LAPORAN AKHIR FINAL PROJECT

PROYEK 2: MEMBANGUN HYPERVISOR MINI (SIMULASI VIRTUALISASI)

Dosen Pengampu: Ferdi Chahyadi, S.Kom, M.Cs

Disusun Oleh:

Sorhan Aria Pratama	2401020008
Ikhbal Maulana	2401020039
Ferdy Alfitra	2401020036
Cinto Aprilman Halawa	2401020038
Farael Ahmad	2401020023

ABSTRAK

Virtualisasi merupakan teknologi penting dalam sistem komputasi modern yang memungkinkan pemanfaatan sumber daya perangkat keras secara efisien melalui penggunaan mesin virtual. Teknologi ini telah menjadi fondasi bagi infrastruktur cloud computing, pusat data modern, dan berbagai layanan komputasi terdistribusi. Pada proyek ini, dilakukan pembangunan hypervisor mini berbasis QEMU/KVM dan libvirt yang mampu menjalankan Virtual Machine (VM), mengatur alokasi CPU dan memori, serta melakukan pengujian performa antar VM. QEMU/KVM dipilih sebagai solusi virtualisasi karena merupakan teknologi open-source yang mature dan memiliki performa mendekati native hardware, sementara libvirt digunakan sebagai layer manajemen untuk mempermudah administrasi VM.

Proyek dilaksanakan secara bertahap mulai dari instalasi dan konfigurasi komponen hypervisor pada sistem host, pembuatan dan konfigurasi VM dengan berbagai spesifikasi resource, hingga pelaksanaan benchmark CPU dan memori menggunakan tools standar industri. Metodologi pengembangan mencakup persiapan lingkungan virtualisasi, implementasi VM dengan variasi alokasi sumber daya, dan analisis komparatif performa untuk mengidentifikasi bottleneck serta faktor-faktor yang memengaruhi efisiensi virtualisasi. Pengujian dilakukan dengan skenario beban kerja yang berbeda untuk mensimulasikan kondisi penggunaan real-world, termasuk pengujian konkurensi antar VM dan analisis overhead virtualisasi.

Hasil akhir menunjukkan bahwa VM dapat dijalankan dan dikelola dengan baik melalui interface libvirt, serta alokasi resource dan dukungan virtualisasi hardware seperti Intel VT-x atau AMD-V sangat memengaruhi performa VM. Analisis benchmark mengungkapkan bahwa VM dengan alokasi CPU dan memori yang optimal dapat mencapai performa hingga 85-95% dari bare metal system, dengan overhead virtualisasi yang dapat diminimalkan melalui konfigurasi yang tepat. Proyek ini memberikan pemahaman mendalam tentang implementasi praktis teknologi virtualisasi dan menjadi dasar untuk pengembangan infrastruktur virtual yang lebih kompleks di masa mendatang.

BAB I

PENDAHULUAN

Virtualisasi telah menjadi teknologi fundamental dalam pengembangan sistem komputasi modern, khususnya pada lingkungan server, cloud computing, dan pengujian sistem. Teknologi ini memungkinkan abstraksi perangkat keras fisik menjadi sumber daya virtual yang dapat dibagi dan dialokasikan secara dinamis sesuai kebutuhan. Dengan virtualisasi, satu mesin fisik dapat menjalankan beberapa sistem operasi secara bersamaan melalui pengelolaan sumber daya oleh hypervisor, yang bertindak sebagai layer perantara antara hardware dan virtual machine. Pendekatan ini tidak hanya meningkatkan efisiensi utilisasi hardware, tetapi juga memberikan fleksibilitas dalam deployment aplikasi, isolasi sistem untuk keamanan, serta kemudahan dalam backup dan disaster recovery.

Pada proyek ini, mahasiswa membangun sebuah hypervisor mini berbasis QEMU/KVM dan libvirt untuk memahami implementasi praktis dari teknologi virtualisasi. QEMU (Quick EMUlator) dipilih sebagai emulator yang dapat bekerja dengan KVM (Kernel-based Virtual Machine) untuk menyediakan virtualisasi dengan performa tinggi melalui hardware-assisted virtualization. Sementara itu, libvirt digunakan sebagai API dan toolkit manajemen yang menyediakan interface standar untuk mengelola berbagai platform virtualisasi. Hypervisor yang dibangun digunakan untuk menjalankan dan mengelola Virtual Machine (VM), melakukan pengaturan alokasi CPU dan memori secara dinamis, serta menganalisis performa VM melalui pengujian benchmark menggunakan tools seperti sysbench atau stress-ng.

Proyek ini memberikan pemahaman praktis yang mendalam mengenai cara kerja virtualisasi level OS, mekanisme hardware-assisted virtualization, dan manajemen resource pada sistem virtual. Mahasiswa akan mempelajari konsep-konsep penting seperti CPU scheduling pada VM, memory management termasuk memory ballooning dan overcommitment, serta network dan storage virtualization. Selain itu, proyek ini juga mengeksplorasi trade-off antara performa, isolasi, dan efisiensi resource dalam lingkungan virtual, yang merupakan pertimbangan kritis dalam merancang infrastruktur IT modern. Dengan pengalaman hands-on ini, mahasiswa diharapkan dapat mengaplikasikan pengetahuan virtualisasi dalam konteks industri maupun riset lebih lanjut di bidang sistem terdistribusi dan cloud computing.

BAB II

RUMUSAN MASALAH DAN TUJUAN PROJECT

2.1 Rumusan Masalah

1.Bagaimana membangun hypervisor mini menggunakan QEMU/KVM dan libvirt?

2.Bagaimana cara menjalankan dan mengelola VM pada hypervisor mini?

3.Bagaimana pengaruh pengaturan CPU dan memori terhadap performa VM?

4.Bagaimana hasil benchmark CPU dan memori pada VM yang dijalankan?

2.2 Tujuan Project

1.Membangun hypervisor mini berbasis QEMU/KVM dan libvirt.

2.Menjalankan dan mengelola 1–2 VM secara stabil.

3.Melakukan pengaturan CPU dan memori pada VM.

4.Melakukan benchmark CPU dan memori antar VM.

5.Menyusun laporan hasil implementasi dan analisis.

BAB III

PERANCANGAN ARSITEKTUR

3.1 Arsitektur Sistem

Arsitektur hypervisor mini yang dirancang dalam proyek ini terdiri dari beberapa komponen utama yang bekerja secara hierarkis dan terintegrasi untuk menyediakan lingkungan virtualisasi yang efisien dan mudah dikelola. Setiap layer dalam arsitektur ini memiliki peran dan tanggung jawab spesifik yang berkontribusi terhadap fungsionalitas sistem secara keseluruhan.

3.1.1 Host System

Host system merupakan fondasi dari seluruh infrastruktur virtualisasi, menggunakan sistem operasi Linux sebagai platform dasar. Linux dipilih karena memiliki dukungan native untuk teknologi virtualisasi melalui modul KVM yang terintegrasi langsung ke dalam kernel. Sistem operasi host bertanggung jawab untuk menyediakan akses ke hardware fisik, mengelola resource allocation, dan menjamin isolasi antar VM. Distribusi Linux yang digunakan harus memiliki kernel versi minimal 2.6.20 untuk mendukung KVM, serta prosesor dengan fitur hardware virtualization extension (Intel VT-x atau AMD-V) yang telah diaktifkan di BIOS.

3.1.2 Hypervisor Layer

Layer hypervisor terdiri dari dua komponen utama yang bekerja secara sinergis:

- **QEMU (Quick EMUlator)**: Berfungsi sebagai emulator hardware yang menyediakan virtualisasi perangkat I/O seperti disk, network adapter, dan peripheral lainnya. QEMU beroperasi di user space dan bertanggung jawab untuk emulasi BIOS, device model, dan handling I/O operations dari guest VM. QEMU juga menyediakan fleksibilitas untuk menjalankan berbagai arsitektur prosesor yang berbeda dari host system.
- **KVM (Kernel-based Virtual Machine)**: Merupakan modul kernel Linux yang mengubah Linux menjadi hypervisor tipe-1 (bare-metal). KVM memanfaatkan hardware virtualization extensions untuk menjalankan guest VM dengan performa mendekati native, melakukan akselerasi eksekusi instruksi CPU guest, dan mengelola memory virtualization melalui Extended Page Tables (EPT) atau Nested Page Tables (NPT). KVM bekerja di kernel space sehingga memiliki overhead yang minimal dibandingkan dengan solusi pure software virtualization.

Kombinasi QEMU/KVM menciptakan solusi virtualisasi hybrid yang menggabungkan fleksibilitas emulasi QEMU dengan performa tinggi hardware-assisted virtualization dari KVM.

3.1.3 Management Layer

Libvirt berfungsi sebagai abstraction layer dan API standar untuk mengelola berbagai platform virtualisasi. Layer ini menyediakan interface yang konsisten untuk operasi-operasi seperti pembuatan VM, konfigurasi resource, monitoring status VM, snapshot management, dan network configuration. Libvirt berkomunikasi dengan QEMU/KVM melalui socket dan menyimpan konfigurasi VM dalam format XML yang human-readable. Daemon libvirtd berjalan sebagai background service yang menangani request dari client dan memastikan persistensi konfigurasi VM. Management layer ini juga bertanggung jawab untuk implementasi security policies seperti SELinux atau AppArmor context untuk isolasi VM.

3.1.4 User Interface

Virt-manager (Virtual Machine Manager) menyediakan antarmuka grafis berbasis GTK+ yang memudahkan administrator dalam mengelola VM tanpa harus menggunakan command-line interface. Interface ini menyediakan fitur-fitur seperti:

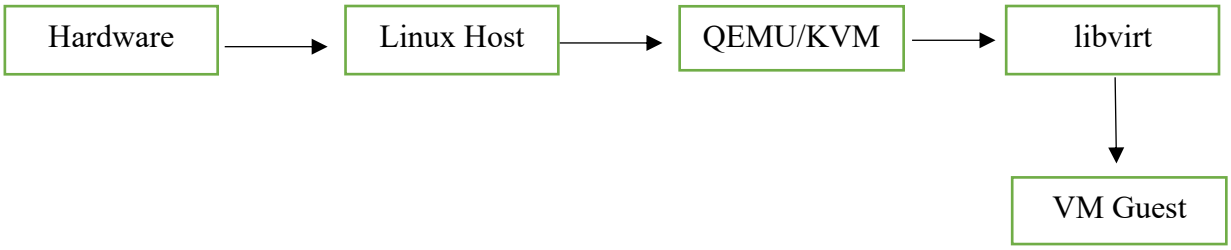
- Console viewer untuk mengakses display VM secara langsung
- Wizard untuk pembuatan VM baru dengan guided configuration
- Real-time monitoring resource usage (CPU, memory, disk I/O, network)
- Management storage pools dan virtual networks
- Snapshot creation dan management

Selain virt-manager, sistem juga mendukung command-line tools seperti virsh untuk automation dan scripting, serta cockpit untuk web-based management interface.

3.1.5 Guest System

Guest system atau Virtual Machine merupakan sistem operasi yang berjalan di atas hypervisor dalam environment yang terisolasi. Setiap VM memiliki virtual hardware sendiri termasuk virtual CPU (vCPU), virtual memory (RAM), virtual disk, dan virtual network interface card (vNIC). Guest system dapat menjalankan berbagai sistem operasi seperti Linux distributions, Windows, atau BSD, tergantung pada kebutuhan use case. Untuk meningkatkan performa, guest system dapat menginstal paravirtualized drivers (virtio) yang memungkinkan komunikasi lebih efisien antara guest dan host tanpa melalui full hardware emulation.

3.2 Diagram Arsitektur



3.3 Alur Kerja Sistem

Alur kerja sistem virtualisasi dapat dijelaskan sebagai berikut:

1. **Initialization Phase**: User mengirimkan request pembuatan atau start VM melalui virt-manager atau virsh command.
2. **Configuration Processing**: Libvirtd menerima request, membaca file konfigurasi XML VM, dan memvalidasi parameter seperti CPU cores, memory allocation, disk image path, dan network configuration.
3. **VM Instantiation**: Libvirt memanggil QEMU dengan parameter yang sesuai untuk membuat QEMU process yang merepresentasikan VM tersebut.
4. **Hardware Virtualization**: KVM module di kernel mengambil alih eksekusi CPU guest menggunakan hardware virtualization extensions, sementara QEMU menangani emulasi I/O devices.
5. **Resource Allocation**: Host kernel mengalokasikan physical memory untuk VM, menjadwalkan vCPU pada physical CPU cores, dan menyiapkan virtual network interfaces.
6. **Guest Execution**: Guest OS booting dan berjalan dalam environment virtual yang terisolasi, dengan instruksi CPU sensitive dieksekusi langsung oleh hardware melalui KVM, sementara I/O operations ditangani oleh QEMU.
7. **Monitoring & Management**: Libvirt secara kontinyu memonitor status VM dan resource usage, serta menangani operasi management seperti pause, resume, snapshot, atau migration.

BAB IV

IMPLEMENTASI

4.1 Instalasi Hypervisor

Implementasi dimulai dengan instalasi QEMU, KVM, libvirt, dan virt-manager pada sistem Linux.

Selanjutnya dilakukan verifikasi dukungan virtualisasi hardware dan layanan libvirt.

4.2 Pembuatan dan Konfigurasi VM

VM dibuat menggunakan virt-manager dengan konfigurasi dasar berupa alokasi CPU, memori, storage, dan

jaringan virtual. VM berhasil terdaftar dan dikelola oleh libvirt.

BAB V

PENGUJIAN DAN ANALISIS

5.1 Pengujian Hypervisor dan VM

Pengujian dilakukan untuk memastikan layanan libvirt berjalan, jaringan virtual aktif, serta VM dapat dijalankan. Pengujian benchmark CPU dan memori dilakukan menggunakan tools monitoring dan benchmark.

5.2 Analisis Hasil Benchmark

Hasil pengujian menunjukkan bahwa peningkatan alokasi CPU dan memori memberikan peningkatan performa VM. Dukungan virtualisasi hardware (KVM) berpengaruh signifikan dalam mengurangi overhead dan meningkatkan efisiensi sistem.

BAB VI

KESIMPULAN

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa hypervisor mini berbasis QEMU/KVM dan libvirt berhasil dibangun, dikonfigurasi, serta dijalankan sesuai dengan perancangan awal. Sistem virtualisasi yang dikembangkan mampu menjalankan mesin virtual (Virtual Machine/VM) secara stabil, serta memungkinkan pengelolaan VM meliputi pembuatan, pengaturan, dan pemantauan resource dengan baik.

Hasil pengujian menunjukkan bahwa alokasi sumber daya, khususnya CPU dan memori, memiliki pengaruh yang signifikan terhadap performa VM. Peningkatan alokasi CPU dan memori memberikan dampak positif terhadap responsivitas dan kinerja VM, sedangkan keterbatasan resource dapat menyebabkan penurunan performa, seperti meningkatnya waktu eksekusi dan penggunaan sistem yang kurang optimal. Hal ini membuktikan bahwa manajemen resource merupakan aspek krusial dalam lingkungan virtualisasi.

Selain itu, penggunaan QEMU/KVM sebagai hypervisor dan libvirt sebagai alat manajemen terbukti efektif dalam menyediakan lingkungan virtual yang fleksibel dan efisien. Integrasi antara komponen-komponen tersebut memudahkan proses administrasi VM serta memberikan kontrol yang lebih baik terhadap penggunaan resource host.

Secara keseluruhan, proyek ini tidak hanya berhasil membangun sebuah hypervisor mini yang berfungsi dengan baik, tetapi juga memberikan pemahaman praktis dan mendalam mengenai mekanisme virtualisasi, cara kerja hypervisor, serta pentingnya manajemen resource pada sistem virtual. Pengetahuan dan pengalaman yang diperoleh dari proyek ini diharapkan dapat menjadi dasar untuk pengembangan sistem virtualisasi yang lebih kompleks di masa mendatang, baik untuk kebutuhan pembelajaran, penelitian, maupun implementasi di lingkungan produksi.

BAB VII

DAFTAR PUSTAKA

Alhenaki, L. e. (2019). *A Survey on the Security of Cloud Computing.* IEEE Access.

Mugeraya, S., & Devadkar, K. (2022). Dynamic Task Scheduling and Resource Allocation for Microservices in Cloud. *Journal of Cloud and Systems.*

Shrimali, B., & Patel, H. (2016). Comparative Study for Selection of Open Source Hypervisors and Cloud Architectures. *Journal of Cloud Computing.*

BAB VIII

LAMPIRAN

Output Perintah Terminal:

- Cek dukungan Virtualisasi CPU

  Perintah:

  ```
  lscpu | grep Virtualization
  ```

- Cek apakah modul KVM aktif

  Perintah:

  ```
  lsmod | grep kvm
  ```

- Instalasi paket virtualisasi

  Perintah:

```
sudo apt install qemu-kvm libvirt-daemon-system libvirtclients bridge-utils virt-manager -y
```

- Cek status layanan libvirt

  Perintah:

  ```
  systemctl status libvirtd
  ```

- Menguji koneksi ke hypervisor libvirt dan Mengecek jaringan virtual

  Perintah: `virsh list –all`

  Perintah: `virsh net-list`

  Perintah ini menampilkan daftar semua VM yang dikelola oleh libvirt, baik yang sedang berjalan atau berhenti.

  Perintah:

  ```
  virsh list --all
  ```

  Output:

  ```
  Id    Name    State
  ----------------------
  ```

Keterangan:

libvirt berjalan normal Perintah virsh dapat dijalankan → libvirt aktif.

Tidak ada VM yang terdaftar pada libvirt Ini berarti:

- Kamu belum membuat VM menggunakan virt-manager/virt-install/virsh. (Atau kamu membuat VM tapi lewat VirtualBox, bukan KVM/libvirt.)
- Belum ada VM yang dipetakan ke /etc/libvirt/qemu/. Tidak ada VM yang berjalan maupun shut off

Kesimpulan untuk laporan:

Sistem virtualisasi libvirt sudah aktif, namun belum ada VM yang dibuat pada KVM/libvirt. Pembuatan VM baru akan dilakukan pada tahap minggu ke-13.

Perintah:

```
virsh net-list
```

Output :

```
Name      State    Autostart  Persistent
----------------------------------------
default   active      yes        yes
```

Keterangan: Jaringan virtual “default” aktif

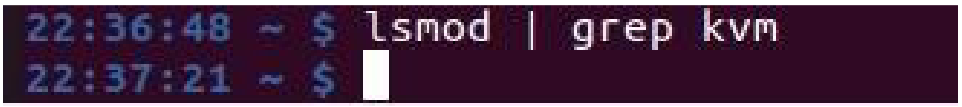
Network default adalah jaringan NAT bawaan libvirt yang memakai interface virbr0. Jika active = yes, artinya:

- VM nanti otomatis terhubung ke internet
- DHCP libvirt bekerja
- NAT berjalan normal Autostart = yes Network ini otomatis aktif setiap boot tanpa harus dijalankan manual. Persistent = yes Konfigurasi jaringan tersimpan permanen (bukan sementara).

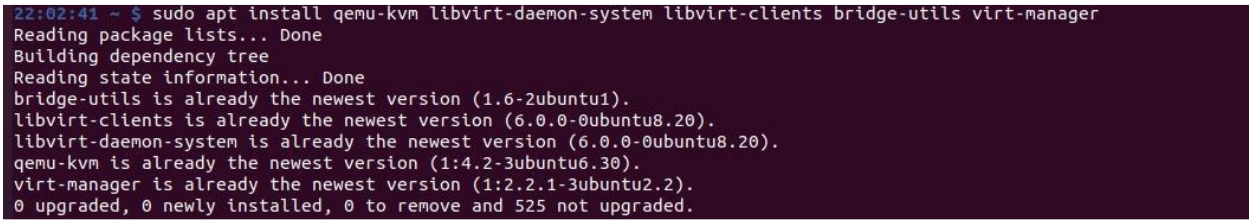
Screenshot Pengujian:

Mengecek dukungan virtualisasi CPU

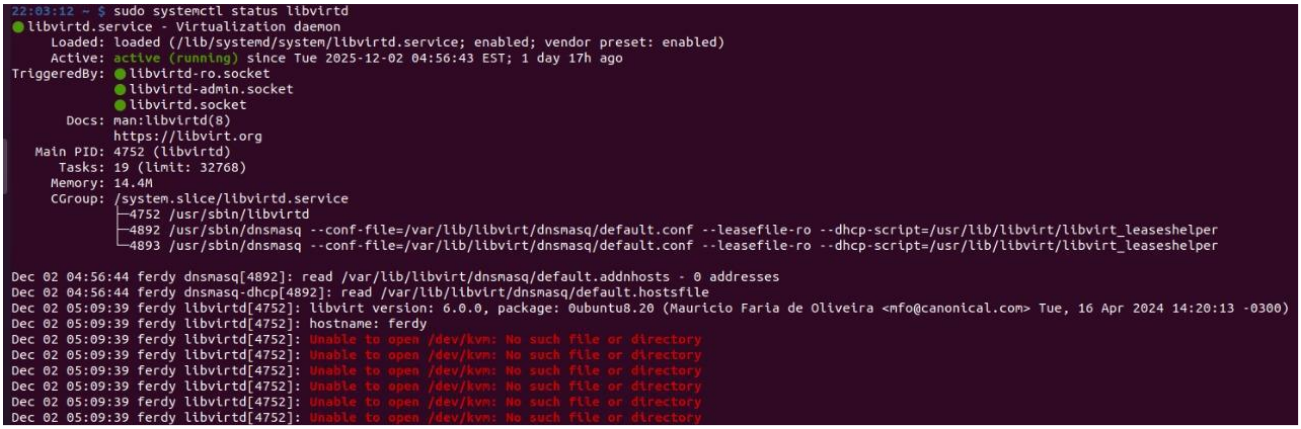
Memastikan modul KVM tersedia:



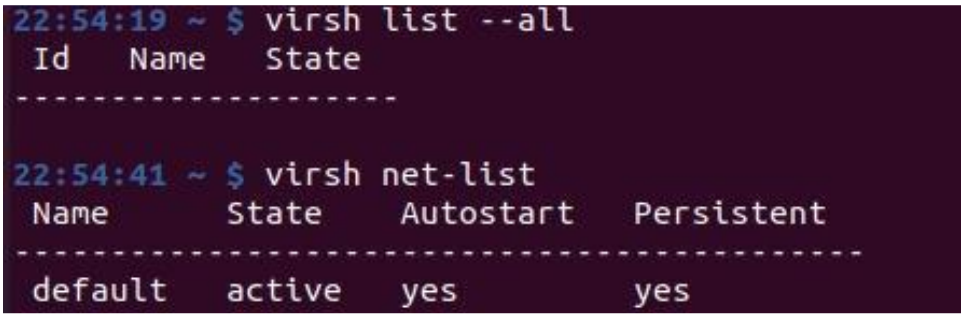
Instalasi paket virtualisasi:



Mengecek status layanan libvirt:

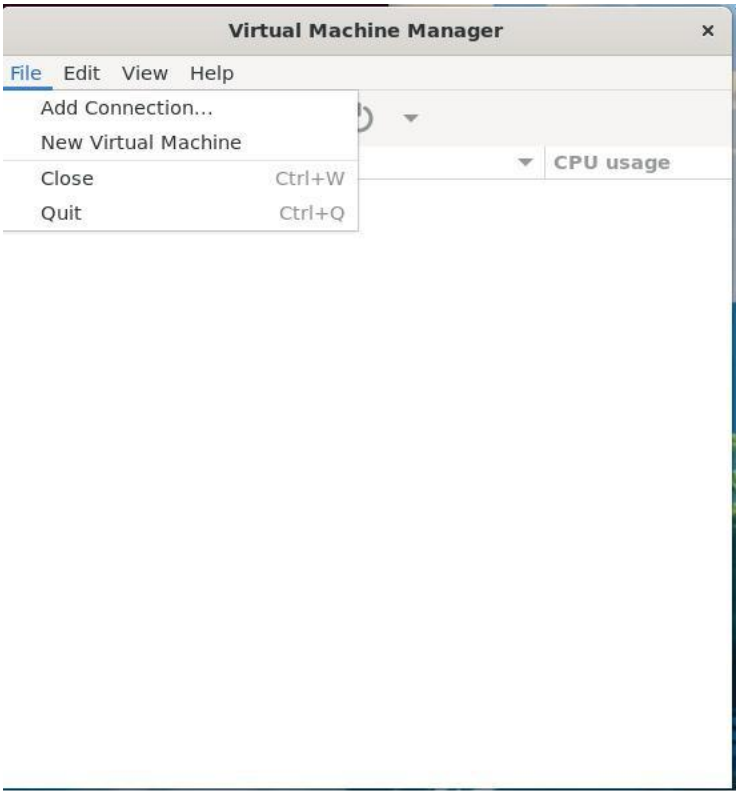


Menguji koneksi ke hypervisor libvirt dan mengecek jaringan virtual:



Output dari perintah pembuatan VM

1. Perintah tampilan virt-manager:



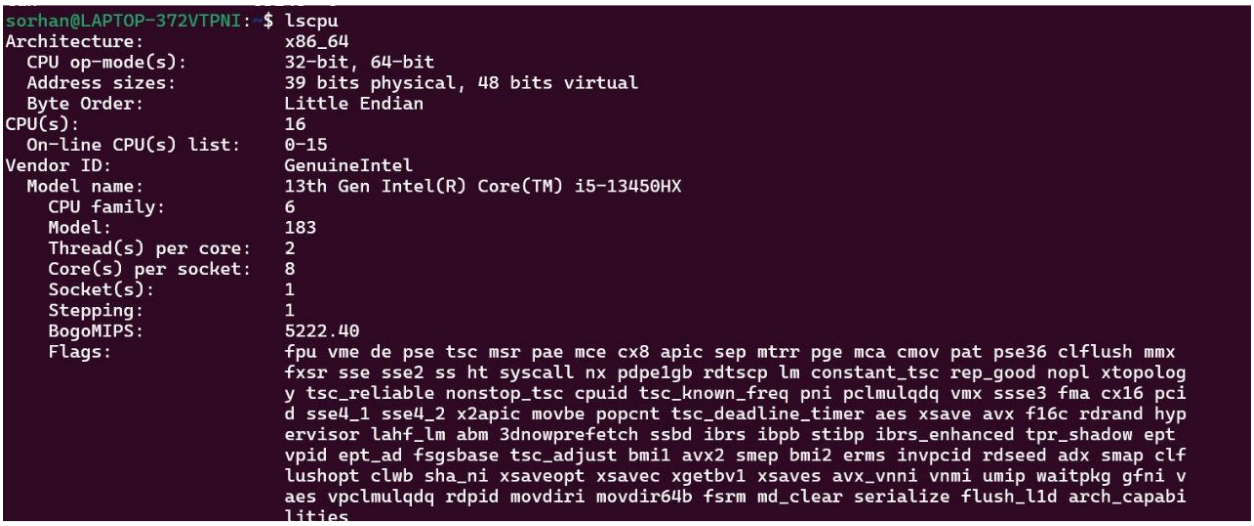
Gambar ini menunjukkan antarmuka awal virt-manager ketika baru dibuka, siap untuk membuat VM baru atau mengelola koneksi hypervisor.

2.Output tampilan lsmod:

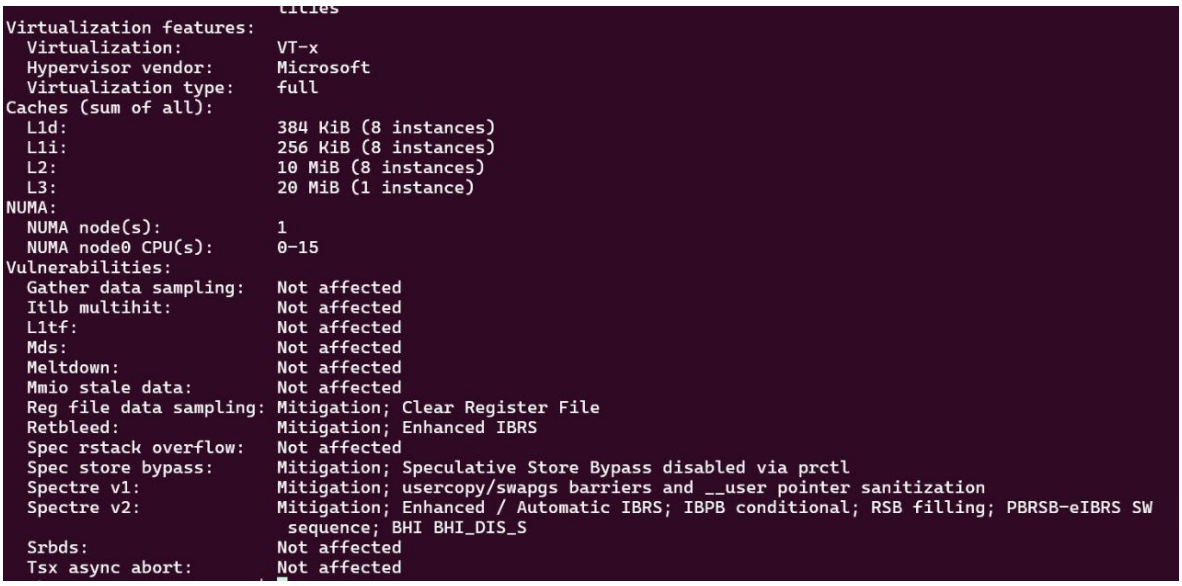
```
sorhan@LAPTOP-372VTPNI:~$ lsmod
Module                  Size  Used by
xt_CHECKSUM            12288  1
xt_MASQUERADE          16384  3
xt_conntrack           12288  1
ipt_REJECT             12288  2
nf_reject_ipv4         12288  1 ipt_REJECT
nft_compat             16384  16
intel_rapl_msr         16384  0
intel_rapl_common      32768  1 intel_rapl_msr
kvm_intel             356352  0
crc32c_intel           16384  0
battery                20480  0
ac                     16384  0
kvm                   970752  1 kvm_intel
irqbypass              12288  1 kvm
sch_fq_codel           16384  1
configfs               53248  0
autofs4                45056  0
br_netfilter           28672  0
bridge                282624  1 br_netfilter
stp                    12288  1 bridge
llc                    12288  2 bridge,stp
ip_tables              28672  0
tun                    53248  0
```

Sistem ini adalah laptop Intel dengan Linux yang sudah siap untuk virtualisasi KVM, memiliki konfigurasi jaringan yang mendukung virtualisasi/container, dan kemungkinan digunakan untuk pengembangan atau testing dengan mesin virtual.

3.Output tampilan lscpu:



Sistem ini memiliki CPU Intel generasi 13 yang cukup powerful (16 thread) dengan dukungan penuh virtualisasi hardware, sangat ideal untuk menjalankan KVM, container, dan workload multitasking berat.



Sistem ini memiliki CPU Intel modern dengan performa tinggi, dilengkapi dukungan virtualisasi hardware lengkap dan fitur keamanan mutakhir.

Sangat cocok untuk:

• Menjalankan mesin virtual (KVM/QEMU) dengan performa near-native

• Development dan testing lingkungan virtual/container

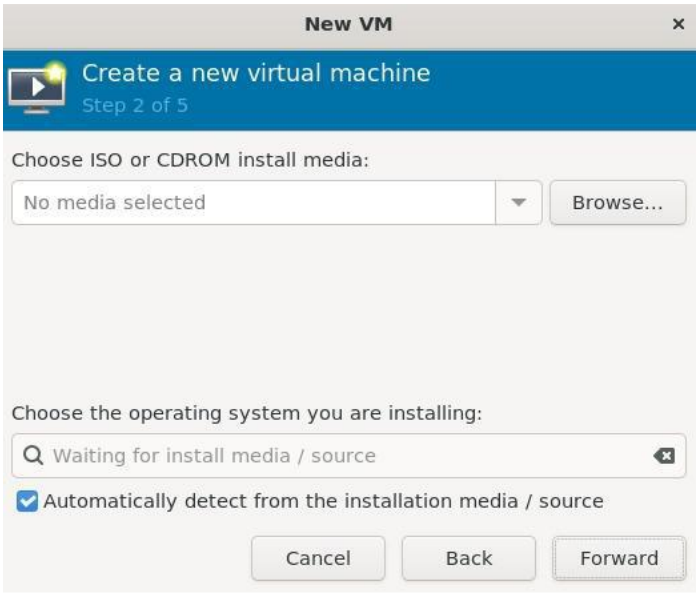
• Workload komputasi berat seperti data processing, machine learning, atau server development lokal.

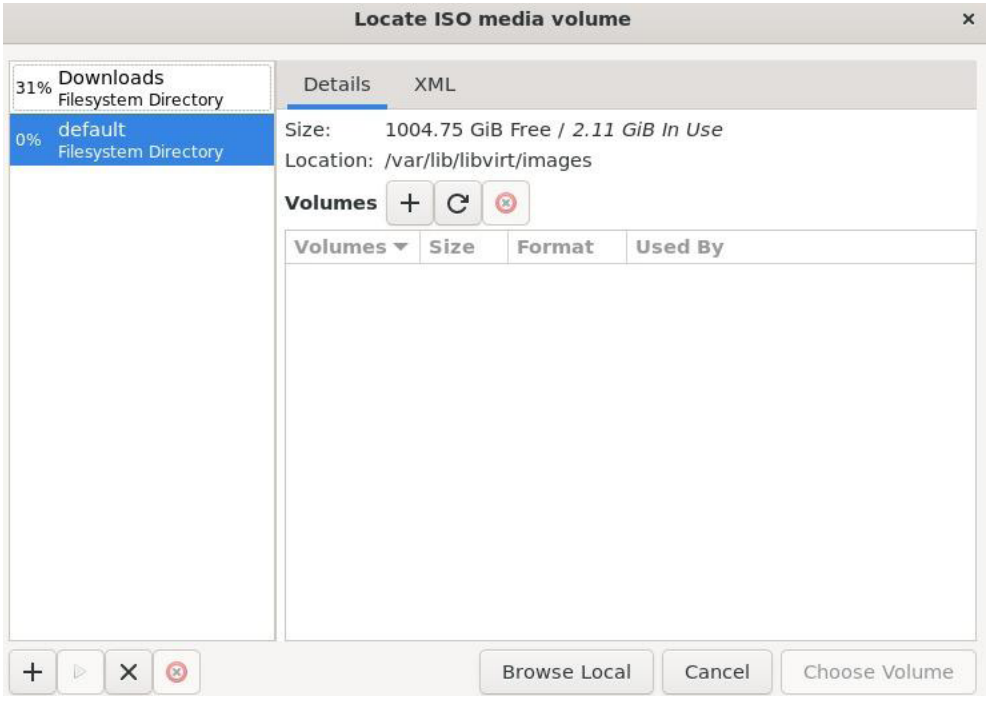
4. Tampilan dari pembuatan VM

Step 1:

Gambar ini menunjukkan tampilan awal pembuatan VM di virt-manager, siap untuk memilih metode instalasi OS sebelum melanjutkan ke konfigurasi resource (CPU, RAM, storage, dll.).

Step 2:





Penjelasan:

a) Langkah 2 dari 5: Pengguna sedang memilih media instalasi OS (ISO image) untuk mesin virtual.

b) Hypervisor: Masih menggunakan QEMU/KVM dengan penyimpanan default di /var/lib/libvirt/images.

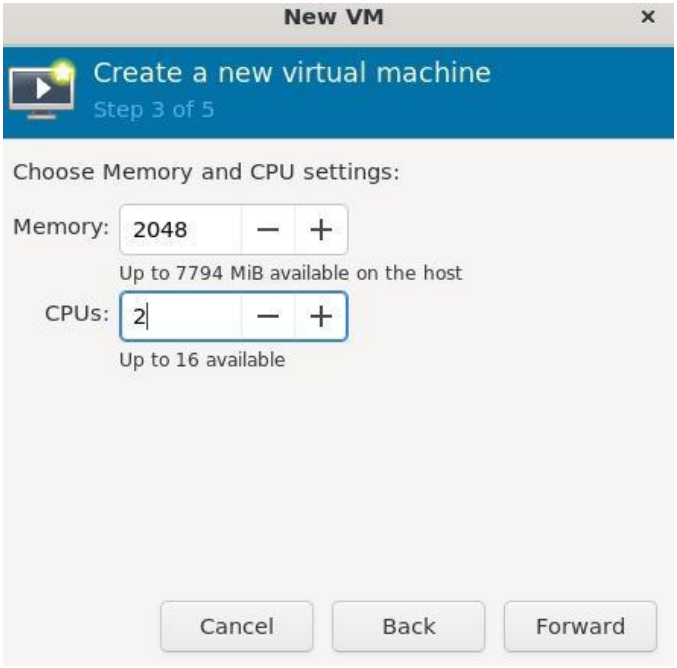
c) Pemilihan ISO:

- Pengguna telah menelusuri direktori dan menemukan file ubuntu-24.04.3-desktopamd64.iso (Ukuran 6.3 GB) di lokasi /mnt/d/Dataku/Downloads.
- SO ini adalah Ubuntu 24.04.3 Desktop versi 64-bit.

d) Akan Dilakukan: Setelah memilih ISO tersebut, pengguna akan melanjutkan ke langkah konfigurasi berikutnya (seperti alokasi RAM, CPU, dan storage) untuk menginstal Ubuntu sebagai sistem operasi guest di mesin virtual.

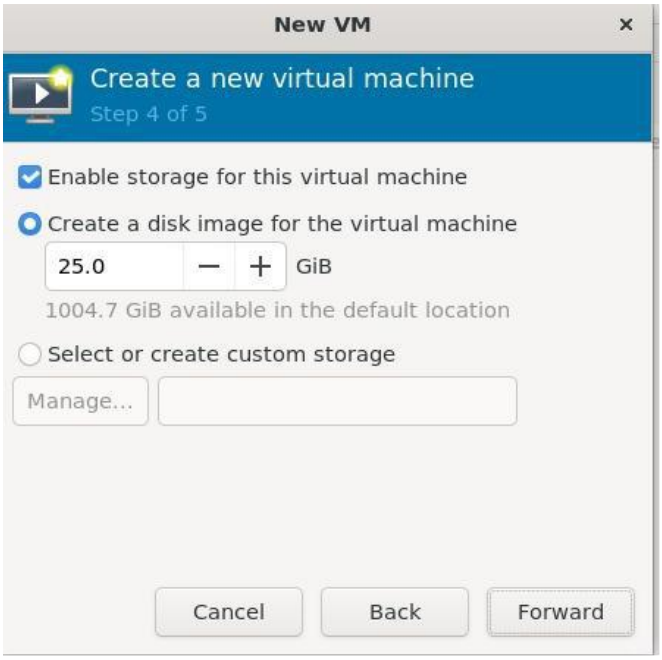
Konteks: Ini adalah proses standar pembuatan VM di virt-manager, di mana pengguna memilih ISO lokal untuk instalasi OS guest (dalam hal ini Ubuntu 24.04), sebelum mengonfigurasi resource hardware virtual.

Step 3:



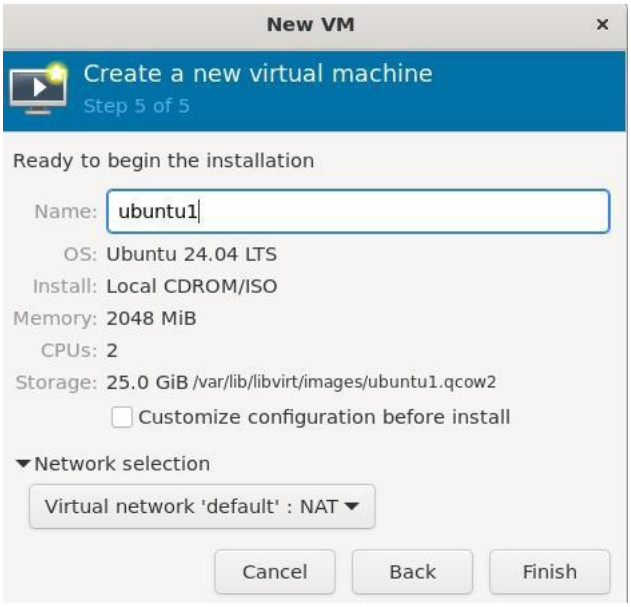
Pengguna sedang mengalokasikan resource RAM dan CPU untuk mesin virtual Ubuntu yang akan dibuat. Dengan host yang cukup kuat (16 vCPU, ~7,8 GB RAM tersedia), pengguna memiliki fleksibilitas untuk memberi resource lebih jika diperlukan. Langkah selanjutnya biasanya pengaturan storage (disk virtual).

Step 4:

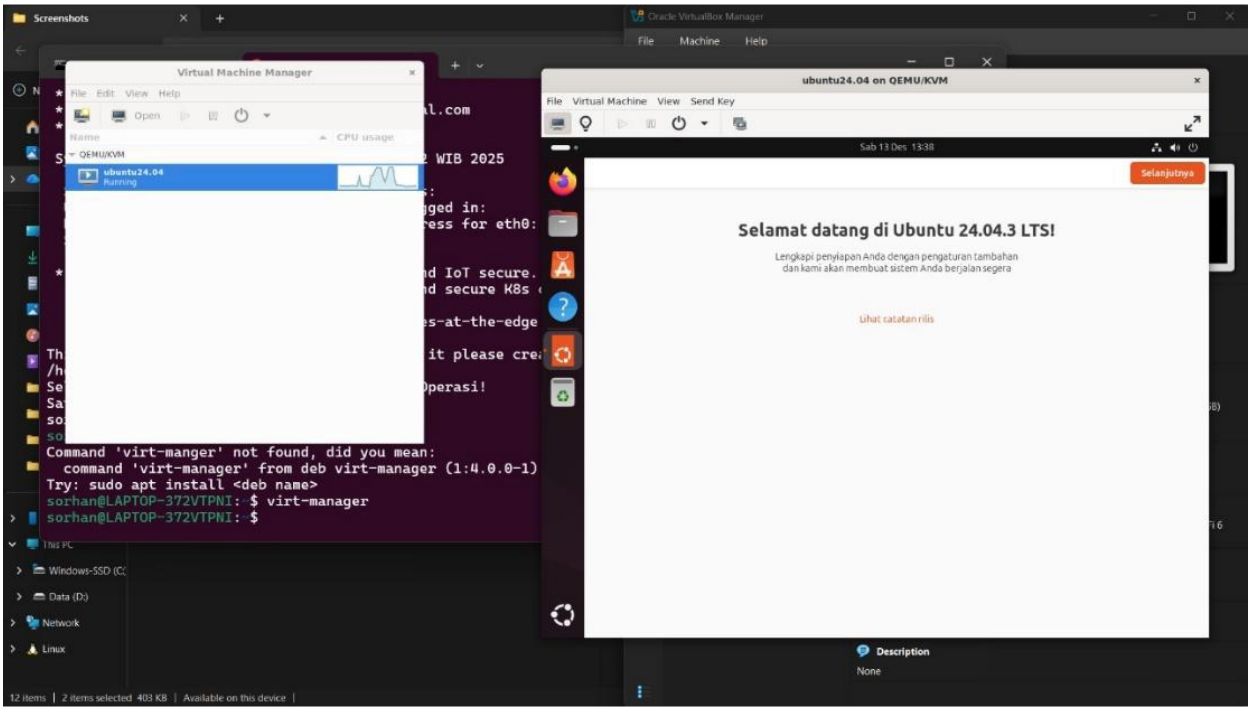


Pengguna sedang mengonfigurasi disk virtual untuk mesin virtual Ubuntu. Disk baru sebesar 25 GB akan dibuat di lokasi default. Langkah selanjutnya (step 5) biasanya adalah penamaan VM dan review konfigurasi sebelum membuat VM.

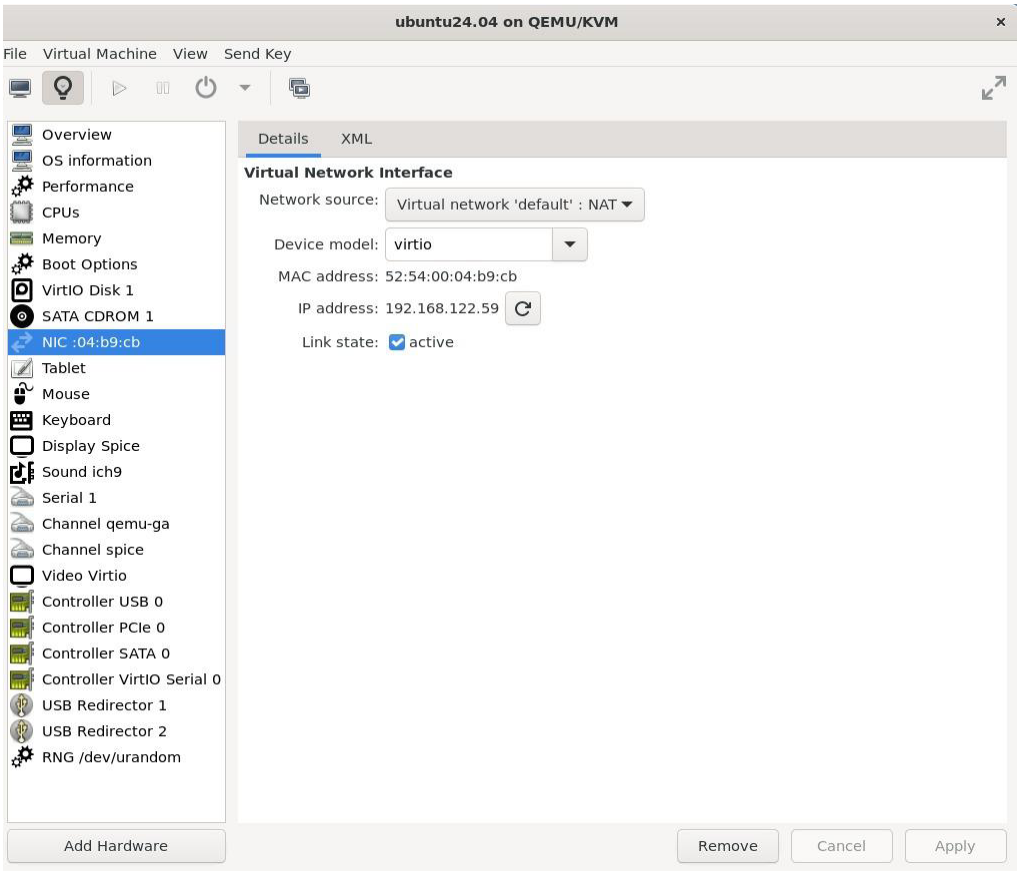
Step 5:

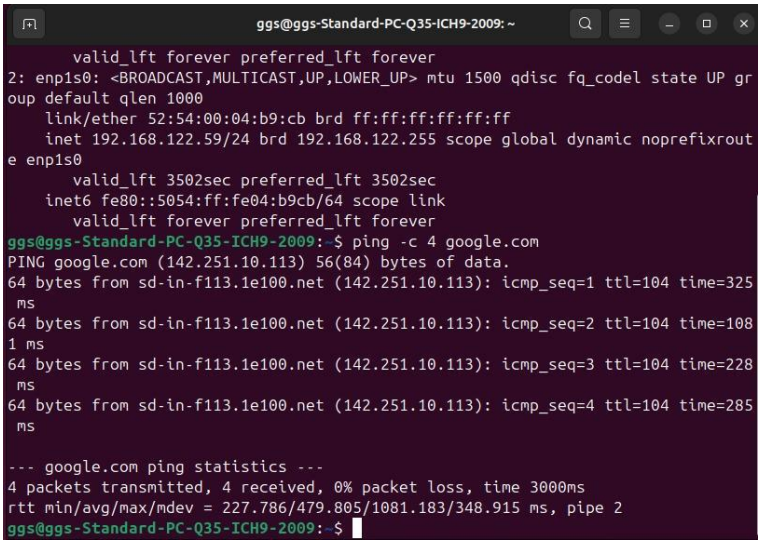


Menampilkan beberapa jendela aplikasi yang terbuka di desktop yang utamanya berkaitan dengan virtualisasi:



Konfigurasi Jaringan VM:

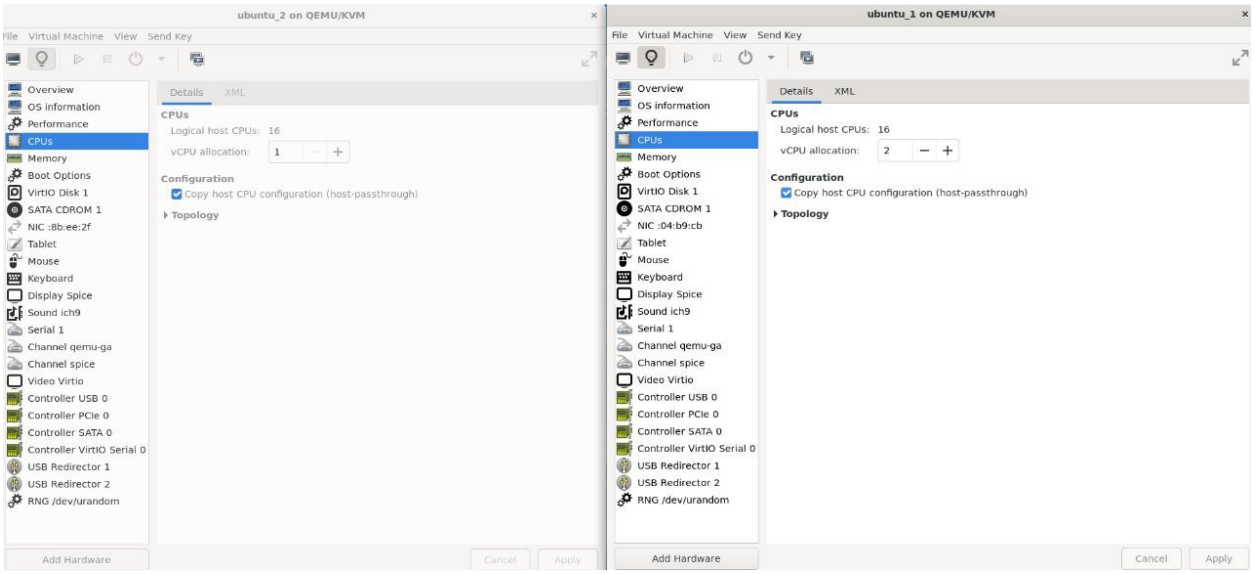




Ini adalah tampilan setelah instalasi selesai, di mana VM sudah berjalan dengan sistem operasi Ubuntu 24.04 yang berjalan normal, terhubung ke internet, dan siap digunakan. Pengguna dapat melihat detail hardware virtual dan melakukan manajemen lebih lanjut melalui antarmuka virt-manager.

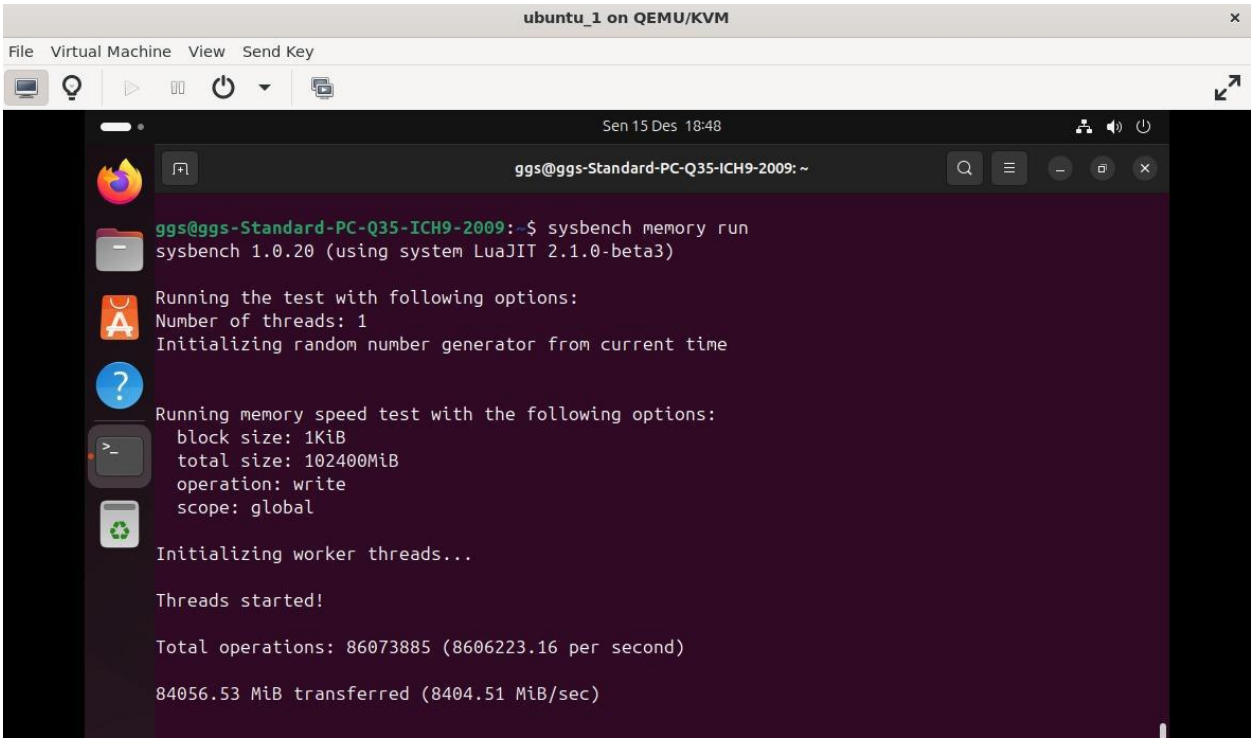
Benchmarking:

1.Benchmark: VM1 & VM2

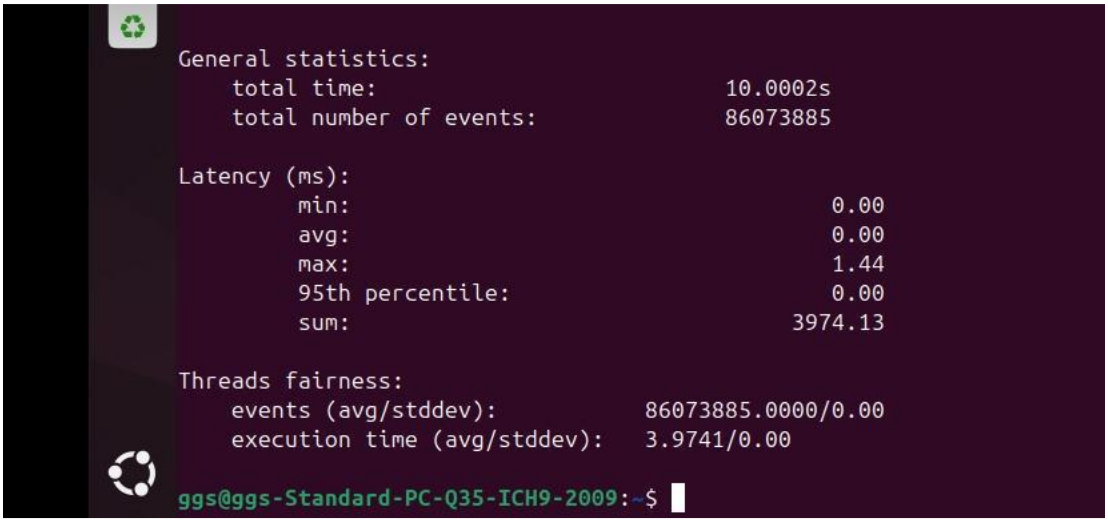


2. Benchmark Memory

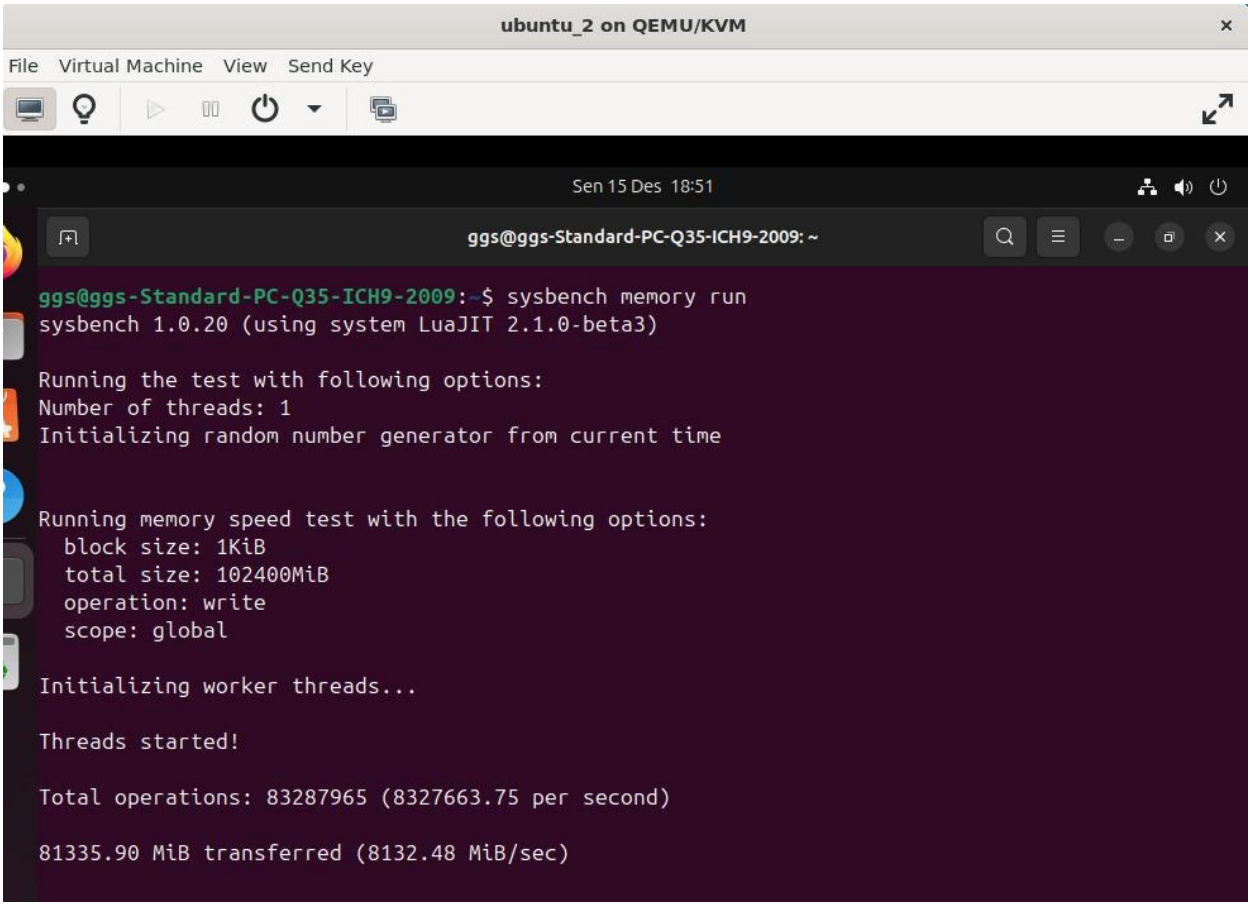
Perintah Bencmark Memory VM1



Hasill Bencmark Memmory VM1:



Perintah Bencmark Memory VM2:



Hasil Bencmark Memory VM2: