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.

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	:	**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	:		( )
Penguji	:		( )
Penguji	:		( )

Ditetapkan di	:	Depok
Tanggal	:	Desember 2023

KATA PENGANTAR

Dengan mengucap syukur kehadirat Allah SWT, saya dapat menyelesaikan skripsi ini sebagai salah satu syarat untuk menyelesaikan program studi Teknik Komputer di Universitas Indonesia. Penulisan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak, yang dengan tulus saya ucapkan terima kasih.

Ucapan terima kasih yang sebesar-besarnya saya persembahkan kepada kedua orang tua saya, yang telah memberikan doa, dukungan moral, dan finansial selama proses pembuatan skripsi ini. Tak lupa kepada dosen pembimbing, Bapak Yan Maraden, S.T., M.T. atas bimbingan, arahan, dan kritik yang membangun selama proses penulisan skripsi ini.

Saya juga ingin mengucapkan terima kasih kepada teman-teman di Teknik Komputer Universitas Indonesia yang telah memberikan semangat dan dukungan. Pengalaman dan ilmu yang saya peroleh selama proses pembelajaran di universitas ini sangat berharga dan menjadi motivasi dalam penyelesaian skripsi ini.

Skripsi ini berjudul Analisis Performa Tuning Hypervisor KVM *Streaming SIMD Extension* (SSE) untuk Kompresi Video pada *Virtual Machine* di Apache Cloudstackyang merupakan upaya saya untuk memberikan kontribusi dalam bidang cloud serta sebagai aplikasi dari teori yang telah saya pelajari. Saya berharap skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan, khususnya bagi para peneliti dan mahasiswa yang berkecimpung dalam bidang yang sama.

Akhir kata, semoga skripsi ini dapat memberikan manfaat dan inspirasi bagi kita semua. Kritik dan saran yang membangun selalu saya harapkan untuk perbaikan di masa yang akan datang.

Depok, 29 Desember 2023

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 Non-eksklusif 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 : 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.

Penelitian ini bertujuan untuk menganalisis performa dari tuning Hypervisor Kernel-based Virtual Machine (KVM) dengan fokus pada Streaming SIMD Extensions (SSE) dalam konteks kompresi video pada Virtual Machines (VMs) yang berjalan di Apache CloudStack. Kajian ini secara khusus meneliti pengaruh penambahan flag SSE3, SSE4.1, dan SSE4.2 terhadap performa kompresi video. Dua aspek utama yang menjadi fokus adalah kecepatan kompresi dan kualitas video yang dihasilkan.

Untuk menilai kecepatan kompresi, penelitian ini membandingkan waktu kompresi antara Hypervisor KVM yang telah di-tuning dengan flag SSE tambahan dan Hypervisor KVM default. Metode yang digunakan untuk mengukur kecepatan ini adalah dengan menghitung waktu yang diperlukan untuk kompresi video pada kedua konfigurasi hypervisor tersebut.

Di sisi lain, kualitas video yang dikompresi dievaluasi menggunakan metrik Structural Similarity Index Measure (SSIM). SSIM digunakan untuk mengukur kesamaan visual antara video asli dan video hasil kompresi, memberikan gambaran tentang seberapa baik video dikompresi tanpa kehilangan kualitas visual.

Penelitian ini menghasilkan temuan penting mengenai efektivitas tuning SSE pada Hypervisor KVM dalam konteks Apache CloudStack, memberikan wawasan baru tentang optimisasi performa untuk kompresi video di lingkungan cloud. Hasil ini diharapkan dapat membantu dalam pengembangan praktik terbaik untuk konfigurasi Hypervisor KVM dalam meningkatkan efisiensi dan efektivitas kompresi video pada VM.

Kata kunci:
Tuning Hypervisor KVM, Kompresi Video, Cloud Computing, Apache CloudStack

ABSTRACT

Name	:	Ariq Pradipa Santoso
Study Program	:	Teknik Komputer
Title	:	Performance Analysis of KVM Hypervisor Tuning *Streaming SIMD Extension* (SSE) for Video Compression on *Virtual Machine* on Apache Cloudstack
Counsellor	:	Yan Maraden S.T., M.T.

This study aims to analyze the performance of tuning the Kernel-based Virtual Machine (KVM) Hypervisor, focusing on Streaming SIMD Extensions (SSE) in the context of video compression on Virtual Machines (VMs) running in Apache CloudStack. Specifically, this research examines the impact of adding SSE3, SSE4.1, and SSE4.2 flags on video compression performance. The main focuses are the speed of compression and the quality of the resulting video.

To assess compression speed, this study compares the compression time between the KVM Hypervisor tuned with additional SSE flags and the default KVM Hypervisor. The method used to measure this speed involves calculating the time required for video compression on both hypervisor configurations.

On the other hand, the quality of the compressed video is evaluated using the Structural Similarity Index Measure (SSIM) metric. SSIM is used to measure the visual similarity between the original video and the compressed video, providing an insight into how well the video is compressed without losing visual quality.

This research yields significant findings regarding the effectiveness of SSE tuning on the KVM Hypervisor in the context of Apache CloudStack, offering new insights into performance optimization for video compression in cloud environments. These results are expected to assist in developing best practices for configuring the KVM Hypervisor to enhance efficiency and effectiveness in video compression on VMs.

Key words:
KVM Hypervisor Tuning, Video Compression, Cloud Computing, Apache CloudStack

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 beberapa tugas komputasi, seperti kompresi video, enkripsi, dan juga validasi integritas data. Selanjutnya, waktu yang diperlukan untuk menyelesaikan proses komputasi 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 konfigurasi default dari Hypervisor KVM sudah dibuat dengan optimal dan efisien?
2. Apakah tuning Hypervisor KVM untuk *Virtual Machine* di Apache Cloudstack memberikan efek yang signifikan terhadap performa tugas komputasi seperti kompresi video, enkripsi AES, dan validasi integritas data dengan CRC32?
3. Bagaimana konfigurasi Hypervisor KVM dapat diatur secara optimal untuk meningkatkan efisiensi atau kecepatan dalam menangani tugas komputasi tertentu 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 Ryzen 7 4800H x64.
- RAM: 16 GB.
- Penyimpanan: 1TB pada SSD.

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, waktu enkrispsi dengan AES, dan waktu validasi integritas data dengan CRC32 antara hypervisor default dan hypervisor yang telah dituning.

1.3 Tujuan

Penelitian ini berfokus pada analisis performa hypervisor KVM setelah proses tuning, khususnya dalam penggunaan Apache CloudStack. Dengan tujuan sebagai berikut:

- Evaluasi Konfigurasi Default Hypervisor KVM

  Menilai apakah konfigurasi default hypervisor KVM telah diatur secara optimal dan efisien.

- Pemahaman Dampak Tuning terhadap Performa Sistem Operasi *Guest*

  Memahami bagaimana tuning hypervisor KVM mempengaruhi performa sistem operasi *guest*.

- Panduan Optimalisasi Hypervisor KVM

  Menyusun rekomendasi untuk optimalisasi hypervisor KVM dalam rangka meningkatkan kinerja layanan Cloud Computing Infrastructure as a Service (IaaS).

Hasil penelitian ini diharapkan tidak hanya memberikan panduan praktis tetapi juga kontribusi akademis dalam bidang optimalisasi infrastruktur cloud.

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, enkripsi AES, dan juga mengenai validasi integritas data dengan CRC32 serta evaluasi keuntungan dan kerugian yang mungkin muncul akibat hasil tuning ini.

2. Pengumpulan Data

   Pada penelitian ini, akan dilakukan tiga pengujian

   (a) Kompresi Video

   Kompresi video akan dilakukan menggunakan ffmpeg 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.

   (b) Enkripsi AES

   Enkripsi AES akan dilakukan dengan program benchmark yang dibuat Penulis dengan python pada kedua sistem. Pada masing-masing sistem, enkripsi AES akan dilakukan sebanyak 10 kali. Setelah itu, waktu rata-rata enkripsi AES pada kedua sistem akan dibandingkan dan dianalisis.

   (c) Validasi CRC32

   Validasi CRC32 akan dilakukan dengan program benchmark yang dibuat Penulis dengan python pada kedua sistem. Pada masing-masing sistem, validasi CRC32 akan dilakukan sebanyak 10 kali. Setelah itu, waktu rata-rata validasi CRC32 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 PARAMETER HYPERVISOR KVM DENGAN CLOUDSTACK AGENT

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

- BAB IV PENGUJIAN TUNING PARAMETER HYPERVISOR KVM DENGAN CLOUDSTACK AGENT

  Bab IV menjelaskan hasil penelitian yang terdiri dari pengujian tuning parameter hypervisor KVM, dan dilakukan analisis performa hypervisor KVM setelah proses tuning.

- BAB V PENUTUP

  Bab V menjelaskan kesimpulan dari penelitian dilakukan dan saran.

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].

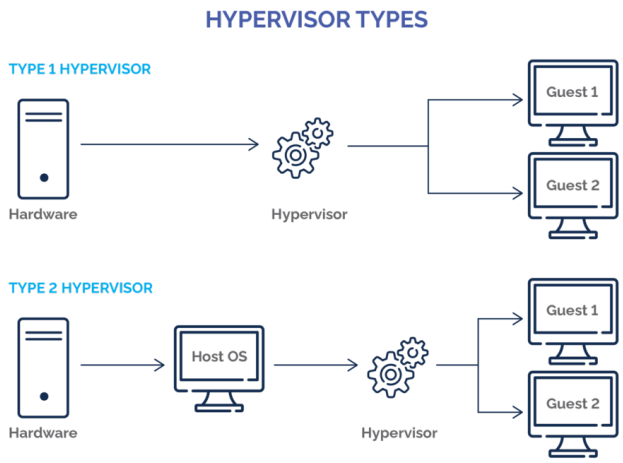
Pengguna dapat mengelola cloud mereka dengan antarmuka web yang mudah digunakan, alat *commandline*, dan/atau API RESTful yang lengkap. Selain itu,

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].

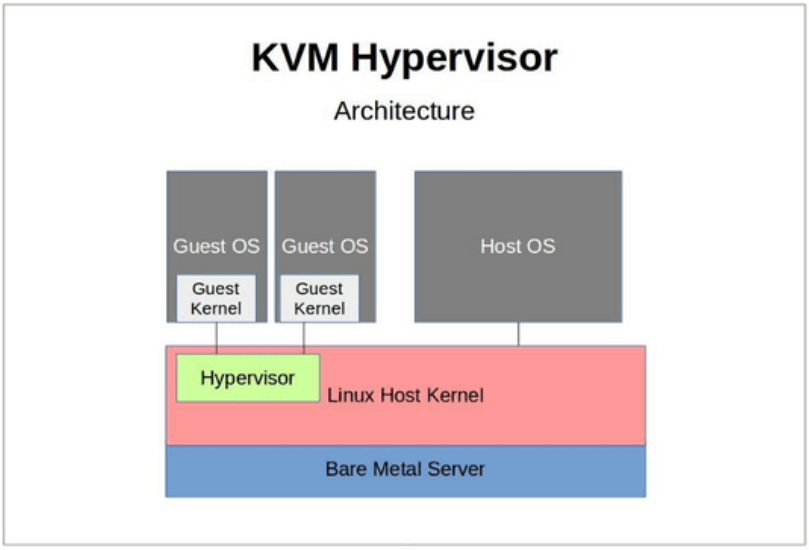
2.4 KVM *(Kernel-Based Virtual Machine)*

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].

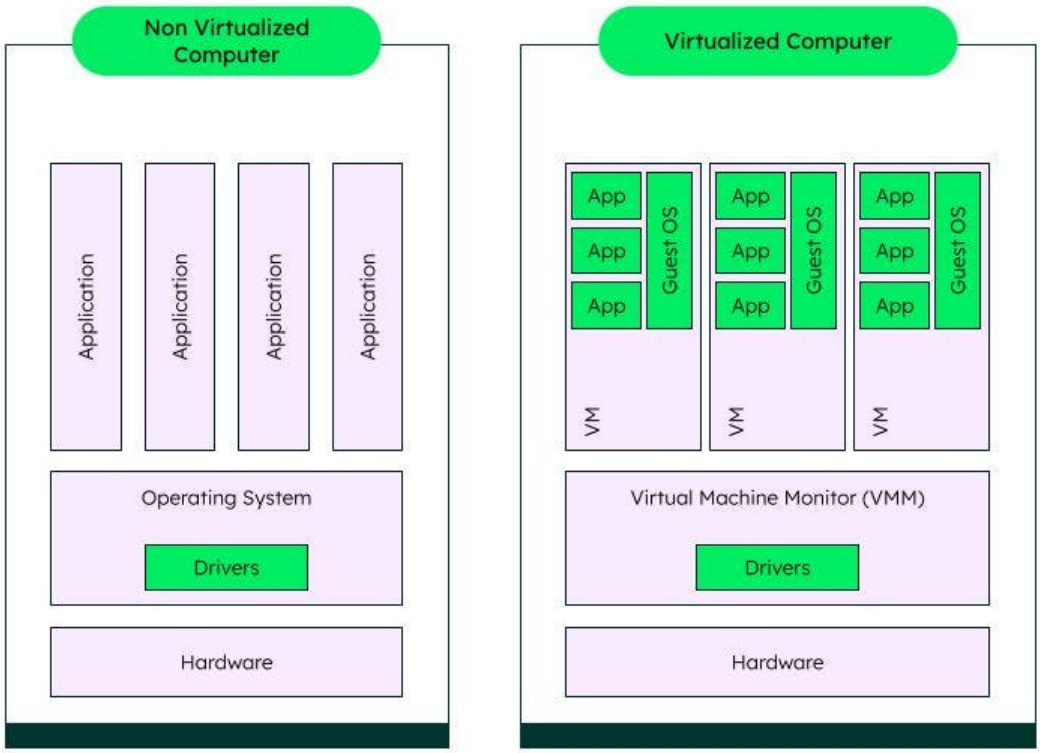


Gambar 2.3: Arsitektur Hypervisor KVM

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[13].

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[14].

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[14].

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[14].

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

penting.

2.7.1 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[15]. Perbedaan utama dan peningkatan dalam SSE2 dibandingkan dengan pendahulunya SSE meliputi:

1. 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).

2. 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.

3. 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[15].

2.7.2 SSE3

SSE3 (Streaming SIMD Extensions 3) merupakan ekstensi dari set instruksi SSE2, yang mana memperluas teknologi SSE (Streaming SIMD Extensions) asli. Perbedaan antara SSE3 dengan SSE2 antara lain:

1. 13 Set Instruksi Baru

   SSE3 menambahkan tiga belas instruksi baru ke set instruksi SSE/SSE2 yang ada[16].

2. Dukungan untuk Model Eksekusi SIMD:

   Dari 13 instruksi ini, sepuluh dirancang untuk mendukung model eksekusi SIMD (Single Instruction, Multiple Data). Model ini merupakan aspek penting dari komputasi paralel, di mana beberapa titik data diproses secara bersamaan menggunakan satu instruksi. Ini sangat bermanfaat untuk tugas-tugas seperti pengolahan grafis, perhitungan ilmiah, dan aplikasi multimedia, di mana operasi pada set data besar adalah hal yang umum[16].

3. Konversi Floating-Point ke Integer:

   Salah satu instruksi SSE3 secara spesifik mempercepat pemrograman *x87-style*. Instruksi ini berfokus pada peningkatan efisiensi dalam mengkonversi nilai floating-point menjadi integer. Hal ini bisa sangat berguna dalam skenario di mana perhitungan melibatkan tipe data floating-point dan integer[16].

4. Akselerasi Sinkronisasi Thread:

   Dua instruksi lain dalam set SSE3 ditujukan untuk meningkatkan sinkronisasi thread. Peningkatan ini sangat penting dalam aplikasi multi-thread, di mana pengelolaan eksekusi dan koordinasi beberapa thread adalah kunci untuk kinerja yang baik dan efisiensi[16].

Secara singkat, SSE3 memperluas kemampuan set instruksi SSE dan SSE2 dengan penekanan pada peningkatan eksekusi SIMD, konversi floating-point ke integer, serta sinkronisasi thread. Selain itu, SSE3 tetap mempertahankan kompatibilitas dengan jenis data dan lingkungan pemrograman yang sudah ada. Oleh karena itu, SSE3 menjadi alat yang sangat berguna bagi pengembang yang bekerja pada tugas komputasi berkinerja tinggi, terutama yang melibatkan pemrosesan paralel dan jenis data yang kompleks.

2.7.3 SSE4

SSE4, yang merupakan singkatan dari Streaming SIMD Extensions 4, adalah serangkaian instruksi yang diperkenalkan oleh Intel dalam prosesor 64-bitnya yang dibuat dengan teknologi proses 45 nm. SSE4 terdiri dari dua subset: SSE4.1 dan SSE4.2[17].

Fitur	**SSE4.1**	**SSE4.2**
Pengenalan	Bagian dari prosesor Intel 45 nm	Diperkenalkan dengan arsitektur mikro Nehalem
Instruksi	47 set instruksi baru	7 instruksi tambahan
Fokus	Peningkatan kinerja pada media, pengolahan citra, dan beban kerja 3D	Pengolahan string dan teks
Penyempurnaan Utama	Peningkatan vektorisasi kompiler, dukungan untuk perhitungan *packed dword*	Kemampuan integer SIMD yang ditingkatkan, fokus pada operasi string
Kompatibilitas	Sepenuhnya kompatibel dengan versi SSE sebelumnya	Juga kompatibel dengan versi SSE sebelumnya
Dukungan OS	Tidak memerlukan dukungan OS baru	Sama seperti SSE4.1

Tabel 2.1: Perbandingan SSE4.1 dan SSE4.2

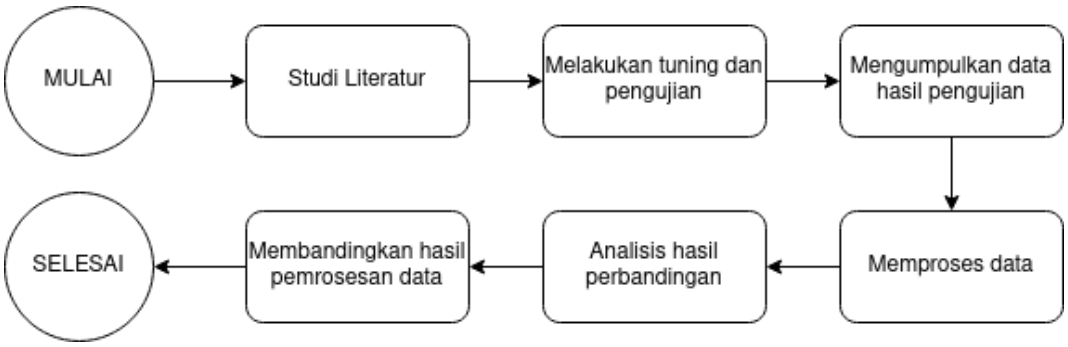
Berdasarkan informasi dalam tabel 2.1, SSE4.1 dirancang untuk meningkatkan performa dalam aplikasi media, pencitraan, dan 3D, serta menghadirkan 47 instruksi baru. Instruksi-instruksi ini berkontribusi pada peningkatan vektorisasi kompiler dan memberikan dukungan untuk perhitungan *packed dword*. Di sisi lain, SSE4.2, yang pertama kali diperkenalkan dalam prosesor dengan mikroarsitektur Nehalem, menambahkan tujuh instruksi tambahan. Fokus utama dari instruksi-instruksi tambahan ini adalah pada pemrosesan string dan teks, serta peningkatan dalam kemampuan integer SIMD[17].

SSE4 tidak memperkenalkan tipe data baru tetapi mencakup berbagai tipe instruksi seperti perkalian dword, *floating-point dot-product*, *streaming load hint*, *packed blending*, dan instruksi MIN/MAX *packed integer*. Teknologi ini bertujuan untuk meningkatkan dukungan untuk perhitungan *packed dword* dan meningkatkan throughput memori untuk jenis memori tertentu. SSE4 sepenuhnya kompatibel dengan perangkat lunak dari generasi sebelumnya dari prosesor Intel dan tidak memerlukan dukungan OS baru di luar apa yang diperlukan untuk Streaming SIMD Extensions (SSE) sebelumnya[17].

BAB 3
METODE PENGUJIAN TUNING PARAMETR HYPERVISOR KVM DENGAN CLOUDSTACK AGENT

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



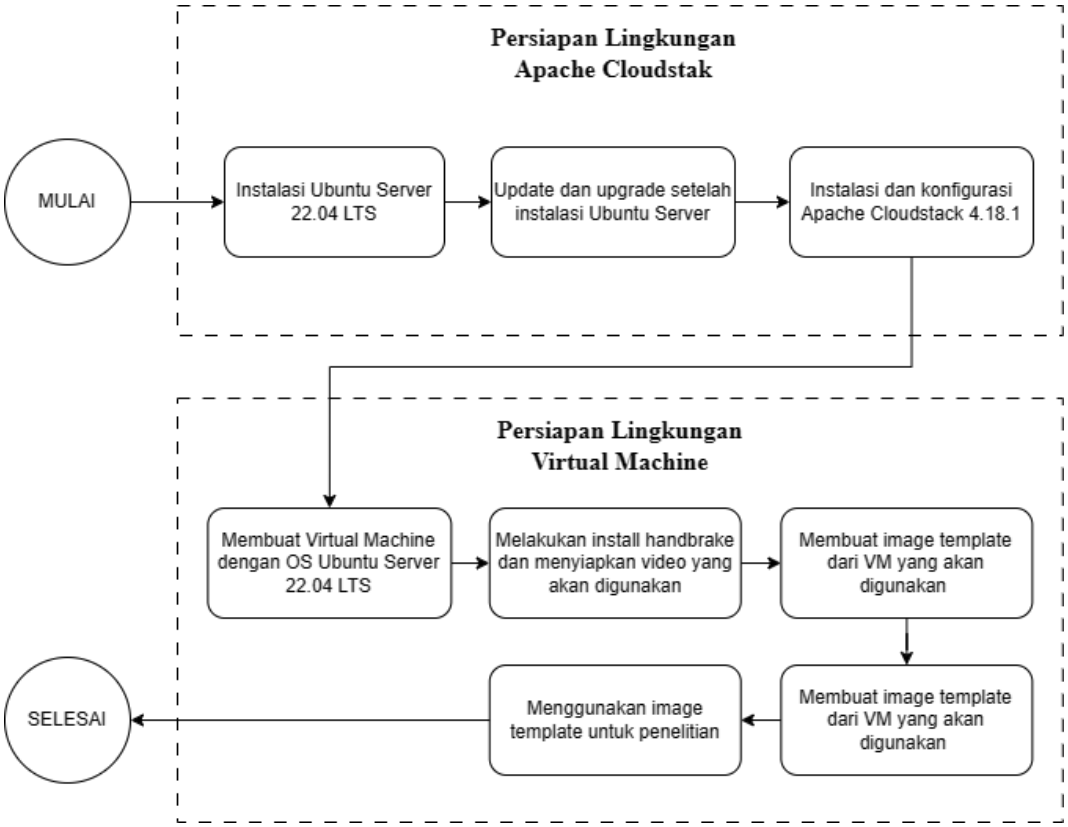
Gambar 3.1: Tahapan Penelitian

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 Persiapan Lingkungan untuk Pengujian dan Analisis



Gambar 3.2: PersiapanLingkunganPenelitian

Proses persiapan lingkungan untuk pengujian dan analisis yang tergambar dalam gambar 3.2 menggambarkan serangkaian langkah sistematis yang penting untuk memastikan integritas hasil penelitian. Proses ini dimulai dengan tahap inisiasi yang meliputi instalasi Ubuntu Server 22.04 LTS, sebuah sistem operasi yang stabil dan sering digunakan untuk keperluan server. Instalasi ini menjadi landasan dasar bagi infrastruktur yang akan dikembangkan.

Setelah sistem operasi terinstal, dilakukan pembaruan dan peningkatan sistem operasi tersebut untuk memastikan semua komponen sistem berada pada versi terkini serta keamanan sistem terjaga. Langkah ini krusial untuk menutup potensi kerentanan yang mungkin ada pada sistem. Selanjutnya, Apache Cloudstack versi

4.18.1 diinstal dan dikonfigurasi. Apache Cloudstack berfungsi sebagai platform manajemen cloud yang memungkinkan pembuatan dan pengelolaan infrastruktur cloud yang kompleks, yang merupakan elemen kunci dalam penelitian ini.

Dalam pembuatan lingkungan virtual, dibuat *Virtual Machine*yang menggunakan sistem operasi yang sama, yaitu Ubuntu Server 22.04 LTS. Pembuatan VM ini memungkinkan simulasi berbagai skenario penelitian dalam lingkungan yang terisolasi. Selanjutnya, untuk keperluan analisis dan pengujian, Handbrake diinstal, dan video yang akan digunakan pun disiapkan.

Tahapan selanjutnya merupakan pembuatan image template dari VM yang telah dikonfigurasi. Proses ini memungkinkan duplikasi VM dengan cepat dan efisien untuk pengujian berulang atau skenario analisis yang berbeda, menjamin konsistensi lingkungan penelitian. Image template ini kemudian digunakan sebagai standar untuk penelitian lebih lanjut, memastikan bahwa setiap VM yang dibuat untuk tujuan pengujian memiliki konfigurasi yang identik.

Setelah semua dilakukan hal ini menandakan bahwa lingkungan penelitian telah siap untuk diuji dan dianalisis. Kesiapan lingkungan ini esensial untuk memastikan bahwa pengujian yang dilakukan dapat diulang dan hasilnya valid.

3.3 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 3.3: Seluruh flag instruksi di Host

Terlihat pada gambar 3.3 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.

Penulis dapat mengaktifkan flag tersebut untuk digunakan oleh *Virtual Machine* yang menggunakan hypervisor KVM. Untuk melakukan tuning Penulis 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 Penulis dapat menambahkan fitur tersebut sehingga *Virtual Machine*dapat menggunakan flag instruksi tersebut. Perubahahan yang akan Penulislakukan 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

Jika sudah dilakukan penambahan fitur seperti di kode 3.2 sekarang *Virtual Machine*yang menggunakan Hypervisor KVM sudah bisa mendukung set instruksi tersebut.

3.4 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[18]. Dalam pengujian ini, video yang akan digunakan adalah ”COSTA RICA IN 4K 60fps HDR (ULTRA HD)” yang diunduh dari YouTube dengan resolusi video yang digunakan adalah 2K. 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:

- **Cloud Platform**: Apache CloudStack 4.18.1
- **OS**: Ubuntu Server 22.04 LTS
- **CPU**: 1 CPU x 1.00 Ghz
- **Memory**: 1024 MB

Perintah yang digunakan untuk melakukan kompresi video dengan handbrake adalah sebagai berikut:

```
./HandBrakeCLI -i /path/to/input.mov -o /path/to/output.mp4
-e x264 -q 28 -r 15 -B 64 -X 1280 -O
```

Dengan perintah ini Penulisakan melakukan kompresi video dengan codec x264. Codec yang dikembangkan oleh VideoLAN yang sudah banyak digunakan dikarenakan mampu untuk melakukan kompresi video yang baik dan rendahnya tingkat penurunan kualitas gambar.

3.5 Analisis Pengujian

Dalam analisis ini, waktu kompresi untuk Hypervisor KVM yang belum di-tuning akan disebut sebagai T_{default}, sementara waktu kompresi untuk Hypervisor KVM yang telah di-tuning akan disebut sebagai T_{tuned}. Waktu kompresi ini akan diambil

dari logs HandBrake yang telah melakukan kompresi video. Untuk mengetahui apakah terdapat perbedaan yang signifikan, Penulis dapat menggunakan formula speedup.

$$\text{Speedup} = \frac{T_{\text{default}}}{T_{\text{tuned}}}$$

Jika nilai speedup lebih dari 1, ini menandakan bahwa proses kompresi dengan Hypervisor KVM yang sudah di-tuning lebih cepat dibandingkan dengan yang belum di-tuning.

Pengujian akan dilakukan sebanyak kurang lebih sepuluh kali dan nantinya akan ditampilkan dalam bentuk grafik. Akan terdapat grafik *line chart* dengan dua garis dimana masing-merepresentasikan T_{default} dan T_{tuned}.

Selain menganalisis waktu kompresi video, Penulis juga akan melakukan analisis untuk menilai kualitas video yang telah dikompresi. Analisis ini akan melibatkan perbandingan nilai dari *Structural Similarity Index Measure* (SSIM) antara video yang dikompres menggunakan Hypervisor KVM default dan Hypervisor KVM yang telah di-tuning. SSIM adalah metrik yang mengukur kesamaan visual antara dua video. Semakin tinggi nilai SSIM, semakin mirip video yang dikompres dengan versi aslinya. Oleh karena itu, nilai SSIM yang lebih tinggi menunjukkan kualitas kompresi yang lebih baik, dimana integritas visual dari video asli lebih terjaga.

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] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

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

[14] 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].

[15] 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].

[16] M. Hassaballah, S. Omran, and Y. B. Mahdy. A review of simd multimedia extensions and their usage in scientific and engineering applications. *The Computer Journal*, 51(6):630–649, January 2008.

[17] Intel. Sse4 programming reference, 2007. Intel 64 and IA-32 Architectures.

[18] 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.