Cara Kerja Hadoop MapReduce

Apa itu MapReduce

MapReduce adalah suatu software framework dan programming model yang digunakan untuk pemrosesan jumlah data yang besar. Program **MapReduce** bekerja dalam dua fase, yaitu Map dan Reduce. Tugas Map berurusan dengan splitting dan mapping dari data sedangkan tugas Reduce melakukan shuffle dan reduce terhadap data.

Hadoop mampu menjalankan program MapReduce yang ditulis dalam berbagai bahasa: Java, Ruby, Python, dan C++. Program MapReduce sifatnya parallel, jadi sangat berguna bagi pelaksanaan analisis data skala besar menggunakan banyak mesin di dalam cluster.

Inputan atau masukan untuk setiap fase adalah pasangan **key-value**. Setiap programmer harus menetapkan dua fungsi: fungsi **map** dan fungsi **reduce**.

Dalam tutorial ini kita akan mendiskusikan:

- Apa itu MapReduce di dalam Hadoop?
- Bagaimana MapReduce Bekerja? Proses lengkap
- Arsitektur MapReduce Secara Rinci
- Bagaimana MapReduce Mengelola Kerja?

Bagaimana MapReduce Bekerja? Proses Lengkap

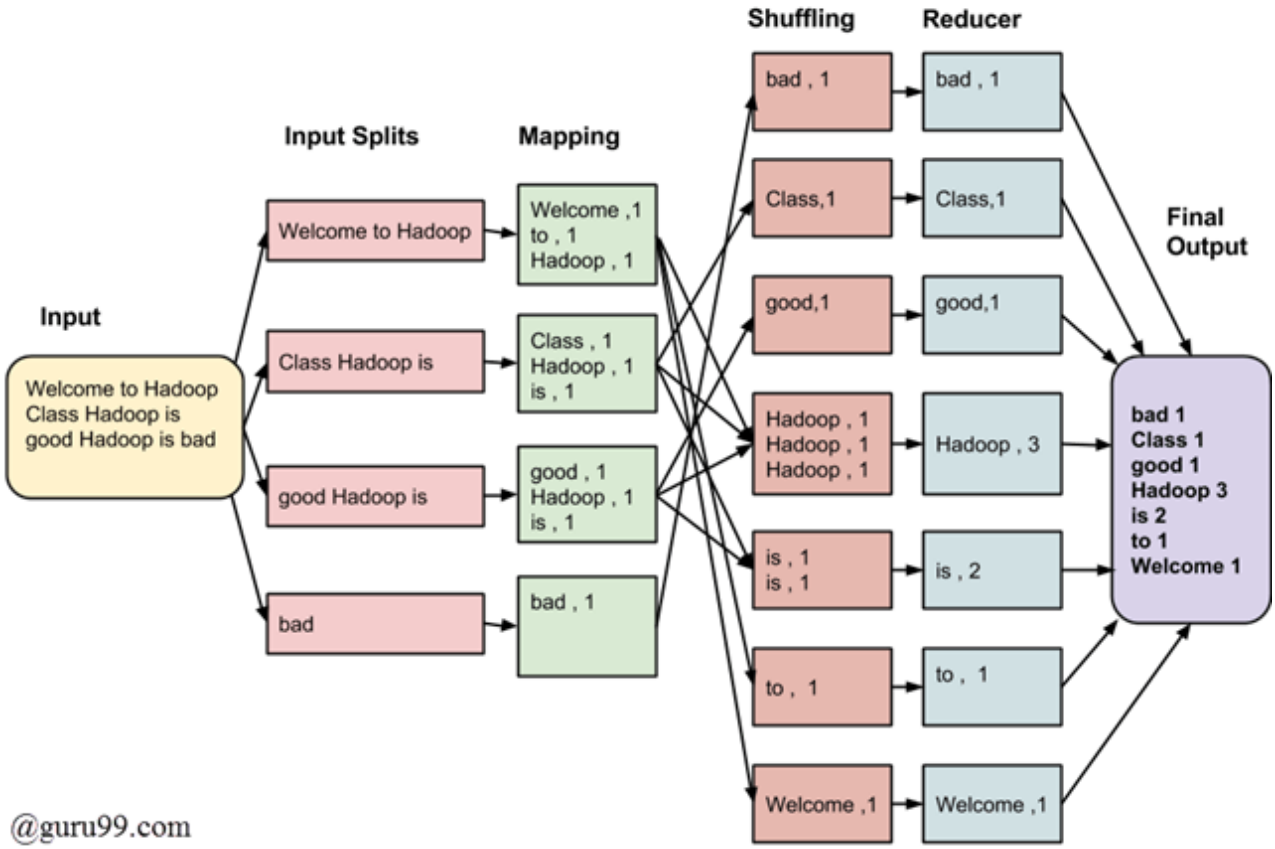
Proses keseluruhan bergerak melalui empat fase eksekusi, yaitu splitting, mapping, shuffling, dan reducing.

Mari kita memaha mi ini dengan suatu contoh umum, yaitu menghitung jumlah kemunculan kata dalam suatu dokumen.Katakanlah kita mempunyai data input untuk program MapReduce seperti di bawah ini

```
Welcome to Hadoop Class
Hadoop is good
Hadoop is bad
```

Proses ***splitting*** (memecah input menjadi segmen kecil atau split), ***mapping*** (pemetaan jumlah kemunculan kata dalam setiap split), ***shuffling*** (menata atau mengurutkan kata sesuai abjad) dan ***reducing*** (menyimpulkan) diperlihatkan pada Gambar 1. Hasil akhir dari pekerjaan MapReduce ini adalah

bad	1
Class	1
good	1
Hadoop	3
is	2
to	1
Welcome	1



Gambar 1. Arsitektur MapReduce

Data bergerak melalui fase-fase berikut:

- **Input Splits:**

  Suatu input ke kerja MapReduce dibagi ke dalam potongan-potongan berukuran tetap bernama **input splits** Input split merupakan potongan input yang akan diproses oleh suatu map tunggal.

- **Mapping**

  Ini merupakan fase paling awal dalam eksekusi program map-reduce. Dalam fase ini data dalam setiap split dilewatkan ke fungsi mapping untuk menghasilkan nilai-nilai output. Dalam contoh kita, suatu kerja fase mapping adalah menghitung jumlah kemunculan setiap kata dari input splits (lebih rinci mengenai input-split dibahas di halaman selanjutnya) dan menyiapkan suatu daftar (list) berbentuk <kata, frekuensi>.

- **Shuffling**

  Fase ini menerima output dari fase Mapping. Tugasnya adalah menggabungkan record-record yang relevan dari output fase Mapping. Dalam contoh di atas, kata yang sama dikumpulkan bersama dengan frekuensinya masing-masing.

- **Reducing**

  Dalam fase ini, nilai output dari fase Shuffling diaggregasikan (digabungkan). Fase ini mengkomb inasikan nilai-nilai dari fase Shuffling dan mengembalikan suatu nilai output tunggal. Singkatnya, fase ini merangkum dataset lengkap.

Dalam contoh di atas, fase ini menggabungkan nilai fase sebelumnya, yaitu menghitung kemunculan total dari setiap kata.

Arsitektur MapReduce Secara Rinci

- Satu tugas map dibuat untuk setiap split yang kemudian mengeksekusi fungsi map untuk setiap record di dalam split tersebut.
- Adalah selalu bermanfaat untuk mempunyai banyak split karena waktu yang diperlukan untuk memroses suatu split adalah kecil dibandingkan waktu yang dihabiskan untuk memroses input keseluruhan. Pada saat the split lebih kecil, pemrosesan adalah lebih baik untuk di-load balance-kan karena kita memroses split-solit secara paralel.
- Akan tetapi, juga tidak diinginkan mempunyai split terlalu kecil ukurannya. Saat split terlalu kecil, *overload* dari pengelolaan split dan pembuatan tugas map mulai mendominasi waktu eksekusi pekerjaan total.
- Untuk kebanyakan pekerjaan, adalah lebih baik membuat ukuran split yang sama dengan ukuran blok HDFS (yaitu 64 MB, secara default).
- Eksekusi tugas map mengakibatkan penulisan output ke disk lokal pada node masing-masing dan tidak ke HDFS.
- Alasan pemilihan disk lokal daripada HDFS, adalah menghindari replikasi yang terjadi pada saat operasi penyimpanan HDFS.
- Output Map merupakan output tengahan (luaran antara) yang diproses oleh tugas reduce untuk menghasilkan output akhir (*final*).
- Segera sesudah pekerjaan selesai, output map dapat dibuang. Sehingga menyimpannya dalam HDFS dengan replikasi menjadi *overkill*.
- Jika terjadi kegagalan node, sebelum output map diterima oleh tugas reduce, Hadoop menjalannkan-ulang (rerun) tugas map pada node lain dan membuat ulang output map.
- Tugas reduce tidak bekerja pada konsep lokalitas data. Suatu output dari setiap tugas map dimakan oleh tugas reduce. Output map ditransfer ke machine dimana tugas reduce sedang berjalan.
- Pada mesin ini, output digabung dan kemudian dilewatkan ke fungsi reduce yang *user-defined*.
- Tidak seperti output map, output reduce disimpan dalam HDFS (replikasi pertama disimpan pada node lokal dan replika lain disimpan pada node-node *off-rack*). Jadi penulisan output reduce.

Bagaimana MapReduce Mengatur Pekerjaan?

Hadoop membagi pekerjaan dalam tugas-tugas. Ada dua jenis tugas, yaitu:

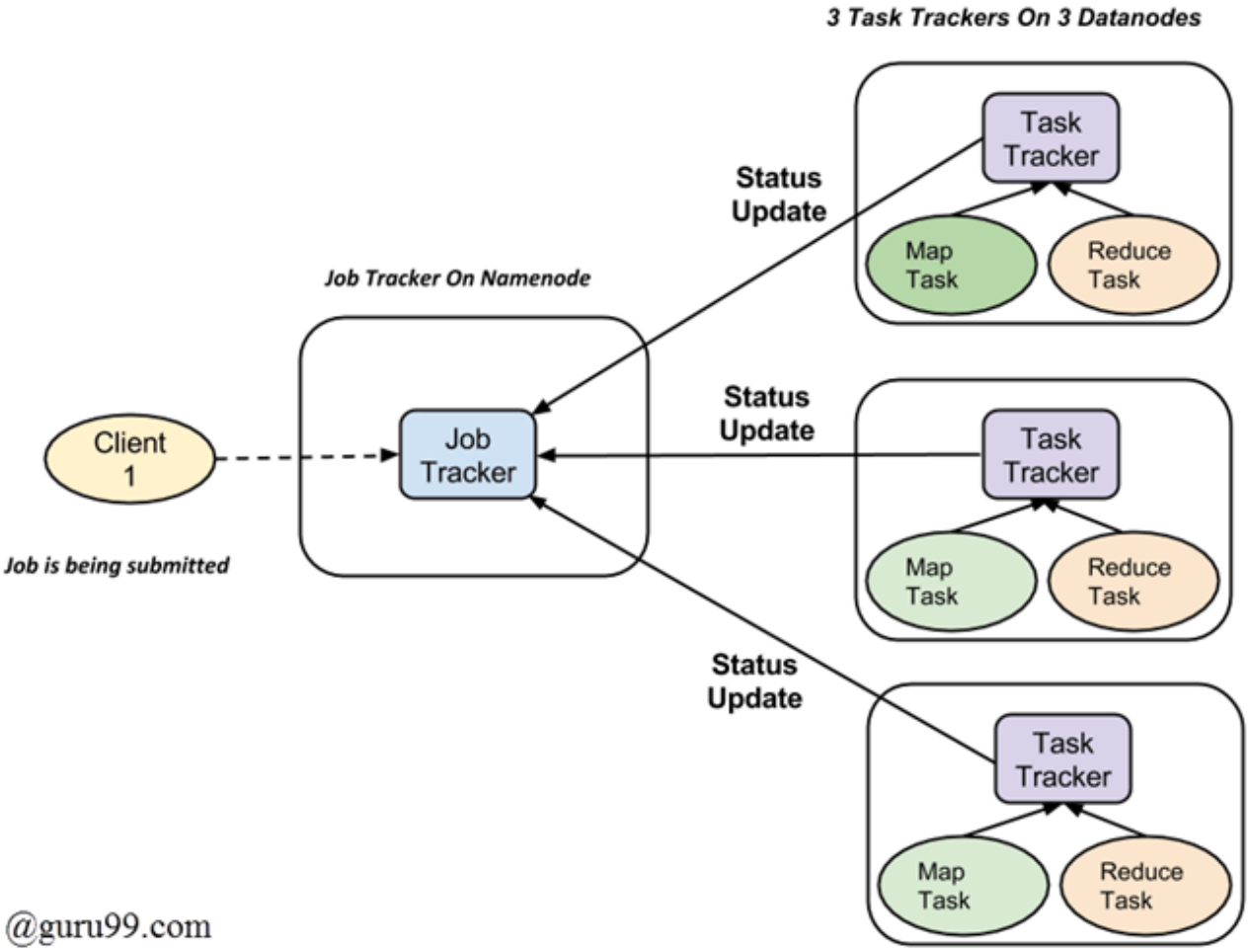
1. **Map tasks** (Splits & Mapping)
2. **Reduce tasks** (Shuffling, Reducing)

Sebagaimana telah dibahas di atas.

Proses eksekusi lengkap (eksekusi tugas Map dan Reduce) dikendalikan oleh dua jenis entitas bernama:

1. **Jobtracker**: bertindak seperti **master** (bertanggungjawab menyelesaikan eksekusi pekerjaan yang disubmit)
2. **Multiple Task Trackers**: Bertindak sebagai **slaves,** masing-masing melaksanakan pekerjaannya.

Untuk setiap pekerjaan yang disubmit untuk eksekusi dakam sistem, Ada **satu Jobtracker** yang berada pada **Namenode** dan ada **banyak tasktrackers** yang berada pada **Datanode**.



- Suatu pekerjaan dibagi ke dalam banyak tugas yang kemudian berjalan pada banyak node data dalam suatu cluster.
- Adalah tanggungjawab dari job tracker untuk mengkoordinasikan aktifitas dengan menjadwalkan tugas-tugas untuk berjalan pada node-node data berbeda.
- Eksekusi tugas individu kemudian dilaksanakan oleh task tracker, yang berada pada setiap node data yang mengeksekusi bagian dari pekerjaan.
- Tanggungjawab task tracker adalah mengirimkan laporan progress kepada job tracker.
- Task tracker juga secara periodik mengirimkan sinyal **'heartbeat' (denyut)** ke Jobtracker untuk mengabari status terkini dari sistem.
- Jadi job tracker memelihara jejak (track) dari progress keseluruhan (overall progress) dari setiap pekerjaan. Jika terjadi kegagalan tugas, job tracker dapat menjadwal ulang (reschedule) tugas tersebut pada task tracker berbeda.