

# **Seleksi Bagian A Laboratorium Sistem Terdistribusi**

**"Siklus Samsara"**



Dipersiapkan oleh  
Sister '21

**Dikerjakan oleh:**  
13522158 - Muhammad Rasheed Qais Tandjung

**Waktu Mulai :**

Jumat, 12 Juli 2024, 00.00 WIB

**Waktu Akhir :**

Minggu, 21 Juli 2024, 23.59 WIB

# I. Latar Belakang

"Terima kasih Oniichan! Da-da~!



Dari kecil, kamu diajarkan bahwa manusia hanya memiliki lima indera - penglihatan, pendengaran, pengecapan, penciuman, dan perabaan.

"Oh, seberapa salah mereka," pikirmu sembari jatuh di ruangan terlarang antara ruang dan waktu, meresapi ketakhinggaan sebelum akhirnya hilang kesadaran akibat betapa luar biasanya hal tersebut.

\*\*\*



Labtek V

Kamu terbangun di depan Labtek V, di kampus ITB yang indah, di saat mobil-mobil sudah mulai keluar dari tempat parkir mereka, dan mahasiswa-mahasiswa mulai berpulang - entah ke tempat-tempat tinggalnya, atau ke tempat-tempat rapatnya. Hijau dedaunan yang

mengayun dengan lembut dalam harmoni dengan hembusan angin menyoraki langit biru-merah jambu. Kicauan burung-burung cabak yang hidup tanpa peduli, bersimfonii dengan obrolan-obrolan para akademis yang (setidaknya, merasa) menggendong seluruh dunia pada pundaknya. Deruan petir dari tempat yang jauh di mata, ditemani oleh aroma petrikor dan kelembaban di lidahmu yang sama-sama membentarkan badi yang akan datang..

Kepalamu sakit. Seindah-indahnya dan semenyegarkannya dunia di sekitarmu, entah mengapa, kamu merasa kamu akan pingsan akibat penyergapan yang luar biasa ini pada indera-inderamu. Namun, entah mengapa juga, badanmu merasa bahwa ini jauh lebih baik dari sebuah pengalaman lampau yang kamu tidak dapat ingat. Alhasil, kamu bertahan.

Kemudian, kamu merasakannya. Sebuah sensasi yang hampir alien bagi dirimu. Sebuah getaran yang datang dari kantong kanan celanamu.

*Smartphone-mu berdering.*

Layaknya sebuah reflek primal, kamu mengeluarkan *smartphone-mu* dari kantong dalam hitungan milidetik. Di benak pikiranmu, lagi-lagi muncul - entah mengapa - rasa bahwa kamu kamu sudah lama tidak melakukan hal ini muncul kembali.

[!! H-1 Open House Lab HMIF !!]

"Open House Lab? Bukannya hal tersebut dilakukan di semester 4?" ucapmu, kebingungan.

"Sebentar, mengapa kalender menunjukkan Juni 2024?"

\*\*\*



Meme Mba Kapka Setelah Ditangkap POLRI

Kamu mencoba menggali kembali ingatanmu. Terakhir kali kamu sadar, kamu sedang berada di GAMEZONE, mendorong Mba Kapka ke lantai supaya ia tidak dapat menggunakan senjatanya.

Itu merupakan satu semester yang lalu.

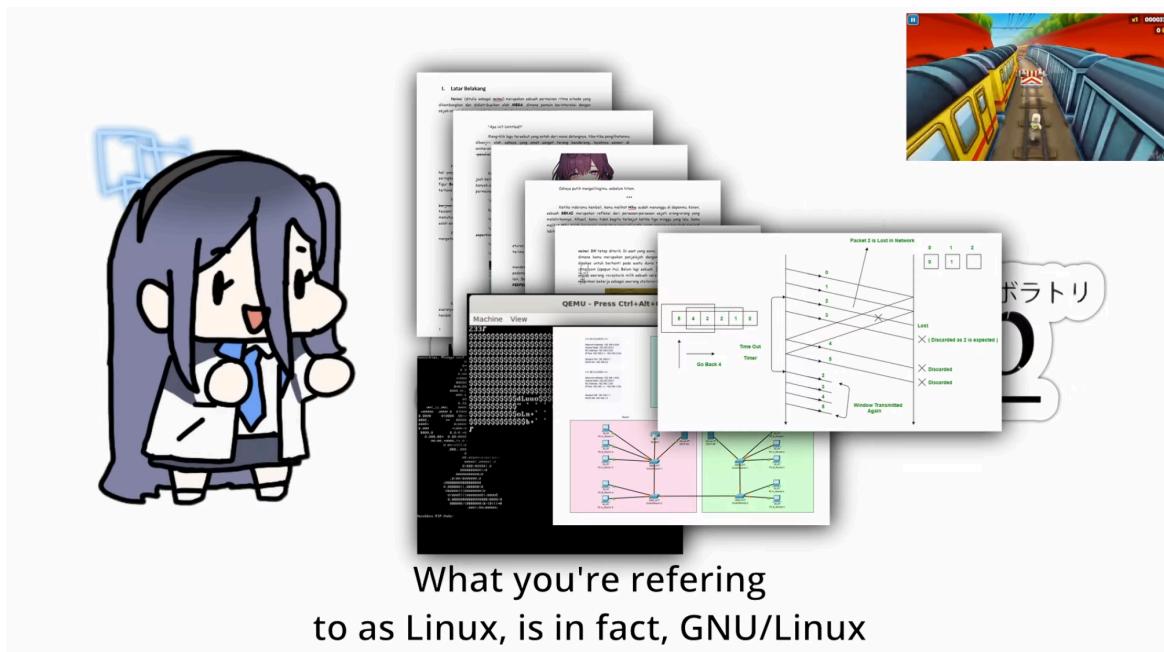
Melihat smartphone-mu, kamu menemukan bahwa antara tanggal keramat tersebut dan hari ini, kehidupanmu berlangsung seperti biasa. Tugas-tugas besar dan ujian tetap dilakukan, dan indeks tetap muncul di SIX. Tanggung jawab di berbagai organisasi kemahasiswaan tetap terlaksana dengan baik, dan komunikasi dengan teman-teman serta keluarga tetap berjalan. Bahkan, nainai pun masih dimainkan, dan catatan menunjukkan bahwa Matsune Hiku masing sering dikunjungi.

Hanya saja, kamu tidak ingat sedikitpun dari itu. Seolah-olah waktu terpotong, layaknya video pada perangkat lunak penyuntingan.

Video. Oh ya, video. Video-video OHL sudah rilis. Mungkin ini saat yang baik untuk menontonnya, berhubung memikirkan apapun fenomena ini yang terjadi padamu hanya akan memperburuk sakit kepalamu.

Waktunya pulang.

\*\*\*



Cuplikan Video Lab Sister

Sesampainya di rumah, kamu langsung membuka laptop-mu sebelum kemudian membuka tautan ke *playlist* video *teaser* OHL. Sejak kamu ditempatkan di program studi IF/STI, kamu sering menemukan dirimu menonton video-video *teaser* tahun-tahun sebelumnya. Lantas, diunggahnya video-video tahun ini merupakan suatu hal yang cukup kamu tunggu-tunggu. Dalam hati, kamu panjatkan syukur bahwa kamu "kembali ke dunia ini" sebelum OHL berlangsung.

Kamu mengecek video di *playlist* tersebut satu per satu. Secara diam-diam, kamu me-review setiap video. Video Lab \*\* normal seperti biasa, Lab \*\*\* dan \*\*\*\* sayangnya terlalu flat, dan Lab \*\*\*\*\* membuat sakit kepalamu semakin parah. Sementara itu, video Lab \*\*\*\*\* cukup berbeda dari tahun-tahun sebelumnya, Lab \*\*\*\*\* menggunakan formula yang sama tapi sangat menghibur, dan sayangnya video Lab \*\*\* tidak memiliki *gimmick* seperti tahun lalu.

Terakhir, kamu sampai di video Lab Sister.

\*\*\*

"Video macam apa ini...?" tanyamu setelah menontonnya. Memang, video Lab Sister dari tahun ke tahun selalu terkesan *absurd*, namun tidak pernah ada yang menyamai tingkat esoterisme video tahun ini. "Lab Sister, s-nya benar-benar stres kayaknya..."

Tiba-tiba, muncul notifikasi di *smartphone*-mu. Untitled. Mengekliknya, kamu panik karena inderamu tiba-tiba diserbu oleh cahaya putih, sebelum hitam. Ketika inderamu kembali, kamu disambut oleh Matsune Hiku - rupanya *timeskip* yang melanda dirimu menyebabkanmu untuk melupakan fungsi dari notifikasi Untitled yang dulu menjadikanmu pemain nainai nomor satu di keseluruhan wilayah Negara Kesatuan Republik Indonesia, dari Sabang sampai Merauke.



"Selamat siang! Aku di sini mau ngajak kamu masuk Lab Sister, seru kan?"

Entah mengapa, teman virtualmu ini rupanya telah menjadi promotor Lab Sister. Sebuah SEKAI konon merupakan cerminan dari perasaan-perasaan sejati pemiliknya; lantas, kamu tidak begitu kaget akan perilaku Hiku. Bagaimana juga, kamu memiliki banyak kenalan senior di Lab Sister, video-videoonya lah yang paling menyentuh hatimu, dan ilmu Lab Sister juga lah yang memungkinkanmu untuk menaikkan rating nainai-mu secepat itu. Meskipun begitu, kamu masih memiliki berbagai pertanyaan, dan melihat jabatan baru Hiku, kamu berharap ia dapat menjawabnya.

"Memang, menjadi asisten Lab Sister sebenarnya bagaimana?"

"Etto-"

Cahaya putih kembali menyelimutimu, sebelum telingamu mendengar musik yang entah mengapa mengingatkanmu atas baja persegi galvanis.

"Video macam apa ini...?" tanyamu...

\*\*\*

"H-h-h-huh? Aku... sudah menonton ini... berapa kali?" ucapmu. Belakangan ini, banyak sekali hal yang tidak kamu ketahui, dan alasan mengapa kamu memiliki perasaan bahwa kamu sudah menonton video OHL Lab Sister berkali-kali merupakan salah satunya.



"???"

"Kita baru saja menyelesaikan siklus ke-168," ujar sebuah gadis kecil. "Tahukah kamu, dalam upaya membuat video ini, editor harus mendengarkan remix Indihome sebanyak itu? Siklus ini akan terus berlanjut sampai kamu masuk Lab Sister."

"Tunggu-!"

"Tidak ada tunggu menunggu, aku sudah mengatakan semuanya yang bisa aku katakan. Aku tahu, seberapa kuat kamu ingin menjadi seorang asisten Lab Sister. Dahan-dahan Irminsul juga telah mengkonfirmasi potensimu. Jadi, apakah kamu ingin keluar dari siklus samsara ini?"

"...Iya-"

Sebelum kamu selesai berbicara, kamu akhirnya mengingat apa yang kamu alami selama timeskip satu semester itu. Kamu mengingat Operating Sister dan apa yang kamu lakukan untuknya. Dan tentunya, kamu mengingat sensasi jatuh dalam ruangan terlarang antara ruang dan waktu - bagaimana tidak, ketakhinggaan saat ini kembali menyerang indera-inderamu, sebelum akhirnya kamu hilang kesadaran akibat betapa luar biasanya hal tersebut.

\*\*\*

Ketika kamu terbangun, kamu menemukan dirimu berada di sebuah pulau yang indah, yang udaranya tidak tercemar oleh asap maupun suara knalpot. Arsitektur dan pemandangannya mengingatkanmu akan negara Jepang, namun terdapat beberapa perbedaan jelas; antara lain, banyaknya pohon-pohon yang menghasilkan listrik. Di tengah pengamatan ini, sebuah batu pun menyadarkanmu bahwa saat ini kamu sedang dibawa di atas sebuah gerobak, menuju sebuah rumah mewah di atas salah satu bukit yang menonjol pada pulau tersebut.

Melewati gerbang rumah tersebut, gerobak pun berhenti, dan para pengawal memintamu untuk turun. Tidak lama kemudian, mereka membungkuk dan memerintahkanmu untuk mengikuti.



### Bangsawan Pecinta Boba

Seorang lelaki dengan rambut biru yang mengenakan baju putih berakses emas mendatangimu. Penampilannya memberitahumu bahwa ia adalah seorang bangsawan, lantas mendorongmu untuk terus membungkuk. Meskipun begitu, kamu tidak berhasil menahan tawa ketika kamu melihatnya mengeluarkan teh boba dari dalam bajunya sebelum menghisapnya dengan santai.

"Selamat datang. Aku rasa kamulah yang dikirimkan oleh mereka di Akademiya untuk memajukan dunia kami secara umum, dan bangsa kami secara khusus. Tenang, aku tahu kamu bingung. Ikuti aku; aku akan jelaskan situasi kamu, dan apa yang harus kamu lakukan."

Lagi-lagi, kamu menemukan dirimu tidak mengetahui apapun; namun, melihat prajurit-prajurit di sekitarmu dan tombak-tombak lancip di genggaman mereka, kamu merasa kamu tidak punya pilihan. Akhirnya, kamu pun mengikutinya; entah mengapa, kamu merasa bahwa setidaknya kali ini pemandangannya indah, waktu berjalan, dan ada bangsawan yang dapat kamu mintai teh boba.

## II. Ketentuan

Dalam tugas kali ini, kalian akan diberikan sejumlah soal yang berkaitan dengan Lab Sister. Berbeda dengan tugas-tugas Lab Sister yang sudah kalian lalui sebelumnya, kalian diminta untuk menjawab soal-soal yang ada dengan kalimat yang kalian susun dalam bahasa Indonesia. Namun, kami menerima jawaban dalam bahasa mesin  jika diperlukan.

Jawablah pertanyaan-pertanyaan dibawah dengan ketentuan sebagai berikut:

1. Kerjakan sebanyak mungkin dan semaksimal mungkin. Setiap soal akan memberikan poin maksimal sesuai angka yang tertera, dengan kemungkinan mendapatkan tambahan poin bonus. Nilai akhir yang kalian dapatkan adalah jumlah seluruh poin yang berhasil kalian peroleh. **KALIAN TIDAK WAJIB MENGERJAKAN SELURUH SOAL, TETAPI PASTIKAN KALIAN MENGERJAKAN SOAL YANG DIWAJIBKAN.**
2. Pahami apa yang kalian tulis.
3. Diperbolehkan untuk mengerjakan & belajar bersama dengan teman.
4. Diperbolehkan untuk mencari jawaban di internet maupun dari *large language model*.
5. Dianjurkan untuk mencantumkan seluruh sumber yang digunakan dalam menjawab; bila memungkinkan, buatlah daftar pustaka yang rapi dan teratur (format IEEE).
6. Kerjakan dengan jujur; **segala bentuk plagiarisme akan ditindaklanjuti**.
7. Kerjakan dengan jelas; **berikan nomor soal dengan jelas untuk setiap jawaban**
8. Kerjakan dengan benar; **pastikan jawaban memenuhi permintaan soal**.
9. Kerjakan dengan *storytelling* yang baik; **pastikan jawaban mudah dibaca dan dipahami, serta tidak terlalu panjang maupun pendek**.
10. Kerjakan dengan mata terbuka 
11. Kerjakan dengan tangan  dan *keyboard*  kalian masing-masing
12. Kerjakan dengan bahagia :D

**Segala kecurangan yang melewati batas (i.e. *rename & submit* jawaban peserta lain, melakukan submisi atas nama peserta lain, dan seterusnya) akan menyebabkan diskualifikasi dari Seleksi Lab Sister 2024. Kerjakan dengan jujur & usaha diri sendiri.**

**~ Good Luck, Have Fun! ~**

Catatan Tambahan :

**Sangat disarankan** untuk membuat *script pengumpulan* terlebih dahulu. Jika belum familiar dengan mekanisme yang digunakan untuk pengumpulan, pembuatan *script* mungkin akan memakan waktu. Silahkan uji *script* karena server tidak terbatas ke 1x pengumpulan.

**Sangat tidak disarankan untuk deadliner.**

### III. Deliverables

Kalian akan mengumpulkan berkas jawaban kalian dalam format **PDF**. Berkas yang kalian kumpulkan harus diberikan nama dengan format:

<NIM>\_<5 karakter pertama sha2-256 sum berkas>.pdf

Untuk menghitung SHA 2-256 *sum*, dapat digunakan kakaes seperti shasum atau get-filehash. Validasi akan dilakukan terhadap *checksum* asli dengan nama *file*. Jika terdapat perbedaan, maka **nilai peserta akan dikurangi 5 poin**.

Pengumpulan berkas dilakukan dengan mengunggah berkas tersebut ke Google Drive atau Github, kemudian mengirimkan tautan berkas tersebut ke API yang disediakan. Jangan lupa untuk mengatur visibility menjadi publik. Teknis pengiriman tautan berkas adalah dengan mengikuti langkah-langkah berikut:

1. Buat sebuah *string JSON* dengan isi sebagai berikut:

```
{  
    "fullname" : "Nama Lengkap",  
    "link" : "link berkas",  
    "message" : "pesan singkat"  
}
```

Isikan bagian yang diwarnai **seperti ini** dengan isi yang sesuai. Sebagai contoh, **fullname** diisi nama lengkap kalian dan **link** diisi tautan ke berkas yang telah kalian unggah. Untuk bagian **message**, kalian dapat mengisinya dengan apapun yang kalian mau disini (mohon untuk tidak mengisi dengan *light novel*).

2. Bacalah sedikit mengenai RFC2617 (terutama bagian 2, mengenai HTTP *Basic Authentication*) serta RFC6238 (mengenai varian HOTP yang menggunakan waktu).
3. Buatlah *request POST* ke <http://sister21.tech:7787/recruitment/submit/a> untuk melakukan submisi. Karena *endpoint* tersebut dilindungi oleh *HTTP Basic Authentication*, maka kalian harus menyertakan *header Authorization* ketika kalian akan mengumpulkan. Kredensial yang kalian gunakan adalah sebagai berikut:

userid : NIM kalian  
password : sebuah OTP 8 digit hasil *Time-based OTP*

Kemudian, untuk membuat OTP, parameter yang digunakan adalah sebagai berikut:

TO	0
<i>x / Time step</i>	30
Algoritma Hash	SHA-256
<i>Shared secret</i>	"seleksister24" + NIM kalian + Waifu/Husbu/Idola (pertama) kalian sesuai dengan resume yang dikumpulkan, tanpa spasi

Sebagai contoh untuk *shared secret*, apabila NIM yang kalian miliki adalah 18221102 dan waifu kalian di resume adalah "Raiden Ei" serta "Yae Miko", maka *shared secret* yang kalian gunakan adalah seleksister2418221102RaidenEi. **Gunakan encoding UTF-8 (bukan ASCII) ketika menghitung TOTP.**

**Note:** Jika kalian mengosongkan pertanyaan waifu/husbu/idola, atau pertanyaan tersebut dijawab dengan jawaban yang bukan merupakan nama (misal, "pacar IRL saya" atau "rahasia"), ganti *field* tersebut dengan nama lengkapmu sendiri.

Contoh *request* yang benar

```
POST /recruitment/submit/a HTTP/1.1
Content-Type: application/json
Authorization: Basic MTgyMjExMDI6ODIxODgyMDY=
Content-Length: 112

{ "fullname": "Duke Frost I Achsien-Sachdie of Sawangan",
  "link": "itb.ac.id", "message": "Soalnya kurang banyak :D"}
```

Akan menghasilkan *response*

```
HTTP/1.1 201 Created
Server: sister20
Date: Sat, 30 June 2023 14:35:56 GMT
Content-Length: 155
Content-Type: text/plain; charset=UTF-8

Congratulations Duke Frost I Achsien-Sachdie of Sawangan on
submitting part A! This is your 1st submission.
« Nec fatum finire te, nec tribulatio potest. »
```

4. Apabila kalian mengikuti langkah-langkah tersebut dengan benar, maka kalian akan mendapatkan HTTP status 201. Pastikan *script* mendukung dan dapat menampilkan karakter-karakter UTF-8 (misal: hiragana dan katakana). Anda dapat melakukan submisi sebanyak mungkin, namun **hanya submisi terakhir yang akan dinilai**. Silahkan gunakan ini untuk menguji *script* pengumpulan ~~dan melihat kemungkinan pesan-pesan random~~, namun mohon jangan lakukan submisi terlalu banyak. **Nilai peserta akan dikurangi 5 poin jika pengumpulan yang berlebih olehnya ditemukan menyebabkan permasalahan pada server maupun kesehatan mental asisten.**

**Untuk catatan tambahan, dilarang untuk melakukan serangan dalam bentuk apapun pada server, baik aktif maupun pasif.**

## IV. Soal

### I. Organisasi dan Arsitektur Komputer



Hello, Warudo!

1. (2.5 poin) Desainlah sebuah algoritma dengan hanya menggunakan operator bitwise (operator = diperbolehkan), dalam *pseudocode* atau bahasa pemrograman apapun untuk:

- a. Menambahkan dua buah 32-bit *integer*.

```
int add(int a, int b) {  
    while (b != 0) {  
        // Calculate carry  
        int carry = a & b;  
  
        // Calculate sum without carry  
        a = a ^ b;  
  
        // Update b to be the carry shifted left  
        b = carry << 1;  
    }  
    return a;  
}
```

- b. Menambahkan dua buah *floating point number* IEEE 754.

2. (2.5 poin) Dibandingkan dengan GPU yang mungkin memiliki ribuan core, CPU seri M (M1, M2, dan seterusnya) memiliki jumlah core GPU yang sedikit. Meskipun

**begitu, mereka terkenal dapat menjalankan *workload* yang umumnya hanya bisa diangkat dengan GPU diskrit dengan cukup kencang. Jelaskan alasan mengapa ini terjadi, selain karena mereka berbasis RISC dan memiliki *unified memory*!**

GPU dalam chip seri M dirancang secara khusus untuk memanfaatkan arsitektur terintegrasi dan integrasi mendalam dengan CPU. Meskipun jumlah core GPU lebih sedikit dibandingkan GPU diskrit, setiap core dirancang untuk efisiensi dan kinerja tinggi. Apple mengoptimalkan arsitektur GPU-nya dengan fokus pada pemrosesan paralel yang efisien, serta fitur-fitur khusus untuk mempercepat berbagai jenis workload, termasuk grafik dan komputasi umum.

Selain itu, Apple mengembangkan CPU dan GPU-nya secara bersamaan dengan optimasi khusus untuk macOS dan iOS. Ini memungkinkan pemrosesan yang sangat efisien karena perangkat keras dan perangkat lunak dirancang untuk saling melengkapi. Selain itu, Apple menggunakan teknologi seperti Metal, API grafik dan komputasi yang memungkinkan akses langsung dan efisien ke GPU, meningkatkan performa bahkan dengan jumlah core yang lebih sedikit.

Meski jumlah core GPU di chip M-series lebih sedikit, masing-masing core GPU dapat melakukan tugas komputasi paralel dengan sangat efektif. Arsitektur GPU yang dioptimalkan dan teknik pengolahan data yang canggih memungkinkan penanganan banyak thread secara bersamaan, meningkatkan performa keseluruhan dalam aplikasi yang memerlukan komputasi intensif.

### 3. (2.5 poin) Jelaskan apa itu *instruction pipeline* dan *instruction cycle*!

Instruction pipeline dan instruction cycle adalah dua teknik untuk eksekusi instruksi pada CPU. Cara kerja CPU melaksanakan instruksi adalah dengan melakukan pengulangan terus-menerus, dalam setiap iterasi melakukan hal-hal tersebut secara sekuensial: Fetch → Decode → Execute → Memory Access → Write Back.

Pada CPU yang menggunakan metode instruction cycle, setiap tahap pada sekuens tersebut dilakukan satu per satu, tanpa ada dua instruksi yang dilaksanakan bersamaan. Pada CPU yang menggunakan metode instruction pipeline, setiap tahap dijalankan oleh bagian yang berbeda pada CPU, sehingga memungkinkan untuk melakukan lebih dari satu tahap pada waktu yang bersamaan. Contohnya ketika sebuah instruksi sedang di-decode, instruksi lain bisa di-fetch, dan instruksi lainnya bisa dieksekusi, semua dalam waktu yang bersamaan. Hal ini menyebabkan jauh lebih banyak instruksi dapat diselesaikan dalam periode waktu tertentu.

### Referensi

- [1] Javatpoint, "Instruction Pipeline," [Online]. Available: <https://www.javatpoint.com/instruction-pipeline>. [Accessed: 21-Jul-2024].
- [2] Wikipedia, "Instruction pipelining," [Online]. Available: [https://en.wikipedia.org/wiki/Instruction\\_pipelining](https://en.wikipedia.org/wiki/Instruction_pipelining). [Accessed: 21-Jul-2024].

**4. (2.5 poin) Jelaskan berdasarkan apa yang kalian pahami tentang:**

**a. Apa itu *cache miss*?**

Ketika sebuah klien (*web browser*, sistem operasi, atau lainnya) ingin mengakses sebuah data, umumnya akan dicek penyimpanan pada *cache* terlebih dahulu. Jika data tersebut ada di *cache*, maka pencarian selesai. Namun jika data tersebut tidak ditemukan pada *cache*, klien tersebut akan mengecek pada penyimpanan memori yang lain. Kejadian gagalnya ditemukan data pada *cache* disebut *cache miss*.

Ketika terjadi sebuah *cache miss*, waktu pencarian data bertambah karena klien harus melanjutkan pencarian di lokasi penyimpanan lain. Setelah data yang inginkan berhasil ditemukan, data tersebut dimasukkan ke *cache* sehingga jika dibutuhkan lagi di masa depan, data tersebut sudah tersedia di *cache* dan waktu pencarian data berjalan lebih cepat.

**b. Beberapa solusi untuk mengurangi *cache miss*.**

- Memanfaatkan *locality of reference* – Sering menggunakan ulang sebuah variabel dan mengakses data secara sekuensial akan mengurangi *cache miss*, karena data yang sama atau berdekatan akan lebih sering berada di *cache*.
- Melakukan *prefetching* – Mencoba memprediksi data yang akan digunakan dan langsung menyimpannya ke *cache* dapat mengurangi *cache miss*.
- Memanfaatkan struktur data *cache-friendly* – Beberapa struktur data dapat mengurangi terjadinya *cache miss*, contohnya penggunaan array sekuensial dibanding *linked list*, karena sebuah array sekuensial akan memanfaatkan prinsip *spatial locality*.

**c. Apa itu prinsip *locality*?**

Prinsip *locality* adalah sebuah konsep pada *computing*, yaitu prosesor pada umumnya akan sering mengakses alamat-alamat memori yang sama dan berdekatan.

Dua jenis *locality* yang umumnya dibicarakan adalah *temporal locality*, yaitu sebuah alamat memori yang pernah diakses kemungkinan besar akan diakses kembali di masa depan, dan *spatial locality*, yaitu ketika sebuah alamat memori diakses, kemungkinan besar alamat-alamat lain yang berdekatan dengan alamat tersebut akan diakses juga di masa depan.

Prinsip *locality* sering dibahas dengan konsep *caching*, karena penulisan program dengan *locality* yang tinggi dapat mengurangi *cache miss*, sehingga meningkatkan performa program. Contohnya ketika sebuah program memiliki

*temporal locality* yang tinggi, maka pengaksesan memori tersebut setelah pengaksesan pertama akan selalu menghasilkan *cache hit*.

- d. Contoh penerapan prinsip *locality* pada proyek/tugas/tubes yang pernah kalian lakukan.

```
// get parent directory cluster
struct FAT32DirectoryTable dir_table;
memset(&dir_table, 0, CLUSTER_SIZE);
read_clusters(&dir_table, request.parent_cluster_number, 1);

// iterate through all entries in the parent folder, check if an entry with the same name exists
for (uint8_t i = 0; i < 64; i++) {
    struct FAT32DirectoryEntry dir_entry = dir_table.table[i];
```

Gambar A4.1. *Spatial* dan *temporal locality* pada pembacaan *filesystem*

Gambar A4.1 menunjukkan cuplikan kode pembacaan sebuah kluster data pada *filesystem* di tugas besar Sistem Operasi 32-bit. Variabel *dir\_table* yang digunakan pada setiap iterasi *for loop* merupakan contoh *temporal locality* karena variabel yang sama digunakan ulang pada setiap iterasi, dan pengaksesan sekuensial array *dir\_table.table* menggunakan indeks *i* merupakan contoh *spatial locality*.

## Referensi

- [1] Wikipedia contributors, "Cache (computing)," in Wikipedia: The Free Encyclopedia, Wikimedia Foundation, June 2024. Available: [https://en.wikipedia.org/wiki/Cache\\_\(computing\)](https://en.wikipedia.org/wiki/Cache_(computing)). [Accessed: July 12, 2024].
- [2] Wikipedia contributors, "Locality of reference," in Wikipedia: The Free Encyclopedia, Wikimedia Foundation, July 2024. Available: [https://en.wikipedia.org/wiki/Locality\\_of\\_reference](https://en.wikipedia.org/wiki/Locality_of_reference). [Accessed: July 12, 2024].

## 5. (2.5 poin) Jelaskan fungsi tiap jenis *register* di *processor x86\_64!*

- General purpose registers (RAX, RBX, RCX, RDX, RSI, RDI, R8-R15): Tipe register ini merupakan register serbaguna yang berfungsi sebagai penyimpanan data umum, seperti manipulasi data dan operasi aritmatika.
- Segment registers (CS, DS, SS, ES, FS, GS): Digunakan untuk menyimpan *segment selector* untuk mengakses segmen-segmen memori dalam mode *real mode* dan *protected mode*.
- Stack registers (RBP, RSP): Tipe register ini pada prosesor 32-bit digunakan sebagai pointer ke lokasi stack pada memori, namun pada prosesor 64-bit seperti x86-64 digunakan sebagai register general purpose.
- Instruction pointer (RIP): Menyimpan lokasi memori untuk instruksi selanjutnya yang akan dieksekusi.

- Flag registers (RFLAGS): Menyimpan *status flags* untuk operasi aritmatika dan logika, seperti zero flag (ZF), carry flag (CF), overflow flag (OF), dst.
  - Control registers (CR0, CR2, CR3, CR4, CR8): Digunakan untuk keperluan sistem operasi, seperti untuk mekanisme *paging*, *context switching*, dan *debugging*.
6. (2.5 poin) Buatlah sebuah kode program dalam bahasa C untuk melakukan hal dibawah ini tanpa menggunakan *for*, *while*, atau pun *do while*!

- a. Jumlah keseluruhan elemen dari sebuah array.

```
#include <stdio.h>

int sum(int arr[], int size) {
    int sum = 0;

    int i = 0;
loop:
    sum += arr[i++];
    if (i < size) goto loop;

    return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5, 6};
    int size = sizeof(arr) / sizeof(arr[0]);

    if (size == 0) return 0;

    printf("Sum: %d\n", sum(arr, size));

    return 0;
}
```

- b. Nilai elemen maksimum dan minimum dari sebuah array.

```
#include <stdio.h>

int min(int arr[], int size) {
    int min = arr[0];
```

```
int i = 0;
loop:
    if (arr[i] < min) min = arr[i];
    i++;
    if (i < size) goto loop;

    return min;
}

int max(int arr[], int size) {
    int max = arr[0];

    int i = 0;
loop:
    if (arr[i] > max) max = arr[i];
    i++;
    if (i < size) goto loop;

    return max;
}

int main() {
    int arr[] = {};
    int size = sizeof(arr) / sizeof(arr[0]);

    if (size == 0) return 0;

    printf("Min: %d\n", min(arr, size));
    printf("Max: %d\n", max(arr, size));

    return 0;
}
```

7. (2.5 poin) Jelaskan keuntungan dan kekurangan arsitektur SOC dibandingkan dengan arsitektur modular.

Arsitektur SOC (System On-A Chip) merupakan tipe arsitektur yang menggabungkan seluruh hardware yang dibutuhkan perangkat dalam satu buah chip.

Hardware ini meliputi CPU, RAM, GPU, dan lain-lain. Kebalikan dari arsitektur SoC adalah arsitektur modular, dimana hardware perangkat tersegregasi menjadi komponen-komponen modular yang memiliki tugas terpisahnya masing-masing. Kelebihan dari arsitektur SoC adalah:

- Ukuran kecil dan compact: Arsitektur SoC mengintegrasikan seluruh komponen pada satu buah chip, sehingga ukuran perangkat dapat dibuat jauh lebih kecil. Hal ini sangat berguna dalam pembuatan perangkat portable seperti mobile phone dan embedded device.
- Latency lebih kecil dan power efficiency: Dengan jarak yang lebih kecil antar komponen, penggunaan energi akan menurun dan waktu untuk perpindahan data akan jauh lebih kecil.
- Cost-effective: Ukuran komponen yang lebih kecil akan berakibat pada harga manufaktur yang lebih kecil.

Sedangkan kekurangan dari arsitektur SoC meliputi:

- Customizability dan upgradability rendah: Dengan seluruh komponen tergabung pada satu buah objek, kontrol pemilik terhadap hardware akan jauh lebih sedikit, karena jika ingin mengubah salah satu komponen harus mengubah seluruh komponen.
- Maintainability: Jika terdapat sebuah masalah pada hardware, troubleshooting terhadap penyebabnya akan lebih sulit pada arsitektur SoC, karena seluruh komponen tergabung dan sistem tidak bersifat modular.
- Modularitas: Dengan setiap komponen tersegregasi dan memiliki fungsinya masing-masing, development dan testing terhadap satu buah komponen akan jauh lebih mudah.

**8. (2.5 poin) Kita mengenal beberapa cara untuk menyimpan data ada *little endian* dan *big endian*. Mengapa beberapa device lebih memilih untuk menyimpan data secara *little endian*?**

- Operasi penambahan: Pada perhitungan aritmatika, metode penambahan dua angka yang kita umumnya gunakan bersifat *little-endian*. Penambahan dimulai dari digit paling kanan (least significant digit), lalu bergeser ke kiri (ke arah most significant digit) dengan membawa *carry* yang dihasilkan. Dengan metode penyimpanan *little-endian*, karena pengurutan byte berawal dari *least significant byte*, maka operasi penambahan cukup bergerak dari memori kecil ke memori besar, tidak perlu dilakukan perhitungan mundur.
- Pointer typecasting: Pada sistem *little-endian*, pointer ke integer 8-bit, 16-bit, maupun 32-bit mengarah ke lokasi memori yang sama, karena semuanya menunjuk ke LSB. Sehingga pada sistem *little-endian*, perubahan tipe pointer tidak membutuhkan perubahan data yang banyak.
- *Truncation* dan *extension*: Pemotongan panjang dan penambahan panjang sebuah angka menjadi cukup sederhana pada sistem *little-endian*. Trunkasi

- angka dari 16-bit ke 8-bit cukup mengabaikan 8-bit terakhir. Ekstensi sebuah angka dari 8-bit ke 16-bit cukup menambahkan byte baru setelah byte tersebut.
- *Legacy*: Prosesor Intel dan arsitektur x86 menggunakan *little-endian*, dan karena popularitas kedua sistem ini, maka banyak device dan hardware lain yang mengikuti sistem *little-endian* agar kompatibel dengan device-device Intel dan x86.

## Referensi

[1] "What is the advantage of little-endian format?", in Software Engineering Stack Exchange, Jul 2011. Available: <https://softwareengineering.stackexchange.com/questions/95556/what-is-the-advantage-of-little-endian-format>. [Accessed: Jul. 13, 2024].

[2] "Why are both little-endian and big-endian in use?," in Stack Overflow, Jan. 2011. Available: <https://stackoverflow.com/questions/4752715/why-are-both-little-and-big-endian-in-use>. [Accessed: Jul. 13, 2024].

**9. (2.5 poin) Jelaskan perbedaan antara arsitektur ARM yang digunakan pada perangkat smartphone (misal, ARM Snapdragon) dengan yang digunakan oleh Apple!**

- Processor Qualcomm Snapdragon umumnya didesain menggunakan tipe core standar, yang merupakan desain umum dan ditujukan untuk dapat dipakai di banyak device. Di sisi lain, Apple umumnya membuat custom core sendiri, seperti Firestorm dan Icestorm. Desain custom ini telah dioptimasi agar cocok dengan ekosistem hardware dan software Apple.
- Processor Snapdragon didesain untuk mencari titik tengah antar performa dengan power efficiency, agar dapat dipakai di banyak jangkauan mobile device dengan beragam harga. Apple mengambil sudut pandang memaksimalkan performa dan efisiensi dengan custom core designs pada processornya.
- Chip Snapdragon umumnya menggunakan big.LITTLE architecture yang menggabungkan high-performance dengan energy-efficient cores. Core custom buatan Apple biasanya memberikan fitur dan optimasi yang umumnya tidak ditemukan di core ARM standar, seperti cache lebih besar, clock speed lebih kencang, dan out-of-order execution.
- Integrasi device pada chip Snapdragon umumnya dapat kompatibel dengan banyak tipe device, komponen, dan pemakaian. Apple memilih untuk melakukan optimasi yang terfokus untuk bekerja dan bersinergi dengan bagus dalam ekosistem iOS dan MacOS.

**10. (2.5 poin) Ketika kita melakukan kompilasi program lalu membedahnya menggunakan decompiler seperti GDB, langkah-langkah program terkadang tidak sama persis dengan langkah-langkah yang kita tulis. Jelaskan mengapa hal ini mungkin terjadi.**

Hal ini terjadi karena adanya konsep *compiler optimization*. Seringkali kode yang ditulis seorang programmer jauh dari optimal jika dilihat di segi low-level. Sebuah compiler akan melakukan optimasi kode tersebut sehingga program berjalan lebih cepat namun melakukan hal-hal dan menghasilkan output yang sama seperti program awal, beberapa caranya yaitu dengan menghapus kode yang tidak dipakai, mengubah urutan kode, melakukan optimasi *looping* serta mengurangi pemanggilan fungsi dan prosedur dengan menggantikan kode pemanggilan dengan kode fungsi tersebut di program utama.

**11. (2.5 poin) Jelaskan bagaimana cara kerja pemrograman menggunakan *punch card*!**

Sebuah *punch card* dapat dianggap sebagai *file source code* pada zaman dahulu ketika pemrograman masih belum bersifat elektronik. Sama seperti sebuah file, *punch card* hanya berfungsi sebagai metode penyimpanan sebuah program yang nantinya dapat dibaca oleh sebuah komputer. Mekanisme penulisan program dalam *punch card* kurang lebih seperti berikut:

- Tuliskan program yang ingin dibuat dalam kertas terlebih dahulu.
- Gunakan sebuah *template* untuk mengkonversi program kertas tersebut menjadi koordinat lubang-lubang yang ingin dibolongi pada *punch card*. Satu buah kartu umumnya memiliki 80 kolom dan 12 baris, dan satu buah kartu merepresentasikan satu baris kode.
- Masukkan sebuah *punch card* kosong ke sebuah *keypunch machine* (alat untuk menulis program pada *punch card*). Isi bolongan-bolongan sesuai dengan yang sudah didapat pada *template* sebelumnya.
- Ketika satu baris sudah ditulis, ambil *punch card* tersebut dari mesin dan masukkan *punch card* selanjutnya untuk diprogram.
- Ketika satu program sudah tertulis pada *punch card* secara lengkap, tumpukkan seluruh *punch card* yang sudah dibuat sesuai urutan barisnya, dan masukkan ke *card reader* yang tersambung pada sebuah komputer. Program pada kumpulan *punch card* tersebut akan dikonversi menjadi instruksi atau data yang dapat diinterpretasikan oleh komputer.

**12. (5.0 poin) Jelaskan cara kerja komputer dengan arsitektur von Neumann dari mulai dibuatnya sebuah program hingga program tersebut dijalankan dengan cara memberikan sebuah contoh penulisan program C hingga program tersebut dijalankan di CPU. Sebutkan juga satu arsitektur lain selain von Neumann dan jelaskan perbedaannya.**

Arsitektur von Neumann merupakan tipe komputer yang berbasis pada konsep stored-program, yaitu instruksi dan data disimpan pada memori yang sama. Arsitektur komputer ini merupakan tipe yang paling umum ditemukan pada komputer saat ini. Sebuah komputer bertipe von Neumann sederhana memiliki satu buah prosesor, memori

untuk menyimpan instruksi dan data, serta mengeksekusi program menggunakan metode fetch-decode-execute cycle.

Komputer berbasis arsitektur von Neumann bekerja dengan melaksanakan fetch-decode-execute cycle, yang tahap-tahapnya adalah sebagai berikut:

- Fetch: Prosesor mengambil instruksi pada memori
- Decode: Control Unit (CU) pada prosesor menerjemahkan instruksi yang didapat menjadi instruksi yang dapat dimengerti oleh ALU.
- Execute: ALU pada prosesor mengeksekusi instruksi yang didapat.
- Memory Access: Jika instruksi memerlukan akses memori, langkah ini akan dilakukan setelah eksekusi.
- Write Back: Hasil dari eksekusi ditulis kembali ke register atau memori jika diperlukan.

Misal terdapat sebuah file C seperti berikut:

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int sum = a + b;
    printf("Sum: %d\n", sum);
    return 0;
}
```

Ketika file tersebut disimpan, file itu berada sebagai bentuk data dalam secondary storage. File ini masih terlalu high-level dan belum bisa dibaca oleh komputer. Jika file tersebut ingin dijalankan, harus dilakukan kompilasi tersebut, contohnya dengan compiler GCC. Misalkan pengguna lalu memasukkan command tersebut ke command prompt:

```
gcc -o main main.c
```

Maka file C diatas, mengasumsikan file tersebut dinamakan main.c, akan dikompilasi oleh compiler GCC menjadi sebuah file executable. File ini sudah dalam bentuk biner dan dapat dibaca oleh CPU, namun masih berperan sebagai data yang disimpan di storage. Jika ingin menjalankan file tersebut, maka pengguna akan memasukkan perintah

```
./main
```

Setelah perintah tersebut dijalankan, maka data tersebut akan dimasukkan ke memori, dan sudah berubah dari data menjadi instruksi. Maka CPU akan melaksanakan fetch-decode-execute cycle untuk setiap instruksi yang ada pada file tersebut.

## Arsitektur Harvard

Selain arsitektur von Neumann, tipe arsitektur komputer lainnya yang umumnya ditemukan adalah arsitektur Harvard. Pada tipe arsitektur ini, data dan instruksi tidak disimpan pada tempat yang sama pada memori, melainkan terdapat komponen memori untuk instruksi dan komponen memori yang terpisah untuk data. Pemisahan ini menyebabkan arsitektur tidak perlu menggunakan bus yang sama untuk transfer data dan instruksi, melainkan transfer kedua hal tersebut bersifat terpisah. Hal ini memungkinkan eksekusi lebih cepat, karena tidak ada konflik atas pengambilan instruksi dan data, kedua hal tersebut dapat dilakukan secara bersamaan.

## Referensi

- [1] Javatpoint, "Von Neumann Model," [Online]. Available: <https://www.javatpoint.com/von-neumann-model>. [Accessed: 21-Jul-2024].
- [2] Wikipedia, "Von Neumann architecture," [Online]. Available: [https://en.wikipedia.org/wiki/Von\\_Neumann\\_architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture). [Accessed: 21-Jul-2024].
- [3] Computer Science, "Harvard Architecture versus Von Neumann Architecture," YouTube, 30-Oct-2021. [Online]. Available: <https://www.youtube.com/watch?v=4nY7mNHLrLk>. [Accessed: 21-Jul-2024].

## II. Sistem Operasi



Arch-Chan 😍

1. (2.5 poin) Jelaskan alur proses *booting* sistem operasi yang menggunakan:

a. *Boot sequence BIOS*.

1. Power-On: Ketika komputer pertama kali dinyalakan, listrik akan mengalir ke seluruh komponen hardware. CPU akan mulai bekerja dan melaksanakan instruksi awal pada ROM BIOS.
2. Power-On Self Test (POST): BIOS akan melakukan serangkaian tes untuk menginisialisasi hardware dan memeriksa apakah seluruh komponen sudah berjalan. Tahap ini akan melakukan pengecekan terhadap memori, CPU, keyboard, hard drive, dan lain-lain.
3. Mencari Bootable Device: BIOS akan mencari perangkat boot yang diperlukan untuk memulai sistem operasi. Komponen yang akan dicek adalah media penyimpanan seperti HDD, SSD, CD/DVD drive, USB flash, ataupun network. BIOS akan terus mencari sampai ditemukan bootable device.
4. Memuat Bootloader: Setelah menemukan bootable device, BIOS akan membaca MBR pada device tersebut untuk memuat bootloader, program yang akan digunakan untuk menjalankan sistem operasi.
5. Menjalankan OS: Kontrol akan diserahkan ke bootloader, yang lalu akan menginisialisasi sistem operasi ke memori lalu memuat dan menjalankan kernel. Kernel akan menginisialisasi subsistem penting lainnya seperti memory manager, process, hardware, dan komponen-komponen lainnya. Pada tahap ini, sistem operasi sudah berjalan secara penuh dan komputer sudah siap digunakan pengguna.

b. *Boot sequence UEFI*.

1. Power-On: Komputer dinyalakan, proses mulai bekerja dan mencari instruksi awal pada firmware UEFI.

2. Power-On Self Test (POST): UEFI akan melakukan pengecekan dan inisialisasi hardware dasar seperti pada BIOS.
  3. UEFI Firmware Initialization: Setelah POST, UEFI akan menginisialisasi sistem dan mengkonfigurasi hardware seperti driver dan configuration settings.
  4. UEFI Boot Manager: UEFI akan mencari tabel EFI System Partition (ESP) untuk bootloader dan boot manager akan mencari bootable device pada HDD, SSD, dan lain-lain.
  5. Menjalankan Bootloader: Setelah bootable device dan ESP sudah ditemukan, UEFI akan menjalankan bootloader.
  6. Menjalankan OS: Bootloader akan melakukan inisialisasi sistem operasi dan menjalankan kernel, yang setelah itu kernel akan melakukan inisialisasi sistem operasi. Setelah seluruh inisialisasi selesai, komputer sudah siap digunakan pengguna.
2. **(2.5 poin) Jelaskan bagaimana *containerization* dan *virtualization* bekerja lalu buatlah perbandingan antara keduanya!**

Containerization adalah metode untuk membungkus sebuah program dengan seluruh dependency yang dibutuhkannya dalam sebuah *container image* yang dapat dipindahkan dan dijalankan di perangkat manapun tanpa perlu mengubah-ubah dan menginstall seluruh dependency yang dibutuhkan.

Containerization bekerja dengan menjalankan sebuah container image pada sebuah container runtime, yang nantinya container image tersebut akan menghasilkan sebuah ekosistem aplikasi sendiri yang sudah lengkap dengan seluruh dependency yang dibutuhkan. Walaupun terbentuk ekosistem yang terpisah dari host device, sebuah container tidak memiliki sistem operasi, sehingga untuk OS dan hardware container tersebut masih menggunakan milik host. Ini menyebabkan semua container yang dijalankan pada satu host menggunakan sistem operasi dan hardware yang sama.

Virtualization konsepnya hampir sama seperti containerization, namun setiap container memiliki sistem operasinya sendiri, sehingga unit yang terisolasi pada virtualization merupakan sebuah mesin tersendiri yang terpisah dari host device secara total. Mesin-mesin tersebut menggunakan hardware dengan bantuan *hypervisor*, sebuah software yang mengalokasikan hardware host device ke mesin-mesin tersebut.

Perbedaan dari containerization dan virtualization adalah sebagai berikut:

- Virtualization melakukan isolasi secara total, sedangkan containerization hanya terisolasi pada level proses.
- Karena virtualization menghasilkan sebuah mesin yang terisolasi total, maka kerusakan pada sebuah mesin tidak akan berpengaruh ke host device sama sekali. Containerization masih terhubung dengan host device pada segi sistem operasi, sehingga terjadinya masalah pada sebuah container dapat mempengaruhi host device.

- Virtualization akan lebih berat dan memakan lebih banyak resource karena melakukan simulasi mesin total. Containerization tidak melakukan simulasi sistem operasi, sehingga umumnya jauh lebih ringan dan cepat.

#### Referensi

[1] "Virtualization vs Containerization," Liquid Web, <https://www.liquidweb.com/blog/virtualization-vs-containerization/>. Accessed: Jul. 13, 2024.

[2] "Containerization vs. Virtualization: Key Differences and Use Cases," Aqua Security, <https://www.aquasec.com/cloud-native-academy/docker-container/containerization-vs-virtualization/#:~:text=Containerization%20vs.,Virtualization%3A%20Key%20Differences%20and%20Use%20Cases,%2C%20storage%2C%20networks%2C%20systems>. Accessed: Jul. 13, 2024.

3. (2.5 poin) Jelaskan masing-masing definisi dari metode *write-through*, *write-back*, *read ahead*, *no read ahead*, dan *adaptive read ahead*. Jelaskan pula kapan metode tersebut lebih baik digunakan (pengaruhnya terhadap *performance*, *memory usage*, etc.) dari metode lainnya (bandingkan dengan metode yang relevan).

- Write-through: Metode caching ketika setiap operasi write ke cache akan selalu dipropagasi ke backing store (memory atau disk). Hal ini akan memastikan bahwa data pada cache dan backing store selalu tersinkronisasi dan konsisten. Metode ini akan meningkatkan integritas dan konsistensi data dan kesederhanaan implementasi dengan mengorbankan waktu eksekusi, karena setiap write harus menunggu sampai sudah ditulis ke backing store. Metode ini unggul untuk sistem yang mengurus data critical yang mengharuskan integritas dan konsistensi data seperti transaksi finansial atau database. Metode ini juga baik digunakan pada sistem yang memiliki toleransi rendah untuk data loss.
- Write-back: Metode caching yang operasi write ke cache tidak dipropagasi ke backing store. Informasi tersebut hanya akan dituliskan ke backing store ketika data tersebut ingin dibuang dari cache. Hal ini akan sangat mengurangi waktu eksekusi, karena penulisan ke backing store hanya ketika cache ingin dibuang. Kelemahannya adalah akan banyak muncul inkonsistensi antar data di cache dengan data di backing store, yang dapat memunculkan masalah ketika terjadi system crash dan data pada backing store belum terupdate. Metode ini juga lebih sulit untuk diimplementasi. Metode ini cocok digunakan untuk sistem yang sangat bergantung pada performa, seperti real-time processing, gaming, dan web caching. Metode ini juga cocok untuk sistem yang memiliki jumlah write yang sangat tinggi.
- Read-ahead: Teknik caching ketika sistem memprediksi data selanjutnya yang akan digunakan dan langsung memasukkannya ke cache sebelum digunakan oleh aplikasi. Tujuannya adalah untuk mengurangi waktu menunggu data diambil dari storage, karena sudah ada di cache. Metode ini akan meningkatkan

performa sistem, namun sistem juga harus memprediksi data selanjutnya yang akan digunakan. Sehingga metode ini unggul pada sistem yang banyak melakukan akses sekuensial dan dapat diprediksi. Kelemahan dari metode ini adalah memory usage meningkat dan akan ada kemungkinan melakukan pembacaan memori yang tidak perlu dan membuang waktu dan resource.

- No read-ahead: Teknik caching yang hanya memasukkan data ke cache ketika diminta secara eksplisit, sehingga tidak ada prefetching dan prediksi data selanjutnya. Hal ini akan mengurangi performa, namun akan menghilangkan akses-akses memori yang tidak diperlukan. Metode ini cocok untuk sistem yang memiliki akses data random dan tidak dapat diprediksi, ataupun sistem dengan memori terbatas yang harus mengurangi akses memori tidak efisien.
- Adaptive read-ahead: Teknik caching gabungan dari read-ahead dan no read-ahead, yaitu melakukan prefetching secara dinamis berdasarkan observasi akses data yang didapatkan. Teknik ini mencoba mengambil keunggulan dari kedua teknik sebelumnya. Efeknya pada performa dan penggunaan memori akan berada di tengah-tengah kedua metode tersebut, dan metode ini cocok digunakan untuk sistem yang tidak terlalu predictable namun tidak terlalu random.

## Referensi

[1] "What Is Write-Back? | Definition from TechTarget," TechTarget. [Online]. Available: <https://www.techtarget.com/whatis/definition/write-back>. [Accessed: 21-Jul-2024].

4. (2.5 poin) Tuliskan dan jelaskan minimal 4 perbedaan dan 4 persamaan aspek teknis (bukan perbedaan sejarah, *license*, *marketing*, etc) dari Windows, MacOS, dan salah satu *distro* Linux (bebas).

### Perbedaan (Windows / MacOS / Arch)

- Filesystem pada Windows dapat terpartisi menjadi banyak drive, seperti C:, D:, dst., sedangkan MacOS dan Arch hanya memiliki satu root directory yang menyimpan seluruh directory lain.
- Windows memiliki shell dengan sintaks unik tersendiri, sedangkan MacOS dan Arch menggunakan Bash sebagai shell default.
- MacOS dan Arch memiliki sebuah package manager secara default, yaitu homebrew untuk MacOS dan Pacman untuk Arch. Windows tidak memiliki package manager secara default.
- Linux (Arch) memiliki keamanan yang paling tinggi, karena bersifat open source sehingga yang memakai dapat mengidentifikasi dan berkolaborasi untuk menghilangkan bug dan security breach. MacOS masih cukup aman karena dijalankan dalam proprietary hardware sehingga tidak sering mendapatkan security breach. Windows merupakan OS yang paling rentan dibobol, karena kompatibel dengan banyak hardware dan juga merupakan OS paling populer,

sehingga kebanyakan malware didesain untuk membobol Windows secara khusus.

### Persamaan (Windows / MacOS / Arch)

- Seluruh sistem operasi memiliki support untuk lebih dari satu filesystem: NTFS dan exFAT di Windows, APFS dan HFS+ di MacOS, dan EXT4, BTRFS, dll. pada Arch.
- Seluruh sistem operasi memiliki support untuk protokol jaringan seperti TCP/IP, Wi-Fi, Ethernet, dan Bluetooth.
- Seluruh sistem operasi memiliki fitur keamanan seperti firewall, user authentication, dan encryption.
- Seluruh sistem operasi memiliki support untuk lebih dari satu akun yang terdaftar pada mesin yang sama.

### Referensi

[1] "Linux vs Windows vs Mac: Which OS Is Better For You?," RedSwitches, <https://www.redswitches.com/blog/linux-vs-windows-vs-mac/>. Accessed: Jul. 14, 2024.

5. (2.5 poin) **Android dan Ubuntu sama-sama dibangun berdasarkan Linux. Meskipun begitu, program pada Ubuntu tidak dapat dijalankan di Android, dan begitu juga sebaliknya. Jelaskan alasannya! (selain karena Android biasanya dijalankan pada arsitektur ARM dan Ubuntu pada arsitektur x86).**

Selain masalah arsitektur, terdapat dua hal lain yang harus sama pada dua buah device jika program untuk device A ingin dijalankan pada device B:

- Distribusi sistem operasi (tools, utilities, dan libraries harus berada pada lokasi yang sama dan memiliki hasil yang sama ketika digunakan)
- Kernel sistem operasi

Android dan Ubuntu dua-duanya memiliki kernel yang sama, yaitu kernel Linux, namun merupakan distribusi sistem operasi yang berbeda, dan memiliki tools dan libraries yang berbeda. Salah satu perbedaan utama antar sistem operasi adalah runtime environment-nya, dengan aplikasi Android berjalan pada Android Runtime (ART), dan aplikasi Linux berjalan pada glibc runtime environment.

### Referensi

[1] "Why don't Android applications work on Linux and vice versa?" Quora. [Online]. Available:

<https://www.quora.com/Why-don-t-Android-applications-work-on-Linux-and-vice-versa>.  
[Accessed: July 14, 2024]

6. (2.5 poin) **Jelaskan konsep - konsep sistem operasi berikut dalam bahasa yang dimengerti anak berumur 5 tahun!**

**a. Kernel**

Googoo gaagaa blerjrhhr blarhsks aksjakjwdj ffrsrsfsfhshhhhhh plepppp bookkkk pajip ffhffhhpopopoppo

(Kernel itu seperti otak yang ada di kepala kita. Otak berfungsi untuk menerjemahkan hal-hal yang kita pikirkan menjadi suatu aksi fisik, seperti berjalan, mengunyah, memukul, dan menendang.

Contohnya ketika kita melihat ada beruang yang besar dan menyeramkan, kita berpikir takut dan ingin melarikan diri, yang otak menerjemahkannya sebagai perintah untuk berlari menggunakan kaki

Sebuah kernel memiliki fungsi yang sama. Dia menerima perintah yang kita berikan melalui keyboard dan mouse, dan menerjemahkannya menjadi operasi yang berinteraksi dengan hardware, seperti prosesor dan memori).

**b. Driver hardware**

Brrrrt skibidi dop dop dop yes yes skibidi dop dop niit niit

(Bayangkan kamu sedang memesan pizza ke Pizza Hub lewat telepon. Kamu memilih pizza yang ingin dipesan, jumlah yang dipesan, dan alamat untuk dikirim ke petugas Pizza Hub. Telepon telah memfasilitasi komunikasi antar kamu dengan petugas Pizza Hub.

Sebuah driver hardware adalah telepon tersebut. Kamu yang memberikan perintah adalah device seperti mouse atau keyboard. Petugas Pizza Hub yang menerima dan melakukan perintah yang kamu berikan adalah komputer).

**c. Shell**

Gedagedagedaoo abimelotago werikikako weigigo dabidebukotolago

(Bayangkan kamu sedang di restoran dan kamu memberikan pesanan yang kamu inginkan ke waiter. Waiter itu lalu akan meneruskan pesanan tersebut ke para koki yang ada di dapur.

Kamu tidak berkomunikasi langsung dengan koki, namun koki yang pada akhirnya akan memasak makananmu. Pesananmu dibawakan ke koki melalui waiter. Waiter tersebut seperti shell pada komputer. Kamu memberikan perintah ke shell, lalu shell akan meneruskan perintah tersebut ke sistem operasi komputer).

**d. Multiprocessing dan cara kerjanya**

skibidi gyatt rizz only in ohio duke dennis did you pray today livvy dunne rizzling up baby gronk sussy imposter pibby glitch in real life sigma alpha omega male

grindset andrew tate goon cave freddy fazbear colleen ballinger smurf cat vs strawberry elephant blud dawg shmlawg ishowspeed a whole bunch of turbulence ambatukam bro really thinks he's carti literally hitting the griddy the ocky way kai cenat fanum tax garten of banban no edging in class not the mosquito again bussing axel in harlem whopper whopper whopper 1 2 buckle my shoe goofy ahh aiden ross sin city monday left me broken quirked up white boy goated with the sauce john pork grimace shake kiki do you love me huggy wuggy nathaniel b lightskin stare biggest bird omar the referee amogus uncanny wholesome reddit chungus keanu reeves pizza tower zesty poggers kumalala savesta quandale dingle glizzy rose toy ankha zone thug shaker morbin time dj khaled sisypus oceangate shadow wizard money gang ayo the pizza here PLUH nair butthole waxing t-pose ugandan knuckles family guy funny moments compilation with subway surfers gameplay at the bottom nickeh30 ratio uwu delulu opium bird cg5 mewing fortnite battle pass all my fellas gta 6 backrooms gigachad based cringe kino redpilled no nut november pokénut november foot fetish F in the chat i love lean looksmaxxing gassy social credit bing chilling xbox live mrbeast kid named finger better caul saul i am a surgeon hit or miss i guess they never miss huh i like ya cut g ice spice gooning fr we go gym kevin james josh hutcherson coffin of andy and leyley metal pipe falling

(Bayangkan jika kamu menyuruh seorang koki untuk memasak 500 pizza. Waktu memasak tersebut akan memakan waktu yang sangat lama. Namun jika melainkan satu koki, yang bekerja adalah 25 koki, waktu yang dibutuhkan akan sangat berkurang. Masing-masing koki hanya butuh memasak 20 pizza, dan seluruh koki dapat memasak secara bersamaan, sehingga waktu yang dibutuhkan hanya waktu untuk memasak 20 pizza.

Pada kasus tersebut, tugas yang butuh waktu sangat lama dibagikan ke lebih dari satu pekerja dan berjalan bersamaan. Multiprocessing konsepnya sama seperti hal tersebut. Sebuah pekerjaan dibagi rata pada sejumlah prosesor dan dijalankan bersamaan agar pekerjaan selesai lebih cepat).

**7. (2.5 poin) Jelaskan apa yang terjadi ketika sebuah program dijalankan pada komponen berikut dan apa peran *operating system* pada kasus ini?**

**a. RAM**

Ketika sebuah program dijalankan, kode program akan dibaca dari non-volatile storage seperti disk, lalu dimasukkan ke RAM agar bisa lebih cepat diakses prosesor. Peran sistem operasi adalah mengatur alokasi memori program pada RAM, seperti menentukan apakah program akan disimpan secara kontigu atau non-kontigu, serta membuat mekanisme proteksi memori agar kode program tidak asal dibaca dan ditulis oleh program lainnya.

**b. SSD/HDD**

Program umumnya akan disimpan pada disk seperti SSD/HDD, lalu ketika komputer mendapat perintah untuk menjalankan program, kode yang ada di disk akan dibaca oleh sistem operasi dan dimasukkan ke memori. Peran sistem operasi adalah membaca program tersebut dari disk, sekaligus melakukan manajemen filesystem dan memfasilitasi operasi read, write, dan delete.

**c. Processor**

Ketika sebuah program berjalan, instruksi program akan dibaca satu per satu oleh prosesor dan dieksekusi. Peran sistem operasi adalah melakukan scheduling atas instruksi-instruksi yang akan dijalankan, memfasilitasi *context-switch* ketika terjadi pergantian proses, serta memberikan mekanisme *interrupt* ke prosesor jika ada event yang perlu direspon oleh prosesor.

**d. Swap memory**

Swap memory adalah storage di disk seperti HDD/SSD yang digunakan sebagai tambahan RAM ketika RAM penuh. Swap memory hanya akan digunakan ketika memori tidak mencukupi untuk dimasukkan program yang ingin dijalankan, sehingga program yang sudah lama tidak digunakan di RAM akan dimasukkan ke swap memory, dan program yang ingin dijalankan dimasukkan ke RAM. Peran sistem operasi adalah memfasilitasi proses *swap-in* dan *swap-out* ke swap memory, serta memastikan kapan harus memanfaatkan swap memory.

**8. (2.5 poin) Pada Linux terdapat istilah *everything is a file*, apa maksud dari istilah tersebut dan berikan contohnya!**

Istilah ini mengatakan bahwa seluruh komponen sistem operasi dapat diakses dan dimanipulasi layaknya file biasa, bahkan devices, directories, dan resource sistem dapat dimanipulasikan seperti file. Contohnya sebuah directory /home/user/ adalah file yang mengandung daftar file dan directory lain, sebuah device seperti hard disk direpresentasikan dalam file dev/sda, dan socket merupakan file yang disimpan di tmp/socket.

**9. (2.5 poin) Jelaskan tentang Ext4 dan FAT32, sebutkan perbedaan antara keduanya.**

FAT (File Allocation Table) merupakan tipe filesystem yang umum ditemukan karena kesederhanaannya. Pada filesystem FAT, storage yang kontigu terpartisi menjadi sejumlah cluster dengan ukuran yang sama besar. Pada cluster pertama, disimpan File Allocation Table yang mendaftarkan cluster yang kosong, terisi, dan cluster selanjutnya jika file/folder berukuran lebih dari satu cluster.

Selain cluster pertama, cluster lainnya dapat berupa sebuah file, folder, atau cluster kosong. Sebuah file tidak memiliki struktur khusus. Sebuah folder merupakan sebuah tabel yang diisi sejumlah entri-entri dengan ukuran sama besar. Entri-entri

tersebut merupakan file dan subfolder yang ada di folder tersebut. Jika ukuran entri folder berukuran 32-bit, maka filesystem FAT ini dinamakan FAT32.

Ext4 merupakan tipe filesystem yang banyak digunakan pada sistem operasi Linux. Filesystem ini bekerja dengan membagi filesystem menjadi beberapa block group, yang masing-masing mengandung inodes, blok data, dan blok bitmap yang digunakan untuk mengalokasi blok. Sebuah inode merupakan struktur data yang menyimpan metadata tentang file seperti pemilik, izin, ukuran, dan pointer ke blok data. Terdapat juga sebuah superblock yang menyimpan informasi penting terkait filesystem, seperti ukuran, status, dan informasi konfigurasi.

Perbedaan antara FAT32 dan Ext4 adalah sebagai berikut:

- Sejarah dan penggunaan: FAT32 dikembangkan oleh Microsoft dan diperkenalkan pada Windows95. Tipe filesystem ini digunakan secara luas pada device seperti USB flash drive, SD card, external hard drive, dan lain-lain karena kemudahan implementasinya. Ext4 umumnya lebih banyak digunakan pada Linux saja.
- Ukuran file dan partisi: FAT32 hanya memiliki support ukuran file maksimum 4GB, dengan ukuran partisi maksimum 8TB. EXT4 memiliki support file maksimum 16TB dengan partisi maksimum 1EB, jauh lebih banyak dari FAT32.
- Performa dan fitur: FAT32 merupakan filesystem yang cukup sederhana dan tidak memiliki banyak fitur. Dibandingkan hal tersebut. EXT4 memiliki banyak fitur yang meningkatkan performa filesystem, seperti journaling, extents, dan delayed allocation.
- Kompatibilitas: Kesederhanaan FAT32 membuat filesystem ini sangat kompatibel dengan seluruh sistem operasi (Windows, MacOS, dan Linux), sedangkan EXT4 hanya kompatibel dengan Linux.

## Referensi

[1] "File Allocation Table," Wikipedia, the free encyclopedia. [Online]. Available: [https://en.wikipedia.org/wiki/File\\_Allocation\\_Table](https://en.wikipedia.org/wiki/File_Allocation_Table). [Accessed: 21-Jul-2024].

[2] "Ext4," Wikipedia, the free encyclopedia. [Online]. Available: <https://en.wikipedia.org/wiki/Ext4>. [Accessed: 21-Jul-2024].

[3] "FAT32 vs EXT4: What's the Difference & Which One Is Better?" MiniTool Partition Wizard. [Online]. Available: <https://www.partitionwizard.com/news/fat32-vs-ext4.html>. [Accessed: 21-Jul-2024].

**10. (2.5 poin) Misalkan terdapat dua sistem yang berbeda, yaitu sistem yang CPU heavy dan sistem real-time. Keduanya memiliki prioritas task yang berbeda sehingga membutuhkan strategi scheduling yang berbeda. Sebutkan dan jelaskan strategi task scheduling yang cocok untuk masing-masing sistem!**

Sebuah CPU-Heavy System dapat menggunakan strategi Multi-Level Feedback Queue (MLFQ). Strategi ini didesain untuk mengelola proses dengan waktu eksekusi dan prioritas yang bervariasi. Metode ini dapat secara dinamis mengubah prioritas berdasarkan perilaku dan kebutuhan proses.

Sistem MLFQ memiliki beberapa queue, masing-masing memiliki nilai prioritasnya. Sebuah task dapat berpindah-pindah queue tergantung perilaku dan kebutuhan sistem. MLFQ juga memiliki sistem aging, yaitu task yang terlalu lama berada di lower-priority queue akan dinaikkan ke queue dengan prioritas yang lebih tinggi. Metode aging ini merupakan solusi yang baik untuk starvation, yaitu ketika sebuah task tidak pernah mendapatkan CPU. Sistem ini juga dapat mendeteksi task yang terlalu banyak memakan CPU time dan menurunkannya ke queue dengan prioritas lebih rendah.

Sebuah Real-Time System dapat memanfaatkan strategi algoritma Rate-Monotonic Scheduling. Algoritma ini didesain untuk sistem real-time, khususnya untuk task periodik. Pada metode ini, setiap task periodik memiliki prioritas berdasarkan periode waktunya, dan task dengan periode lebih pendek akan memiliki prioritas yang lebih tinggi. RMS adalah algoritma preemptive, sehingga task dengan prioritas lebih tinggi dapat menginterupsi task prioritas rendah. Penjadwalan dilakukan dengan mengurutkan task berdasarkan prioritasnya dan menjalankan task dengan prioritas tertinggi.

Manfaat RMS adalah prediktabilitas dan sifat deterministiknya. Semua task akan memiliki prioritas tetap sehingga kapan task itu dieksekusi dapat selalu diprediksi. Hal ini menyebabkan sistem yang menggunakan algoritma RMS bersifat deterministik, sehingga sistem dapat dikontrol dan akan jarang terjadi gangguan-gangguan yang tidak diperkirakan pada sistem.

**11. (2.5 poin) Jelaskan pendapatmu mengapa sistem-sistem operasi berbasis Linux belum berhasil mengalahkan Windows dalam pasar desktop, dan mengapa mereka unggul dalam pasar server.**

Linux belum bisa mengalahkan Windows dalam pasar desktop karena beberapa hal:

- Banyak desktop yang sudah datang pre-installed dengan Windows, membuat Windows sebagai pilihan default untuk banyak pengguna. Sedangkan desktop yang pre-installed dengan distribusi Linux jauh lebih sedikit, sehingga pengguna yang ingin menggunakan Linux harus mencari dan menginstall sendiri.
- Banyak software dan game yang belum kompatibel dengan Linux. Bahkan banyak software yang didesain khusus untuk kompatibel dengan Windows saja.
- Karena Windows dan MacOS sudah lama mendominasi pasar desktop, hal ini membuat sebuah *positive feedback loop* yang membuat pembeli desktop baru lebih memilih menggunakan Windows dan MacOS karena sudah populer dan terbukti bekerja dengan baik untuk banyak pengguna.

- Terdapat banyak distribusi Linux dengan kelebihan dan kekurangannya masing-masing, yang membuat pemilihan Linux sebagai sistem operasi utama jauh lebih sulit.
- Windows dan MacOS merupakan dua sistem operasi yang didesain khusus sebagai sistem operasi yang user-friendly dan market-friendly. Distribusi Linux pada umumnya didesain untuk memaksimalkan di bidang lain, seperti performa, keamanan, customizability, dan lainnya.

Namun, Linux sangat unggul di pasar server. Ini dikarenakan:

- Linux dikenal sebagai sistem operasi yang minimalis dan ringan. Sehingga dalam konteks ratusan komputer yang berjalan bersamaan, keringanan ini akan sangat meningkatkan performa dan meminimalkan penggunaan resource.
- Linux juga dikenal atas keamanannya, sehingga sebuah server dengan ratusan komputer Linux akan lebih sulit untuk dibobol.
- Linux memiliki customizability yang sangat tinggi, sehingga dapat dimodifikasi untuk keperluan-keperluan tertentu.
- Sebuah server tidak membutuhkan interface yang user-friendly, bahkan server umumnya dioperasikan oleh orang-orang yang sangat ahli di bidang komputer, sehingga Linux yang fokus pada performa dan keamanan jauh lebih cocok dipakai.
- Linux juga merupakan sistem operasi yang gratis, sehingga jika ingin membangun server dengan ratusan dan ribuan komputer yang menjalankan Linux tidak perlu ada pengeluaran biaya untuk software.

**12. (5.0 poin) Sebutkan dan jelaskan secara komprehensif 3 *vulnerability/exploit* yang berasal atau terkait dengan *operating system* tertentu.**

#### **EternalBlue**

EternalBlue adalah sebuah exploit yang memanfaatkan vulnerability kritis dalam protokol Server Message Block (SMB) versi 1 di sistem operasi Windows. Pada April 2017, kelompok peretas yang dikenal sebagai Shadow Brokers membocorkan exploit ini ke publik. EternalBlue memanfaatkan kelemahan dalam implementasi SMBv1, memungkinkan penyerang untuk mengirimkan paket yang dirancang khusus ke server Windows yang rentan dan mendapatkan unauthorized access penuh ke sistem.

Exploit ini digunakan secara masif oleh ransomware WannaCry, yang menyebar secara global pada Mei 2017, menyebabkan kerusakan besar dengan mengenkripsi data pengguna dan menuntut pembayaran tebusan. WannaCry menggunakan EternalBlue untuk menyebar secara otomatis dari satu komputer ke komputer lain dalam jaringan yang rentan, menciptakan salah satu serangan ransomware terbesar dalam sejarah. Selain itu, eksplorasi ini juga digunakan oleh serangan ransomware NotPetya dan beberapa malware lainnya.

#### **DirtyCOW**

DirtyCOW (Copy-On-Write) adalah sebuah exploit serius yang ditemukan dalam kernel Linux, yang pertama kali ditemukan pada Oktober 2016. Dikenal juga sebagai CVE-2016-5195 exploit ini, memanfaatkan kelemahan dalam mekanisme copy-on-write (COW) pada manajemen memori kernel Linux. Eksloitasi ini memungkinkan penyerang untuk mendapatkan hak akses penulisan ke memori yang seharusnya hanya dapat dibaca, yang dapat digunakan untuk meningkatkan privilege pengguna biasa menjadi pengguna superuser (root).

Cara kerja DirtyCOW adalah dengan memanfaatkan kondisi race condition dalam cara kernel menangani page memori. Saat dua proses bersamaan mengakses dan menulis ke page memori yang sama, kernel dapat dipaksa untuk memberikan write access ke page memori yang seharusnya dilindungi. Penyerang dapat menggunakan teknik ini untuk memodifikasi file yang dilindungi atau menjalankan kode dengan privilege yang tinggi.

### **Bluekeep**

BlueKeep adalah sebuah kerentanan kritis dalam protokol Remote Desktop Services (RDS) yang mempengaruhi berbagai versi Windows, termasuk Windows 7, Windows Server 2008 R2, dan Windows Server 2008. Terdaftar sebagai CVE-2019-0708, kerentanan ini ditemukan pada Mei 2019 dan memiliki potensi dampak yang sangat besar karena memungkinkan penyerang untuk melakukan eksekusi kode jarak jauh (remote code execution) tanpa authorization dengan mengirimkan paket yang dirancang khusus ke layanan RDS pada komputer yang rentan.

Kerentanan BlueKeep terletak pada cara layanan RDS memproses permintaan untuk koneksi remote desktop. Penyerang dapat mengeksplorasi bug ini untuk memicu kondisi balapan dan menyebabkan kerusakan memori yang memungkinkan mereka menjalankan kode arbitrer di sistem target. BlueKeep dapat digunakan untuk melakukan serangan berbasis worm, yang berarti ia dapat menyebar otomatis dari satu komputer ke komputer lain dalam jaringan yang rentan, mirip dengan cara ransomware WannaCry menyebar melalui eksplorasi EternalBlue.

- 13. (5.0 poin) Belakangan ini, beberapa game mulai menggunakan sistem anti-kecurangan (*anticheat*) yang beroperasi dalam tingkat *kernel* - lebih sering dikenal sebagai *kernel level anticheat* - untuk melawan masalah kecurangan dengan lebih efektif. Di antara game-game tersebut adalah Valorant, *game-game* Hoyoverse seperti Genshin Impact dan Honkai: Star Rail, Rainbow Six Siege, serta Apex Legends.**

- a. Jelaskan cara kerja *kernel level anticheat*.

Kernel-level anticheat bekerja pada mode paling *privileged* pada sistem operasi, yaitu kernel mode atau ring 0. Mode operasi ini berada pada level akses yang lebih tinggi dari mode user yang biasanya digunakan oleh kebanyakan aplikasi, dan menyebabkan software anticheat dapat melihat interaksi pada sisi

hardware dan low level untuk mencari exploit yang umumnya tidak dapat ditemukan dengan software yang beroperasi pada user mode. Kernel-level anticheat juga dapat membuat aturan lebih low-level dan mencegah modifikasi yang tidak terautentikasi terhadap software game yang dimainkan.

- b. **Jelaskan mengapa penggunaan *kernel level anticheat* membuat permainan tidak dapat dijalankan (dengan lancar, aman, dan atau legal) di Linux.**

Penggunaan software dengan kernel-level anticheat yang tidak dapat dijalankan dengan lancar pada Linux merupakan akibat dari Linux sebagai open-source operating system, yang menyebabkan kernel dan sistem operasi sangat fleksibel untuk dimodifikasi pengguna. Kernel-level anticheat butuh pemahaman tinggi atas kernel pada sistem operasi, sehingga sebuah distribusi Linux yang sudah banyak termodifikasi akan sulit dimengerti oleh software anticheat.

Dari segi legalitas, sebuah kernel-level anticheat memiliki akses yang dalam terhadap sistem operasi dan hardware pengguna, dan secara efektif dapat melakukan apapun terhadap perangkat pengguna. Selain berupa sebuah pelanggaran terhadap privasi pengguna, hal ini juga dapat dimanfaatkan pihak jahat untuk memanipulasi dan menyerang perangkat pengguna dengan memanfaatkan hak akses kernel-level pada anticheat.

- c. **Jelaskan (alasan-alasan) mengapa banyak orang yang waspada atau bahkan menentang penggunaan *kernel level anticheat*.**

Kernel-level anticheat membawa masalah yang beragam yang secara wajar akan menyebabkan banyak tantangan dari pengguna. Hal yang paling jelas adalah pelanggaran privasi pengguna, sebab software dengan hak akses kernel dapat melihat aktivitas sistem pengguna pada level hardware, bahkan mengumpulkan data tersebut dan menggunakananya untuk hal yang tidak wajar.

Selain masalah privasi, software yang berjalan pada level kernel dapat memperkenalkan banyak masalah pada keamanan perangkat pengguna, baik secara sengaja maupun tidak sengaja. Secara sengaja, sebuah pihak luar dapat memanfaatkan kernel-level anticheat untuk memanipulasi atau memasukkan berbagai software malicious pada perangkat pengguna. Secara tidak sengaja, sebuah software yang berinteraksi dengan kernel komputer dapat memunculkan bug-bug yang dapat berakibat pada kehancuran hardware pengguna.

### III. Jaringan Komputer



🙏 Bless 🙏

1. (2.5 poin) Jelaskan secara komprehensif apa yang terjadi ketika kita mengirimkan email!

Langkah-langkah pengiriman email di internet dari sejak dikirim sampai diterima adalah sebagai berikut:

- a. Komposisi: Pengirim akan membuat sebuah email menggunakan sebuah klien email seperti Outlook, Gmail, dll. Setelah email sudah selesai dibuat dan akhirnya dikirim, klien email akan mengirimkan email tersebut ke email server milik pengirim menggunakan Simple Mail Transfer Protocol (SMTP).
- b. Transmisi SMTP: Email server menerima email dan metadata dari klien email pengirim dan melakukan pemrosesan email seperti verifikasi address dan spam filtering.
- c. DNS Lookup: Email server akan mengambil domain dari email address pengirim dan melakukan query ke Domain Name System (DNS) server untuk menemukan alamat email server milik penerima.
- d. Pengiriman: Setelah alamat tujuan berupa email server penerima sudah ditemukan, email yang dikirim akan melalui beberapa router menggunakan SMTP.
- e. Diterima di Mail Server: Setelah email sudah sampai di email server, email tersebut akan disimpan di mailbox penerima di email server.
- f. Diterima oleh Pengguna Tujuan: Ketika email sudah ada di mailbox penerima, klien email penerima dapat berkoneksi dengan email server dan menerima email tersebut menggunakan protokol seperti IMAP atau POP3. Setelah email sudah diterima oleh klien email, penerima dapat membaca email tersebut.

#### Referensi

[1] PowerCert Animated Videos, "How does the INTERNET work?," YouTube, 19-Jun-2024. [Online]. Available: [How Email Works](#) [Accessed: 20-Jul-2024].

2. (2.5 poin) Jelaskan apa perbedaan dari **Error-Correcting Code** dan **Error-Detecting Code**. Berikan masing-masing contoh algoritma, **pseudocode**, beserta **use case** dalam kehidupan sehari-hari!

ECC dan EDC adalah mekanisme pada komunikasi digital untuk memastikan integritas sebuah data. Kedua algoritma ini berperan dalam mendekripsi dan membenarkan mutasi data yang dapat terjadi akibat *noise*, interferensi, ataupun gangguan lain dalam perpindahan data.

Error-Detecting Code didesain untuk mengidentifikasi ketika terjadinya error pada transmisi data. Algoritma ini hanya dapat memberitahu ketika terjadi mutasi data, namun tidak memiliki mekanisme untuk membenarkan error tersebut. Error-Correcting Code merupakan peningkatan dari EDC, yang dapat mendekripsi sekaligus membenarkan error pada data. Walaupun ECC lebih membantu, umumnya algoritma pada ECC lebih kompleks sehingga lebih sulit untuk didesain dan dapat memakan waktu yang lebih lama.

Salah satu algoritma EDC yang umum digunakan adalah algoritma checksum. Algoritma ini menjadikan data sebagai sekuens 16-bit word, lalu melakukan penambahan terhadap angka-angka ini menggunakan aritmatika *one's complement*. Nilai checksum yang dihasilkan adalah *one's complement* dari hasil penambahan.

```
function calculateChecksum(data):
    sum = 0

    for i = 0 to length(data) - 1 step 2:
        if i + 1 < length(data):
            word = (data[i] << 8) + data[i + 1] // Combine two bytes to form a 16-bit word
        else:
            word = data[i] << 8 // If odd length, pad the last byte with zeros

        sum = sum + word

    // If sum overflows beyond 16 bits, wrap around (fold)
    if sum > 0xFFFF:
        sum = (sum & 0xFFFF) + 1

    // Take one's complement of the final sum
    checksum = ~sum & 0xFFFF

return checksum
```

Untuk Error-Correcting Code, salah satu algoritma yang dapat digunakan adalah algoritma *Hamming code*. Algoritma ini bekerja dengan menambahkan parity bit pada

posisi angka pangkat dua pada data. Kombinasi dari pengecekan parity bit ini dapat memutuskan lokasi terjadinya error, jika error yang terjadi adalah single-bit error.

```
function calculateHammingCode(data):
    # Step 1: Calculate the number of parity bits needed
    m = length(data)
    r = 0
    while (2^r < m + r + 1):
        r = r + 1

    # Step 2: Initialize the encoded message with parity bits at the correct positions
    encodedLength = m + r
    encoded = array of size encodedLength filled with 0

    j = 0 # Index for data bits
    for i = 1 to encodedLength:
        if (i == 2^(floor(log2(i)))):
            encoded[i - 1] = 0 # Parity bit placeholder
        else:
            encoded[i - 1] = data[j]
            j = j + 1

    # Step 3: Calculate parity bits
    for i = 0 to r - 1:
        parityPos = 2^i
        parity = 0
        for j = 1 to encodedLength:
            if (j & parityPos) != 0:
                parity = parity XOR encoded[j - 1]
        encoded[parityPos - 1] = parity

    return encoded
```

EDC dan ECC merupakan dua metode yang sering digunakan untuk memastikan integritas data pada kehidupan sehari-hari. Contoh penggunaan EDC adalah internet checksum pada header TCP/IP, dan contoh penggunaan ECC adalah di pemberian error pada QR Code dan Barcode.

3. (2.5 poin) Jelaskan pengertian dan perbedaan dari konsep-konsep yang bersangkutan di bawah ini dalam bidang *network security*!

a. *Cryptography, cryptology, dan cryptanalysis*.

- Cryptography adalah ilmu yang mempelajari teknik-teknik untuk menjaga komunikasi antara dua pihak agar tidak dapat dimengerti oleh pihak luar.
- Cryptology adalah ilmu yang mempelajari *secure communication* antara dua pihak secara umum, yang mengandung cryptography dan cryptanalysis.

- Cryptanalysis adalah ilmu yang mempelajari cara membobol komunikasi yang terjaga oleh kriptografi.

Cryptography adalah ilmu menjaga keamanan komunikasi, cryptanalysis adalah ilmu membobol keamanan komunikasi, dan cryptology adalah ilmu yang mempelajari keamanan komunikasi secara umum.

b. *Public key cryptography, private key cryptography, digital signature, dan hash.*

- Public key cryptography: Merupakan nama lain *asymmetric cryptography*, jenis kriptografi ini menggunakan dua tipe kunci: *private key* dan *public key*. Public key merupakan kunci yang dapat diakses siapapun, sedangkan private key merupakan kunci khusus yang hanya boleh diakses oleh pemiliknya.
- Private key cryptography: Merupakan nama lain dari *symmetric cryptography*, jenis kriptografi ini memiliki kunci yang sama untuk pengirim dan penerima. Kunci ini tidak boleh diakses orang lain selain pengirim dan penerima.
- Digital signature: Merupakan aplikasi dari *asymmetric cryptography* untuk melakukan verifikasi pengirim pesan digital. Sebuah pesan dikirim dan diberikan sebuah signature berupa hash yang dihasilkan oleh private key pengirim. Hash ini didekripsi oleh penerima menggunakan public key pengirim untuk memverifikasi pengirim pesan tersebut.
- Hash: Merupakan fungsi yang menerima sebuah input dan memberikan output berupa sebuah string dengan ukuran tertentu. Output yang dihasilkan selalu unik berdasarkan input yang diberikan.

Public key cryptography memiliki dua jenis kunci, sedangkan private key cryptography hanya memiliki satu jenis. Digital signature merupakan aplikasi dari public key cryptography, dan hash merupakan fungsi yang sering digunakan pada cryptography.

c. IPSec dan SSL.

- IPSec: Sebuah protokol yang menjamin keamanan komunikasi IP dengan melakukan autentikasi dan enkripsi pada setiap IP packet di jaringan. Terdapat dua mode operasi untuk protokol ini, yaitu transport mode dan tunnel mode. Transport mode hanya mengenkripsi bagian data pada paket yang umumnya digunakan untuk end-to-end communication, sedangkan tunnel mode mengenkripsi keseluruhan paket termasuk header, umumnya digunakan pada VPN.
- SSL: Sebuah protokol yang menjamin keamanan pada network, dengan melakukan enkripsi, autentikasi, dan integritas data. Protokol ini digunakan dalam web browsing (HTTPS), email, dan VPN.

## Perbedaan

- IPSec beroperasi pada Network layer, sehingga menjamin keamanan paket IP antar host di jaringan, sedangkan SSL beroperasi di antara transport layer dan application layer, menjamin keamanan antar aplikasi.
  - IPSec fokus pada keamanan jaringan, sedangkan SSL fokus pada keamanan antar aplikasi end-to-end.
  - VPN IPsec umumnya perlu menggunakan third-party software pada perangkat pengguna, sedangkan SSL sudah disupport pada web browser sehingga VPN SSL tidak perlu melakukan instalasi software lain.
- d. CBC, PCBC, CFB, dan OFB.
- Cipher Block Chaining (CBC): Block cipher algorithm yang setiap block plaintext di XOR dengan block ciphertext sebelumnya sebelum dienkripsi.
  - Propagation Cipher Block Chaining (PCBC): Mirip dengan CBC, namun block plaintext dan block ciphertext dua-duanya terlibat dalam proses chaining.
  - Cipher Feedback (CFB): Mengubah sebuah block cipher menjadi self-synchronizing stream cipher. Mengenkripsi unit kecil plainteks dan hasil ciphertextnya digunakan untuk enkripsi plaintext.
  - Output Feedback (OFB): Mengkonversi block cipher menjadi synchronous stream cipher.

### Perbedaan

Error pada satu block CBC hanya mempengaruhi block itu dan block selanjutnya. Error pada PCBC mempengaruhi seluruh block pada chain. Error pada CFB hanya mempengaruhi block saat ini dan block selanjutnya. Error pada OFB tidak akan terpropagasi, masing-masing block terenkripsi secara independen.

### Referensi

- [1] H. Walker, "Cryptology vs Cryptography vs Cryptanalysis: Get Your Vocabulary Right," *Hacker Noon*, 13-Jul-2021. [Online]. Available: <https://hackernoon.com/cryptology-vs-cryptography-vs-cryptanalysis-get-your-vocabulary-right-mw3o32w4>. [Accessed: 21-Jul-2024].
- [2] "Cryptography," *Wikipedia*, 21-Jul-2024. [Online]. Available: <https://en.wikipedia.org/wiki/Cryptography>. [Accessed: 21-Jul-2024].
- [3] "IPSec vs SSL VPN: What's the Difference?," *Cloudflare*, 22-May-2023. [Online]. Available: <https://www.cloudflare.com/learning/network-layer/ipsec-vs-ssl-vpn/#:~:text=The%20IPsec%20protocol%20suite%20operates.of%20directly%20encrypting%20IP%20packets>. [Accessed: 21-Jul-2024].

[4] "IPSec vs SSL: Which VPN Protocol is Better?," *N-able*, 15-Aug-2023. [Online]. Available: <https://www.n-able.com/blog/ipsec-vs-ssl>. [Accessed: 21-Jul-2024].

[5] "Block cipher mode of operation," *Wikipedia*, 21-Jul-2024. [Online]. Available: [https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation). [Accessed: 21-Jul-2024].

4. (2.5 poin) Dalam penggunaan sehari-hari, model TCP/IP lebih mirip dengan implementasi yang ada pada dunia nyata. Jelaskan perbedaan dari model OSI dan TCP/IP dan jelaskan pula alasan model OSI lebih sering digunakan dibandingkan model TCP/IP!

Pada model OSI, sebuah jaringan tersegregasi menjadi 7 layer, dari high-level ke low-level: Application, Presentation, Session, Transport, Network, Data Link, dan Physical.

Model TCP/IP merupakan model penyederhanaan dari OSI model yang berbasis dari protokol-protokol yang umumnya dipakai di internet, dan memiliki 4 layer: Application (yang merupakan gabungan dari Application, Presentation, dan Session pada OSI), Transport, Internet, dan Network Access (berkorespondensi dengan Data Link dan Physical pada model OSI).

Model OSI merupakan model yang teoretis, sedangkan model TCP/IP jauh lebih dekat pada sisi praktikal dan aplikatif dari jaringan. Namun model OSI masih lebih sering terlihat digunakan dibandingkan model TCP/IP karena beberapa hal, seperti sifat model OSI yang jauh lebih spesifik dan modular dibandingkan TCP/IP, sifat umum model OSI yang dapat diterapkan di berbagai hal dibandingkan TCP/IP yang jauh lebih spesifik, sifat edukatif dari model OSI yang lebih menjelaskan terkait struktur jaringan, dan alasan sejarah karena model OSI yang lebih sering digunakan dan direferensikan pada literatur akademik.

5. (2.5 poin) Jelaskan cara kerja WiFi beserta masalah-masalah yang muncul serta penanggulangan dari interferensi antara dua jaringan WiFi!

WiFi merupakan *market name* dari protokol IEEE 802.11 yang merupakan teknologi untuk menyambungkan banyak perangkat ke sebuah jaringan secara *wireless* menggunakan gelombang radio.

Cara WiFi bekerja adalah dengan sebuah router yang merupakan *access point* ke internet. Router ini terhubung ke jaringan internet, dan mem-*broadcast* jaringan tersebut ke sekelilingnya menggunakan gelombang radio. Perangkat-perangkat lainnya yang kompatibel dengan WiFi dapat mendeteksi gelombang radio ini dan menyambungkan diri ke gelombang tersebut. Setelah sudah terhubung, perangkat tersebut memiliki akses ke router dan sekaligus ke internet, dan melakukan transfer data melalui gelombang radio tersebut.

Beberapa masalah yang dapat muncul dalam menggunakan WiFi adalah interferensi antara perangkat lain maupun objek fisik, jarak yang terbatas, *congestion* ketika terdapat banyak perangkat yang menggunakan jaringan yang sama, serta masalah keamanan seperti unauthorized access pada sebuah jaringan.

Cara untuk menanggulangi masalah interferensi antara dua jaringan WiFi adalah: Menggunakan *channel* jaringan yang tidak terlalu padat, menggunakan *transmit power* yang lebih kecil sehingga WiFi tidak meraih jarak terlalu jauh dan meminimalkan potensi untuk mengalami interferensi, atau memindahkan router ke lokasi yang lebih strategis yang mengalami interferensi jaringan lebih sedikit.

**6. (2.5 poin) Sebutkan rentang-rentang frekuensi WLAN yang dilarang untuk digunakan di Indonesia dan jelaskan alasan mengapa dilarang!**

Menurut surat edaran yang dikeluarkan Kemkominfo pada tahun 2019 terkait perangkat wireless access point pada pita frekuensi radio 5,8 GHz, frekuensi radio rentang 5570-5670 MHz dilarang digunakan, karena sudah dialokasikan untuk keperluan radar meteorologi.

#### Referensi

[1] Kementerian Komunikasi dan Informatika Republik Indonesia, "Surat Edaran Nomor 3 Tahun 2019 tentang Penggunaan Spektrum Frekuensi Radio," *Direktorat Jenderal Sumber Daya dan Perangkat Pos dan Informatika (SDPPI)*, 2019. [Online]. Available: [https://sdppi.kominfo.go.id/downloads/62/20190923120212-surat\\_endaran.pdf](https://sdppi.kominfo.go.id/downloads/62/20190923120212-surat_endaran.pdf). [Accessed: 21-Jul-2024].

**7. (2.5 poin) Jelaskan perbedaan antara standar WiFi 802.11 b/g/n, 802.11 ac, 802.11 ax, dan 802.11 be!**

- 802.11b (WiFi 1) bersamaan dengan 802.11a merupakan standar WiFi pertama yang dikeluarkan pada tahun 1999. 802.11b menggunakan DSSS (Direct-Sequence Spread Spectrum) untuk mengurangi interferensi jaringan, memiliki frekuensi 2.4GHz, dan memiliki kecepatan sampai 11 Mbps. 802.11b dapat menembus halangan fisik dengan bagus, namun data bergerak lambat dan dapat berinterferensi dengan perangkat dengan frekuensi yang sama, contohnya monitor dan microwave.
- 802.11g (WiFi 3) merupakan lanjutan dari 802.11b yang dikeluarkan pada tahun 2003 dengan kecepatan yang meningkat, memiliki bandwidth sampai 54 Mbps. 802.11g masih menggunakan frekuensi 2.4GHz sehingga masih *backwards-compatible* dengan 802.11b.
- 802.11n (WiFi 4) merupakan standar selanjutnya yang dikeluarkan pada tahun 2009. Standar ini merupakan standar pertama yang menggunakan MIMO (Multiple-Input Multiple-Output), sebuah teknologi yang menggunakan kumpulan antena untuk menerima lebih banyak data dari sebuah perangkat. Standar ini juga standar pertama yang dapat menggunakan dua frekuensi, yaitu 2.4 GHz

dan 5 GHz, sehingga standar ini *backwards compatible* dengan 802.11a/b/g. WiFi 4 dapat memiliki bandwidth sampai 600 Mbps.

- 802.11ac (Gigabit WiFi / WiFi 5) merupakan standar selanjutnya yang dikeluarkan pada tahun 2013. Standar ini juga dapat menggunakan dua frekuensi, dengan bandwidth sampai 1300 Mbps untuk 5GHz, dan sampai 50 Mbps untuk 2.4GHz. Standar ini merupakan standar pertama yang menggunakan Downlink Multi-User MIMO, yang dapat mentransfer informasi ke banyak perangkat pada waktu yang sama. 802.11ac kompatibel dengan 802.11a/b/g/n.
- 802.11ax (WiFi 6 / High-Efficiency WLAN) dikeluarkan pada tahun 2019, dan menambahkan banyak fitur baru seperti teknologi OFDMA, MU-MIMO, 1024-QAM, dan lain-lain. Standar ini memiliki latency yang lebih sedikit, keamanan yang meningkat, dan bandwidth yang lebih cepat, yang secara teoritis dapat menyampaikan 10 Gbps. Pada tahun 2021, dikeluarkan subkategori dari WiFi 6 yaitu WiFi 6E yang dapat mengakses frekuensi 6GHz.
- 802.11be (WiFi 7) merupakan standar WiFi paling terbaru, yang direncanakan untuk dikeluarkan pada akhir 2024. Standar ini memperkenalkan fitur-fitur seperti 4096-QAM, Multi-link operation (MLO), Automated frequency coordination (AFC), dan lain-lain, memiliki perkiraan throughput maksimum sampai 46 Gbit/s.

## Referensi

[1] SignalBoosters, "What Are the WiFi IEEE 802.11 Standards?" [Online]. Available:  
<https://www.signalboosters.com/blog/what-are-the-wifi-ieee-80211-standards/>.  
[Accessed: 17-Jul-2024].

[2] TechTarget, "Wi-Fi 7 (802.11be)," [Online]. Available:  
<https://www.techtarget.com/searchnetworking/definition/Wi-Fi-7>.  
[Accessed: 17-Jul-2024].

8. (2.5 poin) Jelaskan keseluruhan *layer* pada OSI Reference Model. Jelaskan pula proses enkapsulasi dan dekapsulasi serta *Protocol Data Unit* (PDU) untuk masing-masing *layer* pada model tersebut.

- a. Physical: Mengurus komunikasi secara fisik antar perangkat dan transmisi raw bits. Proses enkapsulasi pada tingkat ini yaitu menerima frame dari Data Link Layer dan mengubahnya menjadi sekuens bits. Proses dekapsulasi yaitu menerima bits dan mengubahnya menjadi frame. PDU pada tingkat ini adalah bit.
- b. Data Link: Bertanggungjawab atas transfer data antar node dan error detection/correction. Proses enkapsulasi yaitu membuat sebuah frame dengan menambahkan header dan trailer kepada packet, dan dekapsulasi dengan pembuangan header dan trailer dari frame. PDU pada tingkat ini adalah sebuah frame yang berisi informasi untuk error-checking dan flow control.
- c. Network: Mengatur informasi terkait lokasi device-device pada network, dan menentukan rute terbaik untuk pengiriman data. Enkapsulasi pada tingkat ini

yaitu penambahan header untuk pembuatan paket. Dekapsulasi dengan pembuangan header paket. PDU pada tingkat ini adalah paket yang memiliki informasi logical addressing seperti IP address.

- d. Transport: Berfungsi menjalankan transfer data secara komplit serta error-recovery dan flow control pada end-to-end communication antar device pengirim dan device penerima. Bertanggungjawab atas segmentasi data dan penyusunan ulang data serta menyusun hubungan komunikasi end-to-end. Proses enkapsulasi dengan segmentasi data ke dalam unit yang lebih kecil dan penambahan transport header. Proses dekapsulasi dengan pembuangan transport header dan penyusunan ulang data. PDU pada tingkat ini adalah Segment untuk TCP dan Datagram untuk UDP, dengan header memiliki informasi port number dan informasi sequencing data.
- e. Session: Berfungsi membuat, mengelola, dan mensinkronisasi interaksi antar dua perangkat yang berkomunikasi. Session layer memberikan dialog control dan sinkronisasi transmisi data pada komunikasi di jaringan. Enkapsulasi berupa penambahan session header pada data, dan dekapsulasi berupa pembuangan session header. PDU pada layer ini adalah data yang mengandung informasi session dan dialog control.
- f. Presentation: Layer ini bekerja dengan sintaks dan semantik pada informasi yang dikomunikasikan, dan bertugas sebagai translator data pada komunikasi. Layer ini berperan dalam translasi data, enkripsi/dekripsi data, kompresi/dekompresi data, serta formatting data. Enkapsulasi pada layer ini berupa formatting data berupa enkripsi, kompresi, dan lain-lain, serta dekapsulasi berupa dekripsi, dekompresi, dan penarikan ulang formatting yang sudah dilakukan. PDU pada layer ini adalah data yang terenkripsi dan terkompresi.
- g. Application: Berperan sebagai interface untuk aplikasi dan user untuk berinteraksi dengan jaringan. Layer ini berinteraksi secara langsung dengan aplikasi yang dimiliki pengguna untuk kegunaan komunikasi. Layer ini bertanggungjawab atas application protocol (HTTP, FTP, SMTP, dst.), autentikasi pengguna, file transfer, dan lain-lain. Enkapsulasi di layer ini berupa penambahan header application protocol ke data, dan dekapsulasi berupa pembuangan header tersebut. PDU pada layer ini adalah data yang diberikan dan dikirim oleh pengguna secara langsung.

## Referensi

[1] "OSI Model," javatpoint. [Online]. Available: <https://www.javatpoint.com/osi-model>. [Accessed: 19-Jul-2024].

## 9. (2.5 poin) Jelaskan apa yang kamu ketahui tentang hal-hal di bawah ini!

- a. Apa yang dimaksud dengan DHCP Server?

Sebuah DHCP (Dynamic Host Configuration Protocol) Server adalah sebuah server pada jaringan yang melakukan pemberian IP address dan

konfigurasi jaringan lainnya secara otomatis kepada perangkat-perangkat yang tersambung ke sebuah jaringan. Sebuah DHCP Server umumnya dapat ditemukan pada sebuah router untuk jaringan rumah, dan pada sebuah dedicated server untuk jaringan-jaringan besar.

Contohnya pada sebuah router rumah, router tersebut akan memiliki sebuah DHCP server, dan ketika sebuah perangkat menyambung ke WiFi yang diberikan router tersebut, DHCP Server akan memberikan perangkat tersebut sebuah IP Address dan konfigurasi jaringan lainnya agar perangkat tersebut dapat berkomunikasi dengan perangkat lain pada jaringan.

**b. Tugas serta mekanisme umum dari DHCP Server.**

Sebuah DHCP Server memiliki tugas untuk melakukan konfigurasi jaringan dan IP Address assignment, yaitu memberikan IP address secara dinamis pada setiap perangkat pada jaringan selama sebuah durasi tertentu dalam bentuk *lease*. Selain IP Address, sebuah DHCP Server juga dapat memberikan konfigurasi lain seperti subnet mask, default gateway, dan DNS. Manfaat dari DHCP Server adalah manajemen jaringan tersentralisasi pada satu perangkat, sehingga mempermudah administrasi jaringan.

Mekanisme umum dari sebuah DHCP Server adalah sebagai berikut: sebuah perangkat elektronik pertama akan menyambung ke sebuah jaringan komputer. Perangkat tersebut akan mengirim pesan ke jaringan untuk mencari DHCP Server, yang kemudian server akan menawarkan sebuah IP address pada klien tersebut. Klien lalu mengirim kembali pesan ke jaringan untuk mengambil IP address tersebut, yang lalu DHCP Server akan mengirim pesan *acknowledgement* yang menandakan bahwa IP Address sudah diberikan ke perangkat tersebut.

**c. Empat jenis *messages* yang digunakan oleh protokol DHCP, beserta tipe dari *message* tersebut (*unicast/broadcast/multicast*).**

- DHCP Discover: Sebuah pesan broadcast yang dikirimkan oleh klien yang baru saja menyambung ke sebuah jaringan. Pesan ini digunakan untuk mencari DHCP Server pada jaringan tersebut.
- DHCP Offer: Setelah sebuah DHCP Server menerima pesan DHCP Discover, server akan mengirimkan sebuah pesan unicast ke klien tersebut berupa DHCP Offer untuk menawarkan sebuah IP Address dan informasi konfigurasi lain pada klien tersebut.
- DHCP Request: Setelah mendapatkan tawaran informasi konfigurasi, klien tersebut akan mengirimkan pesan DHCP Request untuk mengambil IP Address tersebut dari server. Pesan ini pada pengiriman pertama merupakan pesan broadcast, namun untuk pembaruan IP Address dikirim secara unicast.

- DHCP Acknowledgment (ACK): Setelah menerima DHCP Request, DHCP Server akan mengirimkan pesan terakhir secara unicast untuk mengkonfirmasi pengambilan IP Address, serta untuk memberikan informasi konfigurasi lainnya secara lengkap. Pesan ini dapat dianggap sebagai finalisasi terhadap proses pemberian IP Address ke sebuah perangkat pada jaringan.

### Referensi

- [1] GeeksforGeeks, "Dynamic Host Configuration Protocol (DHCP)," [Online]. Available: <https://www.geeksforgeeks.org/dynamic-host-configuration-protocol-dhcp/>. [Accessed: 17-Jul-2024].
- [2] Huawei, "DHCP Messages," [Online]. Available: <https://support.huawei.com/enterprise/en/doc/EDOC1100034071/225eec10/dhcp-messages>. [Accessed: 17-Jul-2024].

**10. (2.5 poin) Standar HTTP/3 sampai sekarang masih dikembangkan meskipun sudah dipakai beberapa perusahaan besar seperti Cloudflare. Peningkatan kinerja pada HTTP/3 dibandingkan dengan HTTP/2 salah satunya adalah karena penggunaan protokol QUIC. Jelaskan dan buat perbandingan mengapa HTTP/3 yang didukung dengan protokol QUIC memiliki kinerja yang lebih baik dibandingkan pendahulunya!**

HTTP/3 dapat memiliki performa yang lebih kencang dari HTTP/2 karena memanfaatkan protokol UDP dan QUIC, dibanding dengan TCP yang digunakan HTTP/2. Salah satu masalah terbesar pada HTTP/2 adalah ketika terjadi sebuah packet loss pada sebuah koneksi TCP, seluruh stream pada koneksi tersebut terblokir (head-of-line blocking). Karena HTTP/3 berbasis pada UDP, maka sebuah paket yang hilang hanya memblokir satu stream, sedangkan stream lain tetap berjalan dengan normal.

HTTP/3 juga memiliki support untuk 0-RTT atau Zero Round-Trip Time yang diberikan support oleh QUIC. Sebuah klien yang sudah pernah tersambung pada server dapat melakukan pengiriman data secara instan, karena mengeliminasi TLS acknowledgement yang dilakukan ketika koneksi di-setup.

### Referensi

- [1] "HTTP/3 vs. HTTP/2: The Performance Benefits of the Newest HTTP Version," Cloudflare, 21-Nov-2018. [Online]. Available: <https://blog.cloudflare.com/http-3-vs-http-2>. [Accessed: 19-Jul-2024].

**11. (2.5 poin) Di Indonesia, ketika kita hendak membuka situs yang ilegal tanpa menggunakan layanan VPN atau mengganti DNS, seringkali kita akan diarahkan ke halaman internet positif.**

**a. Jelaskan cara kerja pemblokiran tersebut!**

Ketika kita memasukkan URL sebuah website pada web browser, nama URL tersebut akan ditranslasikan menjadi sebuah IP Address melalui DNS Server, yang umumnya diberikan oleh ISP (Internet Service Provider).

Cara ISP memblokir sebuah situs adalah dengan melakukan DNS redirection dan HTTP redirection. Pada DNS redirection, ketika kita memasukkan URL sebuah website, DNS Server milik ISP akan mendeteksi website tersebut berada dalam list blokir, sehingga DNS Server akan mengalihkan web browser ke halaman Internet Positif. Pada HTTP Redirection, jika kita memasukkan IP Address secara langsung ke web browser, ISP akan mendeteksi HTTP Request ke situs terblokir dan mengarahkan web browser ke halaman Internet Positif.

**b. Jelaskan bagaimana VPN melewati pemblokiran tersebut.**

Sebuah VPN bekerja dengan melakukan enkripsi terhadap seluruh pertukaran informasi yang dilakukan sebuah klien pada internet. Informasi yang dienkripsi tersebut termasuk query DNS pengguna, sehingga server ISP tidak dapat mendeteksi ketika pengguna melakukan DNS query ke website terblokir. Sebagai alternatif, VPN server akan memberikan DNS severnya sendiri, sehingga klien masih dapat menggunakan nama website untuk mengakses website tersebut.

**c. Jelaskan bagaimana penggantian DNS melewati pemblokiran tersebut.**

Pemblokiran sebuah website dilakukan ketika DNS server milik ISP mendeteksi request untuk website terblokir. Sehingga jika pengguna mengganti DNS server, maka pendekripsi pada server ISP tidak terjadi, sehingga ISP tidak dapat mengarahkan web browser ke halaman Internet Positif.

**12. (5.0 poin) Sebutkan dan jelaskan secara komprehensif tiga *vulnerability/exploit* yang berasal atau terkait dengan jaringan komputer!**

**Man-in-the-Middle**

Tipe serangan ini adalah ketika sebuah pihak luar mengintersepsi komunikasi antar dua buah pihak dan memposisikan diri di tengah komunikasi, membuat seolah-olah komunikasi berjalan secara normal. Tujuan dari serangan ini adalah untuk mendapatkan informasi personal seperti login credentials dan nomor kartu kredit. Target umum dari serangan ini adalah bisnis finansial dan situs e-commerce.

Langkah pertama pada sebuah serangan MitM adalah intersepsi, yaitu membobol komunikasi sebelum informasinya sampai ke tujuan. Salah satu cara paling umum untuk melakukan ini adalah dengan memberikan akses ke malicious WiFi hotspot secara gratis. Ketika sebuah pengguna sudah melakukan koneksi, maka penyerang

memiliki akses lengkap pada pertukaran data. Beberapa cara lain untuk intersepsi antara lain:

- IP Spoofing: Penyerang menyamarkan diri sebagai sebuah aplikasi dengan memanipulasi header paket pada jaringan. Sehingga seorang pengguna yang ingin mengakses URL malah tersambung ke website malicious penyerang.
- ARP Spoofing: Penyerang melakukan penyambungan MAC address ke sebuah IP legitimate pada sebuah LAN menggunakan pesan ARP palsu.
- DNS Spoofing: Penyerang melakukan infiltrasi DNS server dan mengganti sebuah website address. Sehingga pengguna yang memasukkan URL ke website tersebut dipindahkan ke website penyerang.

Langkah kedua pada intersepsi adalah melakukan dekripsi data, jika komunikasi yang diintersepsi merupakan two-way SSL traffic. Dekripsi ini perlu dilakukan tanpa menarik perhatian kedua pihak. Hal ini dapat dilakukan dengan HTTPS spoofing, SSL hijacking, ataupun SSL stripping.

Pencegahan serangan MitM dapat dilakukan dengan hal-hal berikut: menghindari koneksi WiFi yang tidak password-protected, menghindari website unsecure, dan tidak menggunakan jaringan publik untuk melakukan transaksi sensitif. Bagi operator website, serangan ini dapat dimitigasi menggunakan protokol keamanan seperti TLS dan HTTPS yang melakukan enkripsi dan autentikasi data yang dikirim.

### **Distributed Denial-of-Service (DDoS)**

DDoS adalah sebuah tipe serangan cyber yang bertujuan untuk mengganggu atau menonaktifkan sebuah server atau service jaringan dengan mengirim internet traffic ke service tersebut sebanyak-banyaknya. Serangan DDoS diraih dengan memanipulasi sebuah jaringan komputer lain untuk membanjirkan server tersebut dengan request. Jaringan ini tersusun atas komputer dan perangkat lain yang telah terinfeksi dengan malware yang menyebabkan mereka dapat diambil alih oleh seorang penyerang. Sebuah device yang terinfeksi ini dinamakan bot atau zombie, dan kumpulan bot ini dinamakan botnet.

Setelah sebuah botnet sudah tersusun, penyerang akan memberikan instruksi ke setiap bot untuk mengirim request ke IP address target. Banyaknya request ini akan menyebabkan server atau network menjadi kewalahan, mengakibatkan denial-of-service kepada jaringan. Karena masing-masing bot merupakan sebuah device asli, memisahkan sebuah bot dengan sebuah klien asli menjadi sangat sulit.

Melakukan mitigasi atas serangan DDoS sangat sulit karena kesulitannya memisahkan traffic asli dengan traffic serangan tersebut, namun beberapa metode yang dapat digunakan adalah: Blackhole routing, rate limiting, web application firewall, dan anycast network diffusion.

### **SQL Injection**

SQL Injection (SQLi) merupakan sebuah exploit ketika seorang penyerang memanipulasi input ke sebuah website sehingga dapat melakukan query ke database secara langsung menggunakan sintaks SQL. Hal ini menyebabkan penyerang tersebut dapat mengakses data sensitif atau bahkan memanipulasi dan menghapus data sensitif tersebut.

Sebuah SQLi yang sukses dapat menghasilkan akses ilegal terhadap data seperti password, kartu kredit, dan data personal pengguna. SQL injection telah sering digunakan dalam high-profile data breach dari tahun ke tahun.

Sebuah serangan SQL injection dimulai dari sebuah metode input yang bisa dimasukkan pengguna, umumnya sebuah input teks seperti tempat memasukkan password atau komentar. Pada input tersebut, penyerang akan memasukkan sebuah SQL query di akhir. Query ini akan dianggap oleh backend server sebagai query ke database, dan akan dijalankan di database website.

Beberapa metode untuk mencegah terjadinya SQL injection adalah dengan validasi input, melakukan parameterized queries yang memisahkan kode dari data, menerapkan least privilege pada website, serta firewall dan intrusion detection.

## Referensi

- [1] Imperva, "Man in the Middle Attack (MITM) - Definition & Examples," *Imperva*, 2024. [Online]. Available: <https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/>. [Accessed: 21-Jul-2024].
- [2] Cloudflare, "What is a DDoS attack?," *Cloudflare*, 2024. [Online]. Available: <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>. [Accessed: 21-Jul-2024].
- [3] PortSwigger, "SQL Injection," *PortSwigger*, 2024. [Online]. Available: <https://portswigger.net/web-security/sql-injection>. [Accessed: 21-Jul-2024].

**13. (5.0 poin) Belakangan ini, muncul kabar bahwa salah satu penyedia internet di Korea Selatan menginstall malware pada perangkat pelanggan-pelanggannya yang melakukan torrenting.**

a. Jelaskan cara kerja torrent.

Torrenting adalah mekanisme untuk mendownload dan mendistribusikan file menggunakan protokol BitTorrent. Cara alternatif untuk mendownload sebuah file di internet adalah memintanya langsung dari sebuah server. Server tersebut akan mentransmisikan keseluruhan file tersebut melalui internet. Namun hal ini dapat mengganggu performa server jika file yang ingin ditransfer besar, atau terdapat banyak pengguna yang ingin file tersebut. Bandwidth pada server akan cepat penuh dan transmisi data akan melambat.

Protokol BitTorrent menanggulangi masalah ini dengan membuat sistem transmisi file yang *decentralized*. Melainkan satu file disimpan dan dikirim secara keseluruhan dari satu buah server (*centralized*), protokol BitTorrent memecah file tersebut menjadi banyak sekali partisi-partisi yang sangat kecil. Jika seseorang ingin mendownload sebuah file, maka transmisi data tersebut akan tersebar melalui banyak komputer, yang masing-masing menyumbang beberapa partisi file tersebut ke pengguna. Sebuah klien BitTorrent di perangkat pengguna lalu akan melakukan penggabungan partisi-partisi tersebut kembali menjadi satu buah file utuh.

**b. Jelaskan bagaimana ISP tersebut berhasil melakukan serangan tersebut.**

Tidak pernah diunggah bagaimana ISP Korea Selatan tersebut berhasil menginfeksi komputer pengguna torrent dengan malware, namun metode yang mungkin dilakukan adalah torrent poisoning dan deep packet inspection.

Torrent poisoning adalah bentuk serangan atau sabotase terhadap jaringan peer-to-peer (P2P), khususnya jaringan BitTorrent. Dalam serangan ini, file berbahaya atau file palsu disebarluaskan ke dalam jaringan P2P untuk mengganggu atau menghambat distribusi file asli. ISP dapat secara aktif mengunggah file yang terlihat sah tetapi sebenarnya rusak atau tidak berguna ke dalam jaringan P2P.

ISP juga dapat memanfaatkan Deep Packet Inspection (DPI), sebuah metode yang digunakan untuk memeriksa data yang dikirimkan melalui jaringan komputer secara mendetail. DPI memungkinkan inspeksi data pada level yang lebih tinggi daripada metode inspeksi jaringan tradisional yang hanya memeriksa header paket. Dengan DPI, ISP Korea Selatan tersebut dapat menginspeksi paket data pengguna torrent, lalu menginfeksi paket data tersebut dengan malware yang kemudian dikirim ke perangkat pengguna.

## Referensi

- [1] C. Pollette, "What is torrenting, and how does it work?," *ZDNet*, 24-Aug-2022. [Online]. Available: <https://www.zdnet.com/article/what-is-torrenting-and-how-does-it-work/>. [Accessed: 21-Jul-2024].
- [2] M. Patel and A. Sharma, "Torrent Poisoning: Anti-Piracy and Anonymity," *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, no. 10, Oct. 2014. [Online]. Available: [https://www.ijert.org/research/torrent-poisoning-antipiracy-and-anonymity-IJERTC\\_ONV3IS10014.pdf](https://www.ijert.org/research/torrent-poisoning-antipiracy-and-anonymity-IJERTC_ONV3IS10014.pdf). [Accessed: 21-Jul-2024].
- [3] Fortinet, "What is Deep Packet Inspection (DPI)?," *Fortinet*, 2024. [Online]. Available:

[https://www.fortinet.com/resources/cyberglossary/dpi-deep-packet-inspection#:~:t ext=Deep%20packet%20inspection%20\(DPI\)%2C,a%20checkpoint%20on%20th e%20network](https://www.fortinet.com/resources/cyberglossary/dpi-deep-packet-inspection#:~:t ext=Deep%20packet%20inspection%20(DPI)%2C,a%20checkpoint%20on%20th e%20network). [Accessed: 21-Jul-2024].

**14. (0.5 poin each) Jelaskan konsep - konsep berikut!**

**a. IP Address (IPv4 dan IPv6)**

Sebuah identifikasi numerik unik yang diberikan ke setiap perangkat pada sebuah jaringan. Alamat ini memungkinkan berkomunikasi antar perangkat di jaringan tersebut dengan mengirimkan data ke alamat yang tepat. IPv4 merupakan alamat 32-bit yang dapat mendukung sekitar 4.3 miliar alamat unik, sedangkan IPv6 merupakan alamat 128-bit yang dapat mendukung  $3.4 \times 10^{38}$  alamat unik.

**b. MAC Address**

MAC Address (Media Access Control Address) adalah alamat unik yang diberikan kepada setiap perangkat jaringan untuk mengidentifikasinya secara individual dalam jaringan lokal. Berbeda dengan IP address yang dapat berubah tergantung pada jaringan, MAC address bersifat tetap dan terintegrasi dalam hardware seperti network card. MAC address terdiri dari 48-bit (6 byte) dan biasanya ditulis dalam format heksadesimal, misalnya 00:1A:2B:3C:4D:5E.

**c. OSPF**

OSPF (Open Shortest Path First) adalah protokol routing yang digunakan untuk menentukan jalur terpendek dalam jaringan. Protokol ini menggunakan algoritma Dijkstra untuk mencari rute terbaik berdasarkan bandwidth.

**d. BGP**

BGP (Border Gateway Protocol) adalah protokol yang digunakan untuk pertukaran informasi antara autonomous systems (AS) di internet. BGP memungkinkan router di berbagai AS untuk berkomunikasi dan bertukar informasi tentang jalur yang tersedia ke tujuan yang berbeda, membantu menentukan rute terbaik untuk lalu lintas data antar jaringan.

**e. IS-IS**

IS-IS (Intermediate System to Intermediate System) adalah protokol routing dinamis yang digunakan untuk pertukaran informasi routing dalam jaringan besar, khususnya di dalam sistem otonom yang sama. IS-IS merupakan protokol link-state yang mirip dengan OSPF, mengandalkan algoritma Dijkstra untuk menentukan jalur terpendek berdasarkan metrik yang ditentukan.

**f. SDN**

SDN (Software-Defined Networking) merupakan sebuah metode pengelolaan jaringan yang memisahkan kontrol dari hardware seperti switch dan router, melainkan kontrol diatur oleh software terpusat yang disebut controller. Controller SDN mengatur aliran data melalui jaringan dengan memberikan instruksi ke hardware, sedangkan peran hardware hanya untuk meneruskan data sesuai instruksi yang diberikan.

**g. *VLAN* dan *Trunking***

VLAN (Virtual Local Area Network) adalah teknologi yang mempartisi sebuah jaringan menjadi beberapa jaringan logical, sehingga perangkat pada VLAN yang berbeda dapat berkomunikasi seolah-olah berada pada jaringan yang berbeda. VLAN berperan pada manajemen jaringan dan keamanan dengan mengurangi congestion dan menyederhanakan desain jaringan.

Trunking adalah metode yang digunakan untuk menghubungkan switch yang mendukung VLAN dengan membawa lalu lintas dari beberapa VLAN melalui satu link fisik. Dalam trunking, setiap frame data diberi tag dengan ID VLAN untuk memastikan bahwa data dari setiap VLAN dapat dikenali dan dipisahkan dengan benar saat melewati link trunk.

**h. *Overlay Network***

- i. *Spine leaf architecture***
- j. *FTP***
- k. *gRPC***
- l. *WebSocket***
- m. *SSH***
- n. *SAMBA***
- o. *PPTP***

## IV. Sistem Paralel dan Terdistribusi



~~Selamat berlibur bersama Sistermo 😊 Bangorin dulu lae diskon 40% di GoFood  
PESANKAN DULU LUE! Katau burger bangor laku kenapa diskon 40% matu mengerjakan!~~

1. (2.5 poin) Sebutkan contoh-contoh operasi yang sebaiknya dikomputasikan secara paralel, dan operasi yang sebaiknya dikomputasikan secara sekuensial. Untuk masing-masing, jelaskan alasannya.

### Sebaiknya Paralel

- Rendering gambar dan video: Contohnya memberikan filter pada setiap pixel di sebuah gambar, atau pemrosesan setiap frame pada sebuah video. Alasannya masing-masing pixel atau frame dapat diproses secara independen, sehingga sangat ideal untuk eksekusi paralel.
- Simulasi, seperti untuk weather forecasting atau dinamika molekuler: Komputasi pada simulasi-simulasi berikut dapat dibagi menjadi pekerjaan-pekerjaan lebih kecil yang berjalan secara independen. Pekerjaan kecil tersebut dapat dijalankan secara paralel dan di akhir hasilnya digabungkan.
- Analisis data dan pembelajaran mesin: Contohnya model training atau large-scale data analytics. Pekerjaan ini dapat dipisah menjadi subset-subset data yang masing-masing dapat dikerjakan secara independen dan paralel.
- Operasi MapReduce, seperti menghitung frekuensi kata dalam set dokumen yang besar: Bagian map dapat dikerjakan secara paralel dengan pemrosesan masing-masing dokumen secara independen, dan bagian reduce akan menggabungkan hasil dari seluruh pekerjaan paralel tersebut.

### Sebaiknya Sekuensial

- Kompilasi Source Code: Proses kompilasi melibatkan berbagai langkah yang dependen, seperti lexical analysis, parsing, dan semantic analysis, yang harus dikerjakan dalam sebuah urutan tertentu.
  - Proses Transaksi pada Database: Sebuah urutan transaksi pada database harus dilakukan dalam urutan yang tepat agar konsistensi data terjaga dan menghindari konflik.
  - Proses Algoritmik yang Dependental: Contohnya menyelesaikan persamaan linear menggunakan metode eliminasi Gauss. Setiap langkah dependen pada langkah sebelumnya, sehingga penyelesaian harus dilakukan secara sekuensial.
2. (2.5 poin) Jelaskan yang dimaksud dengan *level-level of parallelism* dan sebutkan contohnya.

Levels of parallelism mengacu pada tingkat atau granularitas sebuah aktivitas paralel dapat dilakukan. Tingkat ini dimulai dari unit terkecil pada komputasi hingga sistem terdistribusi berskala besar. Beberapa level parallelism umum antara lain:

- Bit-Level Parallelism: Sebuah tingkat parallelism sederhana yang mengacu pada aktivitas yang dilakukan prosesor lebih banyak ketika jumlah bit yang dapat diproses lebih banyak. Contohnya sebuah prosesor 32-bit dapat melakukan dua kali lebih banyak komputasi daripada prosesor 16-bit pada sebuah instruksi.
- Instruction-Level Parallelism: Mengacu pada menjalankan banyak instruksi dari sebuah program secara bersamaan. Contohnya sebuah CPU dapat mengeksekusi banyak instruksi secara paralel menggunakan pipelining.
- Data-Level Parallelism: Melakukan operasi yang sama pada banyak data sekaligus. Hal ini dilakukan dengan SIMD (Single Instruction Multiple Data) pada CPU dan GPU modern. Contohnya adalah mengaplikasikan filter pada sebuah gambar dengan operasi yang sama dilakukan pada setiap pixel.
- Task-Level Parallelism: Mengeksekusi banyak pekerjaan atau fungsi yang dapat bekerja secara independen. Contohnya adalah sebuah web server yang memproses banyak client request secara bersamaan. Tipe ini dapat dibagi menjadi dua sublevel, yaitu Thread-Level Parallelism yang menjalankan banyak thread secara paralel, dan Process-Level Parallelism yang menjalankan banyak proses secara paralel.

## Referensi

- [1] "Introduction to Parallel Computing," GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-parallel-computing/>. [Accessed: 20-Jul-2024].
- [2] "Parallel computing," Wikipedia, The Free Encyclopedia. [Online]. Available: [https://en.wikipedia.org/wiki/Parallel\\_computing](https://en.wikipedia.org/wiki/Parallel_computing). [Accessed: 20-Jul-2024].

**3. (2.5 poin) Jelaskan terkait *multithreading* dan *multiprocessing*, berikan contoh *library* untuk implementasinya, serta jelaskan kelebihan dan kekurangannya.**

Pada multithreading, banyak thread dibuat pada satu buah proses sehingga pekerjaan dapat dikerjakan secara konkuren. Kelebihan dari multithreading adalah fitur shared memory pada sebuah proses sehingga banyak thread dapat berbagi data dengan mudah, serta overhead yang kecil karena pembuatan thread tidak semahal pembuatan proses. Kekurangan dari multithreading adalah shared memory yang dapat menyebabkan race condition, serta sifat multithreading yang hanya konkuren dan tidak paralel. Salah satu library multithreading adalah java.lang.Thread di Java.

Pada multiprocessing, banyak proses dijalankan secara bersamaan, masing-masing dengan memory space-nya sendiri. Masing-masing proses tersebut dijalankan pada core CPU-nya sendiri, sehingga paralelisme eksekusi dapat tercapai dengan multiprocessing. Kelebihan dari multiprocessing adalah paralelisme asli sehingga pengeraaan sebuah pekerjaan dapat diselesaikan lebih cepat, serta isolasi antar proses yang mengurangi terjadinya masalah konkurensi. Kekurangan dari multiprocessing adalah meningkatnya overhead dan pemakaian resource, karena pembuatan dan manajemen proses lebih berat dari thread. Salah satu library multiprocessing adalah modul *multiprocessing* pada Python.

**4. (2.5 poin) Mengapa *thread* (*goroutine*) pada Golang lebih ringan dibandingkan dengan *thread* biasa?**

- Sebuah goroutine memiliki stack size awal yang sangat kecil (2KB) dan dapat berubah ukuran secara dinamis sesuai kebutuhan. Stack pada thread biasa umumnya jauh lebih besar, sekitar 1MB, dan tidak bisa berubah secara dinamis sehingga akan menghabiskan banyak memori jika dibuat banyak thread.
- Go melakukan scheduling goroutine dengan user-space scheduling, yang dapat mengalokasikan banyak goroutine ke jumlah OS thread yang sangat kecil. Sedangkan scheduling pada thread biasa diurus oleh kernel sehingga memiliki efisiensi yang lebih rendah.
- Context switch antar goroutine diatur oleh Go runtime sedangkan context switch pada thread biasa melibatkan kernel-mode context switch. Context switch pada goroutine terjadi secara penuh di user mode, sedangkan pada thread biasa melibatkan banyak perpindahan user-kernel mode yang memiliki overhead yang jauh lebih tinggi.

**5. (2.5 poin) Jelaskan bagaimana cara sinkronisasi proses untuk mencegah terjadinya *deadlock* dan *race condition*. Berikan dan jelaskan 3 metode untuk hal tersebut.**

Mencegah race condition dapat dilakukan dengan mencegah sebuah resource atau data dapat dimanipulasi dan dibaca oleh lebih dari satu proses atau thread secara konkuren. Sehingga jika sebuah data sedang dimanipulasi sebuah proses, proses lain yang ingin membaca atau memanipulasi data tersebut harus menunggu proses pertama

selesai. Namun hal ini mesti dilakukan dengan cermat agar tidak menghasilkan deadlock. Beberapa cara yang dapat dilakukan untuk mencegah masalah sinkronisasi tersebut antara lain:

- Mutex: Fitur mutex bekerja dengan konsep acquire lock dan release lock. Ketika sebuah data yang bersifat kritis ingin dimanipulasi oleh sebuah proses, ia harus mengambil lock untuk data tersebut terlebih dahulu. Setelah lock untuk data tersebut sudah diambil sebuah proses, proses lain yang ingin memanipulasi harus menunggu sampai lock yang dipegang proses pertama sudah dilepas.
- Semaphore: Sebuah semaphore mirip konsepnya dengan mutex, namun tidak terbatas pada status biner true or false saja, melainkan dalam bentuk counter. Sebuah semaphore akan dimulai dengan N buah resource, lalu jika ada sebuah proses yang ingin memakai resource tersebut, jumlah resource berkurang menjadi N - 1. Ketika jumlah resource bernilai 0, proses lain yang ingin memakai resource harus menunggu sampai terdapat proses yang melepaskan resourcennya dan counternya berada di atas 0 kembali.
- Monitor: Sebuah monitor adalah ekstensi dari mutex yang mengenkapsulasi resource dan metode sinkronisasinya dalam sebuah kelas, sehingga proses luar yang ingin memanipulasi data tidak perlu pusing melakukan acquire dan release lock, melainkan hal tersebut sudah otomatis dilakukan oleh kelas monitor.

6. (2.5 poin) Jelaskan apa itu **blockchain** dan bagaimana cara kerjanya. Sebutkan dan jelaskan tiga pemanfaatan **blockchain** di luar **cryptocurrency**.

Blockchain adalah teknologi penyimpanan transaksi yang terdesentralisasi, dengan cara mendistribusikan sejarah transaksi yang sama ke sebuah jaringan komputer yang besar, dengan masing-masing komputer menyimpan sejarah transaksi yang sama. Dengan sejarah transaksi yang disimpan di banyak komputer, cobaan untuk memanipulasi sejarah transaksi menjadi jauh lebih sulit, karena setiap komputer harus diubah sejarah transaksinya.

Blockchain bekerja dengan konsep sebuah block, yang menyimpan daftar transaksi dan memiliki sebuah nilai hash yang unik. Block tersebut juga menyimpan hash dari blok sebelumnya. Hal ini menyebabkan sebuah kumpulan block seperti sebuah *linked list* atau *chain*.

Penambahan sebuah block pada blockchain membutuhkan sebuah mekanisme seperti Proof of Work, yaitu bukti kerja yang membutuhkan sejumlah waktu tertentu. Sistem ini membuat penambahan block pada blockchain dapat terkontrol sehingga data yang tersimpan pada masing-masing node tetap konsisten. Ketika sebuah block sudah ditambahkan ke blockchain, perubahan block tersebut perlu diikuti dengan perubahan block-block lain pada chain, yang menyebabkan memanipulasi data sangat sulit.

Contoh penggunaan blockchain di dunia nyata antara lain:

- Supply Chain Management: Blockchain digunakan untuk menyimpan asal, perpindahan, dan perlakuan terhadap produk pada supply chain. Penggunaan blockchain menambah transparency dan mengurangi kemungkinan terjadinya pemalsuan data dan error terhadap penyimpanan sejarah produk.
- Healthcare: Blockchain dapat digunakan untuk menyimpan sejarah pasien rumah sakit, sehingga data pasien terjaga dari segi privacy dan integrity.
- Voting: Blockchain dapat digunakan untuk sistem pemungutan suara, mengurangi risiko terjadinya pemalsuan data suara dan memastikan setiap suara terdata dan tersimpan secara akurat.

**7. (2.5 poin) Jelaskanlah mengenai hirarki memori pada CUDA dan tentukan apakah beberapa hal berikut mungkin dilakukan, beserta alasannya:**

CUDA memanfaatkan arsitektur dari GPU NVIDIA untuk melakukan *massively-parallel computations*. Pada GPU NVIDIA terdapat sebuah hirarki satuan untuk mengeksekusi instruksi. Unit eksekusi terkecil pada GPU adalah sebuah thread yang dapat melaksanakan sebuah instruksi. Sekumpulan thread lalu dikelompokkan menjadi sebuah block, dan sekumpulan block dikelompokkan menjadi sebuah grid.

Pada arsitektur tersebut terdapat beberapa jenis memori yang berbeda untuk mengakomodasi sejumlah unit eksekusi yang berbeda. Jenis memori tersebut dari unit terkecil sampai unit terbesar adalah:

- Register: Unit memori terkecil dan tercepat, hanya dapat diakses oleh satu thread saja dan memiliki lifetime sampai thread tersebut selesai digunakan. Umumnya digunakan untuk menyimpan variabel lokal untuk sebuah thread.
  - Local memory: Sebuah unit memori yang masih bersifat eksklusif terhadap satu thread, dan juga akan hangus ketika thread yang bersangkutan selesai eksekusi. Ukuran penyimpanan local memory lebih besar dari register, namun memiliki waktu akses yang lebih lambat dari register.
  - Shared memory: Unit memori yang berkorespondensi dengan sebuah block. Seluruh thread pada block tersebut dapat mengakses unit memori ini. Unit ini memiliki lifetime sampai block yang bersangkutan selesai eksekusi, dan umumnya digunakan untuk komunikasi antar-thread.
  - Global memory: Unit memori terbesar yang dapat diakses oleh thread manapun pada GPU. Unit ini memiliki lifetime sampai program yang memanggilnya selesai eksekusi. Karena ukurannya yang cukup besar, tipe memori ini memiliki kelemahan kecepatan akses yang paling lambat.
- a. Akses *local memory* dari *thread* yang berbeda: Tidak bisa. Local memory hanya bisa diakses oleh satu buah thread saja. Thread manapun selain yang memiliki local memory tersebut tidak bisa menggunakan unit memori tersebut.

- b. Akses *shared memory* dari *block* yang berbeda: Tidak bisa. Sebuah *shared memory* hanya berkorespondensi pada satu buah *block*, dan hanya bisa digunakan oleh thread yang ada di *block* tersebut. *Block* lain ataupun thread pada *block* lain tidak bisa mengakses *shared memory* tersebut.
- c. Akses *shared memory* dari *warp* yang berbeda: Bisa, jika kedua *warp* berada pada *block* yang sama. Sebuah *block* dapat memiliki beberapa *warp*, sehingga hal ini memungkinkan.
- d. Akses *global memory* dari *grid* yang sama: Bisa. *Global memory* dapat diakses oleh thread manapun, sehingga akses ke *global memory* akan selalu bisa dilakukan.

### Referensi

[1] J. Holloway, "CUDA Memory Hierarchy Explained," YouTube, 06-May-2019. [Online]. Available: [https://www.youtube.com/watch?v=PJCISyoGpug&list=PLC6u37oFvF40BAm7gwVP7uDzwmW83yHPe&index=6&ab\\_channel=JoshHolloway](https://www.youtube.com/watch?v=PJCISyoGpug&list=PLC6u37oFvF40BAm7gwVP7uDzwmW83yHPe&index=6&ab_channel=JoshHolloway). [Accessed: 20-Jul-2024].

**8. (2.5 poin) Pada *programming* sistem terdistribusi, terdapat 8 *common fallacy*. Jelaskan semuanya!**

- The network is reliable: Fallacy ini mengasumsikan sebuah jaringan selalu dapat dipercaya dan pesan yang dikirim akan selalu lolos tanpa masalah. Kenyataannya adalah sebuah jaringan akan mengalami kegagalan, kehilangan paket, delay, dan masalah-masalah lain. Sehingga selain membangun jaringan, perlu dilakukan usaha untuk fault-correcting seperti retransmission dan error recovery.
- Latency is zero: Fallacy ini mengacu pada asumsi bahwa latency jaringan tidak akan mempengaruhi performa sistem. Kenyataannya latency dapat berpengaruh signifikan, terutama pada sistem terdistribusi yang tersebar secara geografis.
- Bandwidth is infinite: Fallacy ini mengasumsikan terdapat bandwidth tak terbatas untuk transfer data pada jaringan. Kenyataannya adalah bandwidth merupakan resource yang terbatas sehingga sistem harus didesain untuk memanage bandwidth secara efektif menggunakan teknik seperti kompresi data, data format efisien, dan pembatasan transfer data.
- The network is secure: Fallacy ini mengacu pada asumsi bahwa jaringan bersifat aman dan terjaga dari malicious party. Kenyataannya sebuah jaringan akan terekspos ke banyak vulnerabilities dan ancaman keamanan seperti interception, tampering, dan serangan jaringan. Sehingga selain membuat jaringan yang efisien, jaringan harus dibuat aman dan sulit dibobol.
- Topology doesn't change: Fallacy ini mengacu pada pemikiran bahwa bentuk dari jaringan tidak berubah terhadap waktu. Kenyataannya sebuah jaringan dapat berubah melalui kegagalan simpul, scaling, dan reconfiguration.

- There is one administrator: Fallacy ini mengasumsikan hanya terdapat satu administrator yang bertanggungjawab atas keseluruhan jaringan. Kenyataannya adalah sebuah sistem terdistribusi akan tersebar ke banyak domain dengan cara kerja yang berbeda-beda.
  - Transport cost is zero: Fallacy ini mengacu pada pemikiran bahwa biaya untuk transfer data pada jaringan tidak signifikan. Kenyataannya penggunaan bandwidth, energi, dan lain-lain akan mengeluarkan biaya sehingga segi tersebut juga harus diperhitungkan.
  - The network is homogenous: Fallacy ini mengasumsikan bahwa sebuah jaringan bersifat homogen, sehingga seluruh komponen memiliki karakteristik yang sama dalam segi software, hardware, dan konfigurasi. Kenyataannya sebuah sistem terdistribusi akan bersifat heterogen, terbangun dari berbagai macam hardware, software, sistem operasi, dan konfigurasi.
- 9. (2.5 poin) Pada saat membangun sebuah sistem terdistribusi, terdapat sebuah teorema CAP. Apa itu teorema CAP? Sebutkan dan jelaskan secara singkat serta berikan contoh jenis-jenis sistemnya!**

Teorema CAP, kadang disebut Teorema Brewer, merupakan sebuah konsep pada sistem terdistribusi yang mengatakan bahwa sebuah sistem penyimpanan data terdistribusi tidak mungkin memiliki ketiga hal tersebut secara sekaligus:

- Consistency: Every read receives the most recent write or an error, sehingga setiap simpul pada sistem akan melihat data yang sama pada waktu yang sama. Jika sebuah sistem konsisten, maka operasi pembacaan akan selalu mengembalikan penulisan terakhir.
- Availability: Every request receives a non-error response, without guarantee that it contains the most recent write. Konsep ini memastikan bahwa setiap operasi akan selalu mendapatkan respons, tanpa kepastian bahwa data merupakan data terbaru.
- Partition Tolerance: The system continues to operate despite arbitrary partitioning due to network failures. Konsep ini pada sebuah sistem terdistribusi mengatakan bahwa sebuah kerusakan pada jaringan yang menyebabkan pemisahan bagian jaringan tidak akan menyebabkan kegagalan keseluruhan jaringan.

Teorema ini mengatakan bahwa sebuah sistem data terdistribusi tidak mungkin memiliki ketiga hal tersebut, sehingga terdapat tiga tipe sistem data terdistribusi menurut teorema CAP:

- Sistem CA (Consistency and Availability): Sistem yang menjamin consistency dan availability namun tidak dapat menerima network partition, contohnya sebuah SQL database yang berjalan pada satu server. Sebuah sistem terdistribusi umumnya tidak terlepas dari network partition, sehingga sistem yang tidak dapat menerima hal tersebut tidak bisa terdistribusikan.

- Sistem AP (Availability dan Partition Tolerance): Sistem yang mengorbankan consistency. Contohnya Cassandra, sebuah NoSQL database yang menjamin availability tinggi dan partition tolerance, serta consistency yang tidak instan namun dapat dicapai dalam sejumlah waktu tertentu.
- Sistem CP (Consistency dan Partition Tolerance): Sebuah sistem yang mengorbankan availability. Contohnya adalah HBase, sebuah database terdistribusi yang didesain untuk tabel berukuran besar dan menjamin consistency kuat dan partition tolerance.

**10. (2.5 poin) Jelaskan apa itu Kubernetes, dan mengapa ia seringkali dibutuhkan dalam pembangunan dan pengelolaan sistem terdistribusi.**

Kubernetes merupakan sebuah platform open-source untuk pengelolaan sistem terdistribusi. Kubernetes mempermudah pengelolaan banyak host dengan mengotomatisasi deployment, scaling, management dan operation terhadap containerized application yang terdistribusi.

Kubernetes bekerja dengan sebuah Control Plane yang mengatur sistem secara keseluruhan. Control plane memiliki sebuah API Server yang bertugas untuk berkomunikasi dengan worker node dan membagikan kerja secara rata ke worker node. Sebuah worker node adalah sebuah mesin yang menjalankan kerja yang diberikan control plane.

Control plane akan membagikan kerja kepada worker node dalam bentuk sebuah Pod. Pod adalah unit terkecil yang bisa dibagikan oleh Kubernetes. Sebuah pod tersusun atas satu atau lebih container. Sebuah container adalah sebuah aplikasi yang dibungkus dengan dependency yang dibutuhkannya.

Control plane akan berkomunikasi dengan worker node melalui alat bernama kubelet yang berada pada seluruh worker node. Control plane akan memberikan instruksi terkait pod yang dijalankan pada node, dan memastikan bahwa pod dijalankan secara diinginkan. Selain kubelet, terdapat container runtime pada worker node yang menjalankan container pada pod. Terakhir ada kube-proxy yang berjalan pada worker node yang mengatur traffic dan load-balancing pada pod di worker node.

Kubernetes memberikan banyak manfaat yang sangat membantu dalam pengelolaan sistem distribusi, antara lain:

- Automated deployment: Kubernetes meng-automate container deployment dan memastikan bahwa aplikasi ter-setup secara benar dan konsisten pada worker node.
- Skalabilitas: Kubernetes dapat melakukan scale-up dan scale-down aplikasi secara otomatis untuk menyesuaikan dengan workload pada sistem.
- Fault-tolerance dan high availability: Kubernetes melakukan monitor atas kesehatan container dan dapat melakukan restart otomatis atau menggantikan

container yang gagal. Kubernetes juga memastikan aplikasi terdistribusi terhadap banyak node untuk menghindari downtime.

- Load balancing: Kubernetes memastikan bahwa traffic pada jaringan tersebar secara merata kepada seluruh bagian aplikasi, sehingga tidak ada satu bagian yang ter-overwhelmed.
- Resource management: Kubernetes dapat mengoptimasi penggunaan resource pada cluster, sehingga memastikan bahwa container mendapatkan jumlah CPU dan memory yang tepat.

## Referensi

[1] ByteByteGo, "Kubernetes Explained Simply," YouTube, 11-Jan-2023. [Online]. Available:  
[https://www.youtube.com/watch?v=TIHvYWVUZyc&t=93s&ab\\_channel=ByteByteGo](https://www.youtube.com/watch?v=TIHvYWVUZyc&t=93s&ab_channel=ByteByteGo). [Accessed: 20-Jul-2024].

### 11. (2.5 poin) Pernahkah kalian mendengar istilah!

- a. ***Database replication?*** Jelaskan secara singkat!

Database replication adalah proses mengcopy objek-objek database seperti tabel pada banyak database di sebuah sistem terdistribusi. Replikasi ini menjamin bahwa data yang sama terdapat pada banyak lokasi sehingga meningkatkan availability, reliability, fault tolerance, dan accessibility terhadap data. Replikasi ini dapat dilakukan secara real-time (synchronous) maupun setelah transaksi selesai (asynchronous).

- b. CockroachDB merupakan salah satu *provider* yang menyediakan *distributed SQL database*. Mereka menjamin bahwa *replicated database* yang mereka sediakan akan terjamin konsisten. Mengapa mereka yakin bahwa *database* mereka akan selalu konsisten meskipun pada kondisi *server* yang sangat buruk?

CockroachDB mempertahankan konsistensi data menggunakan sebuah metode yaitu Raft Consensus Algorithm. Algoritma ini bekerja dengan membuat sebuah *replication log* yang dibagikan ke banyak node. Algoritma ini memastikan bahwa setiap node menyimpan sekuens operasi yang sama.

Pada algoritma Raft, sebuah node ditetapkan sebagai leader, dan memiliki tugas untuk melakukan log replication. Node-node lain adalah follower yang akan mereplikasikan log dari leader node. Setiap sejumlah durasi waktu, leader baru akan dipilih dari node-node yang ada.

Ketika terdapat sebuah request, leader node akan memasukkan request tersebut ke log. Entri tersebut lalu akan diberikan ke followersnya. Para followers akan memasukkan entri tersebut ke log mereka, dan mengirimkan

acknowledgement ke leader. Setelah mayoritas sudah mengirimkan acknowledgement, entri tersebut di-commit dan terlihat oleh klien.

**c. Jelaskan cara kerja protokol yang mereka gunakan dalam implementasi database sistem terdistribusi.**

CockroachDB mengimplementasikan databasenya dengan melakukan splitting terhadap data menjadi banyak range. Masing-masing range tersebut merupakan sebuah Raft group yang tersusun atas beberapa replika yang tersebar pada banyak node. Salah satu replika dibuat sebagai leader, yang akan memanage penulisan dan mengatur replikasi.

Sebuah operasi write akan ditujukan kepada leader yang memasukkan operasinya ke log dan mengirimkan entri tersebut ke followernya. Setelah para follower sudah mereplika dan mengirimkan acknowledgement, entri tersebut ditulis pada database. Operasi read umumnya akan diberikan oleh leader untuk memastikan konsistensi.

**12. (2.5 poin) Jelaskan konsep skalabilitas dalam sistem terdistribusi. Sebutkan strategi yang biasanya digunakan untuk meningkatkan skalabilitas suatu sistem terdistribusi.**

Skalabilitas pada sistem terdistribusi mengacu pada kemampuan sebuah sistem untuk menangani jumlah kerja yang banyak, atau kemampuannya untuk diperbesar untuk menangani peningkatan tersebut. Sebuah sistem dengan skalabilitas yang baik dapat menahan atau meningkatkan performa walaupun jumlah kerjanya meningkat, umumnya dengan menambahkan resource seperti penambahan nodes, storage, ataupun processing power.

Skalabilitas sistem terdistribusi terbagi menjadi dua, yaitu horizontal scalability dan vertical scalability. Skalabilitas horizontal mengacu pada penambahan node pada sistem untuk mendistribusikan kerja, sedangkan skalabilitas vertikal mengacu pada peningkatan kapasitas dari sebuah node seperti dengan menambah CPU atau memori.

Beberapa strategi yang umumnya digunakan untuk meningkatkan skalabilitas antara lain:

- Load balancing: Mendistribusikan kerja ke banyak node menggunakan teknik seperti round-robin atau least connection. Contohnya adalah mendistribusikan HTTP request ke banyak server.
- Caching: Menyimpan data yang sering diakses ke temporary storage untuk mengurangi kerja yang ditujukan kepada primary data store.
- Sharding: Mempartisi sebuah database menjadi bagian-bagian kecil yang lebih manageable dan terdistribusi ke node-node yang berbeda.
- Replikasi: Menduplikasi data terhadap banyak node untuk meningkatkan performance dan availability.

- Auto-scaling: Melakukan manajemen node yang bekerja secara otomatis berdasarkan demand kerja pada waktu tertentu. Hal ini dapat dilakukan menggunakan alat auto-scaling seperti AWS Auto Scaling atau Kubernetes.

**13. (5.0 poin) Jelaskan perbedaan dan cara kerja serta ilustrasinya dari :**

a. **MPI\_Send dan MPI\_Recv.**

**MPI\_Send**

MPI\_Send digunakan untuk mengirim pesan dari sebuah proses ke proses lainnya.

```
int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag,
MPI_Comm comm);
```

Cara kerja MPI\_Send adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- buf: Pointer ke buffer data yang ingin dikirim.
- count: Jumlah data yang ingin dikirim.
- datatype: tipe data yang dikirim (MPI\_INT, MPI\_FLOAT, dst.).
- dest: rank dari proses tujuan pada communicator.
- tag: Untuk mengidentifikasi pesan.
- comm: communicator handle (MPI\_COMM\_WORLD) yang mendefinisikan proses yang terkait pada komunikasi.

Ilustrasi penggunaan MPI\_Send:

```
int data = 100;
MPI_Send(&data, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
```

Dalam contoh ini, proses mengirimkan data integer dengan nilai 100 ke proses dengan peringkat 1 di komunikator MPI\_COMM\_WORLD.

**MPI\_Recv**

MPI\_Recv digunakan untuk menerima pesan dari proses lain. Proses penerima menentukan buffer untuk menyimpan data masuk, proses sumber, dan parameter relevan lainnya.

```
int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag,
MPI_Comm comm, MPI_Status* status);
```

Cara kerja MPI\_Recv adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- buf: Pointer ke buffer data tempat data yang diterima akan disimpan.
- count: Jumlah maksimum elemen data yang dapat diterima.
- datatype: Tipe data elemen.
- source: Peringkat proses sumber dalam komunikator. MPI\_ANY\_SOURCE dapat digunakan untuk menerima dari proses apa pun.
- tag: Tag pesan yang digunakan untuk mengidentifikasi pesan. MPI\_ANY\_TAG dapat digunakan untuk menerima pesan dengan tag apa pun.
- comm: Handle komunikator.
- status: Penunjuk ke objek MPI\_Status yang berisi informasi tentang pesan yang diterima, seperti sumber, tag, dan jumlah elemen yang diterima sebenarnya.

Ilustrasi penggunaan MPI\_Recv:

```
int data;
MPI_Status status;
MPI_Recv(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &status);
```

Dalam contoh ini, proses menerima data integer dari proses dengan peringkat 0 di komunikator MPI\_COMM\_WORLD. Objek status MPI\_Status akan berisi informasi tentang pesan yang diterima.

### Perbedaan MPI\_Send dan MPI\_Recv

MPI\_Send digunakan untuk mengirim data ke proses, sedangkan MPI\_Recv digunakan untuk menerima data dari sebuah proses.

#### b. MPI\_Bcast dan MPI\_Scatter.

##### **MPI\_Bcast**

MPI\_Bcast adalah operasi komunikasi kolektif yang digunakan untuk menyebarkan pesan dari satu proses (root) ke semua proses lain dalam satu communicator.

```
int MPI_Bcast(void* buf, int count, MPI_Datatype datatype, int root,
MPI_Comm comm);
```

Cara kerja MPI\_Bcast adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- buf: Pointer ke buffer data yang akan dikirim atau diterima.
- count: Jumlah elemen data.
- datatype: Tipe data dari elemen (misalnya, MPI\_INT, MPI\_FLOAT).
- root: Rank dari proses root yang mengirimkan data.
- comm: Komunikator (misalnya, MPI\_COMM\_WORLD).

Ilustrasi penggunaan MPI\_Bcast:

```
// Misalkan ada 4 proses dalam communicator MPI_COMM_WORLD, dan proses dengan rank 0 adalah root:  
  
int data = 100;  
MPI_Bcast(&data, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

Dalam contoh ini, proses dengan rank 0 mengirimkan integer data dengan nilai 100 ke semua proses lain dalam communicator MPI\_COMM\_WORLD.

### **MPI\_Scatter**

MPI\_Scatter adalah operasi komunikasi kolektif yang digunakan untuk mendistribusikan blok-blok data yang berbeda dari satu proses (root) ke semua proses lain dalam satu communicator.

```
int MPI_Scatter(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm);
```

Cara kerja MPI\_Scatter adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- sendbuf: Pointer ke buffer pengirim (di proses root).
- sendcount: Jumlah elemen yang dikirim ke setiap proses.
- sendtype: Tipe data dari elemen pengirim.
- recvbuf: Pointer ke buffer penerima.
- recvcount: Jumlah elemen yang diterima oleh setiap proses.
- recvtype: Tipe data dari elemen penerima.
- root: Rank dari proses root yang mengirimkan data.
- comm: Komunikator.

Ilustrasi penggunaan MPI\_Scatter:

```
// Misalkan ada 4 proses dalam communicator MPI_COMM_WORLD, dan proses dengan rank 0 adalah root yang memiliki array data untuk dikirim:
```

```
int send_data[4] = {1, 2, 3, 4};  
int recv_data;  
MPI_Scatter(send_data, 1, MPI_INT, &recv_data, 1, MPI_INT, 0,  
MPI_COMM_WORLD);
```

Dalam contoh ini, proses root (rank 0) memiliki array send\_data dengan nilai {1, 2, 3, 4}. Setiap proses lain menerima satu elemen dari array tersebut. Misalnya, proses dengan rank 1 menerima 2, proses dengan rank 2 menerima 3, dan seterusnya.

### Perbedaan MPI\_Bcast dan MPI\_Scatter

MPI\_Bcast mengirim pesan yang sama ke semua proses lain dalam communicator. Sedangkan MPI\_Scatter mendistribusikan data yang berbeda ke proses lain dalam communicator. Setiap proses pada MPI\_Scatter akan mendapatkan bagian berbeda dari data yang dikirim.

### c. MPI\_Reduce dan MPI\_Allreduce.

#### **MPI\_Reduce**

MPI\_Reduce menggabungkan nilai-nilai dari semua proses dalam sebuah communicator dan mengembalikan hasilnya ke satu proses root.

```
int MPI_Reduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype  
datatype, MPI_Op op, int root, MPI_Comm comm);
```

Cara kerja MPI\_Reduce adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- sendbuf: Pointer ke buffer pengirim yang berisi data lokal dari setiap proses.
- recvbuf: Pointer ke buffer penerima yang akan menyimpan hasil pengurangan (hanya relevan di proses root).
- count: Jumlah elemen dalam buffer.
- datatype: Tipe data dari elemen (misalnya, MPI\_INT, MPI\_FLOAT).
- op: Operasi pengurangan (misalnya, MPI\_SUM, MPI\_MAX).
- root: Rank dari proses root yang akan menerima hasil.
- comm: Komunikator.

Ilustrasi penggunaan MPI\_Reduce:

```
// Misalkan kita memiliki 4 proses dalam communicator MPI_COMM_WORLD,  
dan kita ingin menjumlahkan nilai integer dari semua proses:  
  
int local_data = 1; // Misalnya, setiap proses memiliki nilai 1  
int global_sum;  
MPI_Reduce(&local_data, &global_sum, 1, MPI_INT, MPI_SUM, 0,  
MPI_COMM_WORLD);
```

Dalam contoh ini, jika ada 4 proses dan setiap proses memiliki local\_data bernilai 1, proses root (rank 0) akan menerima global\_sum dengan nilai 4.

### **MPI\_Allreduce**

MPI\_Allreduce mirip dengan MPI\_Reduce, tetapi hasil pengurangan dikirimkan kembali ke semua proses, bukan hanya ke satu proses root.

```
int MPI_Allreduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype  
datatype, MPI_Op op, MPI_Comm comm);
```

Cara kerja MPI\_Allreduce adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- sendbuf: Pointer ke buffer pengirim yang berisi data lokal dari setiap proses.
- recvbuf: Pointer ke buffer penerima yang akan menyimpan hasil pengurangan di semua proses.
- count: Jumlah elemen dalam buffer.
- datatype: Tipe data dari elemen.
- op: Operasi pengurangan.
- comm: Komunikator.

### Ilustrasi penggunaan MPI\_Allreduce

```
// Misalkan kita memiliki 4 proses dalam communicator MPI_COMM_WORLD,  
dan kita ingin menjumlahkan nilai integer dari semua proses dan  
mendistribusikan hasilnya ke semua proses:  
  
int local_data = 1; // Misalnya, setiap proses memiliki nilai 1  
int global_sum;  
MPI_Allreduce(&local_data, &global_sum, 1, MPI_INT, MPI_SUM,  
MPI_COMM_WORLD);
```

Dalam contoh ini, jika ada 4 proses dan setiap proses memiliki local\_data bernilai 1, semua proses akan menerima global\_sum dengan nilai 4.

### **Perbedaan MPI\_Reduce dan MPI\_Allreduce**

Hasil pengurangan setelah MPI\_Reduce hanya dikirim ke satu proses root yang ditentukan, melainkan hasil pengurangan MPI\_Allreduce akan dikirim ke semua proses pada communicator.

#### **d. MPI\_Gather dan MPI\_Allgather.**

##### **MPI\_Gather**

MPI\_Gather mengumpulkan data dari semua proses dalam sebuah communicator dan mengirimkannya ke satu proses root.

```
int MPI_Gather(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm);
```

Cara kerja MPI\_Gather adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- sendbuf: Pointer ke buffer pengirim yang berisi data lokal dari setiap proses.
- sendcount: Jumlah elemen yang dikirim dari setiap proses.
- sendtype: Tipe data dari elemen yang dikirim.
- recvbuf: Pointer ke buffer penerima (hanya relevan di proses root).
- recvcount: Jumlah elemen yang diterima oleh proses root dari setiap proses.
- recvtype: Tipe data dari elemen yang diterima.
- root: Rank dari proses root yang akan menerima data.
- comm: Komunikator.

Ilustrasi penggunaan MPI\_Gather:

```
// Misalkan kita memiliki 4 proses dalam communicator MPI_COMM_WORLD, dan proses dengan rank 0 adalah root:  
  
int send_data = rank; // Setiap proses mengirimkan rank mereka  
int recv_data[4]; // Buffer penerima di proses root  
MPI_Gather(&send_data, 1, MPI_INT, recv_data, 1, MPI_INT, 0,  
MPI_COMM_WORLD);
```

Dalam contoh ini, setiap proses mengirimkan nilai send\_data (yang diinisialisasi dengan rank mereka) ke proses root. Proses root mengumpulkan nilai-nilai ini dalam array recv\_data.

##### **MPI\_Allgather**

MPI\_Allgather mirip dengan MPI\_Gather, tetapi hasil pengumpulan data dikirimkan kembali ke semua proses, bukan hanya ke satu proses root.

```
int MPI_Allgather(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm);
```

Cara kerja MPI\_Allgather adalah dengan memanggil fungsinya dengan penjelasan parameter sebagai berikut:

- sendbuf: Pointer ke buffer pengirim yang berisi data lokal dari setiap proses.
- sendcount: Jumlah elemen yang dikirim dari setiap proses.
- sendtype: Tipe data dari elemen yang dikirim.
- recvbuf: Pointer ke buffer penerima yang akan menyimpan hasil pengumpulan di semua proses.
- recvcount: Jumlah elemen yang diterima oleh setiap proses dari setiap proses lain.
- recvtype: Tipe data dari elemen yang diterima.
- comm: Komunikator.

Ilustrasi penggunaan MPI\_Allgather:

```
// Misalkan kita memiliki 4 proses dalam communicator MPI_COMM_WORLD, dan setiap proses ingin mengumpulkan data dari semua proses lain:  
  
int send_data = rank; // Setiap proses mengirimkan rank mereka  
int recv_data[4]; // Buffer penerima di semua proses  
MPI_Allgather(&send_data, 1, MPI_INT, recv_data, 1, MPI_INT,  
MPI_COMM_WORLD);
```

Dalam contoh ini, setiap proses mengirimkan nilai send\_data (yang diinisialisasi dengan rank mereka) dan menerima hasil pengumpulan data dari semua proses dalam array recv\_data.

### **Perbedaan MPI\_Gather dan MPI\_Allgather**

Hasil pengumpulan MPI\_Gather hanya dikirim ke satu proses yang ditentukan, sedangkan hasil pengumpulan MPI\_Allgather dikirim ke semua proses dalam communicator.

**14. (5.0 poin) Tantangan utama pada sistem terdistribusi yaitu bagaimana cara untuk setiap replika ataupun *nodes* sepakat satu sama lain.**

- a. Apa nama algoritma yang menyelesaikan permasalahan ini? Dan jelaskan!

Algoritma untuk melakukan hal tersebut adalah algoritma konsensus. Algoritma konsensus adalah mekanisme dalam sistem terdistribusi yang digunakan untuk mencapai kesepakatan di antara beberapa node atau proses yang beroperasi secara bersamaan untuk memastikan bahwa mereka menyetujui keadaan atau keputusan yang sama.

Dalam sistem terdistribusi, node-node mungkin memiliki salinan data yang berbeda atau mengalami kegagalan, sehingga mencapai konsensus menjadi krusial untuk menjaga konsistensi dan integritas sistem secara keseluruhan. Algoritma konsensus memastikan bahwa meskipun beberapa node mungkin mengalami kegagalan atau ada latensi jaringan, semua node yang berfungsi tetap sepakat tentang hasil dari operasi tertentu, seperti penulisan data atau pengambilan keputusan.

- b. **Semakin majunya perkembangan zaman, munculah algoritma-algoritma baru untuk menyelesaikan permasalahan ini. Dahulu terdapat Paxos dan sekarang muncul Raft dan Zab. Jelaskan bagaimana ketiga algoritma tersebut bekerja dan berikan contoh mengapa algoritma tersebut bekerja serta berikan contoh kasus dimana algoritma tersebut berhasil menyelesaikan permasalahan ini pada jaringan yang buruk!**

### Paxos

Algoritma Paxos adalah sebuah algoritma konsensus yang dirancang untuk memastikan bahwa sekelompok node terdistribusi dapat mencapai kesepakatan tentang nilai tertentu meskipun beberapa node mungkin gagal atau ada latensi jaringan.

Algoritma ini terdiri dari tiga komponen utama: proposers (pengusul), acceptors (penerima), dan learners (pembelajar). Proposers mengusulkan nilai untuk diterima, acceptors kemudian menentukan apakah mereka menerima nilai tersebut, dan learners akhirnya mendapatkan hasil dari kesepakatan yang dicapai.

Proses Paxos dimulai dengan proposers mengajukan proposal dengan nomor unik kepada acceptors. Acceptors, jika belum menyetujui proposal dengan nomor yang lebih tinggi, akan menerima proposal tersebut dan memberikan balasan. Proses ini memastikan bahwa hanya proposal dengan nomor tertinggi yang dapat diterima oleh mayoritas acceptors, sehingga mencapai kesepakatan yang konsisten.

Paxos bekerja karena menggunakan mekanisme nomor proposal dan pengakuan dari mayoritas untuk mencegah konflik dan memastikan bahwa semua node yang berfungsi setuju dengan nilai yang sama, meskipun ada beberapa node yang gagal atau mengalami masalah komunikasi.

Contoh penerapan Paxos dalam jaringan yang buruk adalah dalam sistem database terdistribusi yang membutuhkan keputusan konsensus untuk menentukan nilai baru dari data. Misalnya, dalam sebuah sistem penyimpanan terdistribusi, Paxos dapat digunakan untuk memastikan bahwa meskipun beberapa node tidak dapat berkomunikasi karena latensi atau kegagalan jaringan, nilai terakhir yang ditulis ke database akan tetap konsisten di semua node yang tersedia. Jika satu node gagal mengirimkan proposal atau menerima balasan, node lain yang berfungsi akan tetap melanjutkan proses konsensus, menjamin bahwa sistem secara keseluruhan mencapai kesepakatan yang valid dan konsisten mengenai nilai data, meskipun ada gangguan dalam jaringan. Dengan demikian, Paxos memberikan jaminan konsistensi di seluruh node terdistribusi bahkan dalam kondisi jaringan yang tidak ideal.

## Raft

Raft membagi proses konsensus menjadi tiga komponen utama: pemilihan pemimpin, replikasi log, dan keamanan log. Dalam Raft, salah satu node dipilih sebagai pemimpin (leader) yang bertanggung jawab untuk menangani semua permintaan penulisan dan mendistribusikannya ke node lainnya, yang dikenal sebagai pengikut (followers). Pemimpin ini juga bertugas untuk mengelola log yang menyimpan urutan operasi yang dilakukan.

Node-node pengikut akan mereplikasi log dari pemimpin mereka dan mengonfirmasi bahwa mereka telah menerima entri log yang sama. Untuk memastikan kesepakatan, Raft memerlukan mayoritas node (quorum) untuk menyetujui perubahan sebelum mereka diterapkan. Jika pemimpin gagal, proses pemilihan pemimpin baru dimulai, dan node lain dapat dipilih sebagai pemimpin baru, menggantikan pemimpin yang gagal, sambil tetap menjaga konsistensi log di seluruh node.

Raft bekerja dengan efektif karena ia mengorganisir proses konsensus dengan cara yang terstruktur dan jelas, memisahkan tanggung jawab pemilihan pemimpin dan replikasi log, serta menggunakan mayoritas untuk mencapai kesepakatan. Dalam jaringan yang buruk, di mana node mungkin mengalami kegagalan atau latensi tinggi, Raft mampu mengatasi masalah ini dengan cara mengandalkan quorum untuk memastikan bahwa perubahan log hanya diterima jika sebagian besar node setuju.

Sebagai contoh, dalam sebuah sistem penyimpanan terdistribusi yang menggunakan Raft, meskipun ada beberapa node yang mengalami latensi atau tidak dapat berkomunikasi dengan node lain karena gangguan jaringan, sistem masih dapat melanjutkan operasi dengan memilih pemimpin baru jika yang lama gagal, dan memastikan bahwa log yang direplikasi tetap konsisten di seluruh node yang berfungsi. Ini memungkinkan sistem untuk tetap operasional dan konsisten meskipun dalam kondisi jaringan yang tidak stabil.

## **Zab**

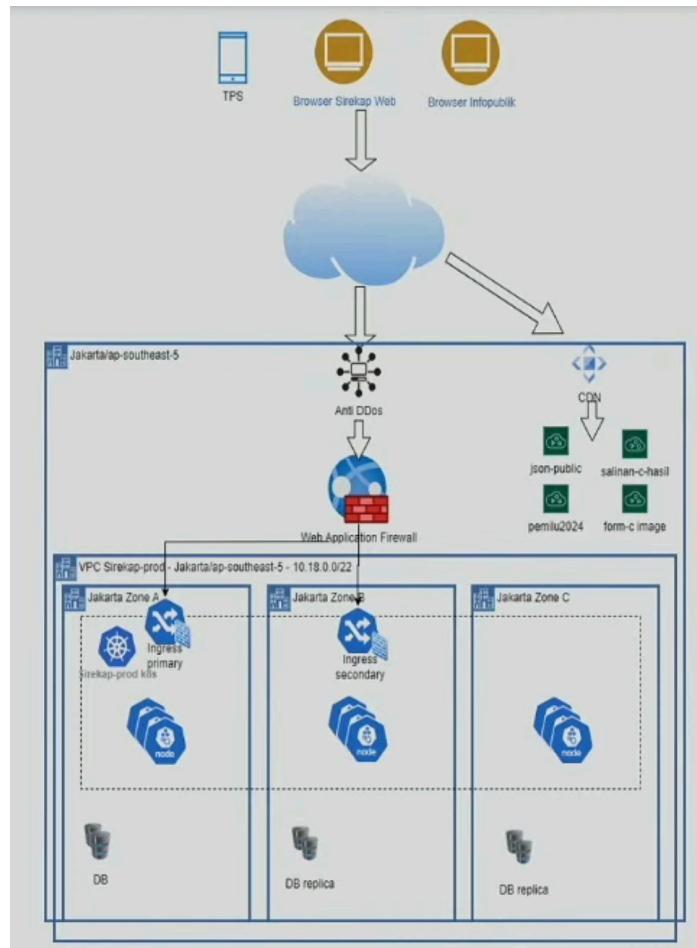
Algoritma Zab (Zookeeper Atomic Broadcast) adalah algoritma konsensus yang dirancang untuk menyediakan layanan broadcast yang konsisten dalam sistem terdistribusi, dan merupakan inti dari Zookeeper, sebuah sistem koordinasi terdistribusi yang digunakan untuk manajemen konfigurasi dan sinkronisasi. Zab berfungsi untuk memastikan bahwa semua node dalam sistem terdistribusi menerima dan memproses pesan dalam urutan yang sama, meskipun ada kegagalan atau latensi jaringan.

Algoritma ini bekerja dengan cara memilih seorang pemimpin (leader) yang bertanggung jawab untuk mengusulkan perubahan dan mengelola proses broadcast pesan. Node lainnya, yang disebut follower, menerima dan meneruskan pesan yang disiarkan oleh pemimpin. Untuk mencapai konsensus, Zab memerlukan mayoritas node (quorum) untuk menyetujui pesan sebelum diterapkan, mirip dengan pendekatan quorum pada algoritma lain seperti Raft.

Zab bekerja dengan efektif karena ia menggunakan pendekatan yang berfokus pada pemilihan pemimpin dan memastikan bahwa pesan yang disiarkan oleh pemimpin diurutkan dan diterima oleh mayoritas node. Jika pemimpin gagal, Zab memulai proses pemilihan pemimpin baru untuk menggantikannya yang lama, sehingga sistem dapat melanjutkan operasinya dengan gangguan minimal. Dalam jaringan yang buruk, di mana node mungkin mengalami latensi tinggi atau kegagalan komunikasi, Zab mampu menangani situasi ini dengan cara terus mengandalkan quorum untuk memastikan konsistensi pesan.

Misalnya, dalam aplikasi Zookeeper yang digunakan untuk manajemen konfigurasi dalam lingkungan terdistribusi, Zab memastikan bahwa perubahan konfigurasi yang dikirimkan oleh pemimpin diterima dan diterapkan secara konsisten di seluruh node yang berfungsi, meskipun beberapa node mungkin mengalami latensi atau kegagalan. Ini menjaga integritas dan konsistensi sistem secara keseluruhan bahkan dalam kondisi jaringan yang tidak ideal.

15. (10.0 poin) Berikut adalah diagram arsitektur Sirekap:



Source: PPT Pak Yudhis di sidang MK

Sirekap memiliki *server* yang tersebar secara geografis di tiga tempat yang berbeda. Setiap *region* memiliki satu *instance database* dan banyak *nodes* aplikasi yang diatur oleh Kubernetes. Misalkan terdapat dua jenis aplikasi, yaitu *backend* utama Sirekap yang menangani fungsionalitas utama seperti *upload* gambar, ambil data, dan lain-lain serta *worker* yang membaca gambar formulir rekapitulasi suara. Selain itu, *cache* rekап data serta gambar publik secara periodik disimpan dan diperbarui pada *content delivery network* (CDN). *Database* yang digunakan adalah *database* relasional dengan satu *database* merupakan *primary database* serta dua *database* sisanya merupakan replika. *Primary database* melayani permintaan *read* dan *write* sedangkan replika hanya melayani permintaan *read*.

**a. Berdasarkan diagram dan penjelasan di atas, sebutkan strategi apa saja yang digunakan oleh Sirekap untuk meningkatkan skalabilitas!**

- Sirekap melakukan distribusi geografis untuk mengurangi latensi pengguna dari berbagai lokasi dan mendistribusikan kerja lebih merata
- Sirekap menggunakan Kubernetes untuk melakukan otomatisasi deployment, scaling, dan container management.

- Sirekap memiliki satu database utama dan dua database replika untuk meningkatkan kapasitas read throughput karena read request dapat diberikan ke replika, mengurangi traffic pada database utama
- Sirekap menggunakan CDN untuk caching agar meningkatkan kecepatan akses bagi pengguna
- Sirekap melakukan segregasi aplikasi menjadi backend dan worker yang memungkinkan optimalisasi resource dan scaling aplikasi berdasarkan kebutuhannya.

**b. Tentukan klasifikasi Sirekap berdasarkan teorema CAP! (AP, CA, atau CP).**

Sirekap merupakan aplikasi CP (Consistency dan Partition Tolerance) karena operasi write hanya dilakukan ke database utama sedangkan database replika hanya bisa melakukan read. Hal ini meningkatkan konsistensi (C) karena setiap perubahan data melalui database utama akan dipindahkan ke replika terlebih dahulu, mengorbankan availability (A) untuk beberapa saat. Penyebaran aplikasi secara geografis membuat Sirekap memiliki toleransi partisi (P).

**c. Misalkan pemindaian gambar rekapitulasi surat suara dilakukan di sisi server dengan menggunakan pola *message queue*. Pekerjaan untuk melakukan pemindaian gambar disebut dengan *job*. Saat gambar diunggah, gambar tersebut akan dimasukkan ke dalam *job queue*. *Worker* akan mengambil *job* dari *job queue* dan menyelesaiakannya. Apa keuntungan menangani pemindaian gambar dengan cara ini dibandingkan langsung saat gambar diunggah?**

- Decoupling: Penggunaan job queue dan worker membuat operasi unggah gambar dan pemindaian gambar terpisah, sehingga kedua operasi dapat dijalankan secara independen. Sehingga jika terjadi lonjakan pengunggahan gambar, cukup melakukan horizontal scaling dengan menambah worker untuk memproses job queue.
- Load balancing: Dengan job queue, pekerjaan dapat didistribusikan secara merata ke seluruh worker, menghindari beban yang lebih pada server.
- Fault tolerance: Jika sebuah worker mengalami masalah, pekerjaan yang gagal dapat dimasukkan kembali ke job queue untuk diambil dengan worker lain.
- Pekerjaan asinkron: Dengan masuknya gambar ke job queue, server bisa langsung memberikan feedback kepada pengguna bahwa gambar sudah masuk job queue, sisa proses pemindaian saja oleh worker.

**d. Database pada kasus ini menggunakan *leader based replication*. Buatlah perbandingan dengan pola lain seperti *leaderless based replication* dan *multi-leader replication*!**

## **Leader-Based Replication**

Sistem ini menetapkan satu node sebagai pemimpin yang menangani semua operasi write. Node lainnya merupakan follower yang mereplikasi data dari leader. Operasi read dapat dilakukan oleh leader atau follower.

### **Kelebihan**

- Konsistensi, data akan konsisten karena write hanya ke leader
- Sederhana, implementasi tidak sulit
- Tidak terjadi konflik karena hanya satu titik penulisan

### **Kekurangan**

- Single point of failure: Jika leader mengalami kegagalan, penulisan tidak bisa dilakukan sampai ditetapkan leader baru
- Skalabilitas write hanya terbatas pada satu leader

## **Leaderless Replication**

Sistem ini tidak memiliki leader node, melainkan semua node dapat melakukan read dan write. Sistem ini menggunakan konsep konsensus untuk mengatasi konflik.

### **Kelebihan**

- Availability tinggi
- Lebih toleran terhadap kegagalan karena tidak ada single point of failure
- Load balancing, terdapat distribusi pekerjaan ke seluruh node secara merata

### **Kekurangan**

- Kompleks dan manajemen konflik lebih sulit
- Konsistensi tidak terjamin

## **Multi-Leader Replication**

Sistem ini memiliki beberapa leader yang dapat menerima penulisan secara bersamaan. Setiap leader mereplikasi penulisan ke leader lain dan followers.

### **Kelebihan**

- Availability untuk write lebih tinggi daripada single-leader replication
- Distribusi geografis, write dapat dilakukan ke leader terdekat

### Kekurangan

- Risiko terjadi konflik tinggi karena banyaknya leader
- Membuat sistem lebih kompleks dan sulit dikelola

- e. Pepatah mengatakan bahwa tidak ada sistem yang sempurna. Pada sistem terdistribusi, kita harus mengambil kompromi dan memahami *trade-offs* yang dipilih ketika mengambil keputusan. Pada permasalahan pemindaian gambar rekapitulasi surat suara, terdapat dua pendekatan yang berbeda, yaitu melakukan pemindaian dari sisi *client* dan sisi *server*. Misalkan hasil pemindaian pada perangkat *client* memiliki akurasi sebesar 95% dan pemindaian pada *server* memiliki akurasi sebesar 99%. Pertimbangkanlah dari segala aspek, mulai dari aspek sosial, biaya, kemampuan komputasi perangkat *client*, hingga akurasi pemindaian. Bila anda menjadi pengembang Sirekap, pendekatan apa yang akan anda ambil? Jelaskan alasan keputusan anda.
- Sosial: **Lebih baik server**, Bisa saja klien tidak memiliki perangkat yang kompatibel untuk melakukan pemindaian. Permasalahan pemindaian akibat perangkat pengguna akan menyebabkan banyak permintaan, pertanyaan, dan kritik kepada pengembang aplikasi terkait hal tersebut.
  - Biaya: **Lebih baik klien**, pemindaian pada perangkat klien akan mengurangi biaya komputasi server dalam angka yang lumayan banyak.
  - Kemampuan Komputasi Client: **Lebih baik server**, karena seperti yang sudah disebutkan, perangkat klien belum tentu kompatibel untuk melakukan pemindaian.
  - Akurasi: **Lebih baik server**, dari angka sudah terlihat.

Sehingga memperhitungkan hal-hal tersebut, lebih baik pemindaian dilakukan di sisi server. Keunggulan akurasi dan kompatibilitas klien saja sudah cukup untuk menentukan hal tersebut terkait aplikasi Sirekap.

Ini lebih lengkap dibahas di matkul PAT kalau tertarik silahkan ambil semester ganjil.

## V. Miscellaneous



1. (1.5 poin) Jelaskan perbedaan *free as in libre* dan *free as in gratis* dalam dunia perangkat lunak.

Free as in libre mengacu pada software freedom, dan umumnya disebut free and open-source software (FOSS). Sebuah software FOSS memberikan kebebasan pengguna untuk menjalankan, mempelajari, memodifikasi, dan mendistribusikan software tersebut. Fokus dari free as in libre adalah pada hak dan kebebasan sebuah pengguna. Harga dari software tersebut bukanlah yang difokuskan, melainkan pada kontrol dan penggunaan dari software tersebut.

Free as in gratis mengacu pada software yang dapat digunakan tanpa pengeluaran biaya moneter. Software gratis dapat di-download dan digunakan tanpa biaya, namun software tersebut belum tentu diberikan kebebasan untuk dipelajari, dimodifikasi, ataupun didistribusikan.

2. (1.5 poin) Sebutkan, jelaskan, dan tunjukkan perbedaan antara tiga lisensi *free-software* yang umum digunakan! Kemudian, untuk setiap lisensi, berikan tiga buah perangkat lunak terkenal yang menggunakannya.

- GPL: GNU General Public License (GPL) adalah lisensi copyleft yang dirancang untuk memastikan bahwa perangkat lunak bebas tetap bebas. Lisensi ini mewajibkan bahwa setiap turunan dari perangkat lunak yang dilisensikan di bawah GPL juga harus dilisensikan di bawah GPL. Dengan kata lain, setiap

- modifikasi atau redistribusi perangkat lunak harus tetap terbuka dan sama bebasnya. **Contoh perangkat lunak:** Linux kernel, GNU Bash, Wordpress
- Apache: Apache License 2.0 adalah lisensi open-source yang memungkinkan pengguna untuk menggunakan, memodifikasi, dan mendistribusikan perangkat lunak dengan lebih fleksibel dibandingkan dengan lisensi copyleft. Lisensi ini juga mencakup persyaratan terkait paten, yang memberikan lisensi kepada pengguna untuk menggunakan paten yang terkait dengan perangkat lunak. **Contoh perangkat lunak:** Apache HTTP Server, Hadoop, Spark
  - MIT: MIT License adalah salah satu lisensi open-source yang paling permissif dan sederhana. Lisensi ini mengizinkan pengguna untuk melakukan hampir semua hal dengan perangkat lunak, termasuk penggunaan, modifikasi, dan redistribusi, dengan syarat utama bahwa salinan lisensi dan hak cipta disertakan dalam distribusi. **Contoh perangkat lunak:** jQuery, Ruby on Rails, Node.js

3. (1.5 poin) Banyak orang yang berpendapat bahwa Free Software Foundation (FSF) terlalu berlebihan dalam pendekatannya mereka akan *free software*.

a. Jelaskan mengapa FSF mendapatkan reputasi ini.

Free Software Foundation (FSF) mendapatkan reputasi sebagai organisasi yang terlalu agresif dalam pendekatannya terhadap free software karena penekanan mereka yang kuat pada prinsip kebebasan perangkat lunak dan penolakan terhadap perangkat lunak proprietary.

FSF secara tegas mempromosikan filosofi bahwa semua perangkat lunak harus bebas dalam arti "libre", yaitu memberikan pengguna hak untuk menjalankan, memodifikasi, dan mendistribusikan perangkat lunak tanpa batasan. Mereka sering kali mengkritik model perangkat lunak proprietary dan bersikeras bahwa perangkat lunak proprietary mengancam kebebasan pengguna dan hak akses terbuka.

Pendekatan ini, yang termasuk pengawasan ketat terhadap lisensi dan model distribusi perangkat lunak, terkadang dianggap sebagai ekstrem atau kurang fleksibel oleh sebagian orang, terutama oleh mereka yang melihat keuntungan dalam solusi proprietary atau model bisnis yang berbeda.

b. Jelaskan pendapatmu sendiri terhadap pendekatan yang dimiliki FSF.

Menurut saya, free software tidak selalu unggul dengan proprietary software, ada beberapa kondisi dimana proprietary software memiliki keunggulannya. Sehingga menurut saya pemikiran bahwa free-software selalu lebih baik merupakan hal yang terlalu ideologis. Namun saya masih mengapresiasi terhadap tekad FSF untuk mendorong free-software, dan mendukung pandangan mereka untuk memunculkan lebih banyak free-software.

4. (1.5 poin) Menurutmu, apakah *free and open-source software* unggul di atas *proprietary/closed- source software*? Berikan alasanmu; tinjaulah dari beberapa aspek.

Software FOSS dan proprietary memiliki pro dan kontranya masing-masing, dan memutuskan tipe yang lebih unggul merupakan hal yang sangat bergantung pada konteks dan aplikasi yang dibicarakan.

Dari segi harga, software FOSS akan lebih umum memiliki harga yang lebih kecil atau bahkan gratis. Software FOSS akan jauh lebih tinggi dalam fleksibilitas dan kontrol bagi pengguna, karena kodennya dapat dilihat dan dimodifikasi oleh pengguna. Software FOSS juga memberikan transparansi software, dan jika terdapat exploit keamanan dapat dideteksi dan diperbaiki dengan cepat melalui komunitas pengguna.

Namun proprietary software umumnya diurus oleh sebuah tim besar yang terpartisi menjadi bagian-bagiannya masing-masing. Ini menyebabkan banyak aspek dari software dapat di-develop dan ditest secara paralel sehingga menghasilkan software dengan kualitas yang tinggi dan komplit di seluruh aspek. Proprietary software juga umumnya dikelola oleh perusahaan ternama yang memiliki budget yang besar untuk dimasukkan ke pengembangan software.

Sehingga memutuskan FOSS atau proprietary merupakan hal yang bergantung pada konteks dan kualitas aplikasi serta pengembangnya. Software FOSS memiliki keunggulan transparansi atas aplikasi dan fleksibilitas dalam modifikasi, serta umumnya diutamakan membuat kualitas software yang tinggi dibanding software yang marketable. Namun software proprietary umumnya akan memiliki budget dan jumlah pekerja lebih tinggi, memberikan sebuah software yang lengkap dalam waktu yang lebih cepat.

5. (1.5 poin) Menurutmu, apakah XAMPP boleh digunakan untuk *production*? Jelaskan.

XAMMP adalah software open-source yang ditujukan untuk kebutuhan testing dan development pada sebuah device lokal. Oleh karena itu, XAMMP didesain agar mudah digunakan pengguna. Kemudahan ini mengorbankan keamanan dan skalabilitas, dua hal yang penting untuk sebuah sistem production. Salah satu kelemahan XAMMP adalah database yang mudah diakses. Namun bahkan tanpa kelemahan keamanannya, menggunakan sebuah perangkat lunak yang didesain untuk testing pada production merupakan hal yang kurang cermat dilakukan, karena terdapat alternatif lain yang dapat digunakan yang khusus didesain untuk production. Oleh karena itu, menurut saya XAMMP sebaiknya tidak digunakan untuk production, dan jika digunakan, sebaiknya dilakukan modifikasi untuk meningkatkan keamanannya.

## Referensi

- [1] "Reasons Why You Should Never Use XAMPP on a Production Server," *MakeUseOf*, [Online]. Available:

<https://www.makeuseof.com/reasons-why-you-should-never-use-xampp-on-production-server/>. [Accessed: 21-Jul-2024].

[2] “Why is XAMPP Not Suited for Production?” *Stack Overflow*, [Online]. Available: <https://stackoverflow.com/questions/26297731/why-is-xampp-not-suited-for-production>. [Accessed: 21-Jul-2024].

6. (2.5 poin) Bayangkan kamu sedang bekerja sebagai seorang konsultan IT untuk sebuah perusahaan atau institusi pemerintah. Sebagai konsultan, kamu mengusulkan agar organisasi tersebut menggunakan OS berbasis Linux dan *free software* (seperti LibreOffice) untuk semua komputer kantor. Yakinkan petinggi perusahaan tersebut - yang awam akan teknologi, namun memiliki pikiran terbuka - mengapa hal tersebut sebaiknya dilakukan.

Saya akan meyakinkan petinggi perusahaan dengan menyebutkan keunggulan-keunggulan penggunaan Linux dan FOSS berikut:

- Kehematian biaya lisensi dan maintenance: Penggunaan Linux dan free-software tidak perlu mengeluarkan biaya lisensi, yang tentunya jauh lebih baik daripada proprietary software yang harus bayar, terutama jika perlu digunakan pada jumlah komputer yang banyak. Linux juga dikenal terhadap stabilitas dan keamanan yang akan mengurangi biaya maintenance.
- Keamanan: Linux dikenal akan keamanannya yang tinggi, sehingga penggunaan Linux pada komputer kantor akan mengurangi biaya dan usaha untuk mencegah dan menangani serangan digital.
- Fleksibilitas dan kontrol: Kustomisasi tinggi Linux membuat perusahaan sangat fleksibel akan memodifikasi komputer untuk kebutuhan mereka. Komputer perusahaan dapat dioptimalkan dan disesuaikan dengan kebutuhan spesifik perusahaan. Hal ini juga membuat perubahan konfigurasi di masa depan lebih mudah, jika ternyata terjadi sesuatu yang membutuhkan perubahan yang banyak.
- Privasi: Dengan menggunakan Linux dan free-software, perusahaan memiliki kontrol yang lebih kuat atas data mereka. Hal ini akan mengurangi terjadinya kebocoran data sensitif perusahaan ataupun dipasangnya iklan dan pelacak pada komputer perusahaan.
- Mendukung Bisnis Berkelanjutan: Dengan menggunakan free software, perusahaan akan memberi dukungan kepada model bisnis yang berfokus pada kolaborasi, inovasi, dan transparansi. Dukungan ini dapat membantu mendatangkan kemajuan pada dunia IT.

7. (2.5 poin) Sebutkan dan jelaskan tahap-tahap peretasan.

Tahap-tahap umum pada peretasan (hacking) adalah sebagai berikut:

- Reconnaissance (Information Gathering): Pada tahap ini, penyerang hanya mencoba mengumpulkan informasi sebanyak-banyaknya terkait targetnya. Tahap

- ini akan membantu penyerang mengerti sistem target, arsitektur, dan potensi kelemahan. Reconnaissance dapat dilakukan secara pasif, yaitu tanpa berinteraksi dengan target secara langsung (dapat melalui sosial media, public record, atau DNS), ataupun secara aktif dengan network scanning atau probing.
- Scanning: Pada tahap ini, penyerang akan aktif mencoba mencari kelemahan sistem target. Tahap ini dapat dilakukan dengan port scanning, vulnerability scanning dengan software, atau network scanning.
  - Gaining Access: Tahap ini melibatkan memanfaatkan kelemahan yang ditemukan untuk mendapatkan akses ke sistem target. Hal ini dapat dilakukan dengan menggunakan kode exploit, brute force attacks, ataupun social engineering.
  - Maintaining Access: Setelah penyerang sudah mendapatkan akses, langkah selanjutnya adalah pemasangan tools dan metode agar mereka dapat mengakses kembali sistem ini di masa depan. Hal ini bermanfaat jika target yang dibobol dapat memberikan keuntungan jangka panjang. Pelaksanaan tahap ini dapat dilakukan dengan menggunakan backdoors atau rootkits.
  - Covering Tracks: Tahap terakhir adalah menghapus jejak-jejak telah terjadinya penyerangan atau peretasan. Penyerang dapat melakukan hal seperti log tampering, file deletion, atau system modification untuk menghapus bukti-bukti bahwa telah terjadi unauthorized access.

## Referensi

- [1] “Phases of Hacking,” GreyCampus, [Online]. Available: <https://www.greycampus.com/opencampus/ethical-hacking/phases-of-hacking>. [Accessed: 21-Jul-2024].
- [2] “5 Phases of Hacking,” GeeksforGeeks, [Online]. Available: <https://www.geeksforgeeks.org/5-phases-hacking/>. [Accessed: 21-Jul-2024].

8. (2.5 poin) Beberapa orang memiliki kepercayaan atau pendapat bahwa keamanan informasi terbatas pada keamanan aplikasi/perangkat lunak dan data.

a. Apakah kamu setuju dengan pendapat tersebut? Jelaskan pendapatmu.

Tidak setuju, karena walaupun keamanan aplikasi dan data merupakan bagian dari keamanan informasi, masih terdapat konsep-konsep lain yang terkait dengan keamanan informasi. Bidang-bidang lain yang dicakup dalam keamanan informasi adalah:

- Keamanan fisik: Bidang ini membahas terkait pengamanan informasi secara fisik, seperti menjaga akses fisik ke ruangan server, data center, dan perangkat-perangkat informasi lainnya.
- Keamanan jaringan: Bidang ini membahas terkait menjaga integritas dan penggunaan jaringan, seperti penggunaan firewall, IDS, dan protokol-protokol keamanan jaringan.

- Keamanan sosial: Bidang ini membahas bagaimana kita bisa meningkatkan pemahaman dan best-practice terkait keamanan informasi, seperti pembelajaran mengelola password, cara mendeteksi malware dan kelemahan keamanan, dan lain-lain.

**b. Sebutkan dan jelaskan area-area keamanan informasi.**

- Application security: Bidang yang membahas menjaga keamanan sebuah aplikasi agar tidak dapat dimanipulasi oleh pihak jahat.
- Data security: Bidang yang membahas menjaga data dari akses tidak terautentikasi, korupsi, pencurian, dan lain-lain. Memanfaatkan konsep-konsep seperti enkripsi dan kriptografi, akses kontrol, dan data masking.
- Network security: Membahas tentang penjagaan keamanan sebuah jaringan dan informasi yang ditransmisikan pada jaringan tersebut. Memanfaatkan konsep seperti firewall, VPN, dan protokol keamanan jaringan.
- Physical security: Membahas terkait penjagaan perangkat-perangkat informasi secara fisik dari pencurian, vandalisme, akses tidak terautentikasi dan lain-lain. Memanfaatkan penjagaan fisik, surveillance, dan lain-lain.

**9. (5.0 poin) Sebutkan dan jelaskan minimal 3 jenis malware. Untuk setiap jenis, berikan dan jelaskan satu contoh kasus yang terkenal, lengkap dengan kronologi singkatnya dan cara kerjanya.**

**Worm**

Computer Worm adalah jenis malware yang mampu mereplikasi dirinya sendiri dan menyebar ke komputer lain tanpa memerlukan bantuan pengguna. Berbeda dengan virus yang memerlukan file atau program host untuk menyebar, worm dapat bergerak secara mandiri melalui jaringan, mengeksplorasi kerentanan dalam perangkat lunak atau sistem operasi.

Ketika worm menginfeksi suatu sistem, ia tidak hanya mengganggu kinerja dengan menghabiskan sumber daya sistem seperti bandwidth dan memori, tetapi juga dapat memasang payload berbahaya seperti backdoor, ransomware, atau trojan horse. Penyebarannya yang cepat dan kemampuan untuk menginfeksi banyak sistem dalam waktu singkat membuat worm menjadi ancaman serius dalam keamanan komputer dan jaringan.

Worm komputer bekerja dengan cara mereplikasi dirinya sendiri dan menyebar secara otomatis ke komputer lain tanpa memerlukan interaksi pengguna atau file host. Proses ini dimulai ketika worm memanfaatkan kerentanan dalam sistem operasi atau perangkat lunak yang terhubung ke jaringan. Setelah berhasil masuk ke sebuah komputer, worm menginstal salinan dirinya sendiri dan mulai mencari sistem lain yang

rentan untuk diinfeksi. Ia sering menggunakan metode seperti email, file berbagi, atau eksploitasi jaringan untuk menyebar. Dengan cepat, worm dapat menggandakan dirinya dan menyebar ke ribuan komputer, mengganggu operasi jaringan, mengurangi kinerja sistem, dan berpotensi menambahkan payload berbahaya seperti backdoor atau ransomware.

Salah satu kasus worm paling terkenal adalah Morris worm, salah satu computer worm paling tua yang terdistribusi di internet. Malware ini dibuat dan didistribusikan oleh Robert Tappan Morris pada November 2, 1988. Seorang teman dari Morris mengatakan bahwa ia membuat malware ini hanya untuk mencoba apakah bisa dilakukan.

Morris worm bekerja dengan memanipulasi exploit pada sistem, antara lain: hole pada debug mode di Unix sendmail, buffer overflow di finger network service, atau login tanpa password pada rexec/rsh. Setelah berhasil memasuki sebuah komputer, worm ini akan menyalin dirinya sendiri ke sistem lain dengan mencoba berbagai teknik, termasuk brute force pada kata sandi. Namun, desain worm ini menyebabkan infeksi berlipat ganda karena ketidakmampuan untuk mendeteksi apakah sistem sudah terinfeksi, sehingga banyak komputer yang mengalami infeksi ganda dan overload.

### **Ransomware**

Ransomware adalah jenis malware yang mengenkripsi data pada komputer korban dan meminta ransom untuk mendekripsi dan mengembalikan akses ke data tersebut.

Ketika ransomware menginfeksi sistem, malware ini akan menyusup ke file dan data penting, mengunci mereka dengan enkripsi yang kuat sehingga pengguna tidak dapat mengaksesnya. Setelah enkripsi selesai, ransomware menampilkan pesan tebusan yang biasanya meminta pembayaran dalam bentuk cryptocurrency, seperti Bitcoin, untuk mendapatkan kunci dekripsi yang diperlukan untuk membuka kunci file yang terenkripsi.

Serangan ransomware sering kali dimulai melalui phishing email, lampiran berbahaya, atau eksploitasi kerentanan perangkat lunak. Dampak dari ransomware dapat sangat merusak, menyebabkan kehilangan data, gangguan operasional, dan kerugian finansial, baik untuk individu maupun organisasi.

Salah satu kasus ransomware paling terkenal adalah WannaCry. Ransomware WannaCry pertama kali muncul pada 12 Mei 2017 dan segera menyebar dengan kecepatan yang sangat cepat, menginfeksi ratusan ribu komputer di seluruh dunia dalam waktu singkat. WannaCry memanfaatkan kerentanan dalam sistem operasi Microsoft Windows yang dikenal sebagai EternalBlue, yang awalnya ditemukan oleh NSA (National Security Agency) dan bocor ke publik oleh grup hacker Shadow Brokers. EternalBlue adalah eksploitasi untuk kerentanan SMBv1 (Server Message Block versi 1) yang memungkinkan worm ransomware ini menyebar dengan mudah antar komputer dalam jaringan.

Setelah berhasil menginfeksi sebuah sistem, WannaCry mengenkripsi file yang ada di komputer korban dan mengubah ekstensi file menjadi ".wncry". Selanjutnya, ransomware ini menampilkan pesan tebusan yang meminta pembayaran sebesar \$300 hingga \$600 dalam bentuk Bitcoin untuk mendapatkan kunci dekripsi. Pesan ini juga memberikan tenggat waktu; jika pembayaran tidak dilakukan dalam waktu tertentu, jumlah tebusan akan meningkat atau data akan dihapus secara permanen.

## Virus

Virus komputer adalah jenis malware yang dirancang untuk menyebar dan merusak sistem komputer dengan cara menempel pada file atau program yang valid. Ketika file atau program yang terinfeksi dijalankan, virus ini aktif dan mulai menggandakan dirinya sendiri, sering kali tanpa sepengertahuan pengguna. Tujuan utama virus komputer bisa bervariasi, mulai dari merusak data, mencuri informasi pribadi, hingga mengambil kendali sistem.

Virus komputer sering kali disebarluaskan melalui berbagai saluran, seperti email dengan lampiran berbahaya, situs web yang terinfeksi, atau perangkat penyimpanan eksternal yang terinfeksi seperti USB flash drive. Pengguna yang tidak curiga mungkin mengunduh atau menjalankan file yang terinfeksi, yang memungkinkan virus menyebar ke komputer mereka dan perangkat lain yang terhubung. Selain itu, virus juga dapat menargetkan kerentanan dalam sistem operasi atau aplikasi untuk menembus dan menyebar lebih luas.

Contoh virus yang pernah menjadi berita dunia adalah ILOVEYOU virus. ILOVEYOU Virus, juga dikenal sebagai Love Bug atau Love Letter, adalah salah satu virus komputer paling terkenal yang muncul pada Mei 2000. Virus ini pertama kali menyebar melalui email yang tampak seperti pesan cinta dari seseorang yang tidak dikenal, dengan subjek "ILOVEYOU" dan lampiran bernama "LOVE-LETTER-FOR-YOU.txt.vbs". Pengguna yang membuka lampiran tersebut tanpa curiga sebenarnya menjalankan skrip Visual Basic yang mengandung virus.

Cara kerja ILoveYou Virus dimulai dengan menjalankan skrip yang terlampir. Setelah lampiran dibuka, virus tersebut akan mengeksloitasi kerentanan di sistem Windows dan mulai menjalankan kode berbahaya. Virus ini kemudian membuat salinan dirinya dan mengirimkan email ke semua kontak yang ada di daftar alamat email korban. Setiap email yang dikirim mengandung salinan lampiran virus, memicu infeksi lebih lanjut dan menyebar ke jaringan yang lebih luas.

Selama proses penyebaran, ILoveYou Virus juga melakukan kerusakan pada sistem yang terinfeksi. Virus ini mengakses dan memodifikasi file-file penting di sistem, seperti file gambar, dokumen, dan data penting lainnya, dengan cara mengubah atau menghapusnya. Selain itu, virus ini mengubah pengaturan sistem dan menyembunyikan beberapa file untuk menghindari deteksi. Akibatnya, banyak pengguna melaporkan kehilangan data dan gangguan operasional akibat virus ini.

## Referensi

- [1] "Computer Worm," *Wikipedia*, [Online]. Available: [https://en.wikipedia.org/wiki/Computer\\_worm](https://en.wikipedia.org/wiki/Computer_worm). [Accessed: 21-Jul-2024].
- [2] "Morris Worm," *Wikipedia*, [Online]. Available: [https://en.wikipedia.org/wiki/Morris\\_worm](https://en.wikipedia.org/wiki/Morris_worm). [Accessed: 21-Jul-2024].
- [3] "Ransomware," *Wikipedia*, [Online]. Available: <https://en.wikipedia.org/wiki/Ransomware>. [Accessed: 21-Jul-2024].
- [4] "WannaCry Ransomware Attack," *Wikipedia*, [Online]. Available: [https://en.wikipedia.org/wiki/WannaCry\\_ransomware\\_attack](https://en.wikipedia.org/wiki/WannaCry_ransomware_attack). [Accessed: 21-Jul-2024].
- [5] "Computer Virus Definition," *Fortinet*, [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/computer-virus#:~:text=Computer%20Virus%20Definition&text=A%20computer%20virus%20is%20a,in%20data%20loss%20and%20leakage>. [Accessed: 21-Jul-2024].
- [6] "ILOVEYOU," *Wikipedia*, [Online]. Available: <https://en.wikipedia.org/wiki/ILOVEYOU>. [Accessed: 21-Jul-2024].

**10. (7.5 poin) Pada tanggal 17 Juni 2024, terjadi peretasan pada Pusat Data Nasional yang, antara lain, menyebabkan kebocoran data masyarakat serta menghambat operasi berbagai lembaga pemerintah.**

**a. Buat kronologi singkat dari peretasan Pusat Data Nasional.**

Deteksi awal terjadi pada 17 Juni 2024, pukul 23.15 WIB, ketika Badan Siber dan Sandi Negara (BSSN) mendeteksi upaya penonaktifan fitur keamanan Windows Defender pada PDNS 2. Upaya ini sepenuhnya berhasil pada 20 Juni 2024, pukul 00.55 WIB. Hal ini memberikan akses penyerang untuk memasukkan file berbahaya pada sistem dan memperluas infeksi.

Dampak dari penonaktifan Windows Defender serta serangan pada 20 Juni 2024 tersebut berupa gangguan pada Bandara Internasional Soekarno-Hatta, antrian panjang penumpang, serta layanan keimigrasian.

BSSN melanjutkan investigasi dan menemukan bahwa ransomware yang digunakan adalah Brain Cipher Ransomware, sebuah pengembangan dari ransomware Lockbit 3.0. Investigasi masih berlanjut dengan keterbatasan alat bukti yang terkunci oleh ransomware. Pada 24 Juni 2024, para peretas meminta tebusan sebesar 8 juta dolar AS, namun pemerintah menegaskan tidak akan membayar tebusan tersebut.

Respons dari pemerintah adalah memastikan bahwa Pusat Data Nasional (PDN) yang sedang dibangun di Cikarang tidak terdampak oleh serangan ini, dan

evaluasi dan langkah antisipasi untuk memperkuat keamanan siber di Indonesia sedang dilakukan.

Pihak Brain Cipher berpesan akan meminta maaf dan berjanji akan memberikan kunci data pada 3 Juli 2024, dan mereka juga mengkonfirmasi bahwa penyerangan ini tidak ada kaitannya dengan unsur politik melainkan hanya untuk monetisasi saja.

- b. Sebutkan dan jelaskan dengan singkat faktor-faktor teknis (bukan faktor-faktor non-teknis seperti politik atau generasi) yang (mungkin) menyebabkan peretasan tersebut.**

Beberapa faktor teknis atas terjadinya peretasan:

- Hanya 2% dari data yang ada pada PDNS yang memiliki backup. Hal ini sangat berisiko karena jika data utama terinfeksi atau hilang, tidak ada cadangan yang memadai untuk pemulihan. Menurut aturan BSSN nomor 4/2021, salah satu syarat teknis untuk pusat data nasional adalah melakukan backup informasi dan perangkat lunak secara berkala
- PDN menggunakan Windows Defender sebagai lapisan keamanan utama. Windows Defender adalah software antivirus bawaan sistem operasi Windows yang, meskipun cukup untuk kebutuhan rumah tangga atau industri kecil, tidak memadai untuk sebuah pusat data besar dengan nilai anggaran sebesar Rp700 miliar.
- Terdapat rumor bahwa PDNS menggunakan password “admin#1234” sebagai password sistemnya. Pemilihan password ini sangat buruk, karena merupakan password yang sangat umum dan mudah ditebak oleh sebuah program brute force attack.

- c. Sebutkan dan jelaskan dengan singkat aspek-aspek keamanan informasi yang hilang akibat peretasan tersebut (minimal mencakup *triad CIA*). Untuk setiap aspek, jelaskan juga bagaimana peretasan menyebabkan kehilangannya.**

- Confidentiality: Confidentiality merujuk pada perlindungan data dari akses yang tidak sah. Tujuan utamanya adalah memastikan bahwa hanya pihak yang berwenang yang dapat mengakses informasi sensitif. Dalam kasus peretasan PDN, ransomware seperti Brain Cipher Ransomware dapat mengakses dan mengenkripsi data sensitif. Serangan ini memungkinkan peretas untuk memperoleh akses ke informasi yang tidak seharusnya mereka miliki. Hal ini mengancam kerahasiaan data yang seharusnya hanya dapat diakses oleh pihak yang berwenang.
- Integrity: Integritas memastikan bahwa data tidak diubah atau dirusak tanpa izin. Ini termasuk mencegah modifikasi data yang tidak sah dan memastikan bahwa data tetap akurat dan utuh. Ransomware dapat merusak atau mengubah data yang ada pada sistem. Dalam kasus PDN,

ransomware dapat menginfeksi dan mengenkripsi data, yang berarti bahwa data tersebut tidak dapat diakses dalam bentuk aslinya. Jika file yang penting atau sistem dikompromikan, integritas data dapat terancam karena perubahan yang dilakukan oleh malware, sehingga merusak keakuratan dan keandalan informasi.

- Availability: Availability memastikan bahwa data dan sistem dapat diakses dan digunakan oleh pihak yang berwenang saat dibutuhkan. Ini mencakup perlindungan terhadap gangguan layanan dan kerusakan sistem. Dalam peretasan PDN, serangan ransomware mengakibatkan gangguan signifikan pada layanan imigrasi. Sistem yang terkena dampak tidak dapat digunakan untuk proses pengecekan dan layanan lainnya, menyebabkan antrian panjang dan gangguan operasional di Bandara Internasional Soekarno-Hatta. Hal ini menunjukkan bagaimana ransomware dapat mengurangi ketersediaan layanan penting, mempengaruhi kemampuan organisasi untuk berfungsi secara normal.

**d. Berikan pendapatmu bagaimana ilmu-ilmu yang terkait dengan Lab Sister dapat mencegah terulangnya peretasan seperti ini di masa depan.**

- Mempelajari mata kuliah Organisasi dan Arsitektur Komputer dapat memberi wawasan terkait bagaimana sebuah program dapat meretas sistem dan melakukan serangan ransomware seperti pada kasus peretasan PDN, dan melakukan mitigasi dan peningkatan keamanan untuk mencegah terjadinya hal tersebut
- Mempelajari Sistem Operasi dapat memberi wawasan terkait keamanan sistem dan bagaimana sebuah program berinteraksi dengan hardware dan sistem operasi server pusat data untuk melakukan serangan ransomware
- Mempelajari Jaringan Komputer dapat memberikan wawasan terkait keamanan dan metode penyerangan dalam dan melalui jaringan seperti serangan pada PDN, dan bagaimana cara memitigasi dan menjaga sistem dari serangan tersebut
- Mempelajari Sistem Paralel dan Terdistribusi dapat memberi wawasan terkait sebuah sistem terdistribusi seperti server PDN dan bagaimana meningkatkan keamanan agar tidak terjadi serangan seperti pada retasan PDN

## **Referensi**

[1] "Kronologi Serangan PDN," *Cloud Computing ID*, [Online]. Available: <https://www.cloudcomputing.id/berita/kronologi-serangan-pdn>. [Accessed: 21-Jul-2024].

[2] "Belajar dari Diretasnya PDN: Pentingnya Backup dan Keamanan Server," *Netmarks*, [Online]. Available:

<https://www.netmarks.co.id/post/belajar-dari-diretasnya-pdn-pentingnya-backup-dan-keamanan-server>. [Accessed: 21-Jul-2024].

11. **(10.0 poin - WAJIB)** Sebagai asisten, nantinya kamu akan dituntut untuk memastikan seluruh dan segala bentuk kecurangan akademis dalam mata kuliah-mata kuliah yang diasistensi ditemukan, dilaporkan, dan ditindaklanjuti.

- a. **Sebutkan landasan (bebas; hukum, agama, etika, dan seterusnya) yang mewajibkan asisten untuk melakukan hal tersebut.**

Landasan etika: Sebagai anggota dari salah satu universitas paling ternama di negara ini, sudah seharusnya sebagai mahasiswa bahwa kita semua menjunjung tinggi terkait kejujuran akademik. Sebuah sistem akademik tidak ada gunanya jika yang mengikutinya tidak menjunjung tinggi kejujuran dan pendidikan. Sehingga sebagai pihak yang bertugas untuk menindaklanjuti kecurangan akademik, sudah semestinya asisten melakukan hal yang sesuai terhadap tindakan tersebut.

- b. **Sebutkan tiga alat yang mampu membantumu menemukan dan atau mengonfirmasi tindak laku kecurangan.**

Plagiarism detector, AI detector, turnitin.

- c. **Jelaskan garis yang akan kamu tarik terkait kecurangan dan plagiarisme.**

Jika tugas yang diberikan memperbolehkan diskusi antar teman, kemiripan jawaban masih bisa ditoleransi, namun plagiarisme total sudah mendekati daerah kecurangan. Terkait ujian, kuis, dan kegiatan lainnya yang tidak memperbolehkan diskusi, kemiripan jawaban patut diinvestigasi dan jawaban yang sama bahkan plagiarisme dapat ditindaklanjuti dengan memberikan nilai 0.

- d. **Berikut merupakan beberapa studi kasus. Untuk masing-masing kasus, jelaskan dengan singkat apa saja yang akan kamu lakukan untuk menentukan verdict akhir.**

- i. **Terdapat kelompok tugas besar yang kodennya sangat mirip dengan kode yang tersedia di internet, namun mereka tidak memberikan pengakuan (credits) atasnya.**

Mendiskusikan dengan salah satu anggota kelompok terkait hal tersebut, jika tidak diberikan alasan yang masuk akal, maka pantas untuk mendapat pengurangan nilai.

- ii. **Terdapat kelompok tugas besar yang kodennya sangat mirip dengan kode tugas besar karya kelompok kakak tingkat.**

Sama seperti nomor (i).

- iii. **Terdapat beberapa mahasiswa yang saling berbincang dan tengok menengok ketika ujian.**

Menegur untuk memberitahu bahwa mereka sudah diawasi lebih lanjut. Jika masih melakukan, maka dicatat.

- iv. **Terdapat beberapa mahasiswa yang jawabannya sangat mirip satu sama lain, dan jawaban tersebut bukanlah jawaban standar.**

Jika konteks tugasnya memperbolehkan diskusi, biarkan. Jika konteks kuis/ujian/lainnya, kurangi nilai atau berikan 0.

- v. **Terdapat mahasiswa yang jawabannya sangat mirip dengan mahasiswa lain yang kamu ketahui sering memberikan tutor, dan bukan merupakan seseorang yang sepertinya akan melakukan kecurangan.**

Mungkin saja hanya karena mengikuti ajaran yang sama, jadi biarkan saja atau dikomunikasikan dengan orangnya terlebih dahulu.

- vi. **Terdapat mahasiswa yang terbukti, berdasarkan sebuah detail yang sepertinya lupa diganti, mengumpulkan jawaban temannya.**

Komunikasikan dengan orangnya. Jika tidak memberikan jawaban yang masuk akal, berikan 0 atau dikurangi nilainya.

- e. **Apakah kamu akan meresikokan reputasi personal, hubungan-hubungan interpersonal, maupun kestabilan organisasi non-akademik (seperti himpunan atau unit) untuk memastikan integritas akademik?**

Tentu saja, jika situasi memang mengharuskan untuk melakukan hal tersebut. Karena yang paling utama dalam lingkungan pendidikan ini adalah integritas akademik.