Aplikasi Android Untuk Backup dan Sinkronisasi File Menggunakan Amazon Web Services Simple Storage Service

Brian Stanley, Henry Novianus Palit, Agustinus Noertjahyana
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
rocketparadise@gmail.com, hnpalit@petra.ac.id, agust@petra.ac.id

ABSTRAK

Teknologi cloud storage merupakan solusi dari local storage yang membutuhkan waktu maintenance tinggi dari pihak pengguna. Amazon Web Service (AWS) menyediakan storage yang dapat dimanfaatkan untuk membackup data, namun untuk menggunakan fitur S3 pengguna harus mengakses lewat web browser. Solusi untuk mempermudah penggunaan S3 yaitu memanfaatkan aplikasi mobile yang dapat mengautomasikan proses backup dan restore data.

Untuk mengatasi masalah putusnya koneksi yang mengakibatkan proses transfer harus diulang dari awal, maka menggunakan API *multipart upload* yang disediakan oleh AWS. Ketika sebuah upload mengalami putus koneksi internet, TransferUtility dari AWS automatis memerintahkan resume terhadap upload yang gagal.

Hasil pengujian menyimpulkan bahwa AWS S3 memiliki koneksi internet yang setara dengan *cloud storage* lainnya seperti *Google Drive*, *Dropbox*, dan *OneDrive*. Kelebihan dari S3 yaitu kemampuannya untuk me-resume transfer yang telah gagal akibat koneksi internet yang putus secara automatis tanpa perintah manual terlebih dahulu.

Kata Kunci: *cloud storage, backup & restore, file synchronization, Amazon Web Services, Simple Storage Service*

ABSTRACT

Cloud storage technology is a solution for the local storage which consumes lot of time for maintaining in the customer's side. Amazon Web Service (AWS) provides S3, a storage that can be used for data backup purposes, but to use its features everytime users need to login through web browser. A solution to ease the use of S3 features is to utilize a mobile app that could automate the data backup and restore process.

To solve the problem of unstable connection causing file transfers to start from the beginning each time, AWS provides an API that allows resumable uploads. TransferUtility from AWS will automatically issue a resume operation on the failed upload.

Test results conclude that AWS S3 have a comparable connection to other cloud storage providers such as Google Drive, Dropbox, and OneDrive. The strength of S3 is its ability to automatically resume failed transfers caused by an unstable internet connection.

Keywords: *cloud storage, backup & restore, file synchronization, Amazon Web Services, Simple Storage Service*

1. PENDAHULUAN

Aktivitas *transfer file* di jaman *internet-of-things* sekarang semakin tinggi. Dengan meningkatnya intensitas *transfer file*, frekuensi data yang terunduh ke dalam *storage* lokal bertambah banyak. Banyaknya data yang disimpan ini dapat menimbulkan kurang efisiensinya penggunaan *space storage* lokal, apalagi jika data yang disimpan tidak digunakan secara maksimal dan hanya sekedar arsip yang dibutuhkan ketika perlu. Untuk mengatasi masalah tersebut, vendor seperti Google, Dropbox, Amazon menyediakan layanan *cloud storage*, media penyimpanan berbasis *cloud* yang tidak menggunakan *space* dalam *storage* lokal, melainkan data yang ada disebar ke dalam kumpulan data center.

Cloud storage memperbolehkan pengguna untuk mengakses dan mengunggah file pada saat kapanpun dan dimanapun berada, asalkan tersedianya koneksi jaringan internet yang stabil.

Untuk mengakses *cloud storage* diperlukan sebuah aplikasi yang memiliki kemampuan untuk memanajemen file, seperti melakukan pengunggahan, pengunduhan, menghapus dan mengubah isi file yang tersimpan di dalam *cloud storage* yang terhubung. Beberapa penyedia cloud storage seperti *Dropbox*, *Google Drive*, dan *Microsoft OneDrive* menyediakan aplikasi untuk melakukan hal tersebut. *Dropbox* dan *Google Drive* merupakan kedua aplikasi yang paling populer[7] dalam persaingan aplikasi *cloud storage*. Salah satu penyedia *cloud storage* lain yaitu Amazon Web Services (AWS) dengan Simple Storage Service (S3). S3 memiliki tipe penyimpanan berupa *object storage* yang dianggap handal untuk menyimpan cadangan dan melakukan pengarsipan data[2].

Dalam melakukan manajemen file dalam S3, terdapat AWS Console yang dapat diakses menggunakan web browser. Akan tetapi hal ini kurang praktis bagi pengguna yang ingin melakukan pengunggahan dengan cepat karena diharuskan memasukkan URL dari website AWS Console yang cukup panjang. Sedangkan untuk aplikasi mobile dalam platform Android / iOS yang disediakan hanya dapat melihat daftar *bucket* S3 yang sedang aktif, tidak dapat melihat isi dan mengunggah maupun menghapus file.

Untuk memudahkan penggunaan S3 sebagai metode *backup*, maka diperlukan suatu sistem otomasi yang berjalan dalam *background* sehingga tidak perlu dilakukan secara manual. Dalam mengatasi skenario dimana kondisi internet yang kurang stabil menyebabkan kegagalan proses unggah / unduh, maka file sebelum diunggah/unduh akan terbagi menjadi kumpulan *chunk* yang memiliki jumlah chunk tergantung ukuran asli dari file tersebut. Setelah semua file dichunk maka barulah dilakukan proses unggah/unduh. Ketika semua chunk unggah/unduh berhasil maka akan dilakukan proses penyatuan dari kumpulan chunk tersebut menjadi satu file utama. File yang telah disatukan ini lalu

akan siap diakses seperti layaknya file asli. Setelah file telah terbackup dalam S3, maka pengguna dapat mengakses file tersebut sewaktu-waktu dengan device yang berbeda asal terdapat koneksi internet, ataupun jika pengguna merasa file tersebut perlu diakses secara langsung, maka pengguna dapat melakukan *restore* untuk mendapatkan file yang telah dibackup di dalam smartphone pengguna.

2. LANDASAN TEORI

2.1. *Amazon Web Services*

Merupakan bagian dari perusahaan *Amazon* yang menyediakan platform *on-demand cloud computing* bagi individu, perusahaan, maupun pemerintah, dengan sistem berlangganan berbayar. Teknologi yang disediakan oleh AWS yang paling sering digunakan yaitu Simple Storage Service (S3) dan Elastic Compute Cloud (EC2).Untuk mengakses API yang telah disediakan oleh AWS, maka pengguna API membutuhkan pasangan (*key-pair*) *access-key* dan *secret-key* yang dapat di-*generate* dalam akun pengguna. *Access-key* dan secret-key merupakan hal yang wajib untuk mencegah terjadinya penyalahgunaan *resource* dari pihak luar yang tidak mempunyai hak akses terhadap API AWS. Jika pengguna ingin membagi *resource* yang disediakan kepada pengguna lain namun tak ingin membocorkan informasi mengenai *access-key* dan secret-key, maka AWS menyediakan IAM (*Identity & Access Management*). IAM merupakan web service yang memperbolehkan pengguna untuk membuat akun individu bagi orang yang ingin memanfaat *resource* tanpa memerlukan *root* account. Setiap individu ini akan diberi *security credentials* yang diperlukan sebagai pembuktian identitas sebagai IAM *user*. Selain itu *root* account juga dapat mengatur hak yang dapat diakses oleh tiap *IAM user* menggunakan *group permission*, yang dapat diatur sesuai peran dari tiap *IAM user* missal seperti Administrator, Developer, Accounting, dll.

2.2. *Simple Storage Service (S3)*

Sistem penyimpanan *cloud* berbasis *object storage* yang merupakan salah satu layanan yang disediakan Amazon Web Services. Terdiri dari 4 jenis *class*, yaitu S3 Standard, S3 Standard Infrequent Access, S3 One Zone Infrequent Access, dan Glacier. Kapasitas tidak dibatasi dengan ukuran, namun dengan bertambahnya jumlah *object* dalam S3 maka biaya yang dikeluarkan pengguna per bulan juga akan meningkat. Amazon S3 menyediakan API untuk dapat melakukan manajemen S3. Tersedia pula console untuk melakukan manajemen S3, dimana Console ini juga memanfaatkan S3 API dalam mengirim request ke Amazon S3. Objek dalam S3 disimpan dalam *bucket* yang memiliki nama yang unik secara global, terikat dengan semua akun AWS yang telah terdaftar. Hal ini mencegah pembuatan *bucket* menggunakan nama yang sama dengan *bucket* yang sudah terdaftar. Untuk mengatasi jeda waktu koneksi internet dan mengurangi biaya pemakaian, AWS memberikan kebebasan untuk memilih daerah tempat *bucket* akan dibuat[1]. Misalnya jika berada di daerah Eropa, membuat bucket di daerah EU (*Ireland*) atau EU (*Frankfurt*) dapat meningkatkan performa.

Pada umumnya *storage* yang digunakan EC2 (*Elastic Cloud Compute*) Amazon menggunakan EBS (*Elastic Block Storage*) sebagai media penyimpanan sementara. S3 digunakan sebagai media penyimpanan file yang permanen.

Objek dalam sebuah *bucket* S3 memiliki *key* yang merupakan identifier unik dari objek tersebut. Bucket S3 menggunakan data model berbentuk *flat structure*, dimana tidak terdapat *subfolder* dan *subbucket* di dalam sebuah bucket. Secara teknis, tidak terdapat fitur folder dalam S3, namun S3 *console* menyediakan fitur folder dengan mengelompokkan object yang memiliki prefix yang sama. Untuk mengakses objek melalui URL dari bucket, dapat diakses lewat browser atau via API REST dengan format penulisan berikut:

Syntax:
http://(nama_bucket).s3.amazonaws.com/(nama_object)
Contoh:
http://bucketku.s3.amazonaws.com/objekku.txt

2.2.1. Object Storage

Jenis *storage* yang memperlakukan tiap data sebagai objek, dimana tiap objek memiliki *metadata* mengenai *data* tersebut dan sebuah ID yang unik untuk mewakili. *Object storage* memiliki kemudahan akses dibandingkan *block storage*, karena dapat diakses menggunakan API atau protokol HTTP/HTTPS, dan juga tidak harus terikat kedalam suatu operating sistem[11]. Object storage memperbolehkan penyimpanan metadata secara ekstensif, memudahkan kebijakan seperti kompresi data dan pencarian dengan skalabilitas yang tinggi.

2.2.2. ETAG

Adalah metode S3 untuk memverifikasi integritas objek. Sebuah objek yang tidak diupload menggunakan *multipart upload API* memiliki ETAG yang jika dibandingkan dengan *checksum* MD5 dari file aslinya hasilnya akan sama. Jika *file* diupload menggunakan *multipart upload API*, maka ETAG objek yang dikomputasikan akan berbeda. Untuk mengetahui ETAG hasil *multipart upload API*, pertama bagi file menjadi x bagian sesuai dengan besar tiap part yang diinginkan. Lalu kalkulasikan MD5 hash dari tiap file, dan secara urut, gabungkan kumpulan MD5 tersebut menjadi 1 string. Setelah itu, kalkulasikan MD5 hash dari string tersebut, dan tambahkan symbol "-" dan jumlah part dari file tersebut di akhir MD5 hash.

2.3. *File synchronization*

Merupakan metode untuk menjamin dua *file* tersimpan dalam kumpulan *device* yang berbeda agar memiliki versi yang sama dan terbaru. Umumnya keuntungan file synchronization yaitu :

- Menghemat waktu dan biaya, karena berjalan secara otomatis
- Kebebasan dalam memilih data yang disinkronisasikan
- Jika terjadi perubahan isi data/*file*, maka seluruh file yang sama akan mengalami perubahan setara
- Kemudahan untuk menyimpan versi dari file jika ingin mempunyai backup berlapis-lapis

File synchronization sendiri dibagi menjadi dua jenis metode yaitu *one-way synchronization* dan *two-way synchronization*.

- *One-way synchronization* (disebut juga *mirroring/file replication/file backup*), *file* dianggap hanya mengalami perubahan dalam satu lokasi saja. Untuk mengkonfirmasi perubahan, proses sinkronisasi meng-*copy* melewati satu arah saja. Salah satu lokasi akan ditentukan sebagai *source* dan lokasi lainnya sebagai *target*. *File* akan selalu dicopy dari *source* menuju ke *target*.
- *Two way synchronization* (disebut juga *both-ways/bi-directional synchronization*), proses sinkronisasi berjalan

dari dua arah, *source* ke *target* dan *target* ke *source.* Kedua lokasi ini dianggap setara.

2.4. *Checksum*

Merupakan metode *hash function* untuk melakukan mengecek integritas dari suatu data / file. *Checksum* yang paling sering digunakan adalah MD5 (*Message-Digest Algorithm*) dan SHA (*Secure Hash Algorithm*). MD5 dikembangkan oleh Ronald Rivest di MIT, untuk menggantikan metode *hash function* sebelumnya MD4. MD5 didesain untuk dapat berjalan lebih cepat pada mesin berbasis 32-bit. [12]

2.5. HTTP (*Hypertext Transfer Protocol*)

Adalah protokol *application-level* untuk sistem kolaboratif, terdistribusi, dan informasi hypermedia. HTTP mendefinisikan banyak format dan arti dari pertukaran pesan-pesan di internet yang melakukan transaksi antara komponen Web, contohnya *client* dan *server*. HTTP mendefinisikan syntax dari tiap pesan dan bagaimana bagian per baris tiap pesan dapat diartikan. HTTP sudah banyak digunakan sejak tahun 1990 oleh *World-Wide web global information initiative*.[5]. HTTP merupakan protokol yang *stateless*, dimana client dan server tidak menyimpan informasi mengenai kondisinya saat melakukan pertukaran *request* dan *response*.

2.6. *Chunked transfer encoding*

Mekanisme transfer dalam protokol HTTP 1.1 yang memperbolehkan pengiriman file yang dibagi dalam kumpulan potongan (chunk). Tiap potongan dikirim secara independen dan tidak tergantung terhadap potongan lain. Tiap potongan akan dikirim dengan ukurannya sebagai header, jika ditemukan chunk yang memiliki ukuran 0 byte, maka transmisi akan berhenti dan dianggap selesai.[6]. Metode yang memanfaatkan mekanisme ini disebut *chunking* dan banyak digunakan oleh penyedia service sinkronisasi lainnya, seperti *Google Drive* dan *Dropbox*[4]. *File* yang ditransfer akan melalui proses pembagian menjadi banyak bagian dengan ukuran yang sudah ditentukan (*fixed-sized chunking*) maupun bervariasi (*variable-sized chunking*). Tiap *chunk* akan diunggah secara berurutan, dan ketika ada interupsi jaringan yang menyebabkan kegagalan maka *file* dapat diproses mulai dari setelah *chunk* terakhir yang terunggah.

2.7. *Multipart Upload API*

Merupakan implementasi dari chunked transfer encoding yang diterapkan dalam API AWS S3. Terdiri dari 3 langkah:

1. Inisialisasi *upload*
2. Upload masing-masing part dari file
3. Lakukan *request complete upload*

Setelah menerima request complete upload, S3 otomatis menyusun file baru dari kumpulan part yang diupload, dan file dapat diakses seperti layaknya file lainnya dalam bucket.[3]

2.8. Tinjauan Studi

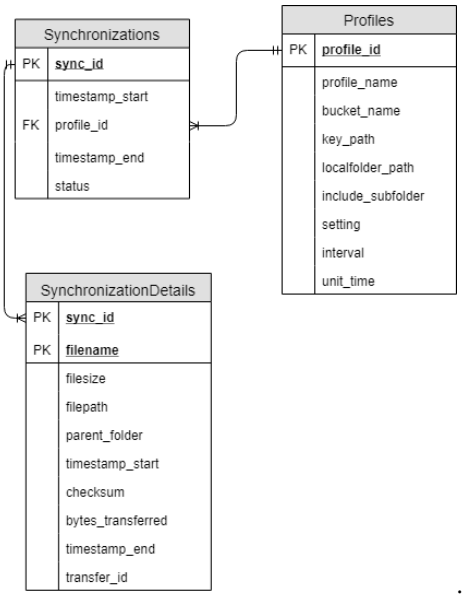
Dalam paper yang dibuat oleh mengenai *Segmented File Transfer* [8], diusulkan metode untuk mentransfer suatu file dalam ukuran yang besar dengan memanfaatkan kumpulan server melakukan proses *download/upload* secara bersamaan. Tiap server mendapat bagian yang berbeda dari *file*. Setiap bagian yang telah di-*download* lalu akan disatukan kembali menjadi file aslinya. Ada kesamaan skripsi ini dengan paper tersebut yaitu pembagian file menjadi banyak bagian, yang lalu disatukan kembali ketika semua bagian selesai ter-*download/upload.* Namun dalam *Segmented File Transfer*, terdapat kumpulan server yang diberi tugas masing-masing untuk men-*download file* tersebut dan dalam hasil yang ingin dicapai adalah peningkatan performa kecepatan, sedangkan dalam proses *Chunking* hanya satu computer saja yang membagi file menjadi beberapa bagian dan melakukan *download* satu persatu, serta hasil yang ingin dicapai adalah *fault tolerance* terhadap konektivitas yang mudah terputus.

Penelitian mengenai peningkatan traffic jaringan yang dihasilkan oleh aplikasi backup sejenis Dropbox dan Google Drive[9]. Dalam mengatasi masalah tersebut peneliti menggunakan mekanisme UDS (Update-batched delayed synchronization) untuk mengurangi overhead yang disebabkan oleh *traffic*. UDS berhasil mengatasi masalah dari overhead dalam traffic dengan beberapa detik delay waktu transfer. Penelitian serupa mengenai peningkatan efisiensi penggunaan traffic dalam cloud storage services.

Dalam percobaannya peneliti memanfaatkan 3 metode yaitu *network-aware chunker*, *redundancy eliminator*, dan *batched syncer* yang berhasil mengurangi waktu sinkronisasi sebanyak 52.9% [4] .

Studi mengenai *cloud storage* dan sinkronisasi, serta kebutuhannya dalam penelitian dan riset pernah dilakukan sebelumnya [10]. Studi ini berfokus mengenai pentingnya sebuah service untuk dapat berinteraksi dengan *service* lain yang ada dan juga performa dalam *cloud storage*.

3. DESAIN SISTEM



Gambar 1. ***Entity Relationship Diagram***

Seperti yang tertera pada Gambar 1, tabel yang digunakan adalah tabel profiles, synchronizations, dan synchronization details.

Tabel synchronizations digunakan untuk menyimpan data keseluruhan dari sinkronisasi, yaitu total ukuran file yang disinkronisasikan, dan waktu mulai dan selesai sinkronisasi.

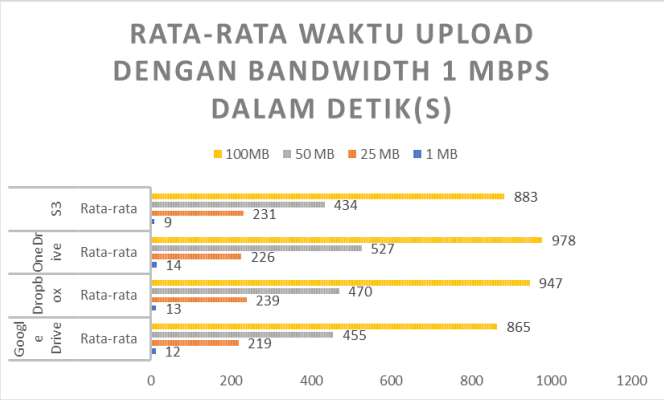
Tabel synchronization details digunakan untuk menyimpan rincian dari sinkronisasi, terdiri dari ukuran / size dari file, waktu /

timestamp nama file dan status sinkronisasi gagal / berhasil. Tabel ini juga mencatat checksum dari file yang mengalami sinkronisasi.

Tabel profile digunakan untuk menyimpan path bucket dan folder yang digunakan, serta preferensi jenis setting yang digunakan, dan konfigurasi bandwidth minimum dan maksimum.

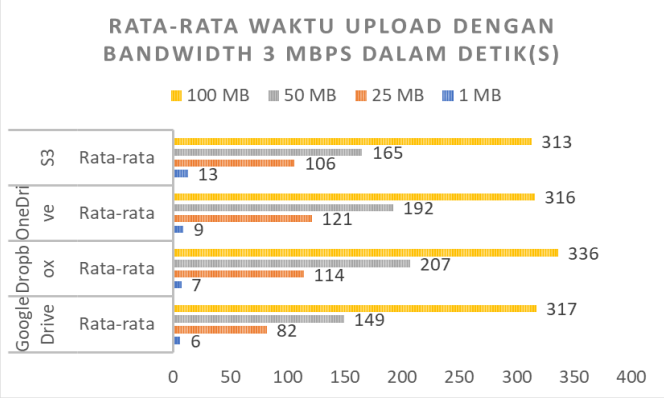
4. HASIL DAN PEMBAHASAN

4.1. Pengujian Upload dan Download



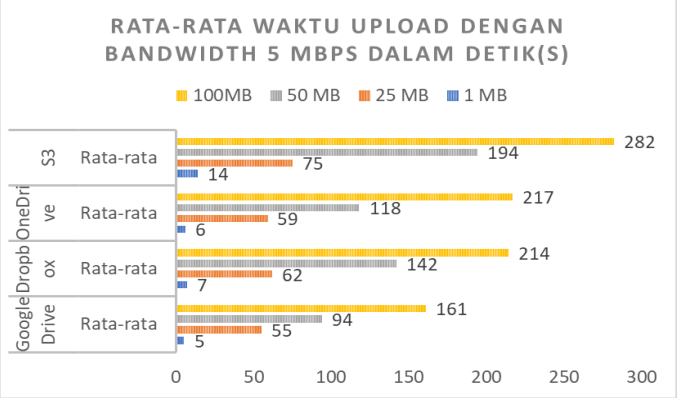
Gambar 2. Chart rata-rata kecepatan *upload* 1 Mbps

Dalam Gambar 2, terdapat grafik yang menampilkan rata-rata kecepatan *upload* dengan *bandwidth* 1 Mbps.



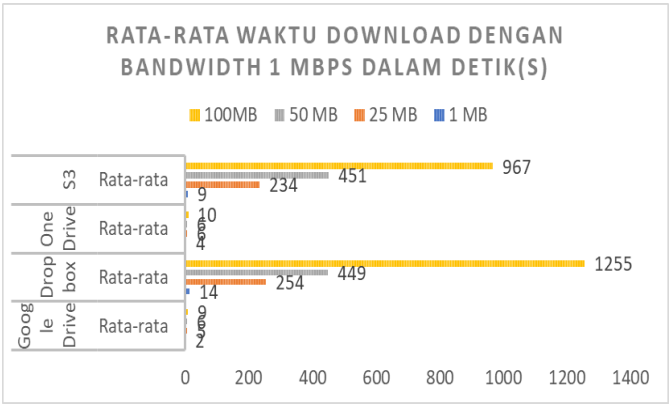
Gambar 3. Chart rata-rata kecepatan *upload* 3 Mbps

Dalam Gambar 3, terdapat grafik yang menampilkan rata-rata kecepatan *upload* dengan *bandwidth* 3 Mbps.



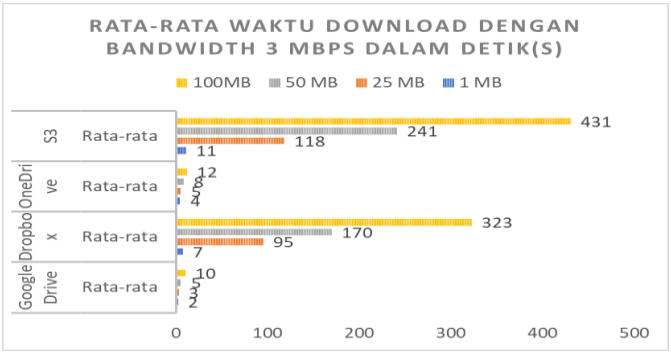
Gambar 4. Chart rata-rata kecepatan *upload* 5 Mbps

Dalam Gambar 4, terdapat grafik yang menampilkan rata-rata kecepatan *upload* dengan *bandwidth* 5 Mbps.



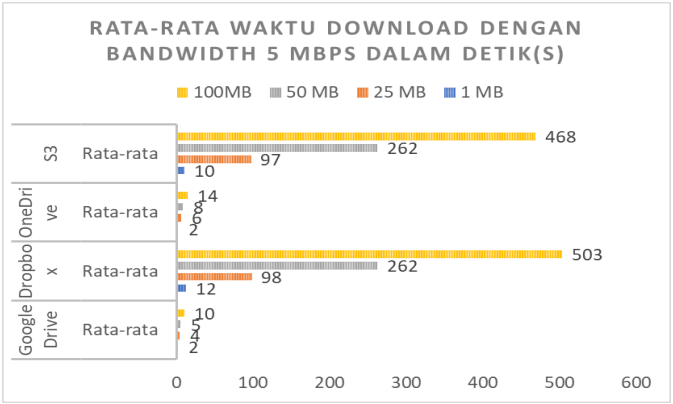
Gambar 5. Chart rata-rata kecepatan download 1 Mbps

Dalam Gambar 5, terdapat grafik yang menampilkan rata-rata kecepatan *download* dengan *bandwidth* 1 Mbps.



Gambar 6. Chart rata-rata kecepatan download 3 Mbps

Dalam Gambar 6, terdapat grafik yang menampilkan rata-rata kecepatan *download* dengan *bandwidth* 3 Mbps.



Gambar 7. Chart rata-rata kecepatan download 5 Mbps

Dalam Gambar 7, terdapat grafik yang menampilkan rata-rata kecepatan *download* dengan *bandwidth* 5 Mbps.

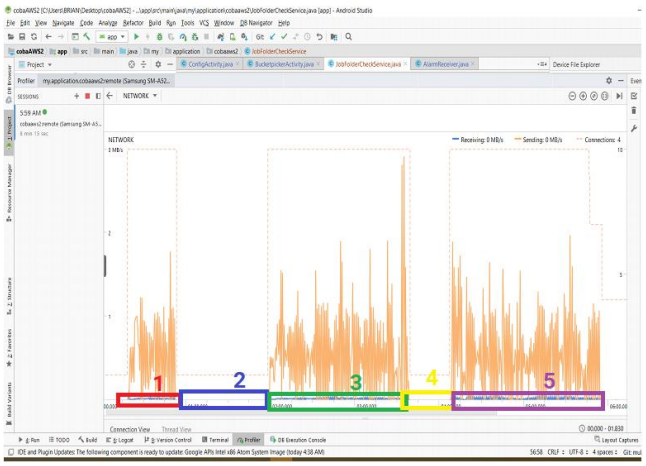
Berdasarkan pengujian yang telah dilakukan, dapat dilihat bahwa *Google Drive* dan *OneDrive* sangat unggul dalam kecepatan download, bahkan di kondisi *bandwidth* yang telah dibatasi. *S3* memiliki kecepatan di atas rata-rata dalam kondisi *bandwidth* yang rendah, dan menurun dibawah rata-rata dalam kondisi *bandwidth* yang standar atau tinggi dibandingkan dengan *cloud storage* lainnya.

4.2. Pengujian Fitur *Resume* Ketika Koneksi Putus

Uji coba berikut dilakukan menggunakan *file* teks 50 MB dengan koneksi paket data Telkomsel Simpati dengan *bandwidth* 42 Mbps.

Gambar 8. Tes *bandwidth* paket data (Sumber: http://www.speedtest.net)

Untuk menampilkan grafik yang menunjukkan terjadinya aktivitas jaringan digunakan Profiler pada Android Studio.



Gambar 9. Tampilan Android Profiler

Penjelasan pada grafik diatas:

1. Menandakan terjadinya aktivitas pada jaringan yang menunjukkan dimulainya proses *upload*. Proses *upload* berjalan seperti biasa, hingga koneksi paket data terputus.
2. Menandakan periode waktu dimana paket data sedang terputus.
3. Menandakan periode waktu dimana paket data mulai tersambung kembali, sehingga aplikasi melakukan proses *resume*.
4. Menandakan periode terputusnya paket data yang kedua kali.
5. Menandakan periode koneksi tersambung kembali hingga proses transfer telah selesai.

Berdasarkan pengujian diatas, fitur *resume* telah dapat berjalan dengan baik.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil pengujian dalam koneksi internet yang kurang stabil, S3 memiliki keunggulan dalam melakukan *upload* dan *download* karena kemampuannya dalam secara automatis meresume transfer ketika koneksi internet sedang terputus. Mengenai apakah S3 unggul dalam faktor kecepatan, sebenarnya S3 memiliki kecepatan yang setara dengan *cloud storage provider* lainnya yaitu *Google Drive*, *Dropbox*, dan *OneDrive*.

5.2. Saran

- Untuk penelitian yang memanfaatkan sistem sinkronisasi pada aplikasi *Android*, dapat menemukan solusi dari limitasi dari sistem operasi *Android* yang menyebabkan file tidak dapat didownload ke dalam kartu SD.
- Untuk penelitian berikutnya penulis menyarankan penggunaan API *JobScheduler* dan *JobService* untuk menghandle operasi yang perlu berjalan secara pararel.

6. DAFTAR PUSTAKA

[1] Amazon. 2006. “Amazon Simple Storage Service Developer Guide.” URI = https://docs.aws.amazon.com/AmazonS3/latest/dev/s3-dg.pdf

[2] Amazon. 2018. “What is cloud object storage”. URI = https://aws.amazon.com/what-is-cloud-object-storage/

[3] Amazon. 2018. Multipart Upload API Overview. URI =https://docs.aws.amazon.com/AmazonS3/latest/dev/mpuoverview.html

[4] Cui, Yong & Lai, Zeqi & Wang, Guanxin & Dai, Ningwei & Miao, Congcong. 2015. QuickSync: Improving Synchronization Efficiency for Mobile Cloud Storage Services. URI = http://dl.acm.org/10.1145/2789168.2790094.

[5] Fielding, R. Reschke, J. 2014. Message Syntax and Routing. URI = https://tools.ietf.org/html/rfc7230#section-4.1

[6] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, 1999. URI = https://www.ietf.org/rfc/rfc2616.txt

[7] Gildred, J. 2018. Dropbox vs Google Drive:the Battle of the Titans. URI = https://www.cloudwards.net/dropbox-vs-google-drive/

[8] Lalit Adithya et. Al 2016, December “Segmented File Transfer”.URI=https://ijcset.net/docs/Volumes/volume6issue12/ijcset2016061201.pdf

[9] Li Z. et al. 2013. “Efficient Batched Synchronization in Dropbox-Like Cloud Storage Services”. In: Eyers D., Schwan K. (eds) Middleware 2013. Lecture Notes in Computer Science, vol 8275. Springer, Berlin, Heidelberg. URI = https://link.springer.com/chapter/10.1007/978-3-642-45065-5_16

[10] Mościcki, J. T., & Mascetti, L. 2017. “Cloud storage services for file synchronization and sharing in science, education and research”. Future Generation Computer Systems, 78, 1052–1054. URI = doi:10.1016/j.future.2017.09.019

[11] Poojary, Nitheesh 2015. “Understanding Block storage and cloud storage use cases”. URI = https://cloudacademy.com/blog/object-storage-block-storage/

[12] Rivest, R., 1992."The MD5 Message Digest Algorithm", RFC 1321 MIT and RSA Data Security, Inc., URI = https://tools.ietf.org/html/rfc1321