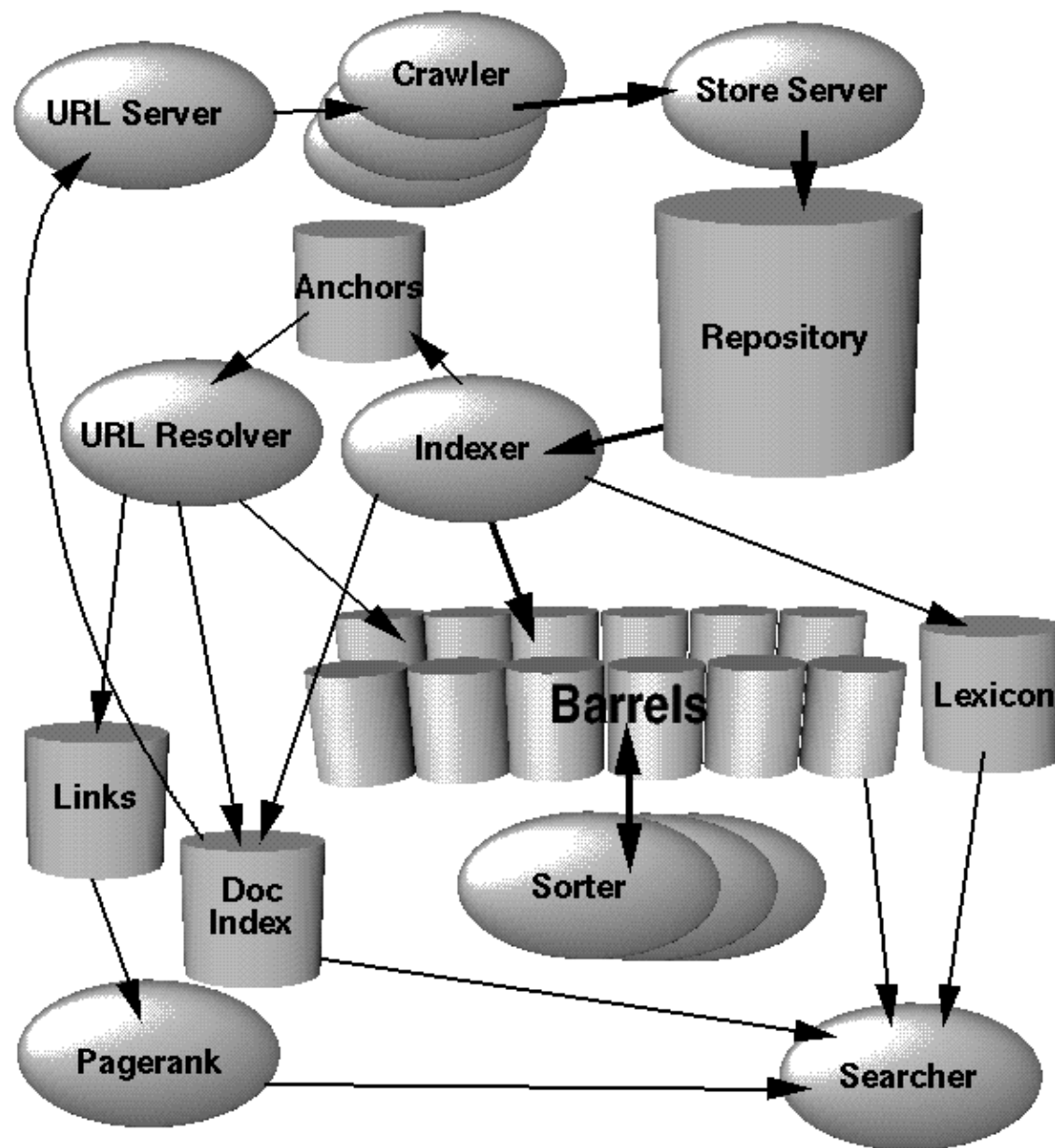


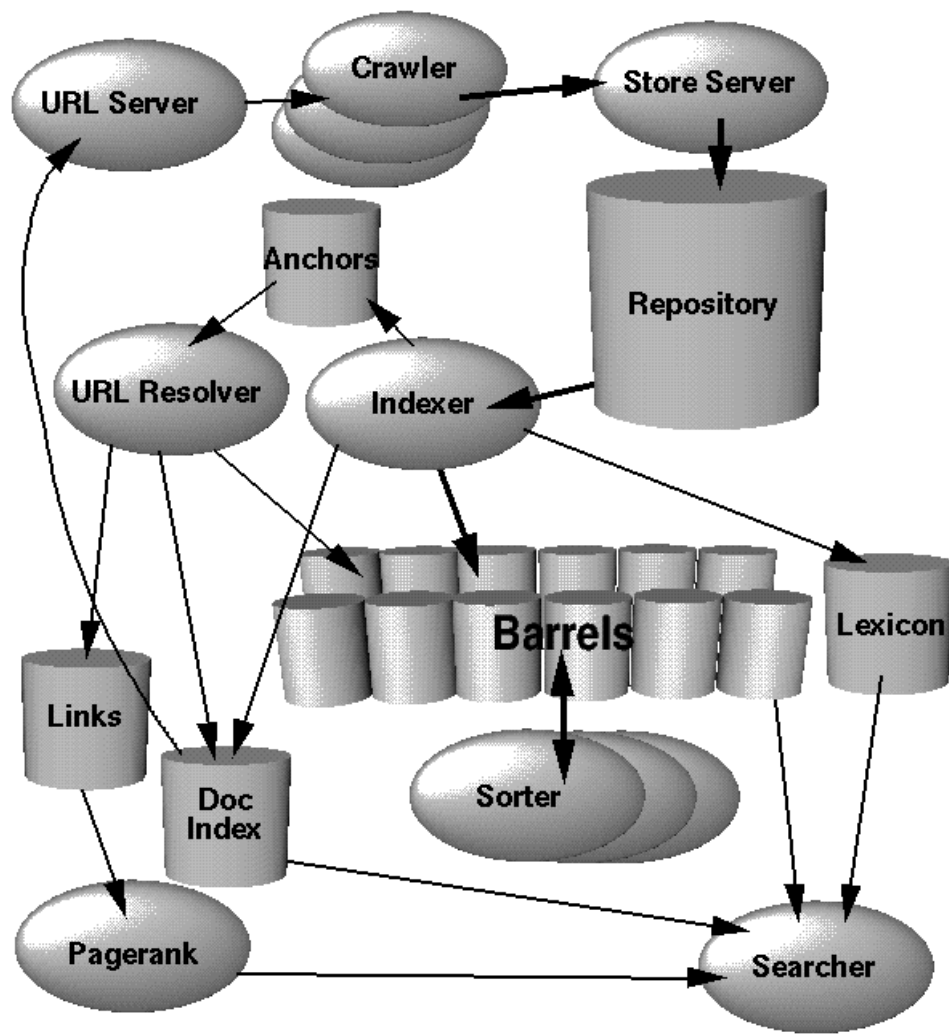
Архитектура на Google

проф. д-р инж. Христо Вълчанов

<http://cs.tu-varna.bg>

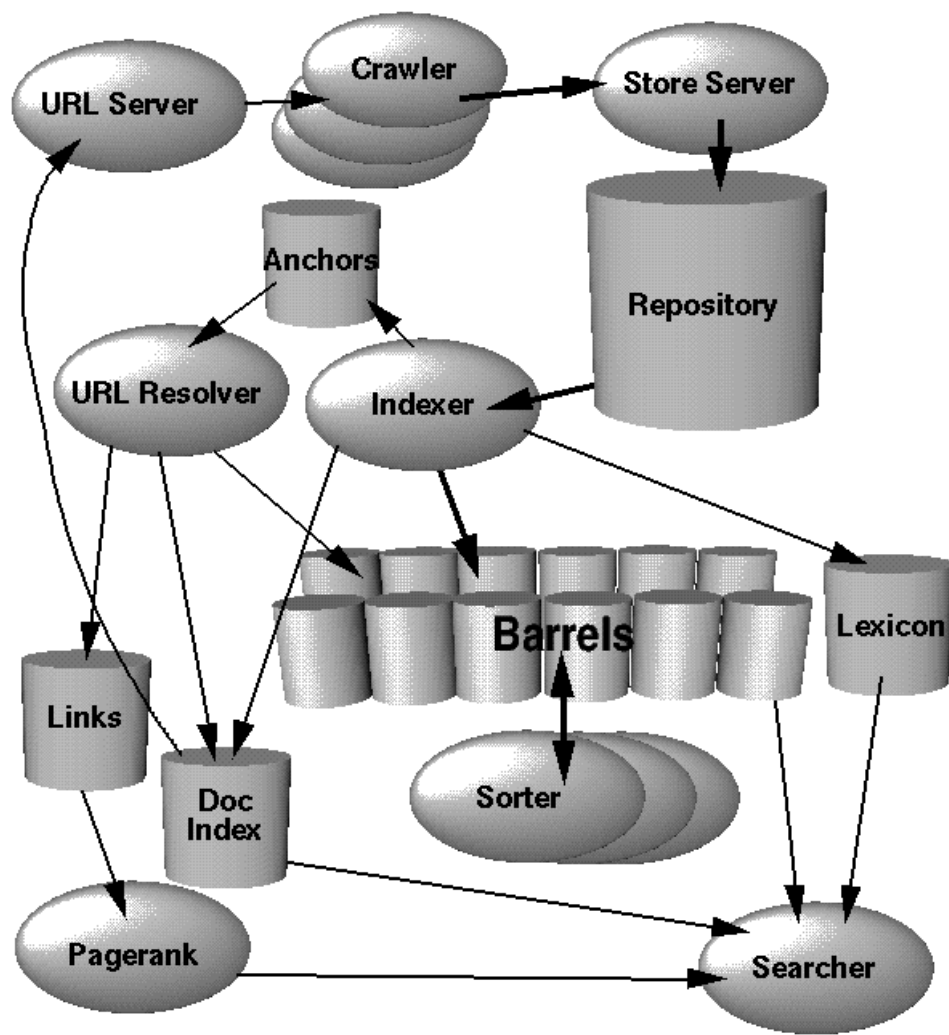


Компоненти - servers



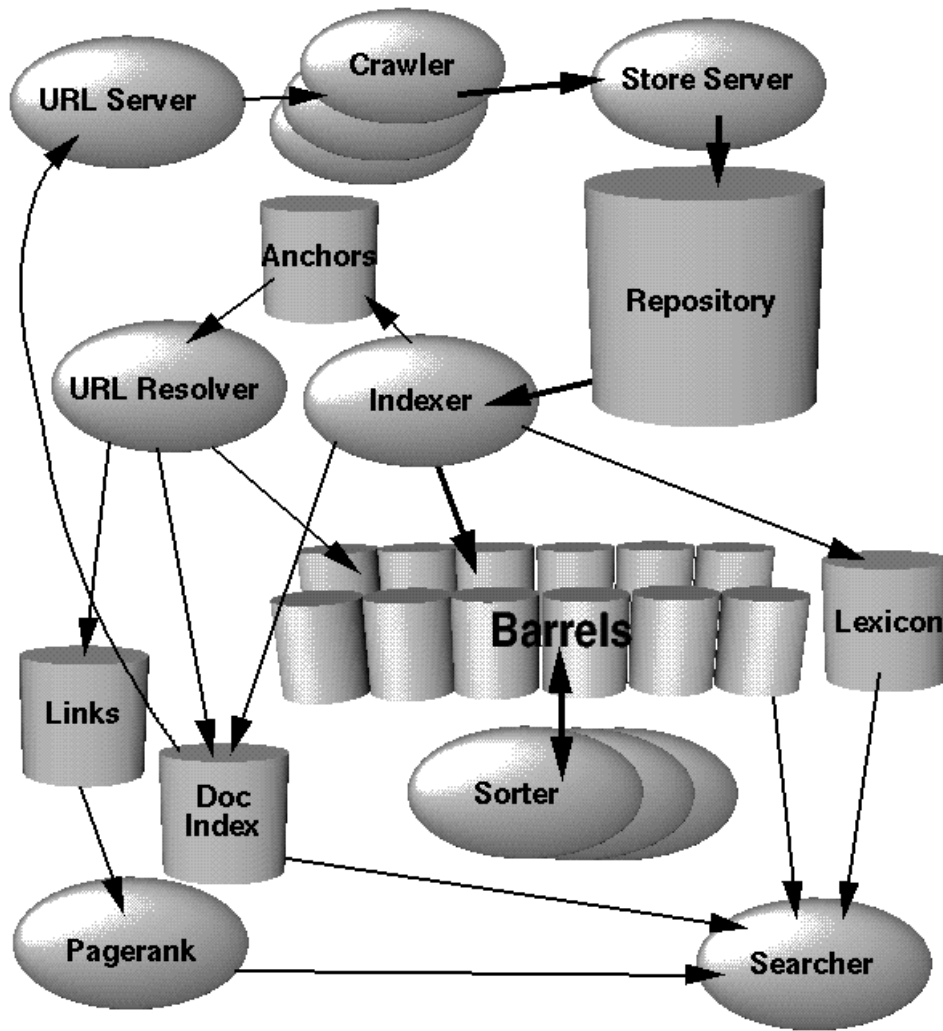
- URL server
- Store server

Компоненти - Repository



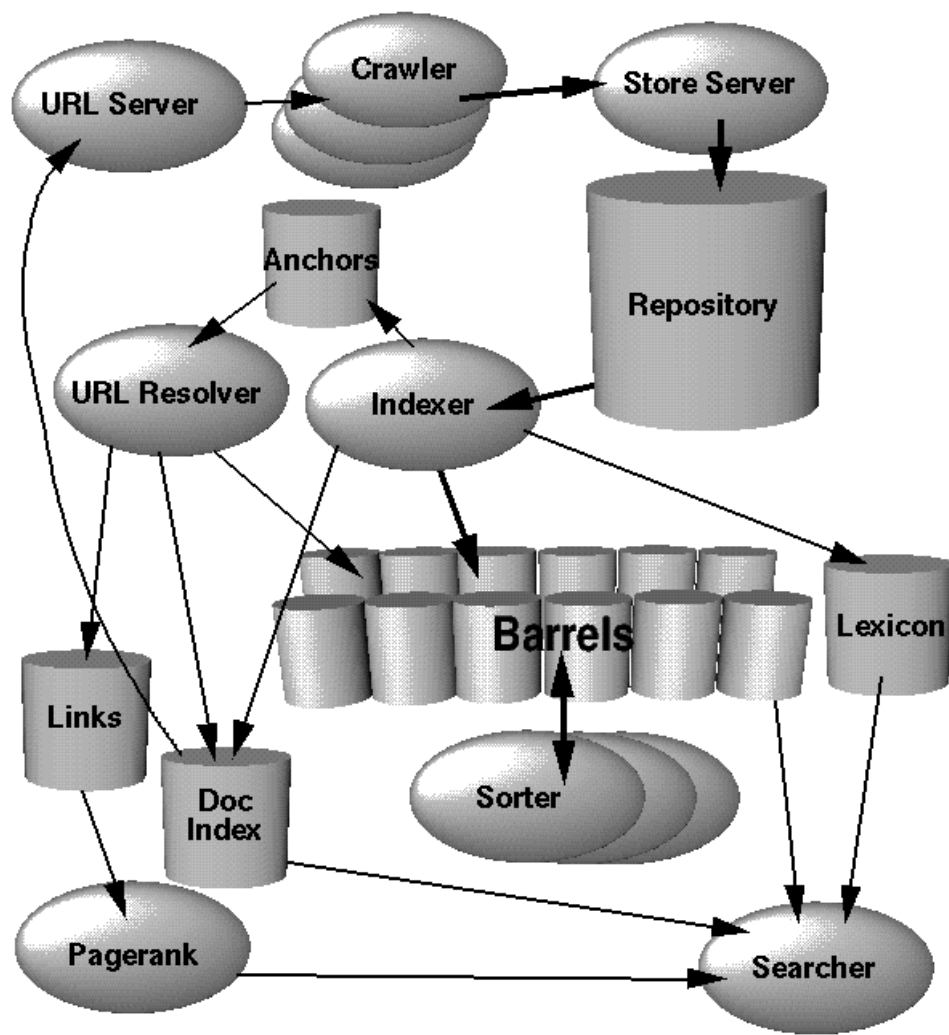
- Съдържа пълен HTML на всяка страница;
- Компресиране с zlib;
- Документите се съхраняват един след друг (docID, Len, URL, page)

Компоненти - Indexer



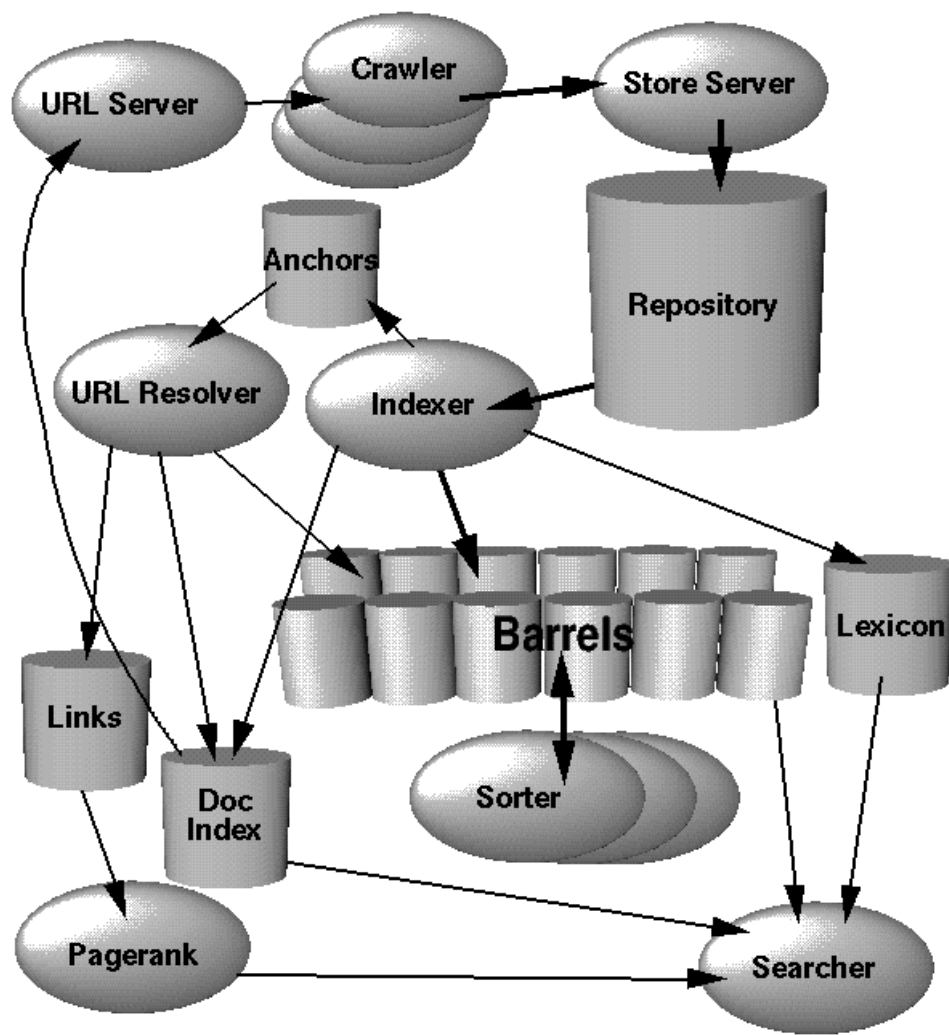
- Декомпресира документите.
- Всеки документ се конвертира в множество появявания на думи – *hits*.
- Hit – дума, позиция, размер шрифт, главни букви.
- Разпределят се в частично сортирани индекси – *barrels*.
- Парсва линковете в *anchor file*.

Компоненти – URL resolver



- Конвертира относителните URL в абсолютни;
- Поставя текста от котвата в индекс с docID, сочен от нея;
- Генерира база от линкове (линк-docID).

Компоненти – Sorter



- Сортира barrels по docID.
- Генерира инвертен индекс.
- Обновява лексикона.

Разпределени файлови системи

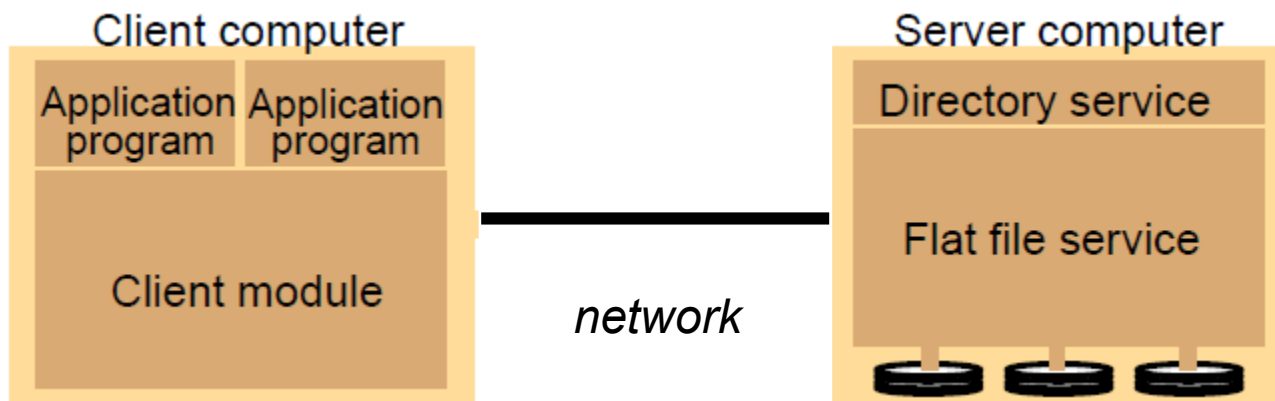
РФС – файлова система, свързваща заедно файлови системи на отделни машини в мрежата.

Файловете се съхраняват на различни машини, но са достъпни от всички в мрежата.

Особености

- Споделяне на данни между множество потребители.
- Мобилност на потребителите.
- Прозрачност на местоположението.
- Независимост на местоположението.
- Прозрачност на разширяването.
- Репликация на файлове.

Архитектура на РФС



Flat file service

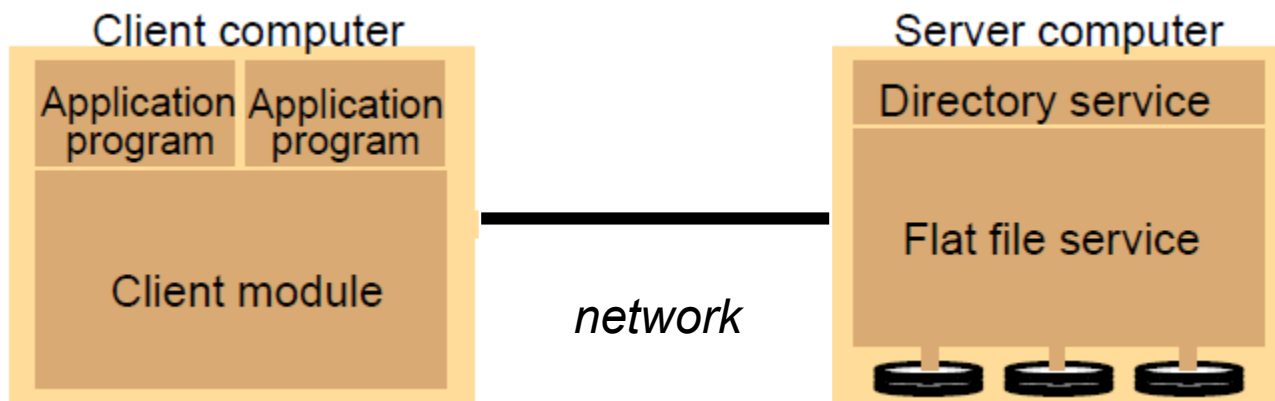
Извършва файловите операции

Използва Unique File ID (UFID) за указване на файлове

Интерфейс

- RPC механизъм за извършване на операциите
- Не се използва нормално от приложенията

Архитектура на РФС



Directory service

Свързва UFID към текстови имена на файлове и обратно

Client module

Предоставя API за файловите операции на приложенията

Примери на РФС

- Andrew File System (IBM, Open AFS)
- NFS (Unix)
- CODA (Carnegie Mellon University)
- DFS (Microsoft)

Google File System (GFS)

- Огромни обеми данни Pb.
 - Множество разпределени сървъри.
 - Хиляди заявки за секунда.
 - Копия на документи.
- Необходимост от: голяма, разпределена и отказоустойчива файлова система

Google File System (GFS)

- Отказоустойчивостта и автоматичното възстановяване трябва да са вградени
- Нов размер на блок
- Добавянето на записи е по-разпространено от чист запис (Record Append)
- Приложенията на Google и файловата система трябва да бъдат съвместно проектирани

Допускания

- Системата се изгражда от нескъпи разпространени компоненти, с възможност за чести откази.
- Системите съхраняват умерен брой големи файлове (няколко милиона, всеки по 100MB+)

Допускания

Типичен обем при операция „Четене“:

- Четене на големи потоци данни
 - Отделни операции четат хиляди Kb, типично 1Gb и повече
 - Операциите четене от един клиент са често от съседни последователни области от файл
- Малко произволни четения
 - Прочитат се няколко Kb от някакво произволно отместване
 - Някои приложения заради производителността групират малки прочитания

Допускания

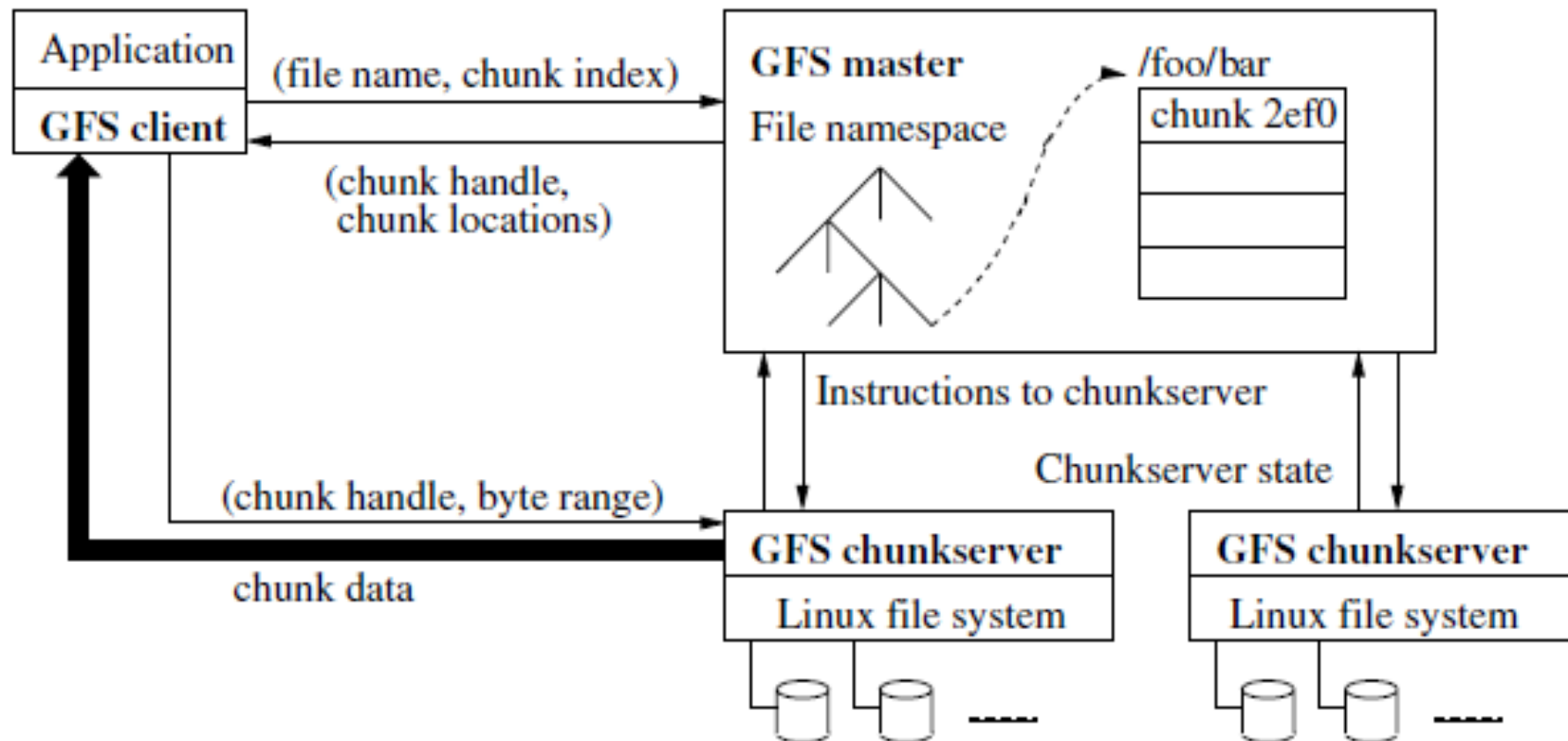
Типичен обем при операция „Запис“:

- Много големи последователни записвания, които добавят данни към файл
 - След като вече веднъж е записан, файлът рядко се модифицира отново
 - Малки записвания в произволни позиции се поддържат, но не са използвани често при работа.

Google File System (аналогия)

- При ФС на една машина:
 - Горните нива поддържат метаданни
 - Долното ниво (диск) съхранява данните в единици – *блокове*
- При GFS:
 - Master процес поддържа метаданни
 - Долното ниво (множество от *chunkservers*) съхранява данните в единици - *chunks*

Google File System - архитектура



Chunk

- Голям блок - 64Mb
- Съхранява се от chunkserver като файл
- Използват се имена за референциране (chunk handle)
- Всеки chunk е репликиран между няколко chunkservers

Master

- Единичен процес, изпълняващ се на отделна машина, съхраняващ *метаданни*:
 - Имената на файловете
 - Съответствията chunk - файл
 - Информация за местоположението на chunk
 - Информация за контрол на достъпа
 - Номер на версия на chunk

Комуникация между Master и ChunkServer

- Комуникират си регулярно (мониторинг) за получаване състоянието на chunk server:
 - Отпаднал ли е сървъра
 - Има ли отказ на диск на сървъра
 - Има ли повредени реплики
 - Кои реплики съхранява сървъра
- Master изпраща инструкции за:
 - Изтриване на съществуващ chunk
 - Създаване на нов chunk

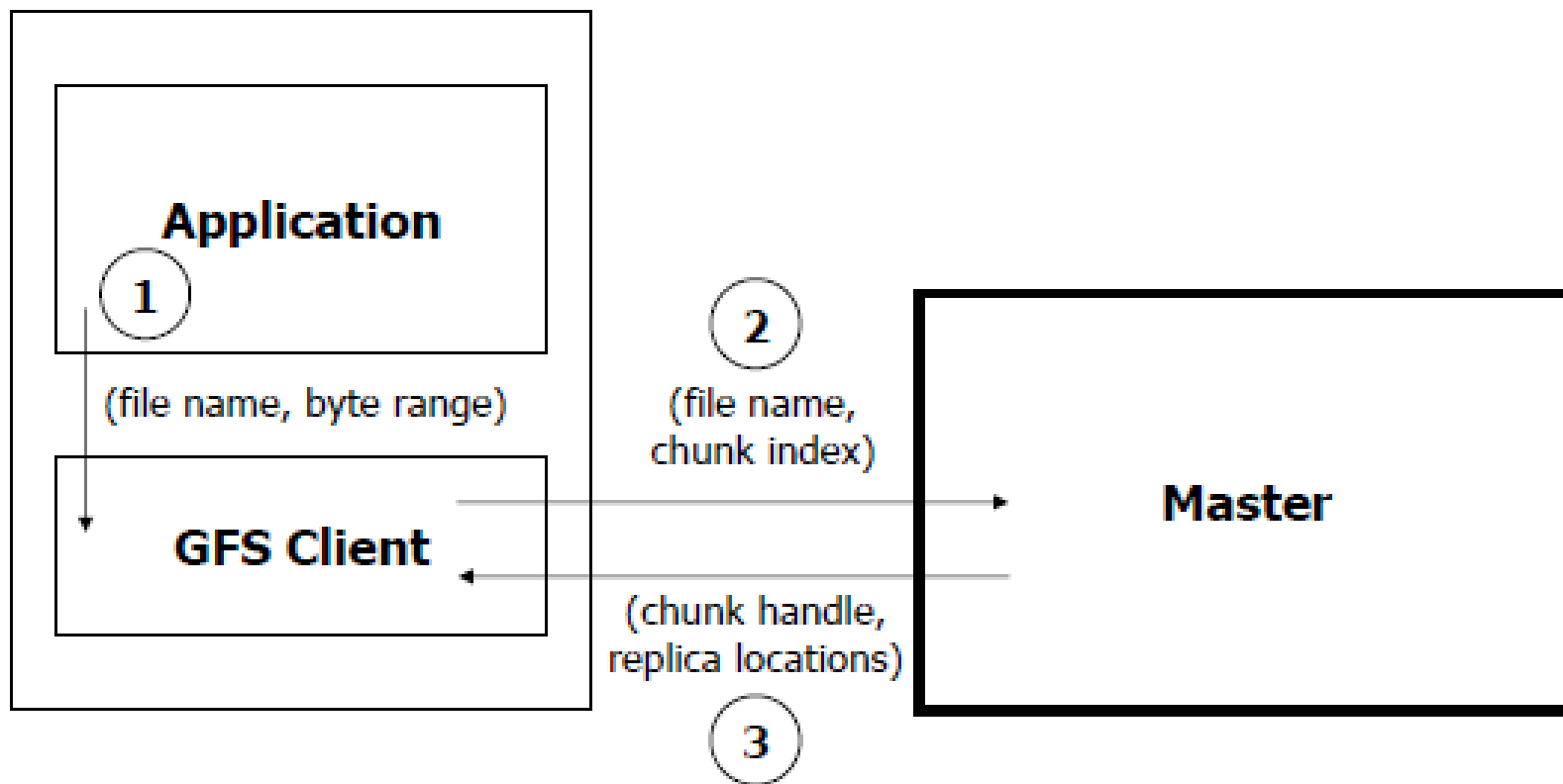
GFS клиент API

- GFS не е файлова система в традиционния вид:
 - Не е POSIX съвместима
 - Не използва VFS на ядрото
- API:
 - `open, delete, read, write`
 - `snapshot`
 - `append`

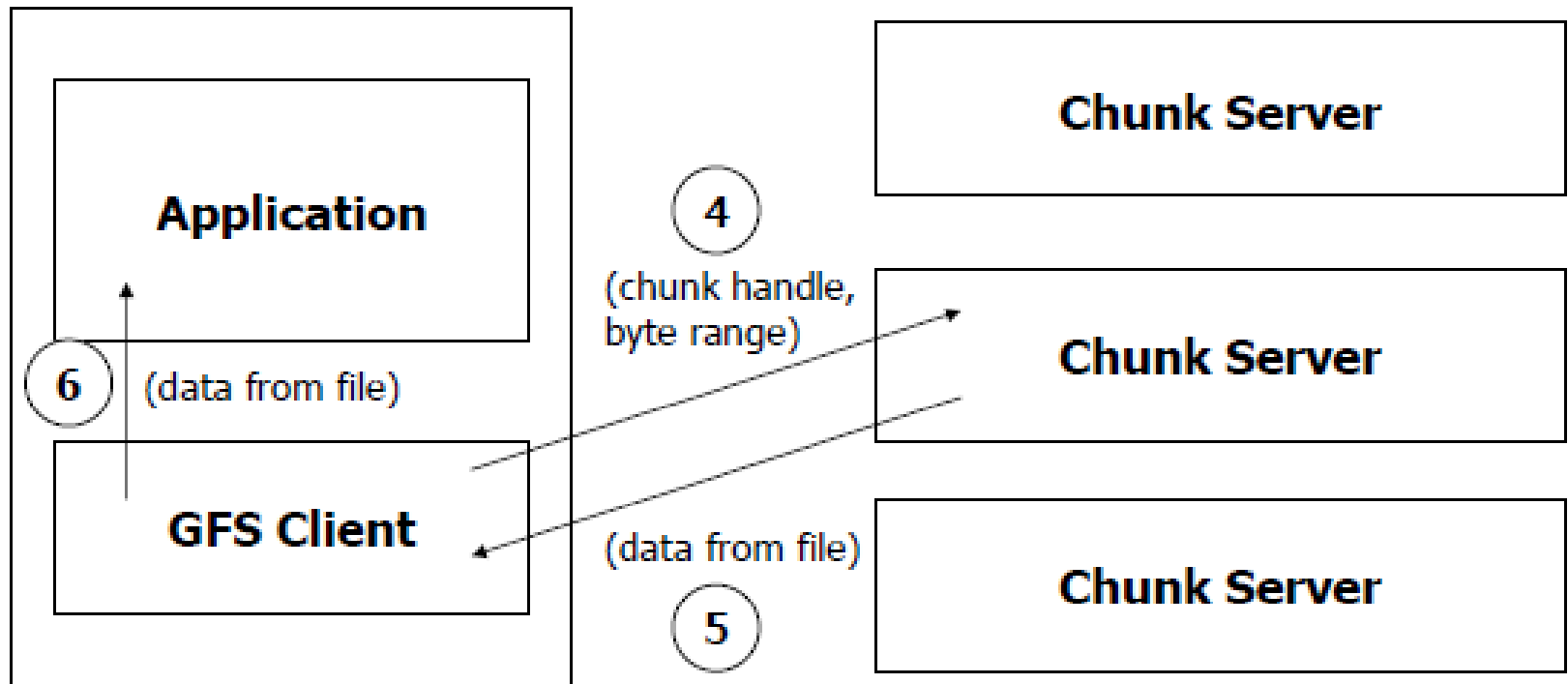
Мутации

- Два типа мутации:
 - **Write** – запис на данни в отместване във файл, дадено от приложението
 - **Record append**:
 - Операции за добавяне на данни към файл
 - Данните се добавят атомарно
 - Отместването се определя от GFS
- Трябва да се извършат за всички реплики
- Цел: минимизиране участието на master
- Потокът данни се отделя от потока за контрол

Обслужване на заявки - Read



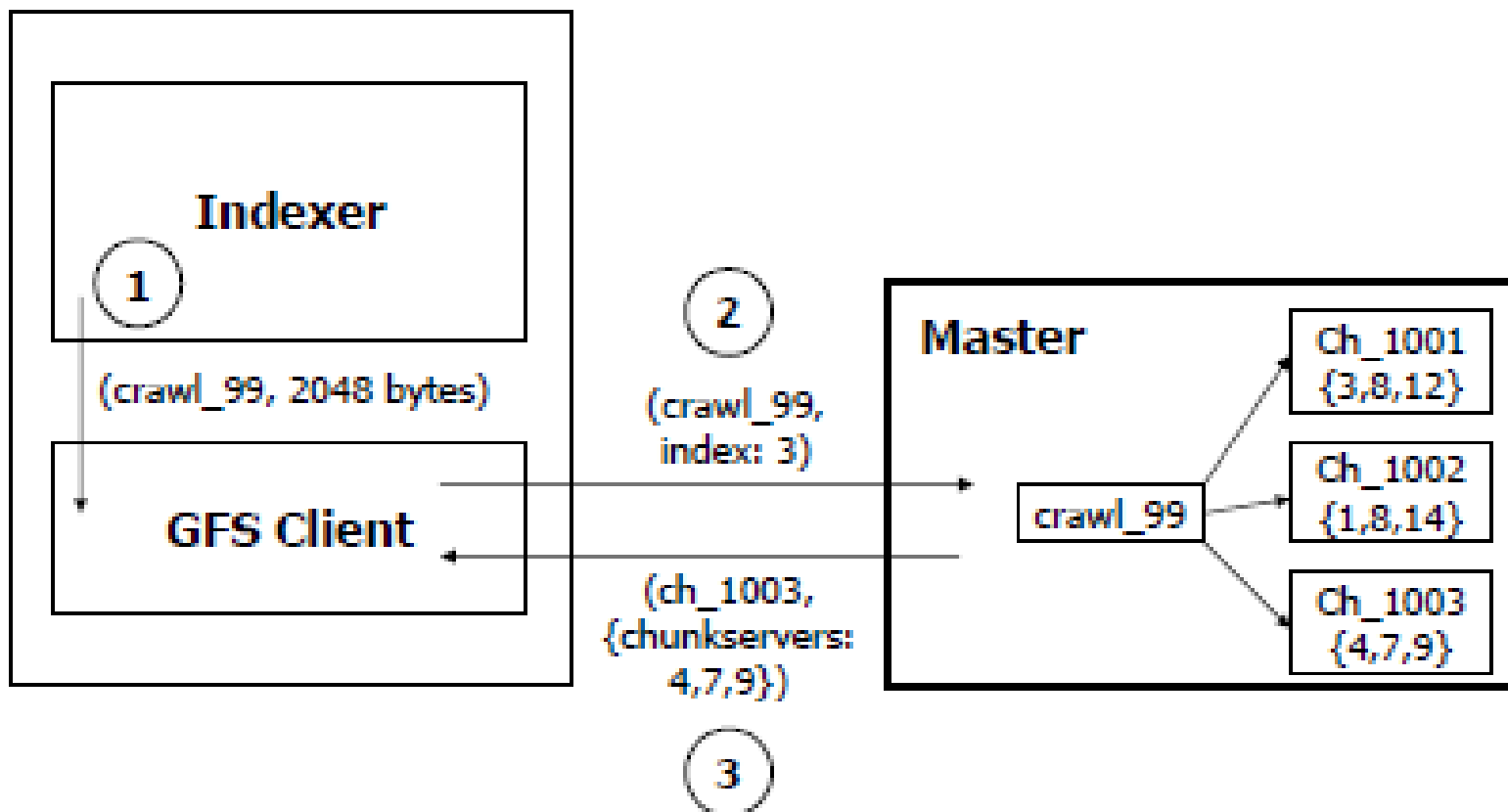
Обслужване на заявки - Read (2)



Read алгоритъм

1. Приложението генерира заявка
2. GFS клиентът транслира заявката от (filename, byte range) -> (filename, chunk index) и я изпраща към master
3. Master отговаря с chunk handle и местоположението на репликите
4. Клиентът изпраща заявка по местоположението
5. Chunkserver изпраща заявените данни към клиента
6. Клиентът препраща данните към приложението

Read алгоритъм - пример

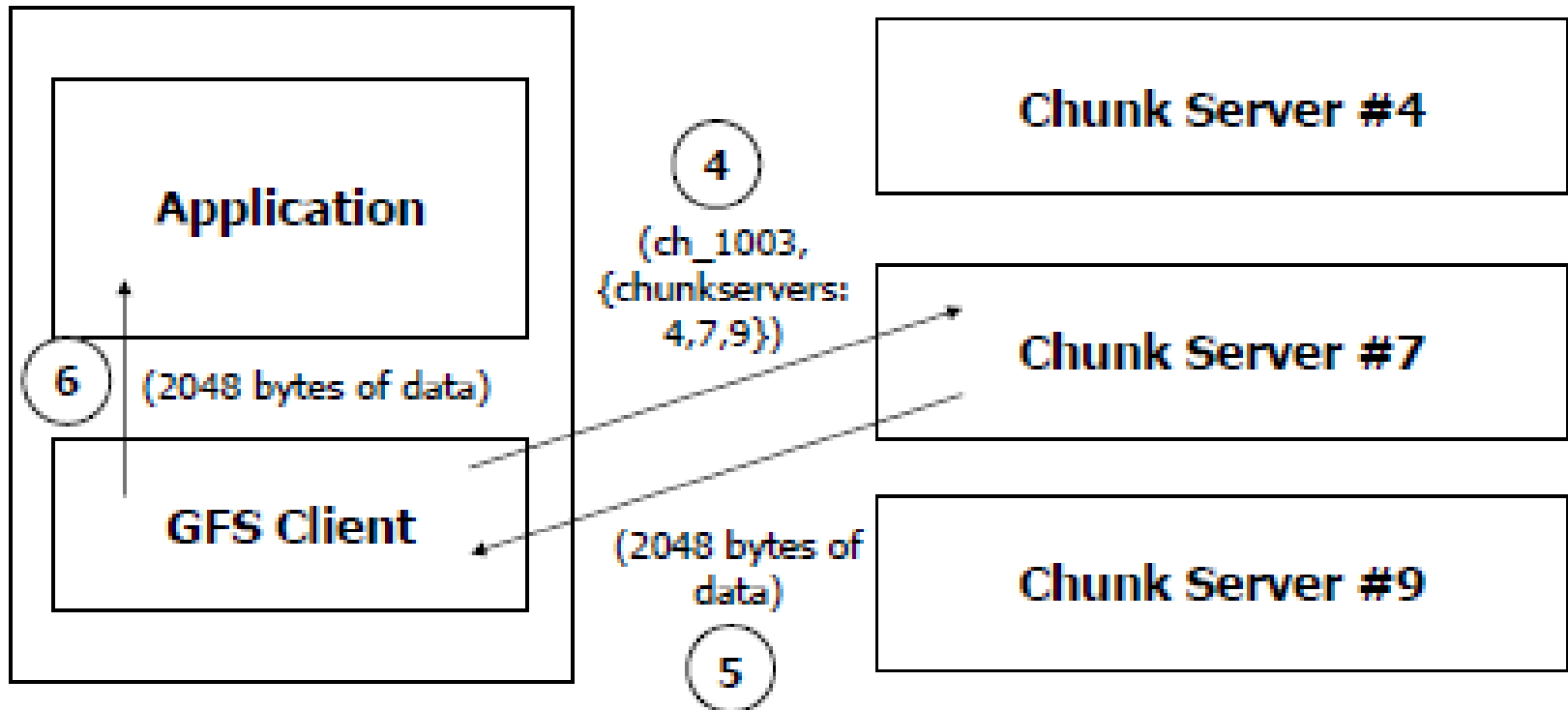


Read алгоритъм - пример

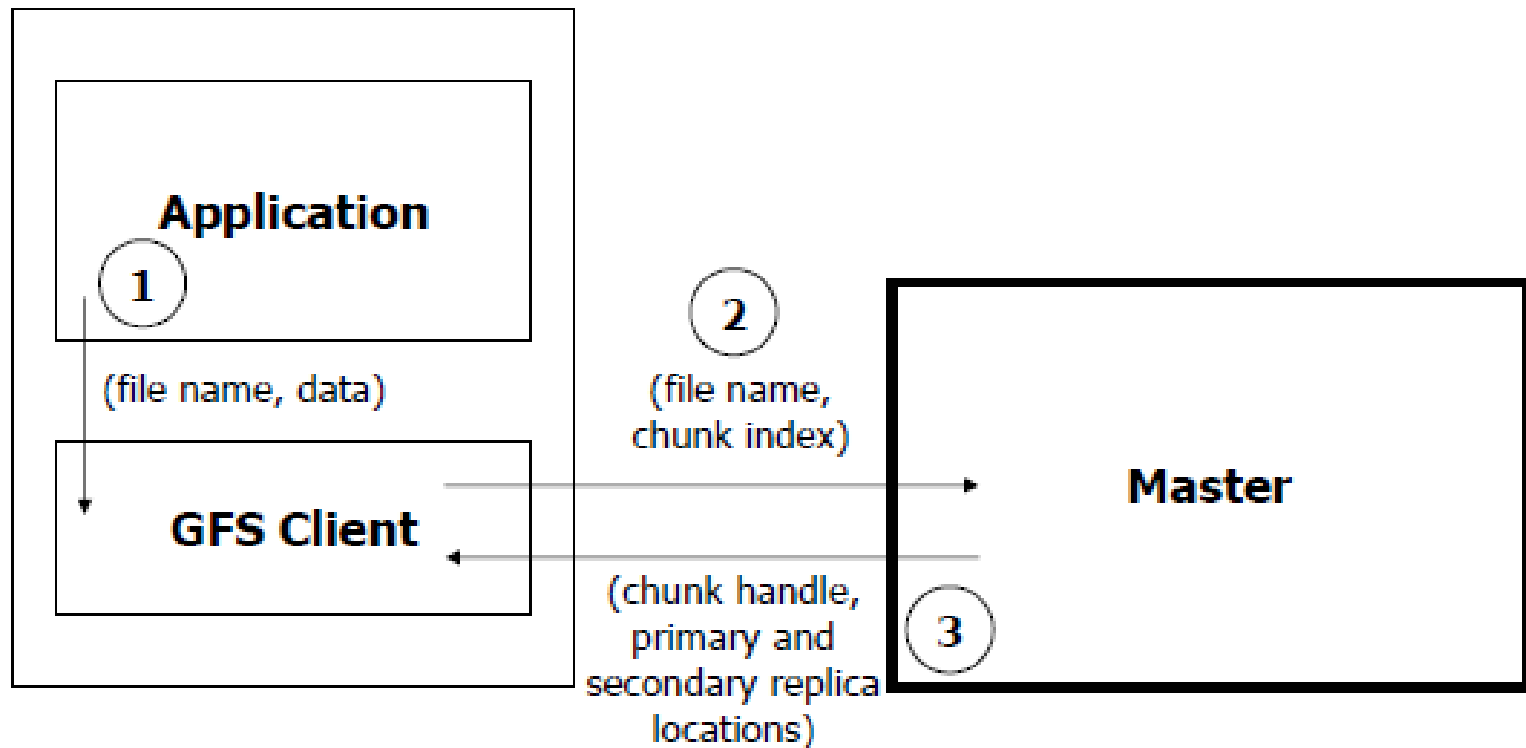
Изчисление на chunk index от byte range:
(Допускане: Позицията на файла е 201 359 161 bytes)

- Chunk size = 64 MB.
- $64 \text{ MB} = 1024 * 1024 * 64 \text{ bytes} = 67,108\,864 \text{ bytes}.$
- $201\,359\,161 \text{ bytes} = 67\,108\,864 * 2 + 32\,569 \text{ bytes}.$
- Клиентът транслира 2048 byte range -> chunk index 3

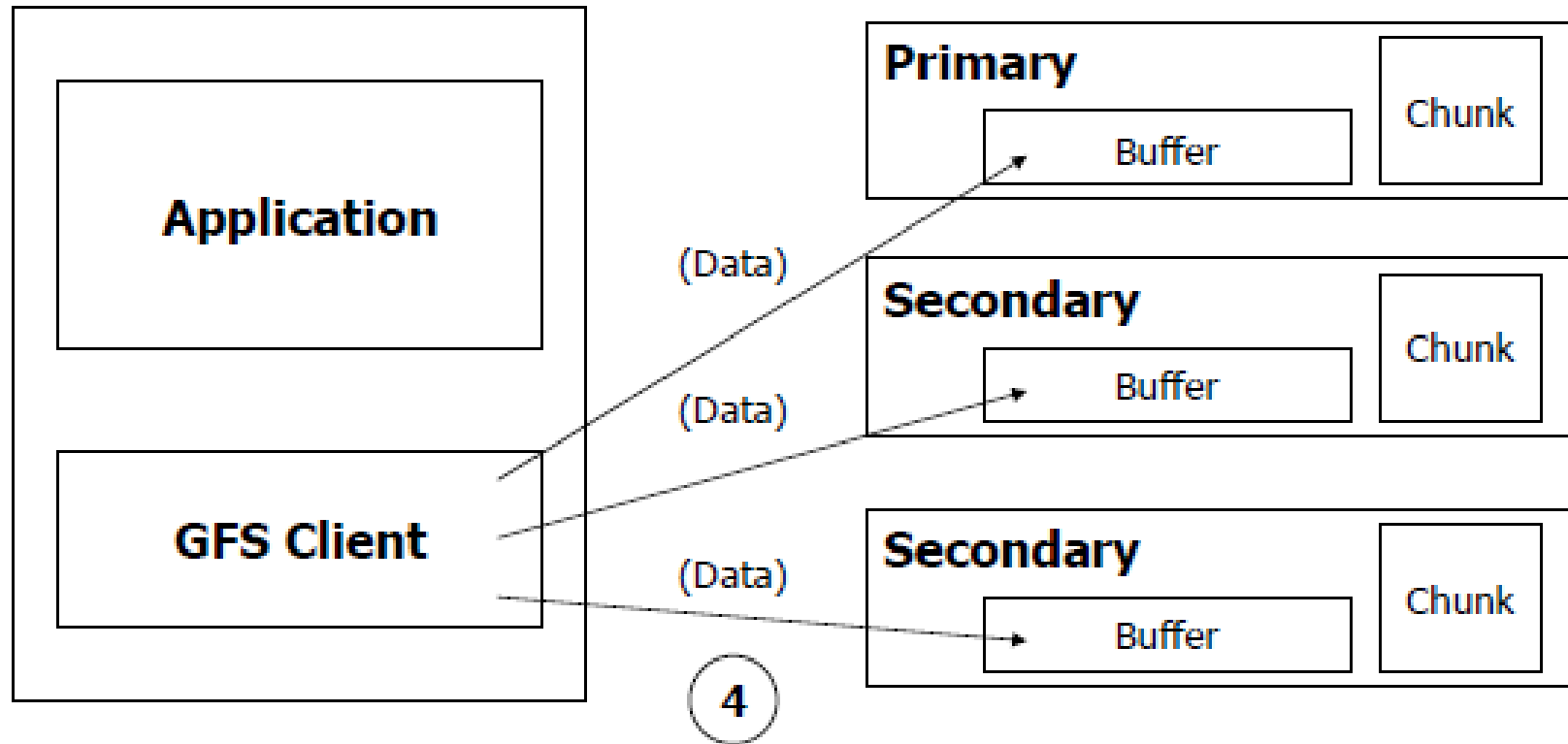
Read алгоритъм - пример



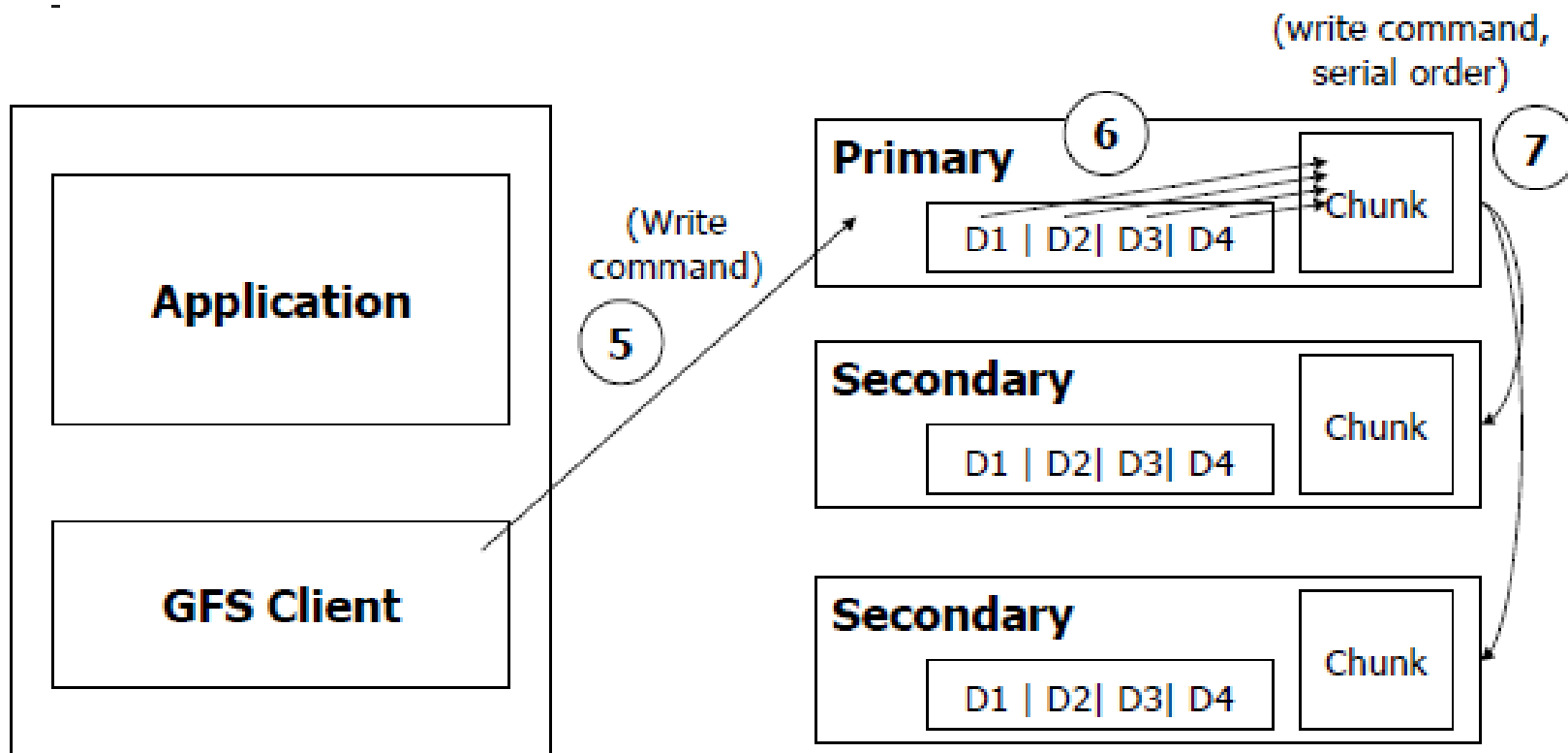
Обслужване на заявки – Write



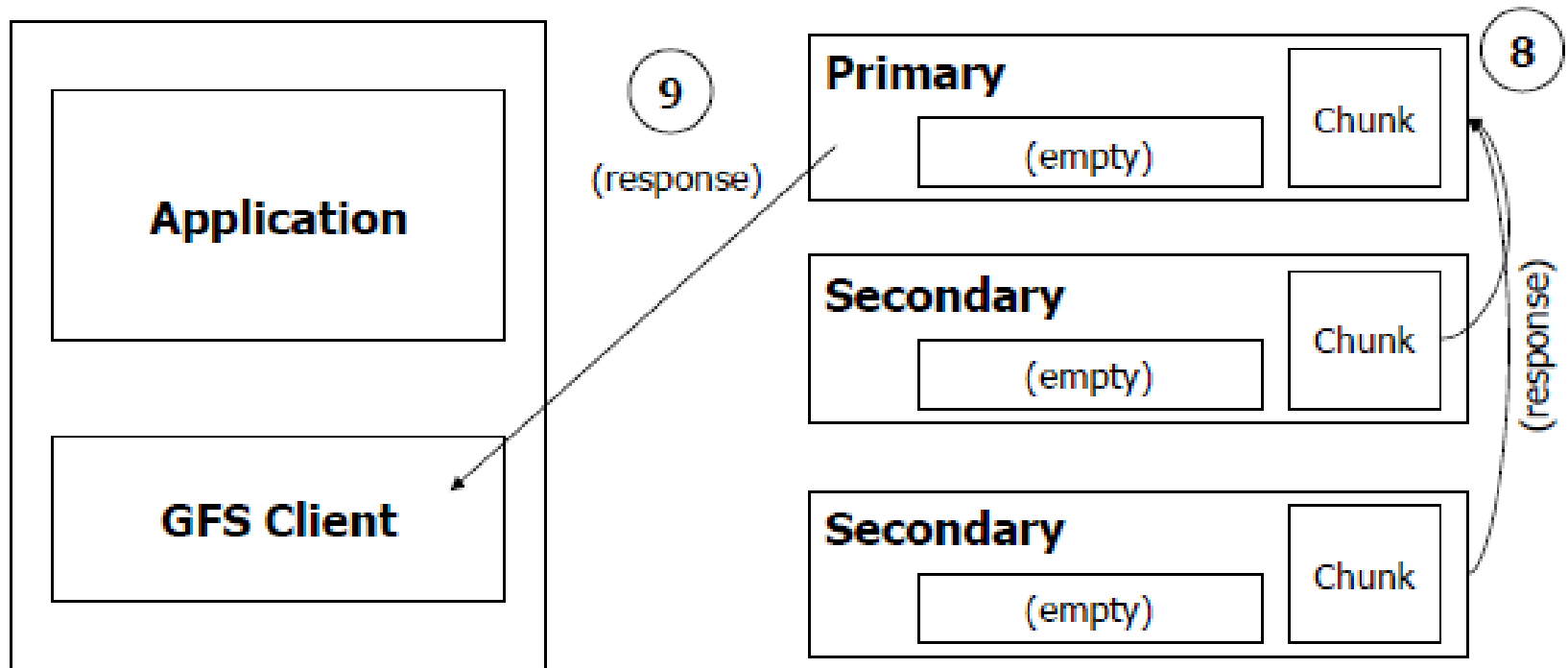
Обслужване на заявки – Write (2)



Обслужване на заявки – Write (3)



Обслужване на заявки – Write (4)



Write алгоритъм

1. Приложението генерира write заявка.
2. GFS клиентът транслира заявката от (filename, data) -> (filename, chunk index) и я изпраща към master.
3. Master отговаря с chunk handle и (primary+secondary) местоположения на репликите
4. Клиентът изпраща данните към всички местоположения. Данните се съхраняват във вътрешни буфери на chunk servers.
5. Клиентът изпраща команда към primary.

Write алгоритъм

6. Primary определя последователния ред за инстанциите на данните, съхранени в неговия буфер и записва тези инстанции в този ред в chunk.
7. Primary изпраща реда към secondaries и им указва да изпълнят запис.
8. Secondaries отговарят на primary.
9. Primary отговаря на клиента.

Ако някой запис не се изпълни, клиентът бива информиран и стартира отново операция запис.

Record Append

Специфична операция за Google:

- Обединява резултати от множество машини в един файл
- Използва файла като опашка от тип *производител-консуматор*
- Клиентът изпраща данни без отместване (GFS добавя данните като последователност от байтове)
- GFS осигурява добавяне на данните атомарно като една непрекъсната последователност от байтове

Отказоустойчивост

- Бързо възстановяване – master и chunkservers са проектирани да се рестартират в рамките на няколко секунди
- Репликация на chunk-овете
- Всички промени по метаданни – в Log

Отказоустойчивост

- Интегритет на данните – всеки 64Kb блок от chunk има контролна сума от 32bits
 - Проверка на контролната сума се извършва всеки път при четене от chunk.
 - По време на периоди на бездействие, chunk server сканира chunk и проверява контролните им суми.

Въпроси?