KATA PENGANTAR

Salam ULTIMA!

Berjumpa lagi dalam Jurnal ULTIMATICS yang saat ini telah memasuki Volume VII No 1. Jurnal ini merupakan jurnal yang menyajikan artikel-artikel penelitian ilmiah dalam bidang analisis dan desain sistem, pemrograman, analisis algoritma, rekayasa perangkat lunak, serta isu-isu teoritis dan praktis terkini, mencakup komputasi, kecerdasan buatan, sistem bergerak, dan topik-topik lainnya di bidang Teknik Informatika.

Pada ULTIMATICS edisi Juni 2015 ini, terdapat sebelas (11) buah karya tulis ilmiah yang berasal dari para peneliti, akademisi dan praktisi di bidang Teknik Informatika, yang mengangkat beragam topik berbeda, seperti pengembangan aplikasi TV dengan protokol *streaming* RTSP/ RTMP, komparasi algoritma *quicksort* dan *bucket sort*, pemanfaatan *augmented reality* pada *marketing communication*, pengembangan aplikasi web pendeteksi gangguan kecemasan dan depresi, penerapan *histogram equalization* pada OCR *preprocessing*, analisis siklus hidup aplikasi web, pengembangan aplikasi diagnosis karies gigi, pengembangan piranti lunak pengelola parameter akuisisi data terowongan angin kecepatan rendah Indonesia, *review* piranti lunak StarUML berdasarkan faktor kualitas McCall, *load balancing* pada *cloud computing* menggunakan penggabungan algoritma ESCE dan Throttled, serta peningkatan performansi *decision tree* dalam pengelompokkan dan prediksi resiko *credit*.

Pada kesempatan ini, kami juga ingin mengundang partisipasi Anda semua, para peneliti, akademisi, maupun praktisi di bidang Teknologi Informasi dan Komunikasi, untuk mengirimkan karya tulis ilmiah yang berkualitas pada International Journal of New Media Technology (IJNMT) maupun jurnal ULTIMATICS, ULTIMA InfoSys, dan ULTIMA Computing. Informasi mengenai pedoman dan *template* penulisan, serta informasi terkait lainnya dapat diperoleh melalui alamat surel umnjurnal@gmail.com.

Akhir kata, kami mengucapkan terima kasih kepada seluruh kontributor dalam jurnal ULTIMATICS edisi Juni 2015 ini. Kami berharap karya tulis-karya tulis ilmiah hasil penelitian dalam jurnal ini dapat bermanfaat dan memberikan sumbangsih terhadap perkembangan penelitian dan keilmuan di Indonesia.

Juni 2015,

Seng Hansun
Ketua Dewan Redaksi

Development of Catch-up TV Application on Internet TV Platform for RTSP/RTMP Streaming Protocol

I Ketut Agung Enriko
Ph.D Student, Departemen Teknik Elektro, Fakultas Teknik, Universitas Indonesia, Depok, Indonesia
agungenr@gmail.com

Diterima 28 April 2015
Disetujui 15 Mei 2015

***Abstract*—With the proliferation of video streaming services, now internet TV has become a popular service on the Internet. Catch-up TV is an attractive service in internet TV, where user who misses a live programme can watch it later. Meanwhile, not all streaming protocols, by default, are able to support catch-up TV application. This paper explains how to develop catch-up TV application in RTSP/RTMP streaming protocol, by doing shell-script programming and using Wowza streaming engine, on Linux-based streaming server. The research result has been implemented in UseeTV, one of the largest commercial internet TV service in Indonesia.**

***Index Terms*—internet TV, video streaming, *catch-up* TV.**

I. PENDAHULUAN

Dewasa ini teknologi *video streaming* di internet semakin berkembang, seiring dengan semakin cepatnya kecepatan akses internet [1]. Layanan internet TV semakin banyak tersedia sebagai sarana informasi dan hiburan, baik gratis atau berbayar, dan diselenggarakan banyak pihak misalnya stasiun TV atau perusahaan telekomunikasi. Salah satu layanan internet TV terbesar di Indonesia adalah UseeTV [2] yang sudah mempunyai lebih dari satu juta pelanggan [3].

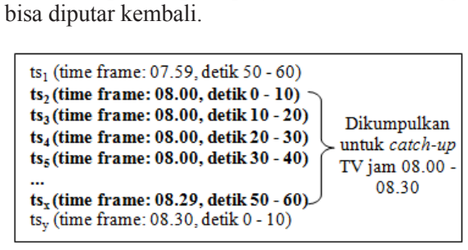
Salah satu fitur yang menarik dalam Internet TV adalah *catch-up* TV, yaitu layanan penayangan acara-acara TV yang sudah lewat, yang bisa dipilih sesuai keinginan pelanggan [4]. Beberapa penyelenggara internet TV sudah menyediakan layanan *catch-up* TV seperti UseeTV, tapi pada umumnya belum.

Aplikasi *catch-up* TV dibuat dengan merekam *video stream source* yang disimpan di *server storage* untuk kemudian di-*stream* juga jika diinginkan oleh pelanggan. Teknologi *streaming* yang mendukung aplikasi *catch-up* TV secara *default* adalah protokol *streaming* Hypertext Transfer Protocol (HTTP) Live Streaming atau HLS. HLS memecah *video source* menjadi *video chunk files* sehingga *video chunks* ini bisa disimpan di *server storage* untuk dijadikan layanan *catch-up* TV [5]. Protokol *streaming* lain misalnya RTSP (Real Time Streaming Protocol) dan RTMP (Real Time Messaging Protocol), secara *default* tidak mendukung *catch-up* TV sehingga harus dilakukan modifikasi pada *platform* Internet TV agar *catch-up* TV bisa tersedia. Tulisan ini akan menjelaskan bagaimana membangun aplikasi *catch-up* TV untuk protokol *streaming* RTSP/ RTMP dengan format video MPEG4/H.264, dengan menggunakan *streaming engine* Wowza [6] yang tersedia di pasaran. Metoda yang dilakukan adalah dengan melakukan *shell-script programming development* pada Linux-*based streaming server* agar *video stream source* dapat "dipecah-pecah", sesuai tabel jadwal acara TV yang dientrikan.

II. PROTOKOL STREAMING

Ada beberapa protokol *streaming* yang digunakan dalam layanan internet TV, misalnya HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP) [7], Real Time Messaging Protocol (RTMP) [8], dan Smooth Streaming [9].

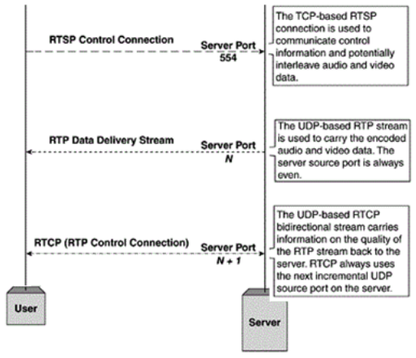
HLS diperkenalkan oleh Apple Inc., yaitu protokol *streaming* untuk men-*deliver multimedia streaming* dengan basis HTTP. Artikel [10] menjelaskan cara kerja HLS yaitu dengan memecah-mecah *audio/video streaming source* menjadi *files* kecil per durasi tertentu (misalnya per 10 detik), dengan format *file transport stream* (.ts). *Files* ini akan diindeks ke dalam *playlist file* sehingga video bisa diputar oleh pengguna internet dengan *video player* yang di-*embed* ke dalam internet *browser*. *Playlist file* tersebut biasanya mempunyai *extension* m3u. Setelah periode tertentu, *files* .ts akan dihapus karena akan memenuhi *storage* di *server*. Untuk keperluan *catch-up* tv, *files* .ts ini tidak dihapus dan dibuatkan *playlists* khusus berdasarkan waktu yang diinginkan. Misalnya, jika diinginkan acara tanggal 23 April 2015 pukul 08.00 sampai dengan 08.30 diputar kembali, maka *files* .ts yang ter-*create* pada *time frame* tersebut akan disimpan dan dikumpulkan pada *playlist* khusus sehingga bisa diputar kembali.



Gambar 1. Teknik membuat *catch-up* TV pada HLS

Walaupun HLS mempunyai beberapa kelebihan, misalnya di-*deliver* di protokol standar internet yaitu HTTP, tetapi belum semua perangkat pengguna mendukung HLS [11], misalnya perangkat Android versi 3 ke bawah, Blackberry OS7 ke bawah, dan Windows Phone, yang masih banyak digunakan di Indonesia.

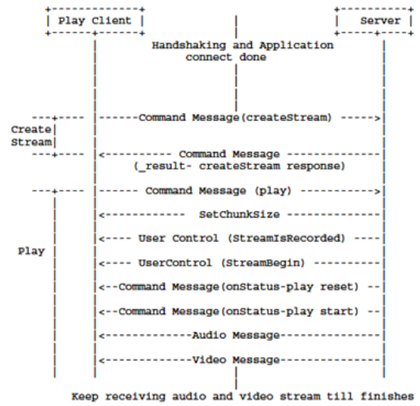
Protokol RTSP adalah protokol yang didesain untuk sistem komunikasi *multimedia streaming*. Artikel [7] menjelaskan bahwa dalam RTSP, *multimedia source* dikirim dari *server*, sedangkan klien menggunakan RTSP untuk *playback control* seperti *play* dan *pause*. RTSP bekerjasama dengan protokol lain yaitu RTP (Real Time Transport Protocol) untuk keperluan *transport*-nya, dan juga RTCP (Real Time Control Protocol) untuk *media stream delivery*. Berbeda dengan HLS, RTSP memerlukan *session* atau me-*maintain state* dalam implementasinya. RTSP tidak memproduksi *video file* secara fisik, sehingga untuk pembuatan *catch-up* TV, diperlukan modifikasi pada *layer* aplikasi. Kelebihan protokol RTSP adalah *streaming*-nya lebih *real time* (*minimum delay*) dibandingkan HLS misalnya, dan protokol ini masih didukung oleh banyak tipe perangkat seperti semua tipe Android, Blackberry, bahkan oleh *platform* IPTV [12]. Cara kerja RTSP secara umum dijelaskan pada Gambar 2.



Gambar 2. RTSP bekerjasama dengan RTP dan RTCP untuk men-*deliver streaming* audio/video [13]

Protokol RTMP diperkenalkan oleh Macromedia untuk keperluan *streaming* audio, video, dan data melalui Internet, di mana *server* akan mengirimkan data dan klien menggunakan Flash *player* agar dapat menikmati *multimedia streaming*. Artikel [8] menjelaskan bahwa RTMP me-*maintain* koneksi dan mempunyai keunggulan *latency* yang rendah. Cara kerjanya, *server* akan memisahkan *stream* dalam bentuk fragmen-fragmen yang akan digabungkan dan dikirimkan dalam koneksi tunggal, dengan *overhead* yang sangat kecil demi efisiensi. RTMP mengenkapsulasi *multimedia stream* dalam format MP3 (MPEG-1 Audio Layer 3) atau AAC (Advanced Audio Coding) untuk audio, serta FLV (Flash Video) untuk video. *Basic flow* dari *video stream* dalam *protocol* RTMP digambarkan pada

Gambar 3.



Gambar 3. *Basic flow* protokol RTMP dalam men-*deliver stream* audio/video [8]

Seperti halnya RTSP, RTMP secara *default* tidak memproduksi *file-file* fisik seperti HLS. Sehingga untuk pembuatan *catch-up* TV juga perlu dilakukan modifikasi di level aplikasi.

III. ANALISIS PEMILIHAN STREAMING ENGINE DAN ARSITEKTUR PLATFORM USEETV

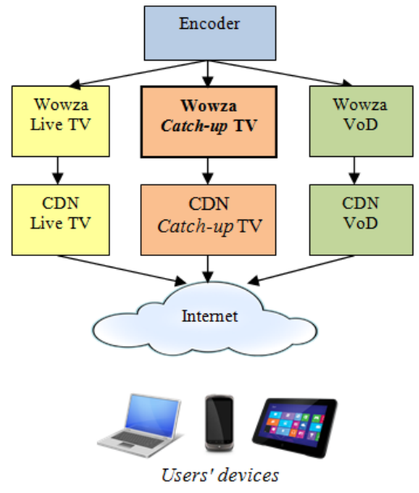
A. Wowza Streaming Engine

Layanan internet TV memerlukan *streaming engine* agar *video source* bisa ditayangkan ke penonton melalui jaringan internet. Salah satu *streaming engine* yang sangat populer adalah Wowza, yang merupakan *commercial off-the-shelf software*. Rujukan [6] mencatat ada lebih dari 80 ribu lisensi telah terjual di lebih dari 150 negara. Wowza mempunyai keunggulan dapat men-*deliver video streaming* dengan banyak format, dari satu format video input saja, sehingga lebih efisien. Beberapa tipe format keluaran yang didukung oleh Wowza adalah: HLS, RTSP, RTMP, Smooth Streaming, dan lain-lainnya. Dengan demikian *video streaming* keluaran dari Wowza dapat ditonton dari berbagai tipe perangkat *user* misalnya: PC/laptop, Apple iPad/iPhone, *smartphone* Android, Blackberry, Windows Phone, bahkan *set-top box*. *Platform* UseeTV menggunakan Wowza sebagai *streaming engine*-nya karena pelanggannya terdiri atas bermacam-macam jenis dan merk *device* sehingga beragam *streaming protocol* harus di-*support* oleh UseeTV.

Untuk keperluan *catch-up* TV, Wowza versi 3.5 ke atas telah menyediakan modul livestreamrecord. Modul ini dapat merekam *video streaming* yang di-*broadcast* ke internet, dan hasilnya berupa *file* berformat MPEG4/H.264 yang ada di *storage server*.

B. Arsitektur Platform UseeTV

Pembuatan aplikasi *catch-up* TV dalam riset ini dilakukan di *platform* internet TV: UseeTV [2]. Arsitektur *platform* UseeTV digambarkan pada Gambar 4.



Gambar 4. Arsitektur *platform* UseeTV

Dalam platform UseeTV, ada 3 layanan *video streaming* yaitu Live TV (siaran TV secara langsung), *catch-up* TV (acara TV yang sudah lewat, atau disebut juga TV on Demand atau TVoD), dan Video on Demand (VoD). *Encoder* memberikan *source* video dengan format MPEG4/H.264 dalam protokol RTMP. Untuk layanan Live TV, *server* Wowza Live TV men-*stream* video secara *live* ke CDN (*content*

distribution network) yang melayani permintaan *streaming* dari *user*. Sedangkan untuk layanan *catch-up* TV, *server* Wowza *catch-up* TV akan melakukan proses rekam sesuai jadwal acara, dan hasilnya akan ditransfer ke CDN untuk melayani *request* dari pengguna. Pada layanan VoD, Wowza VoD mendapat *output file* dari *encoder*, dan akan di-*copy* ke CDN VoD untuk melayani *request* dari pengguna.

IV. IMPLEMENTASI CATCH-UP TV DI WOWZA STREAMING SERVER

Development aplikasi *catch-up* TV di platform UseeTV dilakukan di *server* Wowza *catch-up* TV (Gambar 4) yang berbasis Linux. Aplikasi dibangun dengan pemrograman *shell-scripting*. *Logic flow* dari aplikasi *catch-up* TV di UseeTV adalah sebagai berikut:

1. Rekam *video stream* Live TV dengan modul livestreamrecord.

Pada Wowza *streaming engine* versi 3.5 ke atas tersedia modul livestreamrecord untuk merekam *video stream* yang berasal dari *encoder*, dan *file* fisiknya disimpan di *server* Wowza *catch-up* TV. *File* ini bertambah besar secara kontinyu sampai ada perintah stop pada modul livestreamrecord.

2. *Copy* jadwal acara dari *data base*.

Server Wowza *catch-up* TV mendapatkan jadwal acara dari *data base* acara yang diinput di CMS (*Content Management System*). *Server* Wowza perlu diprogram agar setiap 24 jam sekali melakukan *append* jadwal acara setiap *channel* TV, dan *file*-nya diletakkan di *folder* /schedules sebagai *text file*. Contoh isi dari *schedule file* ini seperti tergambar pada Gambar 5.

2015.04.24 22:00	2015.04.24 22:30	Acara 1
2015.04.24 22:30	2015.04.24 23:00	Acara 2
2015.04.24 23:00	2015.04.25 00:00	Acara 3
2015.04.25 00:00	2015.04.25 01:30	Acara 4
2015.04.25 01:30	2015.04.25 02:30	Acara 5
t_{start}	t_{end}	nama_acara

Gambar 5. Contoh isi *file* jadwal acara TV

3. *Scheduled check* per 1 menit untuk melihat apakah waktu saat ini = waktu akhir sebuah acara ($t_{current} = t_{end(i)}$) untuk melakukan pemotongan *file* video.

Pada Linux *operating system* terdapat modul cronjob untuk menjalankan program secara periodik, berisi perintah yang diset oleh *programmer*. Aplikasi *catch-up* TV menggunakan cronjob per 1 menit untuk melihat apakah waktu saat ini sama dengan waktu akhir sebuah acara. Jika kondisi tersebut terpenuhi, maka dilakukan stop pada livestreamrecord agar Wowza berhenti merekam, kemudian *start* lagi agar Wowza mulai merekam lagi. Hasil perintah stop akan menghasilkan *file* mp4 yang disimpan di *server*, di *folder* /mp4. Ringkasan logikanya tergambar di Gambar 6.

```
if t_current = t_end(i)
   then
      stop record
      save video in folder /mp4
      start record
end if
```

Gambar 6. Logic untuk pemotongan video

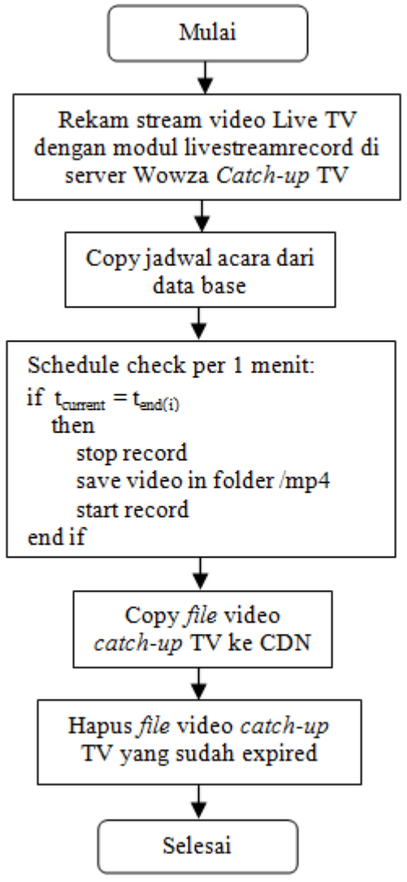
4. Copy *file* video *catch-up* TV ke CDN.

Setelah *catch-up* TV *files* untuk setiap *channel* TV ter-*create*, perlu dilakukan *copy file* ke *server* CDN *catch-up* TV agar video dapat dinikmati oleh pengguna yang menginginkan. Proses *copy* dilakukan dengan metode "rsync" antar *server* Linux.

5. Hapus *file* video *catch-up* TV yang sudah *expired*.

Pada aplikasi *catch-up* TV, perlu dilakukan pembatasan sampai berapa lama *catch-up* TV masih bisa ditayangkan. Di layanan UseeTV, *catch-up* TV diset 7 hari, yang berarti acara TV yang sudah lewat yang bisa disaksikan adalah maksimal 7 hari yang lalu. Jika lebih dari 7 hari, *file* video *catch-up* TV akan dihapus secara otomatis dari *server*.

Secara umum, diagram alir dari aplikasi *catch-up* TV tergambar pada Gambar 7.



Gambar 7. Diagram alir aplikasi *catch-up* TV di *server* Wowza

V. HASIL IMPLEMENTASI

Script command untuk pemotongan *file* video disimpan di *folder* /scripts dan dijalankan oleh cronjob *service* secara periodik per menit. Hasil dari *script* ini disimpan di *folder* mp4 untuk kemudian di-*copy* ke CDN untuk dapat diakses oleh penonton.

```
root:/ mp4/metrotv# ll -h|more
-rw-r--r--   1 root root 160M Apr 19 21:05
metrotv_201504192105.mp4
-rw-r--r--   1 root root  42M Apr 19 21:30
metrotv_201504192130.mp4
-rw-r--r--   1 root root  51M Apr 19 22:00
metrotv_201504192200.mp4
-rw-r--r--   1 root root  51M Apr 19 22:30
metrotv_201504192230.mp4
-rw-r--r--   1 root root  59M Apr 19 23:05
metrotv_201504192305.mp4
```

Gambar 8. Contoh *file-file catch-up* TV di folder /mp4

Catch-up TV *files* ini berformat MPEG4/H.264 yang akan di-*stream* oleh Wowza dalam beberapa protokol yaitu RTSP, RTMP, HLS, dan Smooth Streaming untuk dapat ditayangkan di berbagai macam *client device*: PC/laptop (RTMP), iPad/iPhone (HLS), Android *smartphone*/*tablet* (RTSP), Blackberry (RTSP), dan Windows Phone (Smooth Streaming). Hasil *catch-up* TV ditampilkan di *website*, aplikasi Android, iOS, Blackberry, dan Windows Phone. Gambar 9 adalah contoh tampilan *catch-up* TV di aplikasi Android.

Gambar 9(a)(b). Tampilan *catch-up* TV di aplikasi Android UseeTV

Dari hasil implementasi ini, beberapa jenis *device* (Android versi 3 ke bawah, Blackberry OS7 ke bawah, dan Windows Phone) yang secara *default* tidak di-*support* oleh layanan *catch-up* TV jadi bisa menikmati layanan *catch-up* TV.

Kelanjutan dari penelitian ini kemungkinan akan berfokus ke kualitas *video streaming* dengan *bandwidth* yang lebih efisien. Di mana kita ketahui saat ini teknologi kompresi H.265 sudah mulai berkembang untuk menggantikan H.264. Keuntungan dari teknologi H.265 adalah efisiensi penggunaan *bandwidth* yang mencapai sekitar 40% dibandingkan H.264 [14].

VI. SIMPULAN

RTSP/RTMP merupakan protokol *streaming* yang sudah dipergunakan secara luas, tetapi secara *default* tidak mendukung aplikasi *catch-up* TV. Dengan modifikasi pada *platform* internet TV, menggunakan modul livestreamrecord pada Wowza 3.5 ke atas, dan pemrograman *shell-script* pada Linux *operating system*, maka aplikasi *catch-up* TV dapat dibuat. Aplikasi *catch-up* TV pada tulisan ini sudah diimplementasikan di layanan internet TV UseeTV.

UCAPAN TERIMA KASIH

Terima kasih diucapkan kepada PT Telkom Indonesia, yang juga tempat penulis bekerja untuk mengembangkan layanan internet TV UseeTV.

DAFTAR PUSTAKA

[1] Breitman, Karin, et al. “When TV dies, will it go to the cloud?.” Computer 43.4 (2010): 81-83.

[2] PT Telkom Indonesia, http://www.useetv.com. Diakses tanggal 15 Mei 2015.

[3] Achmad Rouzni Noor, http://inet.detik.com/read/2014/09/07/145134/2683613/328/2/useetv-diharap-bisa-sebesar-netflix. Diakses tanggal 15 Mei 2015.

[4] Cambridge Advanced Learners Dictionary & Thesaurus, http://dictionary.cambridge.org/dictionary/british/*catch-up*-tv. Diakses tanggal 15 Mei 2015.

[5] Pantos, R. (30 September 2011). “HTTP Live Streaming”. Internet Engineering Task Force.

[6] Mantoro, T., M. A. Ayu, and D. Jatikusumo. “Live video streaming for mobile devices: An application on Android platform.” Uncertainty Reasoning and Knowledge Engineering (URKE), 2012 2nd International Conference on. IEEE, 2012.

[7] Schulzrinne, Henning. “Real Time Streaming Protocol (RTSP).” (1998).

[8] Parmar, Ed H., and Ed M. Thornburgh. “Real-Time Messaging Protocol (RTMP) specification.” Adobe specifications, December (2012).

[9] Zambelli, Alex. “IIS Smooth Streaming technical overview.” Microsoft Corporation 3 (2009).

[10] Jordan, Larry (10 June 2013). “The basics of HTTP Live Streaming”. Larry’s Blog. Larry Jordan & Associates.

[11] Andriescu, Emil, Roberto Speicys Cardoso, and Valérie Issarny. “AmbiStream: a middleware for multimedia streaming on heterogeneous mobile devices.” Middleware 2011. Springer Berlin Heidelberg, 2011. 249-268.

[12] Foti, George. “Method and Internet Protocol Television (IPTV) content manager *server* for IPTV servicing.” U.S. Patent No. 7,716,310. 11 May 2010.

[13] Syme, Matthew, and Philip Goldie. Optimizing network performance with content switching: server, firewall, and cache load balancing, Chapter 3. Understanding application layer protocols. Prentice Hall Professional, 2004.

[14] Grois, Dan, et al. “Performance comparison of H. 265/MPEG-HEVC, VP9, and H. 264/MPEG-AVC encoders.” PCS. Vol. 13. 2013.

Komparasi Algoritma Quicksort dan Bucket Sort pada Pengurutan Data Integer

Audy
Program Studi Teknik Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
audytanudjaja@gmail.com

Diterima 31 Maret 2015
Disetujui 08 Mei 2015

***Abstract*—Data sorting is a technique that widely used as a part of a bigger process. Therefore, data sorting should not be the problem of program complexity. This paper gives the reader a comparison between two sorting algorithms, which are comparison based and non-comparison based, in time and space performance. The data type that used in this paper is an integer data type. Testing is carried out by using two types of data’s condition, which are the worst-case condition in each algorithm, and two amounts of data, which represent the maximum and minimum amount data.**

***Index Terms*—quicksort, bucket sort, pseudocode, cost, pivot, bucket, rekursif, integer, divide and conquer**

I. PENDAHULUAN

Perkembangan teknologi dunia sangat pesat. Hal ini terbukti dari kecepatan dan kemudahan dalam penerimaan suatu informasi. Informasi terbentuk dari hasil pemrosesan data. Di dalam buku *Reference Model for an Open Archival Information System (OAIS)* [1], data adalah suatu hal yang dapat diterjemahkan dan direpresentasikan ke dalam bentuk formal agar dapat digunakan untuk komunikasi, interpretasi, atau pengolahan informasi. Seiiring berkembangnya teknologi, perkembangan jumlah data yang dapat kita olah semakin besar sehingga dibutuhkan suatu cara untuk dapat mengolah data secara efisien dan efektif.

Pengolahan data erat kaitannya dengan pencarian data, dimana dalam pencarian tersebut terdapat proses memilah-milah data sesuai kebutuhan. Pencarian data yang efektif dan efisien tidak dapat dilepaskan dari faktor keterurutan data. Data yang sudah terurut akan mempermudah dan mempercepat pencarian data. Oleh karena itu, dibutuhkan suatu algoritma yang dapat mengurutkan data secara benar, efektif, dan efisien.

Menurut Astrachan (2003) [2], algoritma pengurutan, atau yang biasa dikenal sebagai *Sorting Algorithm*, telah muncul sejak tahun 1956. Algoritma tersebut dikenal dengan nama *Sorting by Exchange*. Seiiring berjalannya waktu, berbagai macam metode dalam algoritma pengurutan terus ditemukan sampai saat ini. Canaan dkk. (2011) [3] memberikan beberapa contoh dari algoritma pengurutan yang populer, yakni Bubble Sort, Insertion Sort, Selection Sort, Shell Sort, Merge Sort, Heapsort, Quicksort, dan Bucket Sort.

Setiap algoritma pengurutan memiliki pendekatan dan metode yang berbeda-beda dalam menjalankan fungsinya. Secara garis besar, algoritma pengurutan dapat dikelompokkan menjadi dua kategori, yaitu algoritma pengurutan berbasis perbandingan (*comparison based*) dan tidak berbasis perbandingan (*non-comparison based*). Joshi R., Panwar G.R., dan Pathak P. (2013) [4] menyatakan bahwa pada umumnya, algoritma pengurutan yang tidak berbasiskan perbandingan lebih cepat dibandingkan algoritma pengurutan yang berbasiskan perbandingan. Namun di sisi lain, dalam buku *Data Structures & Algorithms in Java* yang dikarang oleh Lafore (2003) [5] menyatakan bahwa algoritma Quicksort merupakan algoritma pengurutan tercepat secara praktis, dimana algoritma Quicksort merupakan