UNIVERSITAS INDONESIA

ANALISIS PERFORMA TUNING HYPERVISOR KVM *STREAMING SIMD EXTENSION* (SSE) UNTUK KOMPRESI VIDEO PADA *VIRTUAL MACHINE* DI APACHE CLOUDSTACK

SKRIPSI

ARIQ PRADIPA SANTOSO
2006527052

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
2023

UNIVERSITAS INDONESIA

ANALISIS PERFORMA TUNING HYPERVISOR KVM *STREAMING SIMD EXTENSION* (SSE) UNTUK KOMPRESI VIDEO PADA *VIRTUAL MACHINE* DI APACHE CLOUDSTACK

SKRIPSI
Diajukan sebagai salah satu syarat untuk memperoleh gelar
Sarjana Teknik

ARIQ PRADIPA SANTOSO
2006527052

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
DESEMBER 2023

HALAMAN PERSETUJUAN

Judul	:	Analisis Performa Tuning Hypervisor KVM *Streaming SIMD Extension* (SSE) untuk Kompresi Video pada *Virtual Machine* di Apache Cloudstack
Nama	:	Ariq Pradipa Santoso
NPM	:	2006527052

Laporan Skripsi ini telah diperiksa dan disetujui.

XX Desember 2023

Yan Maraden S.T., M.T.

Pembimbing Skripsi

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	:	**Ariq Pradipa Santoso**
NPM	:	**2006527052**
Tanda Tangan	:	
Tanggal	:	**XX Desember 2023**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Ariq Pradipa Santoso
NPM	:	2006527052
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Analisis Performa Tuning Hypervisor KVM *Streaming SIMD Extension* (SSE) untuk Kompresi Video pada *Virtual Machine* di Apache Cloudstack

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing	:	Yan Maraden S.T., M.T.	(	)
Penguji	:	Prof. XXX	(	)
Penguji	:	Prof. XXXX	(	)
Penguji	:	Prof. XXXXXX	(	)

@todo

Jangan lupa mengisi nama para penguji.

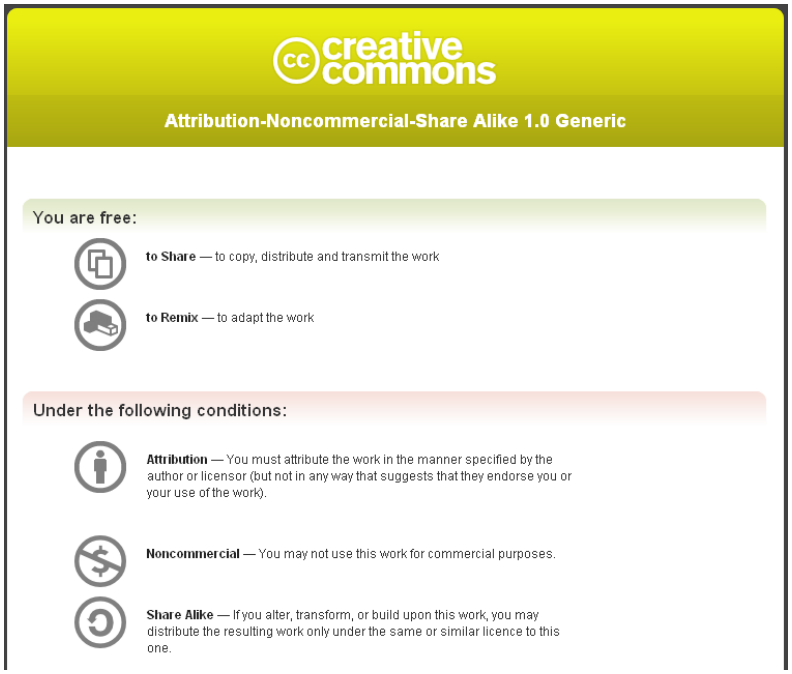
Ditetapkan di	:	Depok
Tanggal	:	XX Desember 2023

KATA PENGANTAR

Template ini disediakan untuk orang-orang yang berencana menggunakan LaTeX untuk membuat dokumen tugas akhirnya. Mengapa LaTeX? Ada banyak hal mengapa menggunakan LaTeX, diantaranya:

1. LaTeX membuat kita jadi lebih fokus terhadap isi dokumen, bukan tampilan atau halaman.
2. LaTeX memudahkan dalam penulisan persamaan matematis.
3. Adanya automatis dalam penomoran caption, bab, subbab, subsubbab, referensi, dan rumus.
4. Adanya automatisasi dalam pembuatan daftar isi, daftar gambar, dan daftar tabel.
5. Adanya kemudahan dalam memberikan referensi dalam tulisan dengan menggunakan label. Cara ini dapat meminimalkan kesalahan pemberian referensi.

Template ini bebas digunakan dan didistribusikan sesuai dengan aturan *Creative Common License 1.0 Generic*, yang secara sederhana berisi:



Gambar 1: *Creative Common License 1.0 Generic*

Gambar 1 diambil dari http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA. Jika ingin mengentahui lebih lengkap mengenai *Creative Common License 1.0 Generic*, silahkan buka http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode. Seluruh dokumen yang dibuat dengan menggunakan template ini sepenuhnya menjadi hak milik pembuat dokumen dan bebas didistribusikan sesuai dengan keperluan masing-masing. Lisensi hanya berlaku jika ada orang yang membuat template baru dengan menggunakan template ini sebagai dasarnya.

Dokumen ini dibuat dengan LaTeX juga. Untuk meyakinkan Anda, coba lihat properti dari dokumen ini dan Anda akan menemukan bagian seperti Gambar 2. Dokumen ini dimaksudkan untuk memberikan gambaran kepada Anda seperti apa mudahnya menggunakan LaTeX dan juga memperlihatkan betapa bagus dokumen yang dihasilkan. Seluruh url yang Anda temukan dapat Anda klik. Seluruh referensi yang ada juga dapat diklik. Untuk mengerti template yang disediakan, Anda tetap harus membuka kode LaTeX dan bermain-main dengannya. Penjelasan dalam PDF ini masih bersifat gambaran dan tidak begitu mendetail, dapat dianggap sebagai pengantar singkat. Jika Anda merasa kesulitan dengan template ini, mungkin ada baiknya Anda belajar sedikit dasar-dasar LaTeX.

Gambar 2: Dokumen Dibuat dengan PDFLatex

Semoga template ini dapat membantu orang-orang yang ingin mencoba menggunakan LaTeX. Semoga template ini juga tidak berhenti disini dengan ada kontribusi dari para penggunanya. Kami juga ingin berterima kasih kepada Andreas Febrian, Lia Sadita, Fahrurrozi Rahman, Andre Tampubolon, dan Erik Dominikus atas kontribusinya dalam template ini.

Depok, 27 Februari 2019

Ariq Pradipa Santoso

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	:	Ariq Pradipa Santoso
NPM	:	2006527052
Program Studi	:	Teknik Komputer
Fakultas	:	Teknik
Jenis Karya	:	Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

Analisis Performa Tuning Hypervisor KVM *Streaming SIMD Extension* (SSE) untuk Kompresi Video pada *Virtual Machine* di Apache Cloudstack

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyatan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : XX Desember 2023
Yang menyatakan

(Ariq Pradipa Santoso)

ABSTRAK

Nama	:	Ariq Pradipa Santoso
Program Studi	:	Teknik Komputer
Judul	:	Analisis Performa Tuning Hypervisor KVM *Streaming SIMD Extension* (SSE) untuk Kompresi Video pada *Virtual Machine* di Apache Cloudstack
Pembimbing	:	Yan Maraden S.T., M.T.

Tesis ini membahas kemampuan mahasiswa Fakultas Psikologi UI dalam mencari dan menggunakan informasi secara efektif dalam konteks *active learning* dan *self regulated learning* selama mereka mengikuti Program Pendidikan Dasar Pendidikan Tinggi. Penelitian ini adalah penelitian kualitatif dengan desain deskriptif. Hasil penelitian menyarankan bahwa perpustakaan perlu dilibatkan dalam pengembangan kurikulum; materi pendidikan pemakai perpustakaan harus dikembangkan sesuai dengan komponen- komponen yang ada dalam *information literacy*; perpustakaan juga harus menyediakan sarana dan fasilitas yang mendukung peningkatan *literacy* mahasiswa.

Kata kunci:
Informasi, *information literacy*, *information skills*

ABSTRACT

Name	:	Ariq Pradipa Santoso
Study Program	:	Teknik Komputer
Title	:	Performance Analysis of KVM Hypervisor Tuning with Video Compression on Apache Cloudstack
Counsellor	:	Yan Maraden S.T., M.T.

The focus of this study is the freshman student of Faculty of Psychology at University of Indonesia experience of acquiring, evaluating and using information, when they enroll in “Program Dasar Pendidikan Tinggi (PDPT)”. The purpose of this study is to understand how freshman students acquire, evaluate and use information. Knowing this will allow library to identify changes should be made to improve user education program at University of Indonesia. This research is qualitative descriptive interpretive. The data were collected by means of deep interview. The researcher suggests that library should improve the user education program and provide facilities which can help students to be information literate.

Key words:
Information literacy, information skills, information

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Cloud Computing adalah sebuah model yang memungkinkan akses yang luas, nyaman, dan sesuai permintaan ke sumber daya komputasi bersama, seperti jaringan, server, dan penyimpanan. Model ini terdiri dari lima karakteristik utama, yakni layanan mandiri *on-demand*, akses jaringan yang luas, pengelolaan sumber daya *(resource pooling)*, elastisitas yang cepat, dan layanan yang terukur. *Cloud Computing* juga melibatkan tiga model layanan, seperti *Software as a Service* (SaaS), *Platform as a Service* (PaaS), dan Infrastructure as a Service (IaaS). Selain itu, terdapat empat model implementasi: *private cloud, community cloud, public cloud,* dan *hybrid cloud,* masing-masing sesuai dengan kebutuhan dan kegunaan dari organisasi yang berbeda[1].

Kita dapat membuat sistem *Cloud Computing* sendiri dengan menggunakan Apache CloudStack. Apache CloudStack adalah perangkat lunak *open source* yang dirancang untuk implementasi dan administrasi jaringan *Virtual Machine*. CloudStack berfungsi sebagai platform *Cloud Computing* Infrastructure as a Service (IaaS) yang *reliable*, CloudStack juga dikenal karena *high availability* dan skalabilitasnya. Apache CloudStack digunakan oleh berbagai penyedia layanan untuk menyediakan layanan cloud publik, CloudStack juga digunakan oleh banyak perusahaan untuk membentuk solusi *private cloud* atau sebagai bagian dari konfigurasi *hybrid cloud*[2].

Untuk menjalankan Apache Cloudstack diperlukan sebuah hypervisor. Hypervisor adalah komponen virtualisasi yang mengawasi dan mengelola sistem operasi *guest* pada sistem *host*[3]. Hypervisor mengontrol komunikasi instruksi antara sistem operasi *guest* dan *hardware* dari *host*[3].

Pada penelitian ini Penulis menggunakan sistem operasi Ubuntu yang berbasis Linux dan menggunakan KVM sebagai hypervisornya. Hypervisor KVM *(Kernel-based Virtual Machine)* adalah sebuah hypervisor berbasis kernel yang memungkinkan virtualisasi pada sistem operasi Linux[4]. KVM memanfaatkan modul kernel untuk menyediakan dukungan langsung untuk virtualisasi dengan menggunakan instruksi *hardware* yang mendukung teknologi virtualisasi, seperti Intel VT atau AMD-V.

Penggunaan KVM sebagai hypervisor untuk menjalankan *Virtual Machine* pada Apache Cloudstack telah menjadi praktik umum. Namun, pertanyaan mendasar muncul: apakah hypervisor ini telah dikonfigurasi secara optimal untuk menjalankan tugas yang diberikan dengan efisien? Penelitian ini bertujuan untuk menganalisis apakah konfigurasi hypervisor KVM telah diatur dengan tepat dan efisien. Pada penelitian ini, penulis akan menguji performa hypervisor dengan melakukan tugas kompresi video menggunakan Handbrake. Selanjutnya, waktu yang diperlukan untuk menyelesaikan proses kompresi ini akan diukur dan dibandingkan dalam rangka evaluasi performa.

1.2 Permasalahan

Pada bagian ini akan dijelaskan mengenai definisi permasalahan yang Penulis hadapi dan ingin diselesaikan serta asumsi dan batasan yang digunakan dalam menyelesaikannya.

1.2.1 Definisi Permasalahan

Berdasarkan latar belakang yang telah dijelaskan pada bab 1.1 bahasan mengenai rumusah yang akan dibahas pada penelitian ini adalah sebagai berikut:

1. Apakah tuning hypervisor KVM pada Apache CloudStack memberikan efek yang signifikan terhadap performa sistem operasi *guest* yang menjalankan tugas kompresi video menggunakan Handbrake?
2. Bagaimana konfigurasi hypervisor KVM dapat diatur secara optimal untuk meningkatkan efisiensi dalam menangani tugas kompresi video pada Apache CloudStack?

1.2.2 Batasan Permasalahan

Karena berbagai kendala yang dihadapi pada skripsi ini dilakukan pembatasan masalah pada penelitian sebagai berikut:

1. Penelitian dilakukan menggunakan perangkat laptop dengan spesifikasi sebagai berikut:
    - CPU: AMD A10-9620P Radeon R5 10 Compute Cores 4C+6G x64.
    - RAM: 8 GB.
    - Penyimpanan: 100 GB pada harddisk.

2. Sistem operasi yang digunakan adalah Ubuntu Server 22.04 LTS.
3. Karena perangkat menggunakan CPU AMD, implementasi teknologi virtualisasi yang diadopsi adalah AMD-V.
4. Parameter yang menjadi fokus analisis dalam penelitian ini adalah rata-rata perbedaan waktu kompresi video antara hypervisor default dan hypervisor yang telah dituning.

1.3 Tujuan

Penelitian ini bertujuan untuk mendalami analisis performa tuning hypervisor KVM, khususnya dalam konteks penggunaan Apache CloudStack, dengan fokus pada tugas kompresi video. Tujuan utama penelitian adalah untuk mengevaluasi apakah konfigurasi hypervisor KVM telah diatur secara optimal dan efisien, serta untuk memahami dampak tuning tersebut terhadap performa sistem operasi guest. Hasil dari penelitian ini diharapkan dapat memberikan wawasan praktis bagi pengelola sistem yang menggunakan Apache CloudStack dalam lingkungan produksi, serta memberikan panduan terkait optimalisasi hypervisor KVM dalam meningkatkan kinerja layanan Cloud Computing Infrastructure as a Service (IaaS).

1.4 Metodologi Penelitian

Beberapa metodologi yang dilakukan pada penelitian ini, antara lain:

1. Studi Literatur

   Studi literatur dalam penelitian ini dilakukan dengan melakukan pencarian, membaca, dan memahami jurnal serta sumber referensi yang terkait dengan *Cloud Computing*. Penelitian ini juga melibatkan eksplorasi literatur yang mendalam terkait dengan metode tuning hypervisor KVM, kondisi saat melakukan pemrosesan kompresi video, serta evaluasi keuntungan dan kerugian yang mungkin muncul akibat hasil tuning ini.

2. Pengumpulan Data

   Pada penelitian ini, kompresi video akan dilakukan menggunakan Handbrake pada dua sistem, yaitu sistem yang menggunakan KVM default dan sistem yang menggunakan KVM yang sudah dituning. Pada masing-masing sistem, kompresi video akan dilakukan sebanyak 20 kali. Setelah itu, waktu rata-rata kompresi video pada kedua sistem akan dibandingkan dan dianalisis.

3. Analisis

   Setelah itu, dilakukan evaluasi terhadap hasil yang diperoleh dan kemudian menyusun kesimpulan.

1.5 Sistematika Penulisan

Pada penulisan skripsi ini terdiri dari 3 bab dengan pokok bahasan yang berbeda-beda untuk setiap bab.

- BAB I PENDAHULUAN

  Bab I membahas tentang pendahuluan penelitian, yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, metode penelitian, dan sistematika penulisan.

- BAB II TUNING HYPERVISOR KVM

  Bab II membahas tentang landasan teori yang digunakan untuk memahami dan menganalisis hasil penelitian.

- BAB III METODE PENGUJIAN TUNING HYPERVISOR KVM

  Bab III Membahas tentang tahapan dan bagaimana metode yang digunakan untuk melakukan tuning dan pengujian dari Hypervisor KVM, dan juga bagaimana analisis akan dilakukan

BAB 2
TUNING HYPERVISOR KVM

2.1 *Cloud Computing*

Cloud Computing, menurut definisi National Institute of Standards and Technology (NIST), merupakan suatu model komputasi yang menyediakan akses dan konfigurasi sumber daya komputasi seperti jaringan, server, penyimpanan, aplikasi, dan layanan secara on-demand, memungkinkan pelepasan yang cepat tanpa interaksi yang kompleks dengan penyedia layanan[1]. *Cloud Computing* bukanlah suatu teknologi spesifik, melainkan sebuah model yang menggambarkan operasional dan ekonomi untuk penyediaan serta penggunaan infrastruktur IT dan layanan terkait. Beberapa definisi cloud memiliki karakteristik umum yang sama, seperti *pay-per-use* (pembayaran sesuai dengan penggunaan), kapasitas yang elastis, layanan mandiri, dan abstraksi atau virtualisasi sumber daya[5]. NIST membagi model layanan cloud menjadi[1]:

1. Software As A Service (SaaS)

   Konsumen dapat memanfaatkan kemampuan dengan menggunakan aplikasi penyedia yang beroperasi di dalam infrastruktur cloud. Aplikasi tersebut dapat diakses dari berbagai perangkat klien melalui antarmuka klien yang ringan, seperti browser web (contohnya, email berbasis web), atau antarmuka program. Pengguna tidak perlu mengelola atau mengontrol infrastruktur cloud yang mendasarinya, termasuk jaringan, server, sistem operasi, penyimpanan, atau bahkan kemampuan aplikasi individual, kecuali mungkin pada pengaturan konfigurasi aplikasi khusus pengguna yang terbatas.

2. Platform As A Service (PaaS)

   Konsumen diberikan kemampuan untuk mendeploy aplikasi yang mereka buat atau peroleh ke infrastruktur cloud, menggunakan bahasa pemrograman, *library*, layanan, dan alat yang didukung oleh penyedia. Pengguna tidak perlu mengelola atau mengontrol infrastruktur cloud yang mendasarinya, termasuk jaringan, server, sistem operasi, atau penyimpanan. Meskipun begitu, pengguna tetap memiliki kendali atas aplikasi yang diimplementasikan dan mungkin dapat mengatur konfigurasi lingkungan hosting aplikasi.

3. Infrastructure As A Service (IaaS)

   Kemampuan yang diberikan kepada konsumen adalah untuk menyediakan sumber daya pemrosesan, penyimpanan, jaringan, dan sumber daya komputasi dasar lainnya di mana konsumen dapat mendeploy dan menjalankan perangkat lunak sembarang, yang dapat mencakup sistem operasi dan aplikasi. Konsumen tidak mengelola atau mengendalikan infrastruktur awan yang mendasari tetapi memiliki kendali atas sistem operasi, penyimpanan, dan aplikasi yang didistribusikan; dan mungkin kendali terbatas terhadap beberapa komponen jaringan tertentu (misalnya, firewall host).

2.2 Apache CloudStack

Gambar 2.1: Logo Apache CloudStack

Apache CloudStack adalah perangkat lunak *open source* yang dirancang untuk mendeploy dan mengelola jaringan besar mesin virtual, sebagai platform komputasi awan berbasis Infrastructure as a Service (IaaS) yang sangat tersedia dan dapat *scaleable*[2]. Pada Gambar 2.1 adalah logo dari Apache Cloudstack. CloudStack digunakan oleh sejumlah penyedia layanan untuk menawarkan layanan cloud publik, dan oleh banyak perusahaan untuk menyediakan penawaran cloud *on-premise* (pribadi), atau sebagai bagian dari solusi cloud hybrid[2].

CloudStack adalah solusi siap pakai yang mencakup seluruh ”stack” fitur yang paling diinginkan oleh organisasi dengan IaaS cloud: orkestrasi komputasi, Jaringan sebagai Layanan, manajemen pengguna dan akun, dan antarmuka Pengguna (UI) yang baik[2].

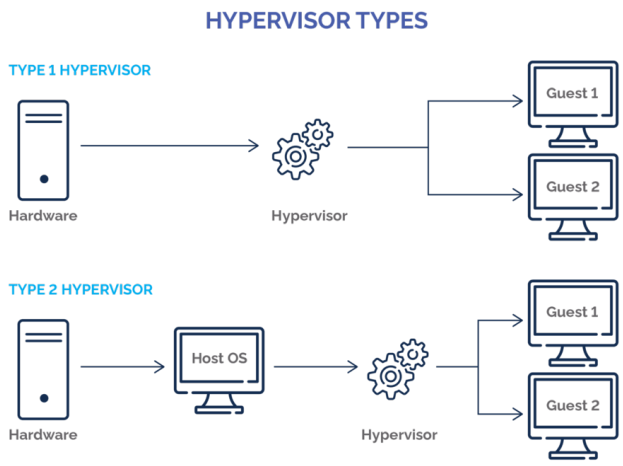
CloudStack saat ini mendukung hypervisor paling populer: VMware, KVM, Citrix XenServer, Xen Cloud Platform (XCP), Oracle VM server, dan Microsoft Hyper-V[2].

Pengguna dapat mengelola cloud mereka dengan antarmuka web yang mudah digunakan, alat *commandline*, dan/atau API RESTful yang lengkap. Selain itu, CloudStack menyediakan API yang kompatibel dengan AWS EC2 dan S3 untuk organisasi yang ingin mendeploy cloud secara hybrid[2].

CloudStack, yang awalnya dikembangkan oleh Cloud.com, diakuisisi oleh Citrix pada tahun 2011 dan diserahkan kepada Apache Software Foundation pada tahun 2012. Pengembangan saat ini diatur oleh Apache Foundation dengan kode yang tersedia di bawah lisensi Apache 2.0[6].

2.3 Hypervisor

Sebuah hypervisor adalah perangkat lunak atau perangkat keras yang digunakan untuk menciptakan dan mengelola mesin virtual. Juga dikenal sebagai *'Virtual Machine Monitor'* atau VMM, hypervisor memungkinkan satu server fisik menjalankan beberapa mesin virtual, mengatasi keterbatasan satu sistem operasi pada satu server. Hypervisor dapat berupa perangkat keras atau program, dan fungsinya adalah menciptakan, memonitor, dan mengelola mesin-mesin virtual[7].



Gambar 2.2: Hypervisor Type 1 vs Type 2

Pada Gambar 2.2 diperlihatkan diagram sederhana perbedaan daripada Hypervisor tipe 1 dan Hypervisor tipe 2. Hypervisor tipe 1 dan tipe 2 merupakan perangkat lunak yang digunakan untuk menjalankan satu atau lebih *Virtual Machine* (VM) pada satu mesin fisik. *Virtual Machine* adalah replika digital dari mesin fisik, menciptakan lingkungan komputasi terisolasi yang pengguna alami sebagai sepenuhnya independen dari perangkat keras yang mendasarinya[8]. Hypervisor mengelola dan mengalokasikan sumber daya fisik ke VM dan berkomunikasi dengan perangkat keras di latar belakang. Hypervisor tipe 1 ditempatkan di atas server tanpa sistem operasi dan memiliki akses langsung ke sumber daya perangkat keras, sehingga dikenal juga sebagai *bare metal* hypervisor. Sebaliknya, hypervisor tipe 2 adalah

aplikasi yang diinstal pada sistem operasi host dan juga dikenal sebagai *hosted* atau *embedded* hypervisor[8].

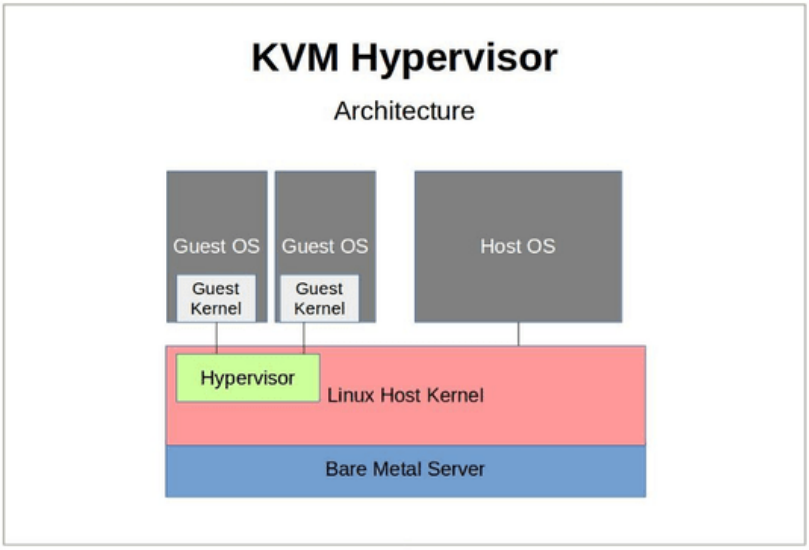
2.4 KVM *(Kernel-Based Virtual Machine)*

KVM (Kernel-based Virtual Machine) merupakan modul virtualisasi *open source* dan gratis dalam kernel Linux yang memungkinkan kernel berfungsi sebagai hypervisor. KVM adalah hypervisor tipe 1 atau biasa juga disebut sebagai hypervisor *bare metal*. KVM memungkinkan mesin host menjalankan beberapa lingkungan virtual terisolasi yang disebut sebagai guest atau *Virtual Machine* (VM).KVM (Kernel-based Virtual Machine) merupakan modul virtualisasi sumber terbuka dan gratis dalam kernel Linux yang memungkinkan kernel berfungsi sebagai hypervisor. Ini adalah tipe-1 (bare-metal) hypervisor yang memungkinkan mesin host menjalankan beberapa lingkungan virtual terisolasi yang disebut sebagai guest atau mesin virtual (VM).

KVM telah disatukan ke dalam kernel Linux utama pada versi 2.6.20, yang dirilis pada 5 Februari 2007. Untuk dapat berjalan, KVM memerlukan prosesor dengan ekstensi virtualisasi perangkat keras, seperti Intel VT atau AMD-V. KVM menyediakan abstraksi perangkat tetapi tidak ada emulasi prosesor. Modul ini mengekspos antarmuka /dev/kvm, yang dapat digunakan oleh mode pengguna untuk menyiapkan ruang alamat VM guest.

KVM mendukung virtualisasi perangkat keras untuk berbagai sistem operasi guest, termasuk BSD, Solaris, Windows, Haiku, ReactOS, Plan 9, dan lainnya. Sebagai bagian dari kernel Linux, KVM langsung mengambil manfaat dari setiap fitur baru, perbaikan, dan kemajuan Linux tanpa perlu rekayasa tambahan.

Dalam pengaturan KVM, CPU virtual dari VM diimplementasikan sebagai thread (disebut ”virtual CPU” atau vcpu) yang dijadwalkan oleh *Linux Kernel Scheduler*. KVM pada dasarnya adalah pengemudi untuk ekstensi virtualisasi processor. Setiap kali *Linux Scheduler* memilih tugas untuk dijalankan pada CPU fisik, dan tugas tersebut dikerjakan oleh CPU virtual dari VM, KVM ”dihubungi” untuk memastikan bahwa yang sebenarnya berjalan di perangkat keras adalah program dari OS guest[9].

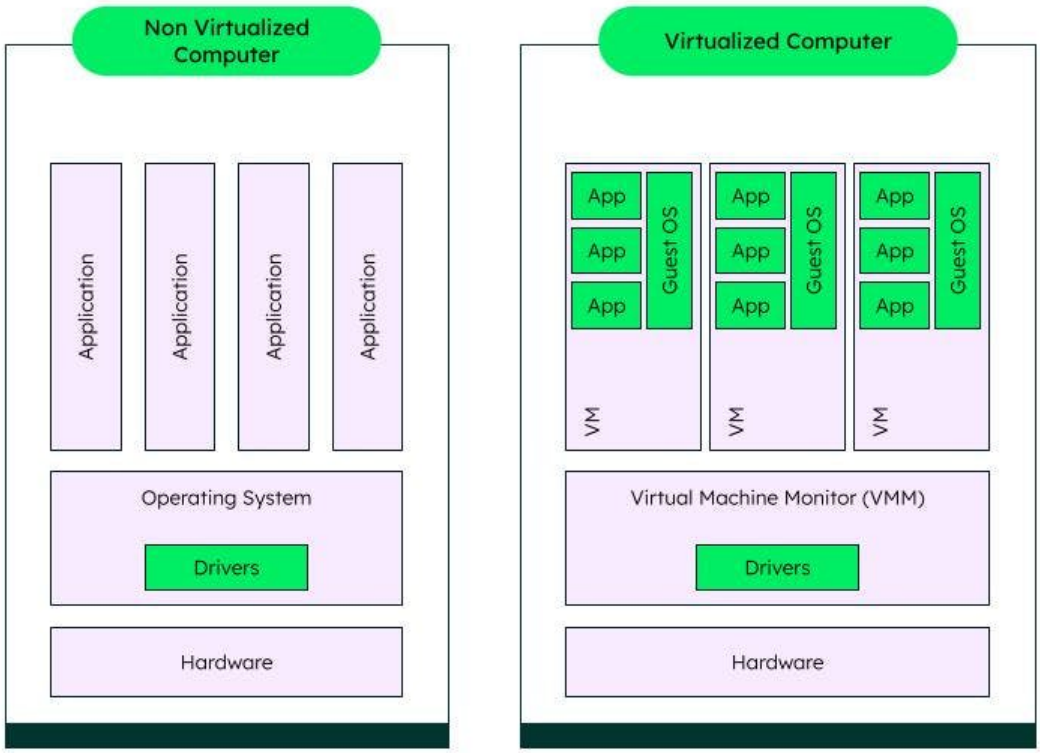


Gambar 2.3: Hypervisor Type 1 vs Type 2

Pada Gambar 2.3 adalah struktur daripada hypervisor KVM. KVM berjalan langsung pada kernel linux host dan membagikan resourcenya kepada guest *Virtual Machine*, sedangkan sistem operasi host berjalan langsung diatas kernel host. Sistem operasi host dapat melakukan konfigurasi kepada *Virtual Machine* yang menggunakan hypervisor miliknya.

2.5 *Virtual Machine*

Virtual Machine (VM) adalah sebuah lingkungan komputasi yang dibuat secara virtual di dalam sebuah komputer fisik (host)[10]. VM ini berfungsi seperti sebuah komputer independen yang dapat menjalankan sistem operasi dan aplikasi seperti komputer fisik biasa, tetapi semuanya berjalan dalam lingkungan virtual yang terisolasi di dalam host fisik. Setiap *Virtual Machine*(VM) menjalankan sistem operasi secara independen dan beroperasi terpisah dari VM lainnya, bahkan ketika semuanya berjalan pada host yang sama. Perangkat lunak yang dikenal sebagai hypervisor atau *Virtual Machine Monitor*(VMM) memungkinkan kita untuk menjalankan berbagai sistem operasi yang berbeda secara bersamaan pada *Virtual Machine*yang berbeda. VM memiliki beberapa keunggulan dibandingkan dengan mesin fisik, termasuk kemampuan untuk menjalankan beberapa lingkungan sistem operasi pada satu komputer fisik, mendukung aplikasi *legacy*, dan menyediakan opsi pemulihan dalam situasi bencana. VM digunakan untuk berbagai keperluan, seperti *Cloud Computing*, pengujian sistem operasi baru, dan menjalankan beberapa aplikasi pada satu mesin fisik[11].



Gambar 2.4: Perbedaan Komputasi Tanpa Virtualisasi dengan Virtualisasi

Terdapat perbedaan dalam cara berjalan antara komputer tanpa virtualisasi (host) dan komputer yang divirtualisasi (guest) atau *Virtual Machine*. Pada Gambar 2.4, dapat diperlihatkan bahwa pada komputer tanpa virtualisasi, sistem operasinya berjalan langsung di atas perangkat keras, dengan driver yang berfungsi sebagai jembatan, sehingga aplikasi dapat berjalan langsung di atas sistem operasi tersebut. Di sisi lain, pada komputer yang divirtualisasi, sistem operasinya berjalan di atas VMM atau Virtual Machine Monitor, di mana satu VMM dapat menjalankan banyak sistem operasi guest.

2.6 virsh

virsh adalah *command line tool* yang merupakan bagian dari *library* libvirt dan secara utama digunakan untuk mengelola dan berinteraksi dengan teknologi virtualisasi pada sistem Linux, khususnya KVM (Kernel-based Virtual Machine) dan QEMU (Quick Emulator), dan mendukung hypervisor lainnya seperti Xen, LXC, OpenVZ, VirtualBox, dan VMware ESX[12].

Secara sederhana penggunaan virsh adalah seperti ini

```
virsh [OPTION]... <command> <domain> [ARG]...
```

Pada command ini, domain adalah ID domain numerik, atau nama domain, atau UUID domain; dan ARGS adalah opsi khusus dari command. Setiap command yang dimulai dengan # dianggap sebagai komentar dan diabaikan oleh sistem, semua perintah yang tidak dikenali akan didiagnosis.

2.7 *Streaming SIMD Extensions*

Streaming SIMD Extensions (SSE) adalah instruksi yang dibawa oleh Intel pada tahun 1999 dengan prosesor Pentium III dengan tujuan untuk meningkatkan kinerja prosesor arsitektur x86. Awalnya dikenal sebagai Katmai New Instructions (KNI) selama pengembangan, SSE secara signifikan meningkatkan kemampuan pemrosesan multimedia dan grafis dibanding pendahulunya, MMX. SSE mencakup 70 instruksi baru yang dirancang untuk meningkatkan tugas seperti pengolahan gambar, video 3D, audio dan video streaming, serta pengenalan suara[13].

Tidak seperti pendahulunya, MMX menggunakan unit floating-point standar yang sama, sedangkan SSE menggunakan unit terpisah dalam prosesor untuk perhitungan floating-point, yang memungkinkan pemrosesan yang lebih efisien. SSE mendukung operasi floating-point SIMD (Single Instruction Multiple Data) single precision, yang sangat penting untuk mengatasi bottleneck dalam pemrosesan grafis 3D. Teknologi ini memungkinkan satu instruksi untuk melakukan hingga empat operasi floating-point secara bersamaan, meningkatkan kecepatan dan efisiensi pemrosesan[13].

Manfaat SSE yang utama terlihat dalam decoding MPEG2, format standar untuk video DVD, memungkinkan prosesor yang dilengkapi SSE untuk menangani decoding tersebut di perangkat lunak dengan kecepatan penuh tanpa perangkat keras tambahan. SSE juga meningkatkan penggunaan CPU dan kinerja perangkat lunak pengenalan suara, memberikan akurasi yang lebih tinggi dan waktu respons yang lebih cepat. Untuk sepenuhnya memanfaatkan kemampuan SSE, aplikasi harus diset secara khusus agar SSE-aware, sebuah fitur yang sekarang umum dalam banyak aplikasi grafis dan berhubungan dengan suara. SSE adalah perluasan dari MMX, yang berarti bahwa prosesor yang mendukung SSE juga kompatibel dengan instruksi MMX, hal ini memastikan *backward compatibility* dengan aplikasi yang mendukung MMX[13].

SSE telah berkembang seiring waktu dengan versi seperti SSE2, SSE3, dan SSE4, masing-masing membawa fitur baru dan meningkatkan kemampuan pemrosesan secara keseluruhan. Ekstensi ini telah banyak diadopsi dalam pengembangan perangkat lunak, terutama dalam bidang di mana pemrosesan paralel sangat pent-

ing.

1. SSE
2. SSE2

   SSE2 adalah perluasan dari kemampuan SIMD (Single Instruction, Multiple Data) yang awalnya diperkenalkan dengan set instruksi SSE (Streaming SIMD Extensions). Pertama kali diperkenalkan oleh Intel dengan prosesor Pentium 4[14]. Perbedaan utama dan peningkatan dalam SSE2 dibandingkan dengan pendahulunya SSE meliputi:

   (a) Rentang Jenis Data yang Lebih Luas

   SSE2 memperluas kemampuan SIMD untuk beroperasi pada data integer 64-bit dan data floating-point presisi ganda (SSE terutama berfokus pada operasi floating-point presisi tunggal).

   (b) Peningkatan Set Instruksi

   SSE2 menambahkan instruksi dan kemampuan baru, meningkatkan pengolahan dan manipulasi berbagai jenis data, termasuk integer terpak (packed integers) dan angka floating-point presisi ganda.

   (c) Peningkatan Kinerja untuk Aplikasi Ilmiah dan Multimedia

   Dengan menyediakan operasi pada tipe data yang lebih besar dan set operasi yang lebih luas, SSE2 meningkatkan efisiensi dan kinerja aplikasi yang berhubungan dengan perhitungan matematika kompleks, grafik 3D, pengkodean video, dan tugas multimedia lainnya.

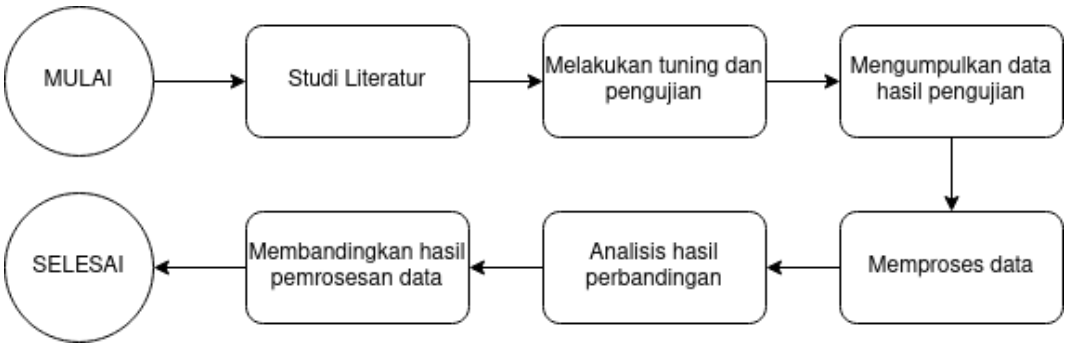
   Penggabungan SSE2 dalam arsitektur AMD64 menyoroti pentingnya dalam komputasi modern, terutama untuk aplikasi yang memerlukan kinerja tinggi dalam pengolahan numerik dan multimedia. Dokumen ABI kemungkinan mengasumsikan keakraban dengan konsep-konsep ini, lebih berfokus pada detail implementasi dalam arsitektur AMD64 daripada menjelaskan perbedaan fundamental antara SSE dan SSE2[14].

3. SSE3
4. SSSE3
5. SSE4 [15]

BAB 3
METODE PENGUJIAN TUNING HYPERVISOR KVM

Penelitian ini bertujuan untuk mengevaluasi konfigurasi Hypervisor KVM yang disediakan secara langsung dalam konteks penggunaan *Virtual Machine* pada Apache Cloudstack. Tujuan utama adalah untuk memastikan bahwa *Virtual Machine* dapat mencapai kinerja yang optimal sesuai dengan spesifikasi sistem yang digunakan. Untuk mencapai tujuan ini, Penulisakan melakukan penyesuaian konfigurasi atau tuning pada Hypervisor KVM dan kemudian membandingkannya dengan kondisi awal yang tidak mengalami tuning. Hasil pengujian akan dianalisis untuk menentukan apakah ada perbedaan signifikan dalam kinerja antara Hypervisor KVM yang telah dituning dengan yang belum dituning.

3.1 Tahapan Penelitian



Gambar 3.1: Tahapan Penelitian

Penelitian ini akan dilakukan dalam beberapa tahap sesuai dengan diagram di gambar 3.1. Tahap pertama melibatkan studi literatur untuk mengumpulkan semua informasi yang diperlukan untuk penelitian ini. Studi literatur ini penting karena memberikan dasar pengetahuan yang kuat tentang topik yang akan diteliti, memungkinkan Penulisuntuk memahami konteks dan kerangka kerja yang relevan.

Selanjutnya, penelitian melibatkan tuning terhadap Hypervisor KVM dan pengujian dilakukan dengan mengkompresi video menggunakan aplikasi Handbrake. Pengujian dengan Handbrake bertujuan untuk mengukur efisiensi dan kinerja Hypervisor KVM setelah tuning. Hasil pengujian Handbrake ini akan digunakan untuk membandingkan waktu kompresi video antara Hypervisor KVM yang tidak dituning dengan yang sudah dituning. Hasil dari perbandingan ini akan memberikan

gambaran mengenai konfigurasi yang paling optimal dan sesuai, serta membantu dalam pemahaman lebih lanjut tentang pengaruh tuning terhadap performa Hypervisor KVM dalam konteks kompresi video.

3.2 Tuning Hypervisor KVM

Tuning Hypervisor KVM yang dilakukan adalah dengan menambahkan flag instruksi yang tidak terdapat pada konfigurasi KVM default, dimana flag default dari KVM yang mendukung kompresi video hanyalah sse, dan sse2.

fpu vme **de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2** ht **syscall nx** mmxext fxsr_opt pdpe1gb rdtscp **lm** constant_tsc **rep_good** acc_power **nopl** nonstop_tsc **cpuid extd_apicid** aperfmperf **pni** pclmulqdq monitor ssse3 fma **cx16** sse4_1 sse4_2 movbe popcnt aes xsave avx f16c **lahf_lm** cmp_legacy **svm** extapic cr8_legacy abm sse4a misalignsse **3dnowprefetch** osvw ibs xop skinit wdt lwp fma4 tce nodeid_msr tbm topoext perfctr_core perfctr_nb bpext ptsc mwaitx cpb hw_pstate ssbd **vmmcall** fsgsbase bmi1 avx2 smep bmi2 xsaveopt arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vmload vgif overflow_recov

Gambar 3.2: Seluruh flag instruksi di Host

Terlihat pada gambar 3.2 adalah keseluruhan flag instruksi dari sistem host, sedangkan text yang ditajamkan adalah flag instruksi bawaan default dari hypervisor KVM. Dari sini flag yang berguna untuk melakukan kompresi video dan ditajamkan hanyalah sse dan sse2, hal ini menandakan bahwa KVM tidak mengaktifkan keseluruhan flag instruksi meskipun pada sistem host sudah mendukung flag instruksinya. Beberapa flag yang berguna untuk melakukan kompresi video dan tidak dinyalakan oleh hypervisor KVM adalah SSE3, SSE4_1, dan SSE4_2.

Kita dapat mengaktifkan flag tersebut untuk digunakan oleh *Virtual Machine* kita yang menggunakan hypervisor KVM. Untuk melakukan tuning kita dapat menggunakan tools virsh untuk mengubah konfigurasi *Virtual Machine*dari CloudStack. Secara default konfigurasi dari *Virtual Machine*di CloudStack akan seperti ini:

```
<domain type='kvm'>

        ...

        <cpu mode='custom' match='exact' check='full'>
                <model fallback='forbid'>qemu64</model>
                <feature policy='require' name='x2apic'/>
                <feature policy='require' name='hypervisor'/>
                <feature policy='require' name='lahf_lm'/>
        </cpu>

        ...

</domain>
```

Kode 3.1: Konfigurasi default dari KVM

Terlihat dari kode 3.1 bahwa feature yang dinyalakan tidak memuat sse3, sse4_1, dan sse4_2. Untuk itu kita dapat menambahkan fitur tersebut sehingga *Virtual Machine*dapat menggunakan flag instruksi tersebut. Perubahahan yang akan kita lakukan akan seperti ini:

```
<domain type='kvm'>

        ...

        <cpu mode='custom' match='exact' check='full'>
                <model fallback='forbid'>qemu64</model>
                <feature policy='require' name='x2apic'/>
                <feature policy='require' name='hypervisor'/>
                <feature policy='require' name='lahf_lm'/>
                <feature policy='require' name='sse3'/>
                <feature policy='require' name='sse4_1'/>
                <feature policy='require' name='sse4_2'/>
        </cpu>

        ...

</domain>
```

Kode 3.2: Konfigurasi tuning KVM

3.3 Pengujian Kompresi Video Dengan HandBrake

Untuk menguji dan membandingkan kinerja Hypervisor KVM sebelum dan sesudah tuning, Penulis akan menggunakan software HandBrake, sebuah software kompresi video yang sudah banyak digunakan. Proses ini melibatkan penggunaan HandBrake untuk kompresi video pada dua lingkungan *Virtual Machine* yang berbeda, yang mana satu menggunakan KVM yang belum di-tuning dan yang lainnya menggunakan KVM yang telah di-tuning. Tujuan dari pengujian ini adalah untuk mengevaluasi sejauh mana tuning Hypervisor KVM dapat mempengaruhi kecepatan kompresi video.

HandBrake dipilih karena kemampuannya yang baik dalam mengkompresi video dan mudah untuk digunakan[16]. Dalam pengujian ini, video yang akan digunakan adalah ”COSTA RICA IN 4K 60fps HDR (ULTRA HD)” yang diunduh dari YouTube. Video ini dipilih karena resolusi tingginya, yang memberikan tantangan yang baik untuk menguji efisiensi kompresi video pada kedua lingkungan *Virtual Machine*.

Parameter penting yang akan diukur dalam pengujian ini adalah:

1. ***Time Elapsed***

   Time Elapsed atau berapa lama waktu yang dibutuhkan oleh masing-masing *Virtual Machine* untuk menyelesaikan proses kompresi video.

2. ***Structural Similarity Index Measure Score***

   Structural Similarity Index Measure adalah cara untuk menilai kesamaan dari 2 buah gambar, untuk melihat apakah terjadi penurunan kualitas atau perbedaan yang besar antara gambar yang belum di compress dengan gambar yang sudah di compress.

Data ini yang akan memberikan gambaran yang jelas mengenai perbedaan kinerja antara KVM yang telah di-tuning dengan yang belum.

Spesifikasi dari *Virtual Machine* yang akan digunakan dalam pengujian ini adalah sebagai berikut:

- **OS**: Ubuntu Server 22.04 LTS
- **CPU**: 1 CPU x 1.00 Ghz
- **Memory**: 1024 MB

3.4 Analisis Pengujian

BAB 4
STRUKTUR BERKAS

@todo
tambahkan kata-kata pengantar bab 1 disini

4.1 thesis.tex

Berkas ini berisi seluruh berkas Latex yang dibaca, jadi bisa dikatakan sebagai berkas utama. Dari berkas ini kita dapat mengatur bab apa saja yang ingin kita tampilkan dalam dokumen.

4.2 laporan_setting.tex

Berkas ini berguna untuk mempermudah pembuatan beberapa template standar. Anda diminta untuk menuliskan judul laporan, nama, npm, dan hal-hal lain yang dibutuhkan untuk pembuatan template.

4.3 istilah.tex

Berkas istilah digunakan untuk mencatat istilah-istilah yang digunakan. Fungsinya hanya untuk memudahkan penulisan. Pada beberapa kasus, ada kata-kata yang harus selalu muncul dengan tercetak miring atau tercetak tebal. Dengan menjadikan kata-kata tersebut sebagai sebuah perintah LaTeX tentu akan mempercepat dan mempermudah pengerjaan laporan.

4.4 hype.indonesia.tex

Berkas ini berisi cara pemenggalan beberapa kata dalam bahasa Indonesia. LaTeX memiliki algoritma untuk memenggal kata-kata sendiri, namun untuk beberapa kasus algoritma ini memenggal dengan cara yang salah. Untuk memperbaiki pemenggalan yang salah inilah cara pemenggalan yang benar ditulis dalam berkas hype.indonesia.tex.

4.5 pustaka.tex

Berkas pustaka.tex berisi seluruh daftar referensi yang digunakan dalam laporan. Anda bisa membuat model daftar referensi lain dengan menggunakan bibtex. Untuk mempelajari bibtex lebih lanjut, silahkan buka http://www.bibtex.org/Format. Untuk merujuk pada salah satu referensi yang ada, gunakan perintah \cite, e.g. \cite{lankton2008introduction} yang akan akan memunculkan [**?**]

4.6 bab[1 - 6].tex

Berkas ini berisi isi laporan yang Anda tulis. Setiap nama berkas e.g. bab1.tex merepresentasikan bab dimana tulisan tersebut akan muncul. Sebagai contoh, kode dimana tulisan ini dibaut berada dalam berkas dengan nama bab4.tex. Ada enam buah berkas yang telah disiapkan untuk mengakomodir enam bab dari laporan Anda, diluar bab kesimpulan dan saran. Jika Anda tidak membutuhkan sebanyak itu, silahkan hapus kode dalam berkas thesis.tex yang memasukan berkas LaTeX yang tidak dibutuhkan; contohnya perintah \include{bab6.tex} merupakan kode untuk memasukan berkas bab6.tex kedalam laporan.

4.7 Penulisan *code* atau *pseudocode* program

4.7.1 *Inline*

Dengan perintah `\verb`: `System.out.println("Hello, World");`
Dengan perintah *custom* `\code`: `System.out.println("Hello, World");` Dengan perintah `\mintinline`: `System.out.println("Hello, World");`

4.7.2 *Multiline*

Dengan perintah `verbatim`:

```
public class HelloWorld {
    public static void main(String[] args) {
        // Prints "Hello, World" to the terminal window.
        System.out.println("Hello, World");
    }
}
```

Dengan perintah `minted`: Kode 4.1

```
def binary_accuracy(y_true, y_pred):
    return K.mean(K.equal(y_true, K.round(y_pred)), axis=-1)


def categorical_accuracy(y_true, y_pred):
    return K.cast(K.equal(K.argmax(y_true, axis=-1),
                          K.argmax(y_pred, axis=-1)),
                  K.floatx())


def sparse_categorical_accuracy(y_true, y_pred):
    # reshape in case it's in shape (num_samples, 1) instead of
    ↪ (num_samples,)
    if K.ndim(y_true) == K.ndim(y_pred):
        y_true = K.squeeze(y_true, -1)
    # convert dense predictions to labels
    y_pred_labels = K.argmax(y_pred, axis=-1)
    y_pred_labels = K.cast(y_pred_labels, K.floatx())
    return K.cast(K.equal(y_true, y_pred_labels), K.floatx())


def top_k_categorical_accuracy(y_true, y_pred, k=5):
    return K.mean(K.in_top_k(y_pred, K.argmax(y_true, axis=-1), k),
    ↪ axis=-1)


def sparse_top_k_categorical_accuracy(y_true, y_pred, k=5):
    # If the shape of y_true is (num_samples, 1), flatten to
    ↪ (num_samples,)
    return K.mean(K.in_top_k(y_pred, K.cast(K.flatten(y_true),
    ↪ 'int32'), k),
                  axis=-1)
```

Kode 4.1: An excerpt from keras: https://github.com/keras-team/keras/blob/master/keras/metrics.py

Konfigurasi tampilan bisa dilakukan di `uithesis.sty` dengan referensi dokumentasi di https://github.com/gpoore/minted/blob/master/source/minted.pdf

BAB 5
PERINTAH DALAM UITHESIS.STY

@todo

Tambahkan kata-kata pengantar bab 5 disini.

5.1 Mengubah Tampilan Teks

Beberapa perintah yang dapat digunakan untuk mengubah tampilan adalah:

- \f
  Merupakan alias untuk perintah \textit, contoh *contoh hasil tulisan*.
- \bi
  Contoh hasil tulisan.
- \bo
  Contoh hasil tulisan.
- \m
  Contoh hasil tulisan: $\alpha \neq \boldsymbol{\alpha}$
- \code
  `Contoh hasil tulisan`.

5.2 Memberikan Catatan

Ada dua perintah untuk memberikan catatan penulisan dalam dokumen yang Anda kerjakan, yaitu:

- \todo
  Contoh:

  @todo

  Contoh bentuk todo.

- \todoCite

  Contoh:

  @todo
  Referensi

5.3 Menambah Isi Daftar Isi

Terkadang ada kebutuhan untuk memasukan kata-kata tertentu kedalam Daftar Isi. Perintah \addChapter dapat digunakan untuk judul bab dalam Daftar isi. Contohnya dapat dilihat pada berkas thesis.tex.

5.4 Memasukan PDF

Untuk memasukan PDF dapat menggunakan perintah \inpdf yang menerima satu buah argumen. Argumen ini berisi nama berkas yang akan digabungkan dalam laporan. PDF yang dimasukan degnan cara ini akan memiliki header dan footer seperti pada halaman lainnya.

Untitled

Ini adalah berkas pdf yang dimasukan dalam dokumen laporan.

Cara lain untuk memasukan PDF adalah dengan menggunakan perintah \putpdf dengan satu argumen yang berisi nama berkas pdf. Berbeda dengan perintah sebelumnya, PDF yang dimasukan dengan cara ini tidak akan memiliki footer atau header seperti pada halaman lainnya.

Ini adalah berkas pdf yang dimasukan dalam dokumen laporan.

5.5 Membuat Perintah Baru

Ada dua perintah yang dapat digunakan untuk membuat perintah baru, yaitu:

- \Var
  Digunakan untuk membuat perintah baru, namun setiap kata yang diberikan akan diproses dahulu menjadi huruf kapital. Contoh jika perintahnya adalah \Var{adalah} makan ketika perintah \Var dipanggil, yang akan muncul adalah ADALAH.

- \var
  Digunakan untuk membuat perintah atau baru.

BAB 6
BAB ENAM

@todo
tambahkan kata-kata pengantar bab 6 disini

BAB 7
KESIMPULAN DAN SARAN

@todo
Tambahkan kesimpulan dan saran terkait dengan perkerjaan yang dilakukan.

7.1 Kesimpulan

7.2 Saran

DAFTAR REFERENSI

[1] P. Mell and T. Grance. The nist definition of cloud computing. *National Institute of Standards and Technology, Information Technology Laboratory*, 2011.

[2] Apache Software Foundation. Apache cloudstack: About. https://cloudstack.apache.org/about.html. Accessed: 2023-11-19.

[3] M. Scarfone, K. Souppaya and P. Hoffman. Guide to security for full virtualization technologies. *National Institute of Standards and Technology, Information Technology Laboratory*, 2011.

[4] Amazon Web Services Inc. What is kvm (kernel-based virtual machine)? https://aws.amazon.com/what-is/kvm/. Accessed: 2023-11-19.

[5] Rajkumar Buyya, James Broberg, and Andrzej M. Goscinski. *Cloud computing: Principles and Paradigms*. Wiley, March 2011.

[6] TechTarget. Cloudstack. https://www.techtarget.com/whatis/definition/CloudStack, July 2019. Accessed: 2023-11-20.

[7] Jordan MacPherson. What is a hypervisor? - types, benefits and how does it work? https://www.parkplacetechnologies.com/blog/what-is-hypervisor-types-benefits/, April 2023. Accessed: 2023-11-20.

[8] Amazon Web Services Inc. What's the difference between type 1 and type 2 hypervisors? https://aws.amazon.com/compare/the-difference-between-type-1-and-type-2-hypervisors/. Accessed: 2023-11-20.

[9] Luca Abeni and Dario Faggioli. Using xen and kvm as real-time hypervisors. *Journal of Systems Architecture*, 106:101709, June 2020.

[10] J.V. Pradilla and C.E. Palau. *Micro Virtual Machines (MicroVMs) for Cloud-assisted Cyber-Physical Systems (CPS)*, page 125–142. Elsevier, 2016.

[11] IBM. What are virtual machines? — IBM — ibm.com. https://www.ibm.com/topics/virtual-machines. [Accessed 30-11-2023].

[12] libvirt: virsh — libvirt.org. https://libvirt.org/manpages/virsh.html. [Accessed 06-12-2023].

[13] SSE (Streaming SIMD Extensions) — Microprocessor Types and Specifications — InformIT — informit.com. https://www.informit.com/articles/article.aspx?p=130978&seqNum=8. [Accessed 17-12-2023].

[14] Andreas Jaeger Mark Mithcell Michael Matz, Jan Hubicka. System v application binary interface amd64 architecture processor supplement. https://refspecs.linuxbase.org/elf/x86_64-abi-0.99.pdf, 2010. [Accessed 17-12-2023].

[15] Shu-Ming Tseng, Yu-Chin Kuo, Yen-Chih Ku, and Yueh-Teng Hsu. Software viterbi decoder with sse4 parallel processing instructions for software dvb-t receiver. In *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*. IEEE, 2009.

[16] Fernando Gomez Folgar, Antonio Garcia Loureiro, Tomas Fernandez Pena, J Isaac Zablah, and Natalia Seoane. Implementation of the KVM hypervisor on several cloud platforms: Tuning the apache CloudStack agent, August 2014.