Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2023106310
Vol. 10, No. 2, Juni 2023, hlm. 475-482 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

MAPREDUCE DALAM LAYANAN TRANSCODING

David Kristiadi*[1], Arum Marwati[2]

[1,2] Sekolah Tinggi Multi Media (MMTC) Yogyakarta, Kabupaten Sleman
Email: *[1]davk@mmtc.ac.id, [2]arum.marwati@mmtc.ac.id
*Penulis Korespondensi

(Naskah masuk: 19 Mei 2022, diterima untuk diterbitkan: 19 Juni 2023)

Abstrak

Penyediaan *file* video dengan *bitrate* bervariasi menjadi syarat utama bagi layanan *Video On Demand* yang menerapkan *adaptive streaming*. Hal tersebut dilakukan dengan *transcoding* yang menghasilkan video dengan *multi-bitrate*. Proses *multi-bitrate transcoding* membutuhkan waktu yang tidak singkat. Lama durasi waktunya sebanding dengan besarnya *bitrate, frame size* dan *frame rate*. Untuk mempersingkat durasi *transcoding*, dibuat sebuah prototipe layanan *transcoding* dengan menerapkan *Mapreduce*. Layanan *transcoding* dengan *Mapreduce* terdiri dari satu komputer *master* dan beberapa komputer *worker* yang terhubung dalam satu LAN. Dengan dikoordinir oleh komputer *master*, komputer-komputer *worker* mengerjakan proses *Map* dan *Reduce*. Di dalam proses *Map* dilakukan *transcoding* terhadap *segmented videos*. Di dalam proses *Reduce* dilakukan penggabungan segmen-segmen video yang telah di-*transcode* dengan parameter/*key* (*bitrate, frame size,* dan *frame rate*) yang sama menjadi satu video yang utuh. Prototipe layanan *transcoding* dibuat menggunakan *Library FFMPEG* untuk *transcoding*, SCP untuk transfer *file*, RPC untuk komunikasi antar komputer. Di dalam pengujian prototipe, jumlah komputer *worker* ditentukan sebanyak 7 buah. Kinerja layanan *transcoding* sangat memuaskan dengan rata-rata efektifitas *transcoding* sebesar 71,3% dibandingkan dengan *transcoding* menggunakan satu komputer.

Kata kunci: *Transcoding, Mapreduce, Multi-bitrate Transcoding*

MAPREDUCE IN TRANSCODING SERVICES

Abstract

The provision of video files with varying bitrates is the main requirement for Video On Demand services that implement adaptive streaming. Transcoding that produces multiple bitrates achieves that. Multi-bitrate transcoding can take a longer time. That duration is comparable with the large of the video's bitrates, frame sizes, and frame rates. A transcoding service prototype was created based on Mapreduce to shorten the duration. Transcoding service using Mapreduce consists of several computers connected to a LAN, one as master and the other as worker computers. The master coordinates the workers to do the Map and Reduce process. In the Map process, workers transcode the segmented videos. In the Reduce process, workers merge all transcoded video segments with the same key or parameters (bitrate, frame size, and frame rate) into a single video. The transcoding service prototype was created using the FFMPEG library for transcoding, SCP for file transfer, and RPC for communication between computers. In the testing stage, the number of workers is 7 computers. Service performance is very satisfactory, with average transcoding effectiveness of 71,3% *compared to transcoding using a single computer.*

Keywords: *Transcoding, Mapreduce, Multi-bitrate Transcoding*

1. PENDAHULUAN

Layanan video berdasarkan permintaan pengguna atau lebih dikenal sebagai *video on demand* (VOD) semakin banyak diminati. Indikator yang dapat diamati adalah banyaknya penyedia layanan VOD dan jumlah kunjungan pengguna pada layanan tersebut. Hasil survei oleh APJII (2020) menunjukkan bahwa sejumlah 49.3% responden tercatat lebih sering mengakses video *online* dibandingkan konten hiburan internet yang lain. Selain itu tercatat dalam sebulan terdapat lebih dari 2 milyar user yang mengakses layanan Youtube dan milyaran jam pemutaran video (Youtube, 2022). Layanan video oleh Youtube adalah salah satu contoh layanan VOD

Sebelum layanan VOD dapat diakses oleh pengguna, video original di-*ingest* ke dalam *server*, dilanjutkan proses *transcoding* (Song et al., 2014). Kemudian setelahnya video tersebut dapat diakses melalui layanan *video streaming*. *Transcoding*

merupakan proses meng-*encode* ulang video ukuran *frame size*, *bitrate* dan format tertentu.

Layanan *video streaming* memiliki beberapa macam protokol seperti protokol tradisional (RTMP, RTSP/RTP) dan protokol *adaptive streaming* yang berbasis HTTP (*Dynamic Adaptive Streaming over* HTTP (DASH), Microsoft *smooth streaming* (MSS), Adobe *Dynamic Streaming* (ADS) HTTP *Live Streaming* (HLS)). *Adaptive streaming* memungkinkan layanan video menyesuaikan dengan kondisi jaringan dan kemampuan hardware pengguna. Dengan model *streaming* ini dapat meningkatkan *Quality of Experience (QoE)*. Dari beberapa jenis protokol *adaptive streaming*, jenis teknologi yang banyak diimplementasikan untuk layanan VOD adalah DASH dan HLS (Reznik et al., 2019).

Untuk memenuhi kebutuhan *file* video dalam layanan *adaptive streaming* diperlukan *transcoding* ke *multi-bitrate* dan *frame size (multi-bitrate transcoding)*. Di dalam proses *transcoding* dilakukan *decoding* dan *encoding* sehingga membutuhkan waktu yang tidak sedikit. Selain waktu juga dibutuhkan *resource* yang tidak sedikit. Beberapa parameter video yang mempengaruhi durasi *transcoding* adalah durasi video, jenis *codec*, ukuran *bitrate* dan *frame size.* Semakin lama durasi video, semakin banyak macam *frame size*, dan semakin besar *bitrate* maka semakin lama waktu *transcoding*. Jenis *codec* yang digunakan juga mempengaruhi waktu *transcoding*. Dalam uji coba yang dilakukan oleh Ozer (2018), dilaporkan bahwa *encoding* dengan *codec* AV1 memiliki durasi *encoding* yang paling lama atau sekitar 45.216 kali dari waktu nyata.

Waktu penyediaan video *multi-bitrate* dan *frame size* dalam layanan VOD berbasis *adaptive streaming* merupakan hal yang krusial di awal layanan ini. Jika waktu *transcoding* lama, maka semakin lama waktu memulai layanan video.

Upaya mempercepat transcoding dapat digunakan model komputasi terdistribusi. Dengan komputasi terdistribusi, komputer - komputer dalam satu jaringan dikoordinasikan supaya menyelesaikan tugas spesifik. Terdapat satu komputer *host* (*master*) yang akan mengkoordinir tugas yang dikerjakan oleh komputer - komputer *slave* (*worker*) yang tergabung. Dengan konsep ini, idealnya suatu tugas akan selesai dengan lebih cepat dari pada hanya jika dikerjakan oleh satu komputer.

Implementasi *transcoding* dengan model komputasi terdistribusi dilaporkan menunjukkan perbaikan waktu yang signifikan dibandingkan transcoding dengan satu komputer (Song et al., 2014; Liu and Yuan, 2018; Liu et al., 2016; Chang et al., 2016; Gutiérrez-Aguado et al., 2020). Implementasi tersebut beberapa dilakukan dengan memanfaatkan *framework* Hadoop (Song et al., 2014; Liu et al., 2016; Sameti, Wang and Krishnamurthy, 2018; Huh, Kim and Jeong, 2019), *cloud computing* seperti AWS (Jiang, Lee and Zomaya, 2019; Jain, Shrivastava and Moghe, 2020) dan layanan *encoding cloud service* seperti Bitmovin (Trattnig, Timmerer and Mueller, 2018).

Salah satu model komputasi terdistribusi adalah komputasi *Mapreduce*. Komputasi ini memiliki dua fungsi utama, yaitu *Map* dan *Reduce*. *Map* mengerjakan proses membagi-bagi pekerjaan menghasilkan hasil pemrosesan sementara (*intermediate result*). Fungsi berikutnya adalah *Reduce* yang mengerjakan hasil pemrosesan sementara (*intermediate result*) menjadi satu hasil yang utuh sesuai dengan kunci yang diberikan (Song et al., 2014).

Implementasi layanan *transcoding* dengan konsep *Mapreduce* sudah banyak dilakukan dengan menggunakan Hadoop (Liu and Yuan, 2018; Sameti, Wang and Krishnamurthy, 2018; Song et al., 2014; Chang et al., 2016). Dari implementasi yang telah dilakukan tersebut, belum ditemui layanan *transcoding* dengan konsep *Mapreduce* yang memanfaatkan FFMPEG *library*, *Remote Procedure Call* (RPC) dan *Secure Copy* (SCP) dalam lingkungan pemrograman Golang. FFMPEG *library* digunakan untuk meng-*convert* data audio video dan *library* ini mampu dijalankan pada sistem operasi personal komputer yang berbeda (Kristiadi and Marwiyati, 2021; Jain, Shrivastava and Moghe, 2020). Komunikasi antar *master* - *worker* menggunakan RPC. Mekanisme RPC digunakan untuk komunikasi dalam jaringan yang kompleks akibat perbedaan mesin, sistem operasi, dan alamat eksekusi (Kiraly and Szekely, 2018). SCP digunakan untuk transfer *file* antar komputer dalam jaringan. Lingkungan pemrograman Golang digunakan untuk *menyatukan FFMPEG library, SCP dan RCP* menjadi sebuah prototipe layanan *transcoding* yang menggunakan konsep *Mapreduce*. *Prototipe* layanan *transcoding* ini selanjutnya dapat dikembangkan untuk keperluan layanan VOD.

2. METODE PENELITIAN

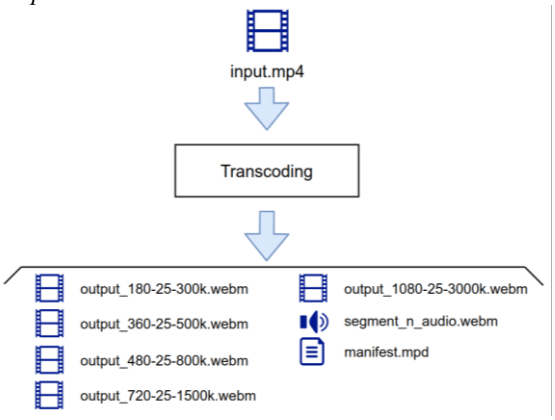
Tahap-tahap penelitian antara lain yaitu pengumpulan bahan dan analisis, perancangan dan implementasi, terakhir uji coba dan evaluasi. Output dari penelitian ini berupa prototipe *transcoding service*.

Bahan - bahan yang akan dikumpulkan untuk mengawali penelitian adalah beberapa referensi tentang penerapan *Mapreduce* menggunakan Golang, isu seputar *Mapreduce* dan *transcoding*, dan referensi bahasa pemrograman Golang. Dari bahan yang telah diperoleh, selanjutnya dilakukan analisis untuk mendapatkan model *transcoding service* yang tepat.

2.1. Rancangan Sistem

Rancangan sistem secara sederhana memenuhi bagan seperti pada Gambar 1. Proses *transcoding* secara umum meng-*encode* ulang video asli ke dalam beberapa format video dengan *bitrate* dan *frame size*

yang bervariasi. Proses *transcoding* dapat dilakukan menggunakan satu komputer maupun dengan multi komputer salah satunya dengan menerapkan konsep *Mapreduce*.



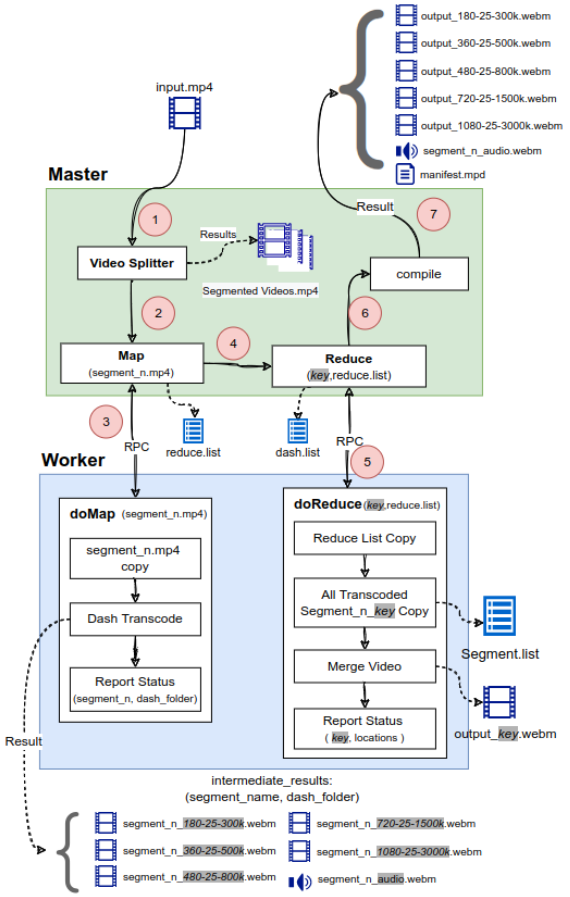
Gambar 1. Proses *Transcoding* Sederhana

Implementasi *Mapreduce* di sistem *transcoding* dirancang seperti pada Gambar 2. Dalam sistem ini, terdiri satu komputer *master* yang mengkoordinasi eksekusi fungsi *Map* dan *Reduce* pada beberapa komputer worker. Komputer *Master* dan *Worker* terkoneksi dalam satu local area network (LAN). Proses komunikasi antar komputer M*aster* dan *Worker* menggunakan *Remote Procedure Call* (RPC), dan untuk keperluan transfer *file* digunakan *Secure Copy Protocol* (SCP) untuk sistem operasi Linux atau *PuTTY Secure Copy Protocol* (PSCP) untuk sistem operasi windows.

Di rancangan Gambar 2, Tahapan *Mapreduce* diawali dengan tahap memotong-motong video oleh *Video Splitter* (1). Hasil dari tahapan tersebut adalah potongan-potongan video atau (*segmented videos*). *Segmented videos* yang diperoleh menjadi input Tahap *Map* (2). Pada Tahap *Map,* komputer master mengirimkan instruksi melalui RPC ke *worker* untuk mengeksekusi Fungsi doMap(3). Fungsi doMap di komputer *worker* menggunakan *pseudocode* seperti pada Gambar 3.

Fungsi doMap mengambil salah satu *segmented video* kemudian mengeksekusi *transcoding* dan menghasilkan beberapa file video (*no audio*) dengan *frame size*, *frame rate* dan *bitrate* yang berbeda-beda dan sebuah file *audio*.

Ketika *transcoding* selesai dikerjakan oleh *worker*, *worker* akan memberikan laporan status *transcoding* yang dikerjakan kepada *master*. Jika status berhasil yang dikirimkan, *master* menulis data ke *file reduce.list*. Data yang ditulis adalah nama *segmented video*, lokasi dash_folder dan alamat IP *worker*. Jika status *transcoding* gagal, *master* akan menginstruksikan *worker* selanjutnya yang ada di dalam daftar *worker,* mengeksekusi Fungsi doMap dengan parameter *segmented video* tersebut.



Gambar 2. Tahapan Detail *Transcoding* Menggunakan *Mapreduce*

```
doMap(video_segment){
  segment := scp_copy(video_segment)
  dash_folder, err :=dashTranscode(segment)
  //report result to master
  if !err
      report(segment_name, dash_folder, true)
  else
      report(segment_name, nil, false)
}
```

Gambar 3. Pseudocode Proses *Map* di Sisi *Worker*

Ketika semua *segmented videos* selesai di-*transcode* oleh *worker* pada tahap *Map*. Master melanjutkan ke tahap *Reduce* (4). Pada tahap ini, master memberikan input kepada *worker* berupa parameter/*key* dan alamat *file* reduce.list (5). *Key* merupakan merupakan kombinasi *frame size*, *frame rate* dan *bitrate* sebagai contoh 1080-25-3000K.

Setelah instruksi *reduce* diterima oleh *worker*, dilanjutkan proses *copy* reduce.list. Proses yang dilakukan selanjutnya seperti pada *pseudocode* Gambar 4. Dengan data yang ada di *file* reduce.list, komputer *worker* akan mengumpulkan potongan-potongan video hasil *transcoding* dengan *Key* yang sesuai ke dalam penyimpanan lokalnya. Apabila semua sudah potongan video dengan key yang sama terkumpul, dilanjutkan dengan penggabungan potongan-potongan video tersebut menjadi satu video yang utuh.

Setelah selesai tahap penggabungan segmen-segmen video dengan *key* yang sama menjadi satu

video utuh, *worker* akan melaporkan hasil eksekusi *reduce* kepada *master*. Jika berhasil master akan menyimpan data *worker* dan lokasi *output* video ke dalam dash.list, jika tidak berhasil *master* akan menugaskan *worker* yang lain untuk mengerjakan tahap *Reduce* pada *key* yang belum berhasil.

```
doReduce(key,reduceLoc){
  reduceFile = scp_copy(reduceLoc)
  segmentRemoteLoc []string  = read(ReduceFile)
  segmentList []string

  i=0
  for remoteLoc := range segmentRemoteLoc {

 // remote folder location example
 // worker@192.168.26.235:/home/segment_001_1080-25-3000k.webm
    segmentList[i] = scp_copy(remoteLoc.dash_folder+
                      "/"+segment_name+"_"+key+".webm")
    i++

  }

  sort(segmentList) //
  videoFile,err = mergeVideo(segmentList)
  //report result to master
  if !err
      report(key, videoFile, true)
  else
      report(key, nil, false)
}
```

Gambar 4. *Pseudocode* Tahapan *Reduce* di Sisi *Worker*

Hasil dari Tahap *Reduce* pada sisi *master* adalah dash.list. Informasi di dalam dash.list kemudian digunakan pada tahap *compile* (6). Pada tahap ini *master* akan mengumpulkan *file-file* video yang utuh hasil dari tahap *Merge* pada tahap Reduce (7). Setelah semua *file-file* video terkumpul dilanjutkan oleh *master* membuat file *media presentation descriptor* atau file *.mpd. File *descriptor* ini berisi parameter-parameter dan lokasi media (audio dan video) yang akan diakses secara *streaming*.

2.2. Implementasi Sistem

Implementasi rancangan yang telah dibuat, dikerjakan menggunakan lingkungan Pemrograman Golang. Lingkungan Pemrograman Golang memiliki fasilitas goroutine. Goroutine ini sangat bermanfaat dalam melakukan *transcoding* secara paralel di komputer *Worker*, membangun *service* keperluan komunikasi secara RPC dan mengeksekusi perintah transfer *file* menggunakan SCP atau PSPC (pada sistem operasi Windows). Dengan Goroutine proses utama tetap berjalan sementara proses pendukung lainnya sedang berjalan.

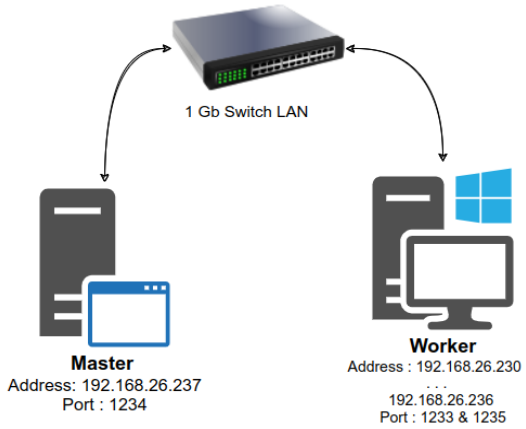
2.3. Uji Coba dan Evaluasi

Pada tahap ini, prototipe layanan *transcoding* diuji coba menggunakan 8 komputer yang memiliki spesifikasi sama dan yang dihubungkan melalui *Local Area Network* (LAN). Dari 8 Komputer, satu komputer sebagai *master* dan 7 lainnya sebagai *worker*. Dari ketujuh komputer *worker* dijalankan dua aplikasi *worker*, sehingga jika ditotal terdapat 14 *worker*. Tabel 1 merupakan spesifikasi komputer yang digunakan untuk uji coba.

Tabel 1. Spesifikasi Komputer untuk Uji Coba

Spesifikasi	Detail
Processor	Intel ® Core (TM) i7-4790 CPU @3.60G.Hz (8CPUs)
Memory	8192 MB
Operating System	Windows 7 Professional 64-bit
Display	NVIDIA Geforce GT 720 Memory 4022MB

Sedangkan konfigurasi LAN yang digunakan seperti pada Gambar 6. Komputer master dan worker berada pada jaringan yang dengan menggunakan 1 Gigabit Switch LAN.



Gambar 6. Konfigurasi Jaringan untuk *Testing*

Selanjutnya dipilih beberapa video yang akan digunakan untuk uji coba prototipe *transcoding* menggunakan *Mapreduce*. Beberapa video memiliki data parameter seperti pada Tabel 2.

Pada saat uji coba akan dilakukan proses *transcoding* secara konvensional yaitu proses *transcoding* dengan menggunakan satu komputer, dan menggunakan *Mapreduce*. Hasil dari uji coba ini kemudian dievaluasi untuk membuat prototipe program yang lebih baik.

Tabel 2. Data parameter Video yang Digunakan untuk Uji Coba

Nama	Ukur-an file (MB)	Durasi	Screen Size	Frame rate (FPS)	Bit-rate (Mbps)	Audio (KHz)
video1.mp4	70,32	28,79s	1920x1080	58.79	20,01	48
video2.mp4	31,74	5m12,86s	640x360	30	0,78	44,1
video3.mp4	27,66	11,17s	1920x1080	29.99	19,62	48
video4.mp4	178,1	3m32,61s	1280x720	29.58	6,74	48
video5.mp4	235,77	4m37,79s	1280x720	30	6,84	48
video6.mp4	446,08	1m14,31s	3840x2160	30	49,00	48
video7.mp4	574,68	3m59,54s	1920x1080	26.03	19,53	48

Eksekusi proses *transcoding* akan menghasilkan beberapa video *no audio* berekstensi *.webm dengan parameter video seperti pada Tabel 3, sebuah *file* audio berekstensi *.webm dengan *sample rate* 48Khz, dan sebuah *file manifest* berekstensi *.mpd

Tabel 3. Data Parameter *Output Transcoding* Video

No	Screen Size	Frame rate (FPS)	Bit- rate (Kbps)	Key
1	1920x1080	25	3000	1080-25-3000K
2	1280x720	25	1500	720-25-1500K
3	854x480	25	800	480-25-800K
4	640x360	25	500	360-25-500K
5	320x180	25	300	180-25-300K

3. HASIL DAN PEMBAHASAN

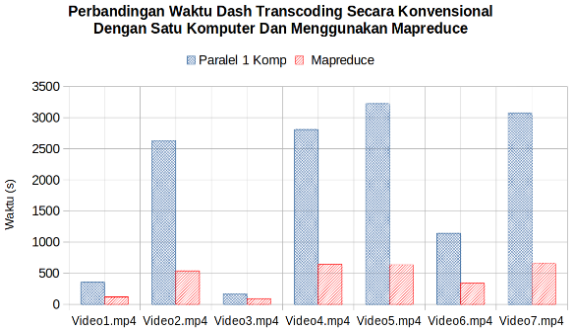
3.1. Efisiensi Total Waktu Layanan *Transcoding*

Melalui uji coba yang dilakukan terhadap prototipe layanan *transcoding* yang telah dikembangkan memiliki efisiensi yang baik. Seperti yang tampil Tabel 4 dan divisualisasikan dengan grafik batang pada Gambar 7 waktu eksekusi prototipe dengan konsep *Mapreduce* memiliki durasi yang lebih cepat dibandingkan dengan eksekusi menggunakan satu komputer.

Tabel 4. Perbandingan Waktu *Transcoding*

Nama File	Metode *Transcode*		Efisiensi *Mapreduce* (%)
	Komputer tunggal	*Mapreduce*	
Video1.mp4	5m53.849s	1m58.395s	67,5
Video2.mp4	43m51.760s	8m56.481s	79,6
Video3.mp4	2m50.179s	1m30.578s	46,8
Video4.mp4	46m47.429	10m41.600s	77,1
Video5.mp4	53m44.417s	10m35.838s	80,3
Video6.mp4	19m0.231s	5m38.644s	70,3
Video7.mp4	51m16.683s	10m53.352s	78,8
rata-rata			71,3

Sebagai contoh pada video7.mp4, waktu *transcoding* memiliki efisiensi hingga 78,8%. Bila menggunakan satu komputer dengan eksekusi dengan komputer tunggal adalah 51 menit 16.683 detik. Bila dengan menggunakan *Mapreduce* membutuhkan waktu sekitar 10 menit 53 detik.



Gambar 7. Grafik Perbandingan Waktu Eksekusi

Rata - rata efisiensi waktu adalah 71,3%. Efisiensi waktu semakin nampak jika durasi video yang lebih lama. Sebagai contoh untuk video3.mp4 dengan durasi 11.17s memiliki tingkat efisiensi waktu 46,8% dibandingkan dengan video7.mp4 dengan durasi video 3m 59.54s memiliki tingkat efisiensi waktu 78,8%. Semakin lama durasi video, maka semakin besar tingkat efisiensi waktu *transcode* menggunakan *Mapreduce*.

3.2. Kecepatan Transfer *File* dengan SCP

Seperti pada Gambar 2, proses *Mapreduce* terdapat beberapa tahapan yang harus dilakukan, tahap *video splitting* (membuat *segmented videos*), tahap *Map* (tahap distribusi *segmented videos* dan dilanjutkan *transcoding* oleh *worker*), tahap *Reduce* (tahap penggabungan segmen video yang telah di-*transcode* menjadi satu video utuh yang diawali dari pengumpulan hasil *transcoding* segmen video dari *worker-worker* yang bertugas), dan tahap *video compiling* (pengumpulan hasil *Reduce* dan pembuatan *manifest*).

Pada setiap tahap kecuali *video splitting* membutuhkan waktu untuk distribusi *segmented videos* dan video utuh hasil *transcoding*. Pada konfigurasi jaringan yang sama didapatkan rata-rata waktu sebesar 14,78 Mbps. Catatan waktu transfer *file* video menggunakan SCP seperti pada Tabel 5. LTR adalah transfer file dari *local to remote*, sedang RTL adalah transfer file dari *remote to local*

Tabel 5. Hasil Ujicoba Transfer *File* Menggunakan SCP

SCP	*File size*(MB)	Waktu (s)			Kecepatan (Mbps)
		LTR	RTL	AVG	
video1.mp4	70,32	4,15	5,98	5,07	111,07
video2.mp4	31,74	2,26	3,64	2,95	86,00
video3.mp4	27,66	2,05	3,24	2,64	83,66
video4.mp4	178,1	8,71	12,2	10,47	136,07
video5.mp4	235,77	9,47	15,13	12,30	153,31
video6.mp4	446,08	26,71	33,80	30,26	117,95
video7.mp4	574,68	28,12	37,85	32,98	139,38
AVG	223.48	11,64	15,98	13,81	118,20

Data kecepatan transfer *file* pada Tabel 5 jika digunakan pada perhitungan waktu di setiap tahap *transcoding* dengan *Mapreduce* dari video 1 akan mendapatkan hasil seperti pada Tabel 6. perhitungan waktu tersebut menggunakan persamaan (1), dengan v adalah kecepatan transfer *file*, s adalah ukuran *file*, dan t adalah waktu transfer *file*.

$$t = \frac{s*8}{v} \quad (1)$$

Tabel 6. Waktu Transfer File Menggunakan SCP pada Proses *Mapreduce* video1.mp4

Tahap *Mapreduce*	*File Size* (B)	Waktu (s)
SCP Tahap MAP (master - worker)	72.993.543	4,71
SCP Tahap REDUCE		
>> Reduce 1080-25-3000K	10.048.685	0,65
>> Reduce 720-25-3000K	4.625.394	0,30
>> Reduce 480-25-800K	2.288.015	0,15
>> Reduce 360-25-500K	1.183.630	0,08
>> Reduce 180-25-300K	398.955	0,03
>> Reduce Audio	155.421	0,01
SCP Tahap *Compiling*	17.950.334	1,16
Total		**7,08 s**

Waktu *transcoding* pada video1.mp4 pada Tabel 4 memiliki total waktu (t_{total}) = 1m58,395s sedangkan total waktu transfer *file* menggunakan SCP (t_{scp}) pada Tabel 6 adalah 7.08s. Dari data

tersebut, didapatkan total waktu untuk proses *Mapreduce* tanpa transfer *file* adalah 1m51,395s.

$$r_{tr} = \frac{t_{scp}}{t_{total}} \quad (2)$$

Dengan menggunakan persamaan (2) diperoleh. Rasio (r_{tr}) transfer *file* dengan SCP dibandingkan dengan keseluruhan durasi *transcoding* adalah sebesar 0,06. Perolehan rasio waktu tersebut sudah cukup memuaskan. Supaya mendapatkan waktu transfer yang lebih cepat dapat ditingkatkan dengan penggunaan infrastruktur jaringan dan *protokol copy file* yang lebih cepat dan aman .

3.3. Durasi Tahap *Map* dan *Reduce*

Untuk mengetahui alokasi waktu pada setiap tahap *Mapreduce*, dilakukan ujicoba proses inti *Mapreduce* yaitu *Map* (*split-transcode) dan Reduce (merge/compile*) pada satu komputer (*single computer*) tanpa ada proses transfer *file* maupun komunikasi RPC. Total yang dibutuhkan dengan ujicoba ini adalah 5m45.7s. Catatan waktu tersebut seperti pada Tabel 7. Dari data tersebut dapat dilihat bahwa proses *Map* memakan waktu yang paling lama pada rangkaian proses *Mapreduce*.

Tabel 7. Waktu yang Diperlukan pada Tahap *Map* dan *Reduce* video1.mp4 dengan Satu Komputer

Tahap *Mapreduce*	Waktu
map output000.mp4	1m1.7973056s
map output001.mp4	59.3631132s
map output002.mp4	59.8249154s
map output003.mp4	57.7711432s
map output004.mp4	59.9889248s
map output005.mp4	46.2365416s
Reduce 180-25-300k	96.6004ms
Reduce 360-25-500k	67.8013ms
Reduce 480-25-800k	67.8012ms
Reduce 720-25-1500k	81.0003ms
Reduce 1080-25-3000k	114.6014m
Reduce audio	119.6017ms
Total Time Execution	**5m45.7073516s**

Tahap *Map* menggunakan waktu yang lebih lama. Hal tersebut dikarenakan di dalamnya terdapat proses *transcoding*. Total waktu yang diperlukan untuk proses *Map* & *Reduce* pada satu komputer mendekati dengan capaian waktu *transcoding* satu *file* video utuh oleh satu komputer.

Dari capaian ini, dapat dimungkinkan juga dengan menggunakan *Mapreduce* dan menambah jumlah *worker* akan mempercepat durasi *transcoding*. Jumlah *worker* yang dibutuhkan untuk mendapatkan waktu *transcoding* yang tercepat adalah sebanyak jumlah *segmented videos*. Jika jumlah *worker* melebihi jumlah *segmented videos*, kelebihan tersebut tidak akan memberikan dampak kepada peningkatan kecepatan durasi *transcoding*. Hal ini dikarenakan setiap *segmented video* ditangani oleh satu *worker* dan kelebihan jumlah *worker* akan *idle* menunggu hingga mendapat tugas dari *master*.

Sebagai contoh pada percobaan yang telah dilakukan dengan jumlah *worker* sebanyak 14 *worker* dan kasus pada *transcoding* video1.mp4 dengan data pada Tabel 4 dan Tabel 7. Pada kasus ini, durasi *transcoding* video01.mp4 dengan *Mapreduce* mendekati waktu transcoding output000.mp4 pada Tabel 7. Output000.mp4 adalah salah satu *segmented video* dari video1.mp4 dengan durasi *transcoding* yang terlama. Fakta dari durasi *transcoding* ini menunjukkan bahwa jumlah *worker* yang berlebih tidak serta merta mempercepat durasi *transcoding*. Karena durasi *transcoding* dengan *Mapreduce* juga ditentukan oleh banyaknya *segmented videos* dan durasi *transcoding* masing-masing *segmented video.* Durasi *transcoding* yang tercepat dengan *Mapreduce* dapat dicapai jika jumlah *worker* sama dengan jumlah *segmented videos.* Tetapi efektifitas dan efisiensi *resource* perlu diteliti lebih lanjut dalam implementasi layanan *transcoding* dengan *Mapreduce* dalam lingkungan produksi.

4. KESIMPULAN

Efisiensi waktu yang lebih baik diperoleh di layanan *transcoding* dengan konsep *Mapreduce* dibandingkan *transcoding* dengan satu komputer. Efisiensi waktu yang diperoleh rata-rata sebesar 71,3% dibandingkan proses *transcoding* oleh satu komputer. Layanan *Transcoding* terdiri dari dua tahap utama yaitu *Map* dan *Reduce*. Pada Tahap *Map* dilakukan *transcoding* terhadap *segmented videos* oleh komputer - komputer Worker. Pada Tahap *Reduce* dilakukan penggabungan segmen-segmen video dengan parameter (*key*) yang sama. Layanan *transcoding* menggunakan *library* FFMPEG, SCP dan RPC. Transfer *file* dengan SCP mengambil porsi sebesar 0,06 dari total waktu *transcoding*. Kecepatan tersebut dimungkinkan bisa lebih cepat dengan infrastruktur jaringan yang lebih baik. Durasi transcoding tercepat dapat dicapai dengan jumlah komputer Worker sama jumlah *segmented videos.*

DAFTAR PUSTAKA

APJII, 2020. Laporan Survei Internet APJII 2019 – 2020. *Asosiasi Penyelenggara Jasa Internet Indonesia*, [online] 2020, pp.1–146. Available at: <https://apjii.or.id/survei>.

CHANG, Z.H., JONG, B.F., WONG, W.J. AND WONG, M.L.D., 2016. Distributed video transcoding on a heterogeneous computing platform. In: *2016 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2016*. Institute of Electrical and Electronics Engineers Inc.pp.444–447.

GUTIÉRREZ-AGUADO, J., PEÑA-ORTIZ, R., GARCÍA-PINEDA, M. AND CLAVER, J.M., 2020. Cloud-based elastic architecture for distributed video encoding: Evaluating H.265, VP9, and AV1. *Journal of Network and Computer Applications*, 171(July), p.102782.

HUH, J., KIM, Y.H. & JEONG, J., 2019. Ultra-High Resolution Video Distributed Transcoding System Using Memory-based High-speed Data Distribution Method. In: *34th International Technical Conference on Circuits/Systems, Computers and Communications, ITC-CSCC 2019*. Institute of Electrical and Electronics Engineers Inc.

JAIN, N., SHRIVASTAVA, H. & MOGHE, A.A., 2020. Production-ready environment for HLS Player using FFmpeg with automation on S3 Bucket using Ansible. *2nd International Conference on Data, Engineering and Applications, IDEA 2020*, pp.26–29.

JIANG, Q., LEE, Y.C. & ZOMAYA, A.Y., 2019. Scalable video transcoding in public clouds. *Proceedings - 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2019*, pp.70–75.

KIRALY, S. & SZEKELY, S., 2018. Analysing RPC and Testing the Performance of Solutions. *Informatica*, [online] 42(4), pp.555–561. Available at: <https://www.informatica.si/index.php/informatica/article/view/1510> [Accessed 13 Sep. 2022].

KRISTIADI, D. & MARWIYATI, 2021. Adaptive Streaming Server dengan FFMPEG dan Golang. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 5(3), pp.413–420.

LIU, Y. & YUAN, J., 2018. Spark platform based video transcoding. *EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM)*, 2017-September.

LIU, Z., CAI, L., CHEN, W. & CHEN, M., 2016. Parallel video transcoding using Hadoop MapReduce. *Journal of Network Computing and Applications*, [online] 1(1), pp.7–11. Available at: <http://www.clausiuspress.com/article/15.html> [Accessed 10 May 2022].

OZER, J., 2018. *AV1: A First Look*. [online] Available at: <https://www.streamingmedia.com/Articles/Editorial/Featured-Articles/AV1-A-First-Look-127133.aspx> [Accessed 9 Apr. 2020].

REZNIK, Y.A., LI, X., LILLEVOLD, K.O., JAGANNATH, A. & GREER, J., 2019. Optimal Multi-Codec Adaptive Bitrate Streaming. *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, [online] pp.348–353. Available at: <https://ieeexplore.ieee.org/document/8795046/>.

SAMETI, S., WANG, M. & KRISHNAMURTHY, D., 2018. Stride: Distributed Video Transcoding in Spark. *2018 IEEE 37th International Performance Computing and Communications Conference, IPCCC 2018*.

SONG, C., SHEN, W., SUN, L., LEI, Z. & XU, W., 2014. Distributed video transcoding based on MapReduce. *2014 IEEE/ACIS 13th International Conference on Computer and Information Science, ICIS 2014 - Proceedings*, pp.309–314.

TRATTNIG, A., TIMMERER, C. & MUELLER, C., 2018. Investigation of YouTube regarding content provisioning for HTTP adaptive streaming. *Proceedings of the 23th ACM Workshop on Packet Video, PV 2018*, pp.60–65.

YOUTUBE, 2022. *YouTube for Press*. [online] Available at: <https://blog.youtube/press/> [Accessed 17 Jan. 2022].

Halaman ini sengaja dikosongkan