Hadoop Distributed File System

Sains Data IKOPIN

GitHub.com/sevinurafni/SD3204

14 Maret 2024

Traditional File System

- Pada akhir tahun 1990-an, Google menghadapi tantangan besar untuk bisa menyimpan dan memproses semua halaman di Internet serta data log web pengguna Google.
- Pada saat itu, Google menggunakan model arsitektur skala atas.
  - Meningkatkan kapasitas sistem dengan menambahkan inti CPU, RAM, dan disk ke server yang sudah ada.

Ada dua masalah utama:

1. Biaya: server yang lebih besar dengan penyimpanan yang lebih banyak menjadi sangat mahal.
2. Batasan struktural: batas dari apa yang dapat dijaga oleh arsitektur skala atas dengan mudah tercapai.

Distributed File System

- Daripada menambah kapasitas ke server yang lebih besar, engineer Google memutuskan untuk memperluas dengan menggunakan kelompok server yang lebih kecil.
  - Mereka terus menambah jika mereka membutuhkan lebih banyak daya atau kapasitas.
  - Alih-alih "satu besar", mereka menggunakan "banyak kecil”.
- Google File System (GFS) dikembangkan.

Ini menjadi inspirasi bagi para engineer yang pertama kali mengembangkan HDFS

What You Can Do with Files in HDFS

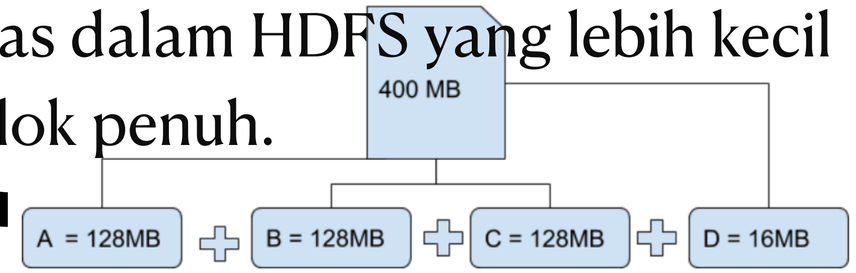
HDFS memiliki model akses data Write Once, Read Often.

Masih banyak yang bisa Anda lakukan dengan file HDFS kecuali memodifikasi file:

- Membuat file baru,
- Menambahkan konten ke akhir file,
- Menghapus file,
- Mengubah nama file,
- Mengubah atribut file seperti pemilik.

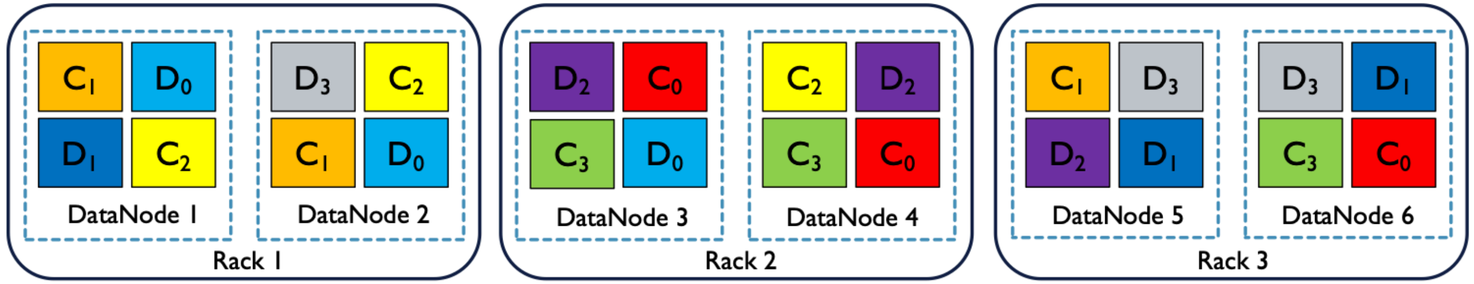
Data Blocks

- HDFS memecah file menjadi sekumpulan blok individual dan menyimpan blok-blok ini di DataNode yang berbeda dalam cluster Hadoop.
- Sebuah file dapat berukuran lebih besar daripada disk tunggal mana pun dalam jaringan.
- Ukuran blok default adalah 128MB, tetapi dapat disesuaikan.
  - 128MB dipertimbangkan untuk data dalam skala petabyte.
  - Ukuran blok pada Linux adalah 4KB.
- Tidak seperti sistem berkas untuk disk tunggal, berkas dalam HDFS yang lebih kecil dari satu blok tidak menempati penyimpanan satu blok penuh.



Data Block Replication

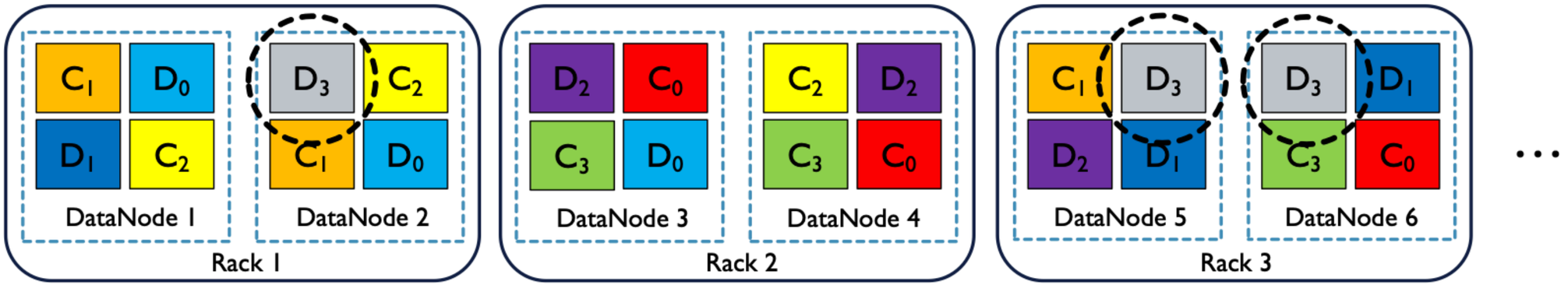
- HDFS dirancang untuk menyimpan data pada perangkat keras komoditas. ¡ Komoditas = murah = tidak dapat diandalkan.
- Dengan asumsi bahwa masing-masing node tidak dapat diandalkan, bagaimana kita dapat membuat cluster dapat diandalkan?
  - Merencanakan bencana, HDFS menyimpan 3 salinan dari setiap blok data di tempat yang berbeda.
  - Keuntungan bonus: lebih banyak kesempatan untuk menempatkan komputasi di dekat data yang dibutuhkan.



Block Placement

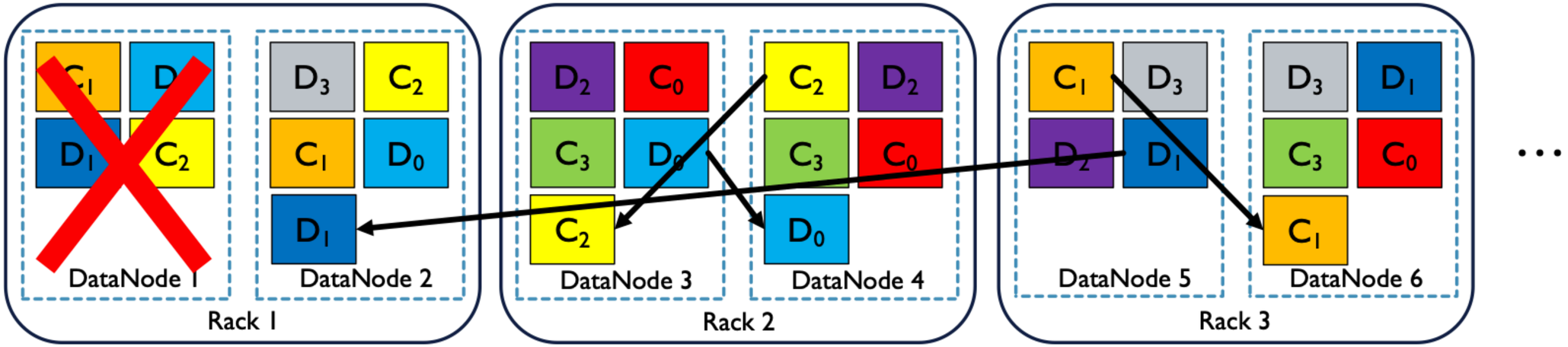
Aturan penempatan replika default HDFS adalah sebagai berikut:

- Tidak ada DataNode yang berisi lebih dari satu replika dari setiap blok.
- Tidak ada rak yang berisi lebih dari dua replika dari blok yang sama.
- Contoh:
  - Ketika D3 perlu disimpan, DataNode 2 dipilih. Salinan pertama dari D3 disimpan di sana.
  - Dua salinan D3 yang tersisa perlu disimpan di rak yang berbeda. Jadi salinan kedua disimpan di DataNode 5, Rak 3.
  - Salinan terakhir disimpan di rak yang sama dengan salinan kedua, tetapi bukan di DataNode yang sama, jadi disimpan di DataNode 6.



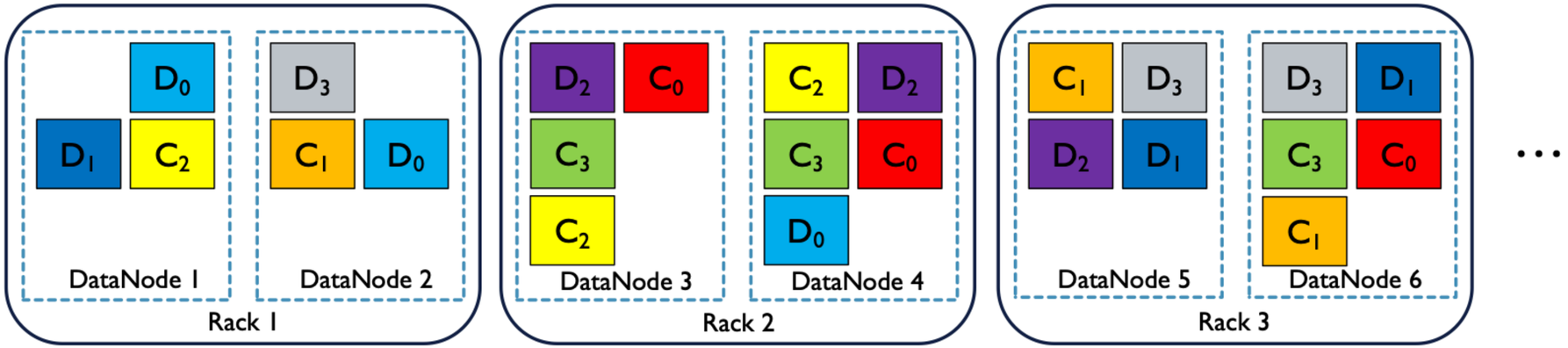
Data Block Replication

- Ketika DataNode 1 gagal, HDFS mengetahui bahwa C1, D0, D1, C2 kekurangan salinan.
- HDFS akan memerintahkan salinan baru untuk mereka.
  - Masih mengikuti aturan penempatan blok.



Data Block Replication

- Ketika DataNode 1 kembali online setelah beberapa jam, HDFS menemukan bahwa C1, D0, D1, C2 tereplikasi secara berlebihan.
- HDFS memerintahkan satu salinannya untuk dihapus.



Data Blocks

- Sebagai pengguna Hadoop, Anda tidak tahu DataNode mana yang memiliki bagian file yang perlu Anda proses. Yang Anda lihat hanyalah daftar file dalam HDFS.
- Anda tidak mengetahui kerumitan bagaimana blok-blok file didistribusikan, dan Anda tidak perlu mengetahuinya.
- Sebenarnya, DataNode sendiri bahkan tidak tahu apa yang ada di dalam blok data yang mereka simpan.
- Siapa yang mengendalikan?

NameNode: Server Metadata Pusat untuk HDFS

metadata: sekumpulan data yang mendeskripsikan dan memberikan informasi tentang data lain.

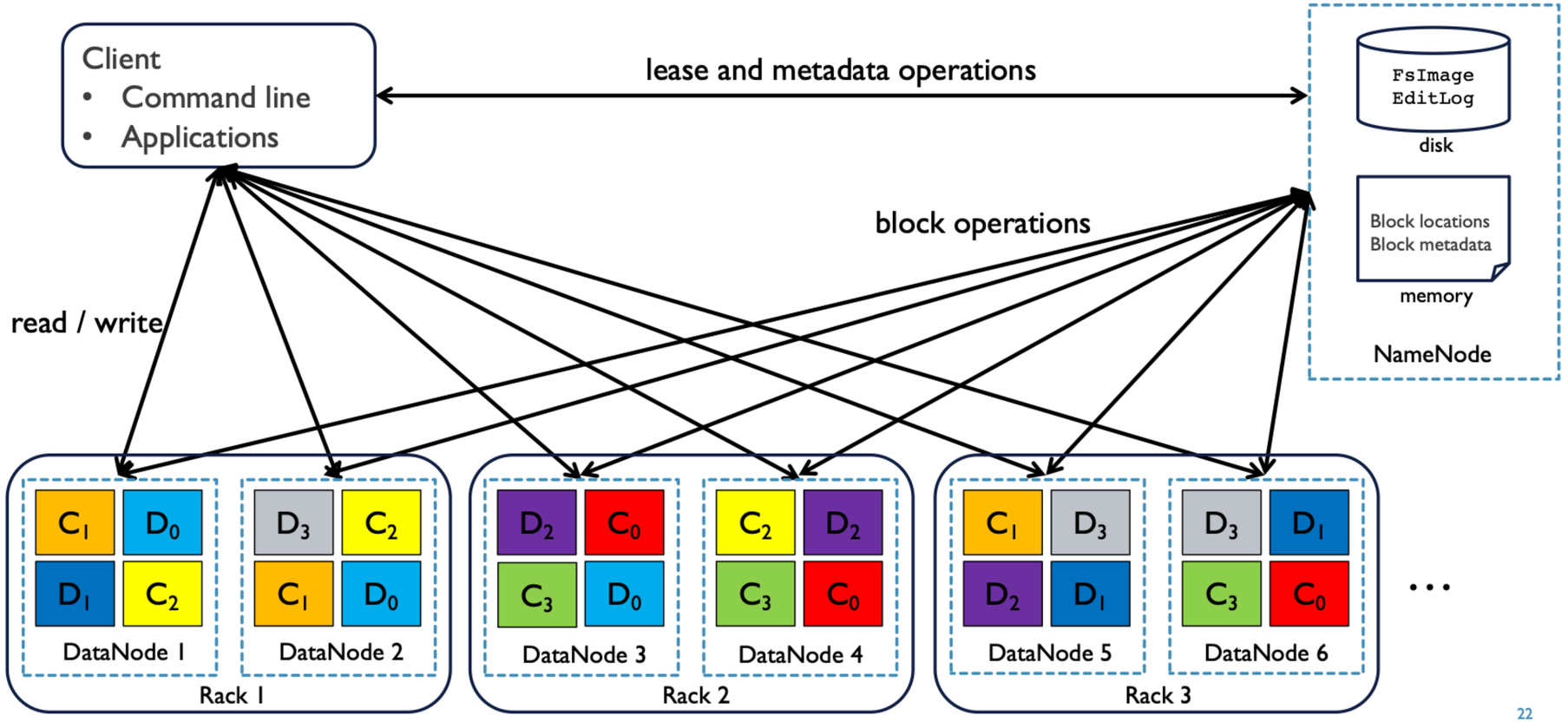
- NameNode bertindak sebagai buku alamat untuk HDFS karena ia tahu
  - blok mana saja yang membentuk berkas-berkas individual,
  - di mana masing-masing blok dan replika disimpan.
- NameNode memelihara dua berkas:
  - FsImage: semua informasi pemetaan yang berhubungan dengan blok data dan berkas-berkas terkait.
  - EditLog: setiap perubahan data sejak pemeriksaan terakhir.

Startup dan Pengoperasian NameNode

daemon: program komputer yang berjalan sebagai proses latar belakang

- Untuk memuat semua informasi yang dibutuhkan NameNode setelah dinyalakan, berikut ini yang terjadi:
    1. NameNode memuat file FsImage ke dalam memori.
    2. NameNode memuat EditLog dan memainkan kembali perubahan yang telah dijurnal untuk memperbarui blok.
    3. metadata yang sudah ada di memori.
    4. Daemon DataNode mengirimkan laporan blok NameNode.
    5. Setelah proses startup, NameNode memiliki gambaran lengkap tentang semua data yang tersimpan di HDFS, dan siap menerima permintaan aplikasi dari klien Hadoop.

Interaksi antara komponen HDFS



Writing Data:Writer's Lease

1. Klien mengirimkan permintaan ke NameNode untuk membuat file baru dan diberikan sewa untuk membuat blok file baru dalam cluster dari NameNode.
   - Hak guna penulis mencegah klien lain untuk menulis berkas, tetapi tidak mencegah pembacaan.
     - HDFS mengimplementasikan model satu penulis, banyak pembaca. Sebuah berkas dapat memiliki banyak pembaca secara bersamaan.
   - Klien penulis secara berkala memperbarui sewa dengan mengirimkan detak jantung ke NameNode.
   - Durasi sewa terikat oleh batas lunak dan batas keras.
     - Jika batas lunak berakhir dan klien gagal menutup berkas atau memperbarui sewa, klien lain dapat mendahului sewa.
     - Jika setelah batas keras berakhir (satu jam) dan klien gagal memperbarui penyewaan, HDFS secara otomatis menutup berkas atas nama penulis, dan memulihkan penyewaan.

Batas Lunak dan Batas Keras

- **Batas Lunak (Soft Limit)**:
  - Batas lunak adalah waktu yang ditetapkan di mana sewa suatu berkas atau sumber daya akan berakhir.
  - Ketika batas lunak berakhir, klien yang memiliki sewa atas berkas atau sumber daya tersebut harus memperbarui sewa mereka. Jika klien gagal melakukannya, maka sewa tersebut menjadi "jangka waktu tidak aktif".
  - Selama jangka waktu tidak aktif, klien lain dapat mengambil alih atau memperoleh sewa atas berkas atau sumber daya yang kadaluwarsa.
  - Batas lunak memberikan kesempatan kepada klien untuk memperbarui sewa mereka sebelum sewa tersebut berakhir sepenuhnya.
- **Batas Keras (Hard Limit)**:
  - Batas keras adalah waktu maksimum yang diizinkan untuk sewa suatu berkas atau sumber daya.
  - Jika batas keras berakhir dan klien masih gagal memperbarui sewa mereka, HDFS secara otomatis akan mengambil tindakan tertentu untuk menangani situasi tersebut.
  - Salah satu tindakan yang dapat diambil oleh HDFS ketika batas keras tercapai adalah menutup berkas atas nama klien penulis (yaitu, klien yang bertanggung jawab untuk menulis atau memperbarui berkas tersebut).
  - Setelah berkas ditutup, HDFS akan memulihkan sewa dan mengambil langkah-langkah yang sesuai untuk mengelola sumber daya yang tersisa.

Writing Data: Repication Pipelining

2. NameNode mengalokasikan blok dengan ID blok yang unik dan menentukan daftar DataNode untuk menampung replika blok.

- Algoritma pemilihan replika mengikuti aturan penempatan blok.
- DataNode membentuk pipa, yang urutannya meminimalkan total jarak jaringan dari klien ke DataNode terakhir.
- Setiap DataNode di dalam pipeline
  - menerima data dalam bentuk paket
  - menulis setiap paket ke tempat penyimpanan lokalnya
  - mentransfer paket tersebut ke DataNode berikutnya.

Writing Data: Acknowledge

3. Setelah daemon DataNode mengakui bahwa blok file telah dibuat, aplikasi klien menutup file dan memberi tahu NameNode, yang kemudian menutup penyewaan terbuka.

- Garis putus-putus pada gambar adalah acknowledge.

Reading Data

Klien mengirimkan permintaan ke NameNode untuk sebuah file.

- NameNode menentukan daftar blok dan lokasi setiap replika blok.
- Lokasi setiap blok diurutkan berdasarkan jaraknya dari pembaca.
- Ketika membaca isi blok, klien mencoba replika yang paling dekat terlebih dahulu. Jika upaya pembacaan gagal, klien akan mencoba replika berikutnya secara berurutan.

Data Integrity

- Ada kemungkinan blok data yang diambil dari DataNode tiba dalam keadaan rusak.
  - Kesalahan pada perangkat penyimpanan, kesalahan jaringan, perangkat lunak yang bermasalah...
- Perangkat lunak klien HDFS mengimplementasikan pemeriksaan checksum pada konten file HDFS.
  - Ketika klien membuat file HDFS, klien menghitung checksum dari setiap blok file dan menyimpan checksum ini dalam file tersembunyi yang terpisah dalam ruang nama HDFS yang sama.
  - Ketika klien mengambil konten file, klien akan memverifikasi bahwa data tersebut cocok dengan checksum.
  - Jika tidak, maka klien dapat memilih untuk mengambil blok tersebut dari DataNode lain yang memiliki replika blok tersebut.

Rebalancing

Seiring berjalannya waktu, data cenderung tidak terdistribusi secara merata di seluruh rak dan DataNode.
Distribusi yang tidak merata ini dapat berdampak buruk pada kinerja karena permintaan pada masing-masing DataNode akan menjadi tidak seimbang.

- Node dengan sedikit data tidak akan digunakan sepenuhnya.
- Node dengan banyak blok akan digunakan secara berlebihan.

Sebuah cluster dikatakan seimbang jika untuk setiap DataNode kondisi berikut terpenuhi:

$$\left|\frac{used\ space\ at\ the\ node}{total\ capacity\ of\ the\ node} - \frac{used\ space\ in\ the\ cluster}{total\ capacity\ of\ the\ cluster}\right| > threshold$$

- Ini secara iteratif memindahkan replika dari DataNode dengan utilisasi yang lebih tinggi ke DataNode dengan utilisasi yang lebih rendah.
- Juga mengikuti aturan penempatan blok.

Decommissioning

- Administrator cluster memiliki daftar alamat host node yang diizinkan untuk mendaftar.
- Ketika sebuah DataNode dihapus dari daftar, prosesnya adalah:
  - DataNode ditandai untuk dinonaktifkan.
  - DataNode tidak akan dipilih sebagai target penempatan replika, tetapi akan terus melayani permintaan baca.
  - NameNode mulai menjadwalkan replikasi bloknya ke DataNode lain.
  - Setelah NameNode mendeteksi bahwa semua bloknya telah direplikasi, node tersebut akan dihapus dengan aman dari klaster tanpa membahayakan ketersediaan data.

Tugas 3

Tujuan HDFS.

Bagaimana data disimpan dalam HDFS.

Apa saja tugas DataNode dan NameNode. ¡ Bagaimana HDFS menangani kegagalan.

Cara menulis/membaca data dengan HDFS.