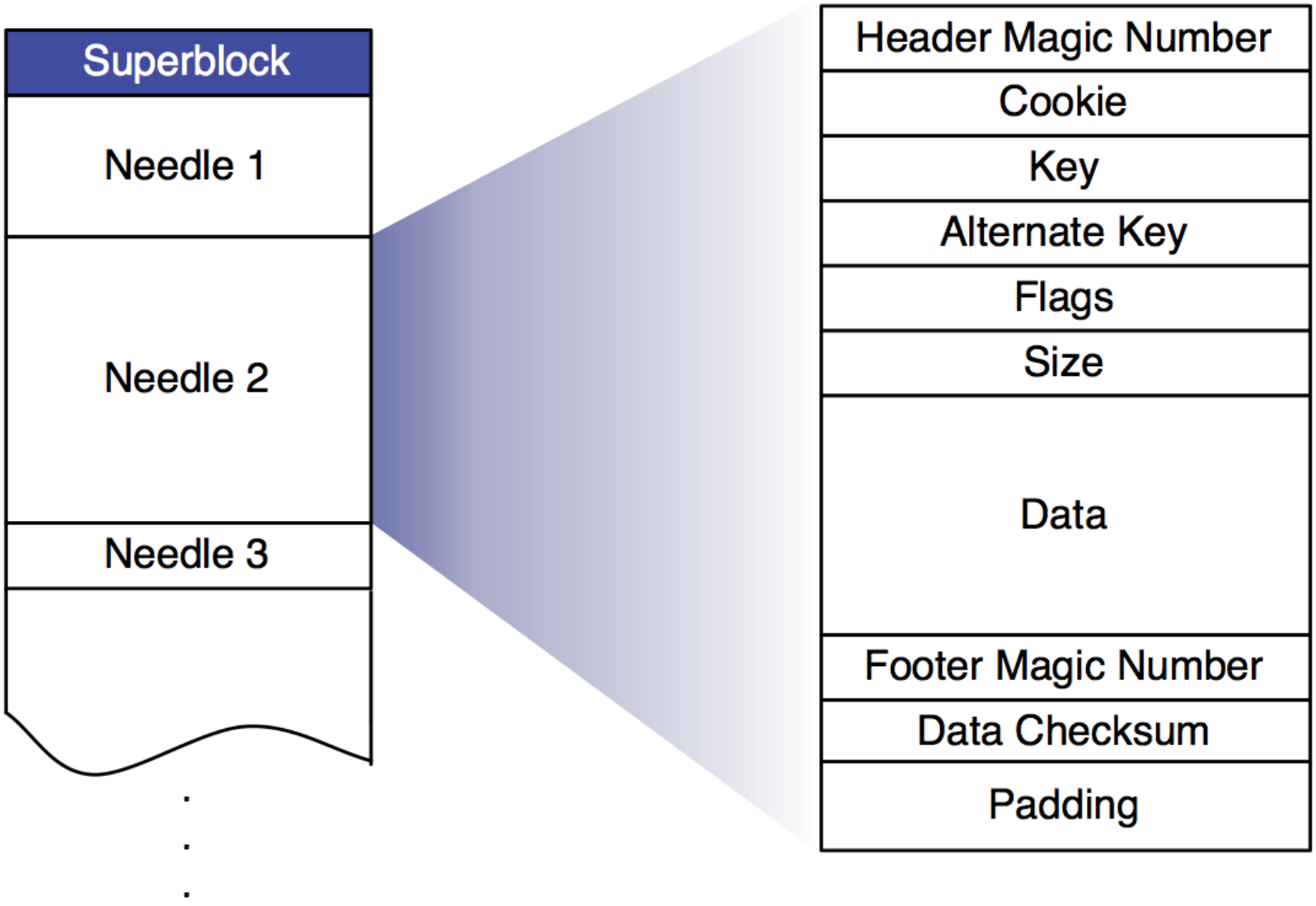
Функции Store

- Простой интерфейс — `get`, `add`, `delete` по `<LV, Photo>`
- На каждом Store множество PV
- Каждый PV содержит миллионы фоток
- PV — большой файл (100 ГБ) `/hay/haystack_<LV_id>`
- Получение имени файла, смещения и размера для фотки не требует дисковых операций
- Открытые файловые дескрипторы для каждого PV
- Отображение в памяти из id фотки в метаданные (файл, смещение и размер)

Store-файл

- Superblock
- Последовательность needle
- Needle — фотка

Формат


Superblock
Needle 1
Needle 2
Needle 3
Header Magic Number
Cookie
Key
Alternate Key
Flags
Size
Data
Footer Magic Number
Data Checksum
Padding

Поля needle

Field	Explanation
Header	Magic number used for recovery
Cookie	Random number to mitigate brute force lookups
Key	64-bit photo id
Alternate key	32-bit supplemental id
Flags	Signifies deleted status
Size	Data size
Data	The actual photo data
Footer	Magic number for recovery
Data Checksum	Used to check integrity
Padding	Total needle size is aligned to 8 bytes

Индекс

- In-memory
- `(key, alternate-key) -> (flags, size, offset)`
- Store может перестроить индекс при сбое

Запросы

- Read
- Write (Append)
- Delete

Read

- Cache передаёт с запросом: `LV_id`, `key`, `alternate_key`, `cookie`
  - `cookie` генерируется и сохраняется в Directory при загрузке фотки
  - `cookie` защищает от атак с подбором URL'ов
- Store извлекает метаданные
- Если фотка не удалена, то читает needle с диска
- Проверяет `cookie` и `checksum`
- Возвращает фотку Cache

Write

- Web server передаёт с запросом: `LV_id`, `key`, `alternate_key`, `cookie` и данные
- Каждый Store синхронно **дописывает** фотку в PV и обновляет индекс
- Модификация фотки:
  - Либо в тот же LV с теми же `key` и `alternate_key`: для Store больший `offset` — свежее версия
  - Либо в другой LV: Directory обновляет метаданные и больше не читает старую фотку

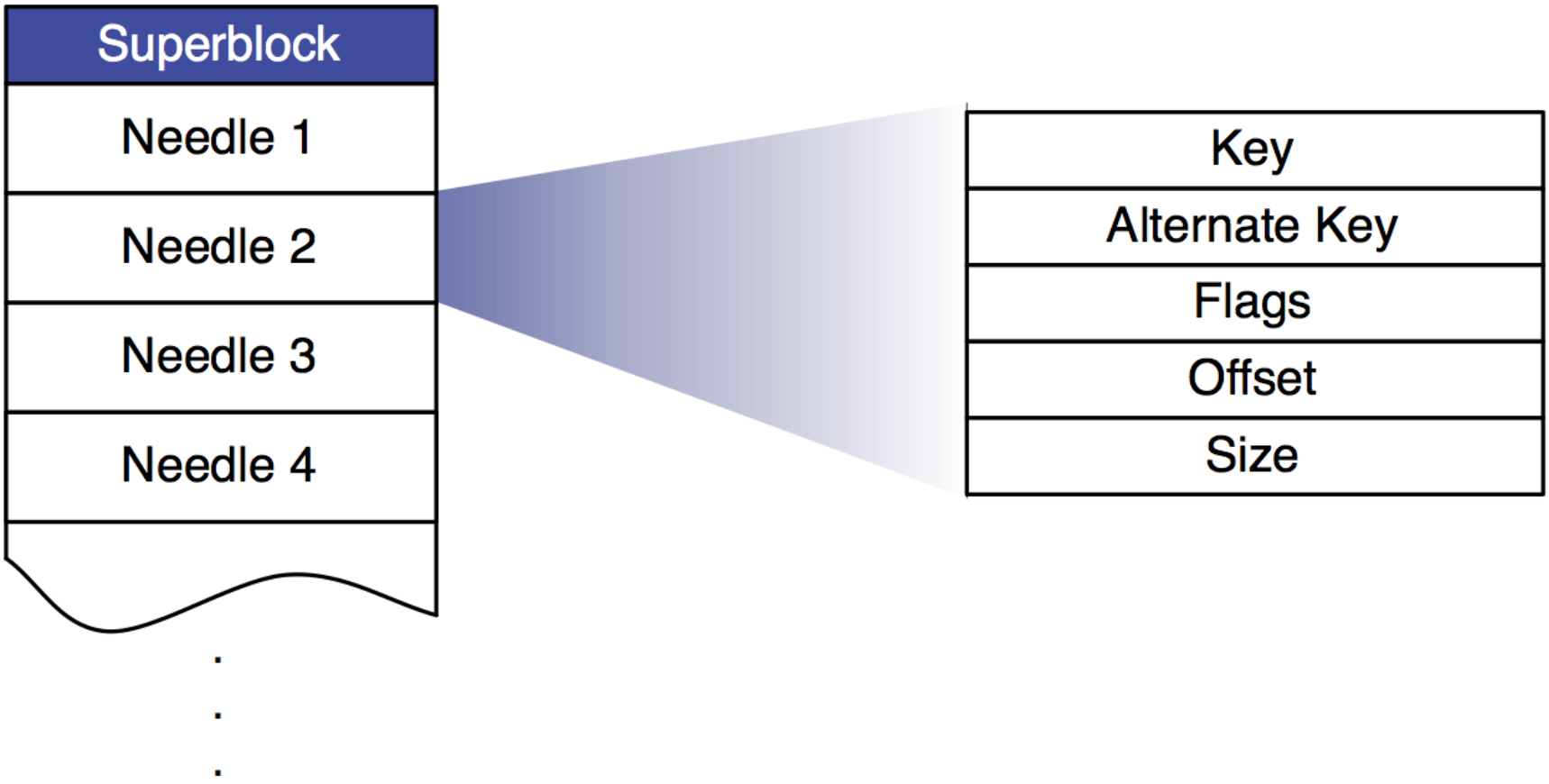
Delete

- Store выставляет флаг в памяти и синхронно на диске
- Запросы на чтение всегда проверяют флаг
- Фотка физически не удаляется

Функции Index

- Оптимизация при перезагрузке Store
- Индекс для каждого тома
- Индекс содержит superblock и индексные записи
- Порядок индексных записей соответствует тому

Формат


Superblock
Needle 1
Needle 2
Needle 3
Needle 4
Key
Alternate Key
Flags
Offset
Size

Обновление

- Write:
  - Синхронное дописывание needle в том
  - Асинхронное — в индекс
- Delete:
  - Синхронно выставляется флаг в томе
  - Индекс не обновляется
- Работает быстрее

Проблемы

Needle есть, а индексной записи нет (orphan)

- «Лечится» при перезапуске
- Последняя запись в индексе — последний non-orphan needle

Индексные записи не отражают факт удаления

- При чтении всегда проверяется флаг deleted
- Обновляется индекс в памяти
- Cache уведомляется, что фотка удалена

Compaction

- Выбрасывание удалённых и дублирующихся needles
- Копирование в новый файл с пропуском мусора
- В это время удаления идут в оба файла
- Дошли до конца — блокируем исходный файл на изменение и атомарно переключаемся

Интересное наблюдение

- Свежие фотки удаляются чаще
- 25% фоток/год

Оперативная память

20% экономии:

- Вместо флагов у удалённых фоток `offset` равен 0
- `cookie` не хранятся в памяти, а проверяются каждый раз после чтения needle

В итоге

10 байт на фотку (в среднем) vs **536 байт** на `xfs_inode_t`

Batch

- Диски любят последовательную запись
- Загрузка альбомов — основной случай

Нагрузка

- 98% чтений — лента новостей и альбомы
- Возраст 2 дня — резкий спад числа обращений
- CDN и Cache довольно эффективны, но длинный хвост
- 80% — Cache hit rate
- Haystack отвечает на 10% нагрузки CDN
- Загрузка каждой фотки — 12 needles (4 размера x 3 реплики)
- См. стресс тесты в оригинальной статье[1]

[1] Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel, Facebook Inc. *Finding a needle in Haystack: Facebook's photo storage*. 2010

Распределение трафика

Operations	**Daily Counts**
Photos Uploaded	~120 Million
Haystack Photos Written	~1.44 Billion
Photos Viewed	80-100 Billion
[*Thumbnails*]	10.2 %
[*Small*]	84.4 %
[*Medium*]	0.2 %
[*Large*]	5.2 %
Haystack Photos Read	10 Billion

Рассмотрели

- Архитектура Haystack
- Основные компоненты: Directory, Cache, Store
- Интересные решения

Уроки

- Особенности реальной нагрузки
- Во что упираетесь
- KISS[2]

[2] http://en.wikipedia.org/wiki/KISS_principle

Вопросы?

- https://polis.mail.ru/blog/view/111/
- https://incubos.org/contacts/
- Не забудьте про ДЗ