BAB 1
PENDAHULUAN

Bab ini menjelaskan latar belakang, permasalahan, tujuan dan ruang lingkup penelitian, serta sistematika penulisan tugas akhir penelitian

1.1 Latar Belakang

Perjalanan kehidupan manusia dipenuhi dengan berbagai kejadian. Manusia memerlukan usaha untuk dapat memenuhi kebutuhan dan menjaga eksistensi dari ancaman, baik itu dari dalam dan dari luar. Salah satu ancaman tersebut adalah penyakit. Sepanjang sejarah, penyakit tidak dapat terlepas dari manusia.Perkembangan ilmu dan daya pikir manusia menyebabkan manusia dapat bertahan dari serangan penyakit, namun penyakit tidak serta merta menghilang. Muncul jenis penyakit baru yang juga memerlukan pengetahuan baru dalam mencegah penyebaran penyakit tersebut.Berbagai pengobatan dan kepecayaan dalam menyembuhkan penyakit bermunculan. Dunia timur muncul dengan pengobatan herbal dan teknik akupuntur. Arkeolog menemukan bukti bahwa pemanfaatan tanaman obat herbal telah ada sejak 60000 tahun silam (masa*Paleolithic*). Tulisan berumur 5000 tahun tentang daftar tanaman obat juga ditemukan. Tulisan ini merupakan list herbal yang dibuat oleh penduduk Sumeria. Tidak hanya Sumaria, dalam sejarah Mesir, Yunani maupun Cina terdaftar sebagai negara yang mengembangkan pengobatan herbal. Cina, tidak hanya mengembangkan herbal, mereka juga mengembangkan teknik akupuntur dalam menyembuhkan penyakit.

Seiring dengan berkembang ilmu pengetahuan dan teknologi manusia khususnya *proteomics* dan *genomics*, manusia dapat mempelajari penyakit pada tingkat molekul. Hal ini memberikan pengetahuan dan kepastian dalam pengembangan obat secara modern. *Drug discovery* merupakan runtutan skenario panjang dalam menemukan calon obat yang berpotensi. Sebelum masuk dalam *drug discovery*, dilakukan *pre-drug discovery*. Dalam tahap ini peneliti mempelajari penyakit tersebut, karaktersitik, bagaimana penyakit tersebut mempengaruhi gen penderita,bagimana gen yang terkena tersebut memproduksi protein, bagaimana protein tersebut berinteraksi dengan lingkungan sekitar, bagaimana protein tersebut mempengaruhi lapisan dimana protein tersebut berperan , dan pada akhirnya bagaimana penyakit tersebut mempengaruhi *host*(tubuh yang terkena penyakit). De-

ngan berbekal pengetahuan dari proses *pre-drug discovery* tersebut, peneliti perlu memastikan bagian yang berupa molekul tunggal (gen / protein) yang berpengaruh dalam penyakit tersebut. Setelah hal tersebut, perlu dilakukan validasi dengan cara menyuntikan ”target” tersebut ke dalam sel hidup atau *speciment* percobaan. Hal ini untuk memastikan bahwa molekul target yang diambil adalah valid. Setelah molekul target valid, peneliti perlu mencari molekul yang dapat membalikan kerja molekul target. Diperlukan waktu yang cukup lama untuk mengidentifikasi molekul pembalik tersebut dan memastikan bahwa molekul tersebut dapat digunakan sebagai obat. Berdasarkan survey, waktu yang diperlukan dalam suatu *drug discovery* sekitar 10-15 tahun. Selain itu biaya yang dikeluarkan juga sangat besar (800 juta - 1 milyar dollar) dan kemungkinan gagal dalam menemukan obat baru yang berpotensi pun tetap ada.

Waktu yang dibutuhkan, biaya yang dikeluarkan, serta kemungkinan gagal dalam menemukan obat baru yang berpotensi menjadi perhatian utama dalam *drug discovery* mula mula. Komputer turut mengambil andil dalam *drug discovery* modern. Pemanfaatan komputer tersebut menggantikan proses uji coba pencarian molekul yang dapat membalikan efek dari molekul target yang sebelumnya dilakukan dengan eksperimen terhadap makhluk hidup digantikan dengan simulasi. *Virtual screening*, yaitu mencari molekul yang dapat mengikat (*binding*) dengan molekul target dari *database* molekul yang tersimpan. Untuk memastikan molekul terikat tersebut, *virtual screening* memanfaatkan perhitungan fungsi / *scoring* untuk menentukan mana molekul yang ”*best-fit*” dengan molekul target. Pada umumnya *virtual screening* tersebut berjalan secara otomatis dengan menggunakan program komputer. Dengan cara tersebut, para peneliti dapat memangkas waktu dan biaya yang dibutuhkan dalam *drug discovery*. Beberapa aplikasi *virtual screening* yang dapat digunakan antara lain : Auto Dock, DOCK , Gold, V Life MDS, Flex X.

Virtual screening yang mencakup evaluasi seluruh koleksi *database* molekul terhadap molekul target membutuhkan tenaga komputasi yang tidak sedikit. Tidak hanya itu, kapasitas memori penyimpanan untuk koleksi data molekul maupun hasil *virtual screening* yang cukup besar, sumber daya listrik yang digunakan untuk menjaga komputer tetap bekerja juga diperlukan. Kemampuan komputasi pada komputer terletak pada *processor*. Komputer akan mengenali *Virtual screening* yang dijalankan sebagai kumpulan instruksi yang harus dikerjakan oleh *processor*. Semakin cepat *processor*, semakin banyak instruksi yang dapat dikerjakan setiap detiknya. *Virtual screening* dapat dilakukan pada sebuah komputer yang memiliki koleksi data molekul yang besar, namun masih terdapat kendala, seberapa lama waktu yang dibutuhkan untuk memproses semua koleksi data molekul tersebut.

Untuk memenuhi kebutuhan tenaga komputasi tersebut, dibutuhkan infrastruktur komputer yang dapat saling bekerja sama untuk melakukan suatu komputasi. Infrastruktur ini dikenal sebagai *supercomputer*. *Supercomputer* terdiri dari beberapa *processor* yang memiliki kemampuan komputasi yang tinggi.Komputer tersebut dapat tersebar diberbagai tempat dan dihubungkan dengan jaringan atau kumpulan komputer yang diletakan saling berdekatan pada suatu tempat. Proses kerja *supercomputer* adalah *centralization* dimana setiap *processor* mengerjakan *task* yang sama dan hasilnya akan kembali diolah oleh suatu *processor* untuk menghasilkan output akhir. Konsumsi energi yang dibutuhkan oleh *supercomputer* tergolong sangat besar dan sebagian besar hasil dari eksekusi *task* adalah panas. Dibutuhkan biaya tambahan untuk pemeliharaan *supercomputer* agar dapat terus bekerja pada suhu yang terjaga.

Besarnya biaya yang diperlukan oleh *supercomputer* merupakan kendala utama. *Grid computing* memberikan alternatif lain untuk permasalahan tersebut. Teknologi ini memanfaatkan sekumpulan komputer biasa secara fisik yang saling terhubung melalui suatu jaringan sebagai suatu kesatuan komputer (*super virtual computer*). Tidak seperti *supecomputer*, setiap komputer bekerja sama untuk menyelesaikan sebuah *task*, yang kemudian *task* tersebut akan hilang jika selesai dan tergantikan dengan *task* yang baru. Masing masing komputer penyusun tersebut akan bekerja secara parallel. Permasalahan kembali muncul ketika pengadaan komputer secara fisik yang banyak dan pengaturan jaringan yang perlu dipahami sebelum digunakan untuk *virtual screening*.

Terlepas dari pengadaan komputer secara fisik, teknologi *cloud computing* hadir sebagai solusi. Pengguna *cloud computing* tidak perlu memikirkan bagaimana biaya dan perawatan dari teknologi tersebut. Pada saat ini, pemanfaatan aplikasi bersifat *cloud*, dimana pengguna hanya disuguhkan dengan tampilan saja, sedangkan proses komputasi berada di ”awan”. Sifat virtualisasi yang menjadi andalan *cloud computing* membuat teknologi ini sangat murah untuk menambah komputer secara virtual. *Cloud computing* seolah olah mampu membuat suatu kumpulan komputer yang saling bekerja sama menyelesaikan *task* seperti *grid computing*. Memaksimalkan virtualisasi merupakan kunci dalam teknologi *cloud computing*

Teknologi virtualisasi yang ada saat ini terbentur dengan abstraksi komputer pada level *hardware*. Pembuatan komputer virtual tersebut membutuhkan *resource* memori penyimpanan dan kemampuan komputasi yang telah ditentukan sebelumnya. Dengan begitu jumlah komputer virtual akan terbatas dengan *resource* yang dimiliki oleh *host*. Disatu sisi, kinerja virtual komputer virtual tidak akan optimal ketika terdapat komputer virtual yang tidak melakukan apa apa, sehingga alokasi

resource komputer virtual tersebut dapat dibagi secara merata kepada komputer virtual lainnya. Docker, sebagai *platform* virtualisasi memberikan pandangan baru dalam mengoptimalkan proses virtualisasi tersebut.

1.2 Permasalahan

1.2.1 Definisi Permasalahan

Berdasarkan latar belakang yang telah dijelaskan, penulis berusaha untuk menganalisis apakah pemanfaatan *platform* Docker pada teknologi *cloud computing* dapat diajukan sebagai solusi alternatif dari *supercomputer* dan apakah performa yang diberikan tidak berbeda jauh dengan *grid computing*. Untuk itu, penulis akan mencoba *virtual screening* dengan menggunakan aplikasi Autodock dan Autodock Vina yang telah terinstall pada *platform* Docker.

1.2.2 Batasan Permasalahan

Pada penelitian ini, penulis lebih berfokus kepada performa Docker dalam menjalankan aplikasi Autodock dan Autodock Vina untuk *virtual screening*. Penulis tidak akan membahas segi *virtual screening* dikarenakan keterbatasan pengetahuan yang dimiliki.

1.3 Tujuan

Penelitian ini secara umum bertujuan untuk mengenalkan pemanfaatan *cloud computing* dengan *platform* Docker dalam virtualisasi komputer untuk menghasilkan performa komputasi dengan harga terjangkau dibandingkan dengan pengadaan *supercomputer* maupun *grid computing* dalam *virtual screening*.

1.4 Posisi Penelitian

Penelitian ini mencari alternatif lain dari *supercomputer* dengan memanfaatkan teknologi *cloud computing*. Hasil dari penelitian ini akan dibandingkan dengan hasil penelitian serupa yang telah dikerjakan oleh Bapak Muhammad Hafizhuddin Hilman, S.Kom., M.Kom. [20] [37]

1.5 Sistematika Penulisan

Penulisan ini terbagi dalam 5 bab :

- BAB 1 PENDAHULUAN
  Bagian ini berisikan latar belakang, permasalahan, tujuan dan ruang lingkup penelitian, serta sistematika penulisan tugas akhir penelitian.
- BAB 2 LANDASAN TEORI
  Bagian ini berisikan penjelasan *drug discovery* dengan cara *virtual screening* memanfaatkan teknologi informasi secara umum. Selain itu, *cloud computing* dan Docker akan dijelaskan secara detail.
- BAB 3 METODOLOGI PENELITIAN
  Bagian ini berisikan detail tahap-tahap penelitian dan data yang digunakan.
- BAB 4 HASIL PENELITIAN DAN ANALISIS
  Bagian ini berisikan hasil dari penelitian yang telah dilaksanakan dan analisis dari hasil yang diperoleh.
- BAB 5 KESIMPULAN DAN SARAN
  Bagian ini berisikan kesimpulan penulis terkait dengan penelitian dan saran penulis dalam penelitian ke depannya.

BAB 2
LANDASAN TEORI

Bab ini akan menjelaskan sekilas tentang *computational drug discovery*, apa itu *cloud computing* , dan Docker sebagai platform virtualisasi komputer.

2.1 *Drug dicovery*

Drug discovery adalah suatu proses menemukan kandidat obat yang berpotensial. Aktivitas tersebut merupakan misi utama yang diemban oleh perusahaan bidang farmasi untuk mengerti cara kerja penyakit sehingga dapat ditemukan obat / penyembuh yang aman untuk dikonsumsi oleh pasien. Para peneliti berusaha memahami cara kerja penyakit pada level gen dan protein.Dalam hal ini, terdapat istilah ”target”, dimana calon potensial obat dapat bereaksi. Berikut merupakan tahapan yang dilewati dalam *Drug discovery* [2] :

- Validasi target
- Menemukan molekul yang dapat berinteraksi dengan target
- Melakukan uji coba terhadap senyawa hasil interaksi tersebut untuk memastikan keamanan dan keberhasilan dari senyawa tersebut
- Mendapatkan izin untuk membuat dan mendistribusikan obat tersebut

Estimasi waktu yang diperlukan dalam proses ini adalah 10 - 14 tahun [3]. Secara garis besar berikut merupakan langkah - langkah yang dilakukan dalam *drug discovery* :

- *Pre-discovery* Pada tahapan ini peneliti bekerja untuk memahami penyakit. Dimulai dari bagaimana gen berubah sehingga mempengaruhih protein yang di*encode*, bagaimana protein tersebut berinteraksi dengan sel-sel hidup sekitar, bagaimana lapisan sel sel hidup yang terpengaruh berubah dan bagaiamana penyakit tersebut mempengaruhi pasien secara keseluruhan. Hasil yang didapat dalam tahapan ini merupakan kunci utama dalam menyelesaikan permasalahan (penyakit).

- *Target Identification* Setelah peneliti memahami penyakit dan elemen yang mempengaruhi penyakit tersebut, peneliti dalam bidang farmasi akan menentukan ”target” untuk calon obat baru. Target merupakan *single* molekul, seperti gen atau protein yang terkait dengan penyakit tersebut. Target tersebut harus dipastikan bersifat *drugable*, dimana dapat berinteraksi dengan dan dipengaruhi oleh molekul obat

- *Target Validation* Setelah target tersebut diidentifikasi, peneliti perlu memastikan apakah target tersebut benar benar terlibat dalam penyakit dan bersifat *drugable*. Peniliti mengujicobakan target tersebut ke dalam sel hidup dan hewan uji coba lab.

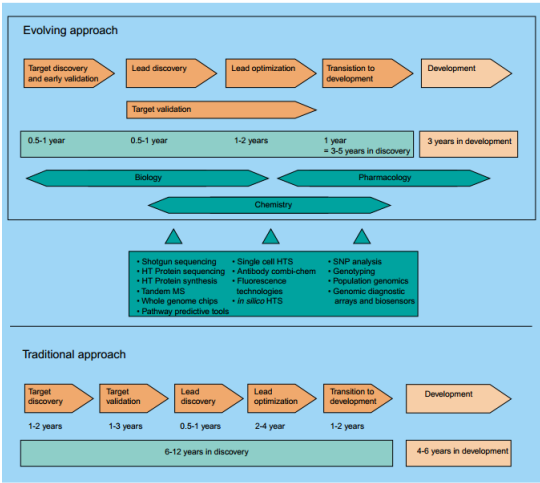
- *Drug Discovery* Dengan berbekal pengetahuan yang didapat pada tahap sebelumnya, para peneliti akan mencari molekul yang akan bereaksi dengan target yang telah ditemukan sebelumnya dan mengembalikan efek yang disebabkan oleh target tersebut. Tahapan ini memakan separuh hingga sebagian besar waktu dari *drug discovery* tersebut. Molekul yang ditemukan dalam tahapan ini bisa menjadi suatu obat baru. Beberapa cara untuk mendapatkan molekul tersebut :

  - ***nature***, dengan memanfaatkan alam dan ikatan yang terkandung didalamnya. Sebagai contoh pemanfaatan bakteri dalam pengobatan
  - ***De Novo***, dengan ilmu kimia dan pemanfaatan teknologi komputer, pembentukan molekul dari awal dengan memanfaatkan komputer untuk memodelkan molekul tersebut dan memprediksi sifat dari molekul tersebut
  - ***High-throughput Screening***, dengan memanfaatkan tenaga komputasi komputer, memungkinkan untuk membandingkan ratusan ikatan molekul dengan target. Ikatan yang diperoleh dari perbandingan tersebut akan dipelajari kembali dikemudian hari untuk dipastikan kebenarannya
  - ***Biotechnology***, dengan teknologi ini para peneliti dapat membuat mensimulasikan suatu organisasi kehidupan untuk menghasilkan molekul yang dapat melawan target.

2.2 *Computational Drug discovery*

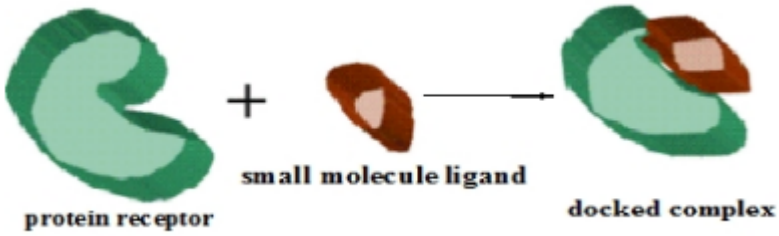
Waktu yang dibutuhkan dalam proses *drug discovery* cukup lama. Tidak hanya itu, proses ini juga sangat beresiko dan membutuhkan dana yang besar

[3].Walaupun demikian, investasi dana dalam *drug discovery* juga berkembang pesat. Namun, investasi tersebut tidak diiringi dengan hasil yang proporsional dengan investasi tersebut. Hal ini disebabkan oleh rendahnya efektivitas dan tingginya kemungkinan kegagalan dalam *drug discovery*. Beberapa pendekatan telah dilakukan dalam meningkatkan efektivitas, salah satu cara tersebut adalah CADD (*Computer Aided Drug Design*).Dengan begitu, penekanan biaya dan siklus waktu penemuan obat juga semakin cepat. Dalam CADD, komputer digunakan sebagai sarana komputasi dan penyimpanan data dalam *drug discovery* [4]. Tidak ketinggalan, CADD menggunakan aplikasi dalam mendesain ikatan kimiawi, memodelkan struktur kimia yang mengandung kandidat calon obat yang berpotensial dan pembuatan *library* yang dapat digunakan untuk riset selanjutnya.[5]



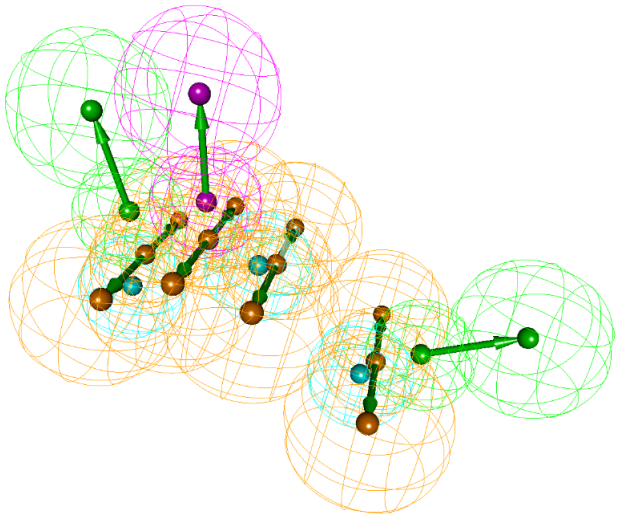
Gambar 2.1: Diagram proses*drug discovery* tradisional dan modern [3]

Dalam *computational drug discovery* (penemuan obat dengan memanfaatkan komputer), pendekatan yang dapat dilakukan dibagi menjadi *structure-based drug design* (SBDD), *ligand-based drug design* (LBDD) dan pendekatan berdasarkan sekuens [5]. Terdiri dari proses *docking* kandidat *ligand* ke target *receptor* kemudian dilakukan penilaian dengan menggunakan *scoring function* untuk dapat menghitung kemungkinan *ligand* tersebut terikat pada target protein dengan daya tarik yang tinggi [6].



Gambar 2.2: Ilustrasi *ligand* yang akan terikat dengan *receptor* [7]

Dalam LBDD, diberikan sekumpulan *ligand* dan *receptor*, dimana model *receptor* dapat dibuat dengan mengumpulkan informasi dari *ligand* [8]. Model ini dikenal dengan *pharmacophore*. Kemudian, kandidat *ligand* akan dibandingkan dengan *pharmacophore*, apakah kompatibel dengan *pharmacophore* dan dapat terikat [7].



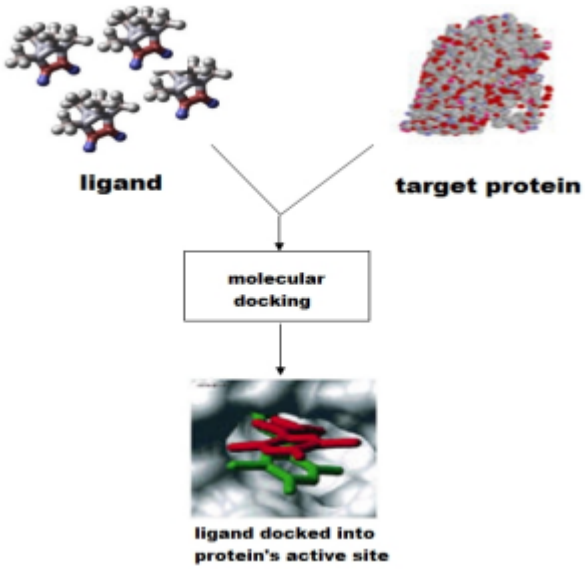
Gambar 2.3: Contoh dari *pharmacophore* [17]

2.3 *Molecular Docking*

Virtual screening berdasarkan *molecular docking* merupakan metode yang sering digunakan dalam SBDD. Penulis juga akan menggunakan metode tersebut yang terdapat dalam aplikasi Autodock dan Autodock Vina. Pemodelan struktur kimia menggunakan *molecular modeling* untuk mempelajari fenomena dari struktur kimia tersebut. Kemudahan dalam *molecular modeling* sekarang ini dibantu dengan aplikasi komputer yang tersedia, namun masih terdapat kesulitan yaitu bagaimana cara mendapatkan model struktur ikatan kimia yang benar dan interpretasi model yang tepat.Secara umum, *molecular modeling* dapat dikatakan sebagai pemanfaatan komputer dalam mengkonstruksi molekul dan melakukan berbagai macam per-

hitungan untuk mempelajari karakteristik dan sifat struktur ikatan kimia. Istilah *molecular modeling* terkadang disamakan dengan istilah *computational chemistry*.

Proses *drug discovery* yang semula berupa *trial and error* berubah menjadi proses yang dibantu dengan perhitungan komputer. Perhitungan tersebut digunakan dalam menentukan struktur ikatan kimia baru berdasarkan struktur protein yang sudah diketahui [4]. Pendekatan ini terbagi menjadi dua : *de novo design*[9] dan *docking*[10]. *Docking* merupakan suatu proses untuk menebak struktur *complex* dari struktur *ligand* dan protein. Dalam *molecular modeling*, *docking* dapat dipandang sebagai suatu metode untuk memprediksi orientasi suatu molekul ketika diikat dengan molekul lainnya untuk membentuk *complex* yang stabil.



Gambar 2.4: *Proses molecular docking* [7]

Molecular docking adalah suatu proses komputasi dalam pencarian *ligand* yang paling cocok baik secara geometri maupun energi ketika diikat dengan suatu *receptor*(protein) yang telah diketahui [7]. Aspek utama dalam *molecular docking* adalah perhitungan energi interaksi dan konformasi dengan menggunakan metode dari kuantum mekanik hingga fungsi empiris energi. Permasalahan *molecular docking* sekilas dapat dilihat sebagai suatu permasalahan *lock and key*, dimana *receptor* protein dapat dipandang sebagai *lock* dan *ligand* sebagai *key*. Namun dalam praktiknya, *molecular docking* akan mencari *ligand* yang dapat menyesuaikan dengan *receptor*. Karena adaptasi tersebut, *molecular docking* dapat dipandang sebagai permasalahan "*best-fit*" *ligand* [11]. Diperlukan *scoring function* untuk dapat membatasi *best-fit* dari proses *molecular docking*. Fungsi tersebut biasanya berdasarkan *force field* yang digunakan dalam mensimulasikan protein. Beberapa *scoring function* juga menambahkan aspek perhitungan lainnya seperti entropi [7].

Dalam percobaan ini penulis memanfaatkan aplikasi *molecular docking* Autodock dan Autodock Vina dikarenakan kedua aplikasi tersebut sering dirujuk dalam karya ilmiah yang berkaitan dengan *docking* [12].

2.4 *Autodock dan Autodock Vina*

Aplikasi ini merupakan aplikasi yang dikembangkan oleh *The Scripps Research Institute*, lembaga riset nonprofit yang terletak di California, Amerika Serikat [30]. Terdapat 2 jenis aplikasi, Autodock dan Autodock Vina. Masing-masing merupakan aplikasi yang saling berbeda dalam segi *scoring function* yang digunakan dan optimisasi komputasi secara paralel oleh Autodock Vina. Autodock memanfaatkan *empirical free energy force field* dengan *Lamarckian Genetic Algorithm* dalam memprediksi koordinat ikatan antara *ligand* dan *receptor* [13]. Sedangkan dalam Autodock Vina, dilakukan optimisasi dengan memanfaatkan *particle swarm optimization* secara global dan *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* secara lokal [14]. Pada percobaan ini penulis menggunakan Autodock versi 4.2 dan Autodock Vina versi 1.1.2.

Autodock versi 4.2 terdiri dari 2 program, Autodock4 dan Autogrid4. AutoGrid4 akan menghasilkan *pre-calculated map* dari suatu *ligand*. Selain itu juga akan menghasilkan *map* tambahan, ”d” untuk *desolvation* dan ”e” untuk *electrostatic*. Pengoperasian Autodock membutuhkan beberapa file, yaitu : *.dpf untuk pengaturan *docking* parameter , hasil *map* berupa *.gpf dari AutoGrid , dan *.pdbqt untuk *receptor* dan *ligand* yang akan diujicoba. Keluaran yang dihasilkan oleh Autodock berupa *.dlg yang berisi hasil perhitungan konformasi posisi dan energi pada setiap *ligand* yang diujicoba. Paket aplikasi Autodock Vina hanya datang dengan sebuah program saja dikarenakan proses *pre-processing docking* dibuat transparan sehingga pengguna tidak perlu tahu dan memahami bagaimana aplikasi tersebut bekerja. Autodock Vina membutuhkan 2 file, *.pdbqt dari *ligand* dan *.pdbqt dari tiap *receptor* yang akan diujicobakan. Keluaran yang dihasilkan juga berupa *.pdbqt yang beisikan konformitas posisi dan energi.

Gambar 2.5: Logo Autodock [15]

2.5 *Kemampuan Komputasi*

Molecular docking yang dilakukan antara suatu *ligand* dan *receptor* akan menentukan apakah *ligand* tersebut merupakan calon dari obat yang sedang dicari. Dengan bermunculan berbagai aplikasi yang membantu dalam proses molecular docking, kesulitan ketika melakukan secara manual dengan menggunakan alat peraga akan berkurang serta perhitungan dan tingkat presisi akan lebih baik dibandingkan dengan perhitungan manusia. Kompleksitas dari *ligand* dan *receptor* juga akan mempengaruhi evaluasi fungsi yang dikerjakan dalam aplikasi *molecular docking*[7].

Virtual screening merupakan suatu teknik komputasi yang dilakukan dalam *drug discovery* dimana secara otomatis akan mengevaluasi kumpulan bank data *receptor* dengan suatu *ligand* untuk menentukan calon obat [18]. Dibutuhkan kemampuan komputasi yang dapat menjalankan eksekusi perintah dengan cepat. CPU, sebagai pusat pengolah data dan instruksi pada komputer membutuhkan *clock rate* yang cepat dan kemampuan menjalankan instruksi per *clock* yang besar. Terlepas dari kompleksitas molekul *ligand* maupun *receptor*, faktor tersebut mengambil peranan penting dalam riset yang memanfaatkan kemampuan komputasi.

Muncul infrastruktur yang dapat dikatakan memiliki kesamaan dengan *supercomputer*, *Computer Cluster*. *Computer Cluster* dapat dikatakan sebagai komputer yang saling berhubung dan dipandang sebagai satu kesatuan. Setiap anggota (*cluster*) akan mengerjakan perintah yang sama diatur dalam *software* penjadwalan. *Cluster* ini dapat membantu peneliti dalam mengerjakan risetnya dengan performa yang tidak berbeda dengan *supercomputer*[**?**]. Berbeda dengan *cluster*, muncul istilah *grid computing*, dimana secara struktur mirip dengan *cluster*, namun untuk setiap bagian komputer mengerjakan tugas yang berbeda beda (namun dalam satu tujuan)[21].

2.6 *Cloud Computing*

Teknologi *cloud computing* sudah berkembang sejak tahun 1990-an. Pada awalnya teknologi ini dikembangkan sebagai solusi dari permasalahan yang dihadapi oleh perusahaan dengan bertambahnya kapasitas data yang perlu disimpan dan pengeluaran yang cukup besar untuk pengadaan perangkat keras beserta konsumsi daya listrik yang dibutuhkan [23]. Definisi dari *cloud computing* dapat dikatakan sebagai virtualisasi dari perangkat komputasi, dimana pengguna mengakses perangkat tersebut dengan menggunakan jaringan internet [22]. Selain itu, teknologi ini memberikan kemudahan dalam penggunaan secara *multiuser*, dimana dapat diakses se-

cara bersamaan.

Terdapat 3 karakteristik dari teknologi *cloud computing*, yaitu virtualisasi, terdistribusi, dan kemudahan dalam pengembangan selanjutnya (*dynamically extendibility*). Virtualisasi merupakan faktor yang utama. Virtualisasi dapat membagi suatu perangkat komputasi secara fisik menjadi beberapa ”virtual” perangkat komputasi. Dengan begitu, kinerja dari perangkat akan dioptimisasi. Alokasi sumber daya perangkat keras tidak ada yang diam secara percuma. Disamping itu, dengan adanya virtualisasi dapat menekan biaya pengeluaran pengadaan perangkat komputasi secara fisik. Yang dimaksud dengan terdistribusi adalah perangkat keras (*physical node*) yang digunakan dalam *cloud computing* digunakan secara terdistribusi. Dalam tahap pengembangan teknologi ini memberikan kemudahan. Kemudahan tersebut merupakan efek dari virtualisasi. Ketika pengguna ingin mengikuti perkembangan yang ada, tidak perlu merubah / membuat kembali dari awal. Pengguna dapat merubah dari level virtual (virtualisasi).

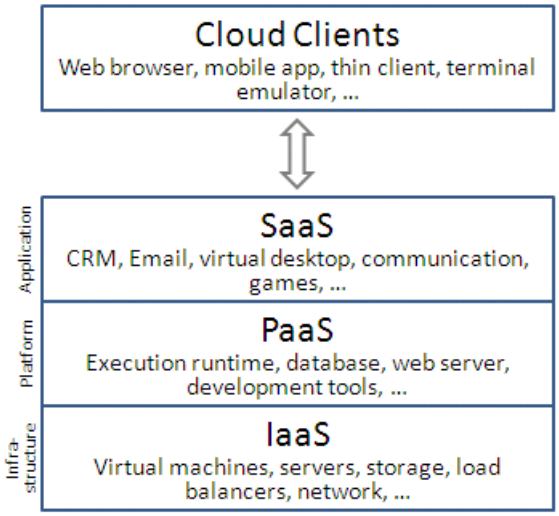
Terdapat 3 jasa yang disediakan dalam *cloud computing* [22] :

- **Software as a Service (SaaS)**
  Layanan jasa ini menyediakan aplikasi dan penyimpanan data (*database*) yang langsung dapat digunakan. Pengguna dapat mengakses aplikasi secara bersamaan dengan pengguna lainnya. SaaS dikenal juga sebagai *on-demand software* dan biaya yang dikenakan oleh penyedia jasa ini merupakan *pay-per-use*. Umumnya, pengguna akan dikenakan biaya bulanan atau tahunan. Pengguna tidak perlu susah - susah untk menginstall aplikasi pada *cloud* dikarenakan penyedia jasa tersebut akan melakukan hal tersebut. Saas memberikan keuntungan bagi perusahaan dengan cara mengurangi biaya operasional IT (Information Technology). Hal ini dikarenakan biaya untuk pemeliharaan dan pengadaan perangkat komputasi ditanggungkan kepada pihak penyedia jasa (*outsourcing*). Contoh dari Saas adalah layanan *e-mail*, media sosial.

- **Platform as a Service (PaaS)**
  Berbeda dengan Saas, layanan ini menyediakan perangkat komputasi yang dapat digunakan untuk menjalankan aplikasi dari pengguna. Layanan ini dapat menyediakan sistem operasi atau *framework* yang dibutuhkan dalam menjalankan apliksi tersebut. Contoh dari layanan jasa ini adalah : Windows Azure, Amazon Web Service, dan Google App Engine.

- **Infrastructure as a Service (IaaS)**
  layanan ini menyediakan infrastruktur IT kepada pengguna sesuai dengan

permintaan yang dapat diakses dengan jaringan internet. Infrstruktur tersebut meliputi perangkat keras seperti *harddisk, memory, firewall*, tipe server, alamat IP, *virtual local area networks*(VLANs), maupun aplikasi yang harus terinstall dalam server



Gambar 2.6: Layanan dalam *cloud computing* [16]

2.7 Virtualisasi

Virtualisasi merupakan suatu konsep dimana suatu sistem komputer secara nyata dapat dijalankan lebih dari satu / *multiple* pada suatu *host*. Virtualisasi bisa berjalan dengan adanya VMM (*Virtual Machine Monitor*) yang mengatur komunikasi antara lingkungan virtualisasi dengan perangkat keras. Virtualisasi juga dapat dipandang sebagai teknik optimisasi perangkat keras sebagai sumber komputasi [33]. Terdapat 3 arsitektur dari VMM :

- **Full Virtualization**
  Pada tipe ini, *hypervisor* dapat menjalankan beberapa sistem komputer virtual secara bersamaan. Setiap komputer virtual tersebut tidak perlu tahu bahwa mereka berjalan dalam lingkungan virtual

- **Para Virtualization**
  Dalam tipe ini, sedikit modifikasi dilakukan ddari Full Virtualization, dimana setiap sistem komputer virtual sadar bahwa mereka berjalan dalam lingkungan virtual dan sadar dengan keberadaan sistem komputer virtual lainnya. Selain itu, untuk setiap aplikasi yang dieksekusi pada masing masing sistem komputer virtual akan langsung diteruskan ke perangkat keras tanpa harus melalui Virtual Machine Monitor

- **OS Partitioning**
  Dalam tipe ini , virtualisasi dilakukan pada level OS, dimana *hypervisor* terinstall pada masing masing OS pada sistem komputer virtual yang diinstall diatas OS host. Batasan dalam tipe ini adalah, OS pada sistem komputer virtual harus mengikuti OS pada host.

Virtualisasi digunakan pada teknologi *cloud computing* dalam mengoptimisasikan kinerja server sehingga tidak ada sumber daya yang tidak terpakai. Virtualisasi yang dimaksud adalah virtualisasi perangkat keras (*hardware*) [24]. Virtualisasi tersebut memungkinkan ”virtual” server yang dapat dibuat sesuai dengan kebutuhan yang diperlukan dan berjalan pada server fisik. Spesifikasi untuk masing - masing virtual server dapat disesuaikan dengan kebutuhan aplikasi yang akan dijalankan diatasnnya dan tidak melebihi spesifikasi server fisik. Masing - masing virtual server akan saling terisolasi satu dengan yang lainnya.

Permasalahan virtualisasi tersebut muncul ketika hendak mengoptimisasikan sumber daya yang ada. Ketika suatu ”virtual” server dibuat untuk menjalankan suatu aplikasi yang spesifik, maka keseluruhan sistem perlu dibuat (Sistem operasi, alokasi sumber daya memori, *processor*) termasuk *library* yang dibutuhkan dalam menjalankan sistem / aplikasi tersebut. ”virtual” server tersebut akan diatur oleh suatu program VMM yang menjembatani komunikasi dengan server fisik. Dengan begitu, kapasitas memori dan alokasi sumber daya pada server tidak optimal dalam menjalankan ”virtual” server secara bersamaan.

2.8 Virtual Machine

Definisi dari *virtual machine* sendiri adalah suatu aplikasi atau *operating system* yang berjalan pada suatu aplikasi yang mengimitasi *hardware* asli yang sesuai dengan lingkungan aplikasi atau *operating system* tersebut. Pada umumnya virtualisasi yang sering digunakan adalah *full virtualization*, dimana terdapat suatu program tersendiri (*hypervisor*) yang mengatur *virtual machine* dengan pembagian *resource* hardware asli (*virtual machine* dapat dikatakan sebagai *guest* dan tempat dimana *virtual machine* diinstall dapat dikatakan sebagai *host*). Masing - masing *virtual machine* yang berjalan pada host akan terisolasi satu dengan lainnya, namun *resource* hardware *host* akan terbagi satu sama lain. terdapat 2 tipe *hypervisor* :

- **Type 1**
  Pada tipe ini, *hypervisor* langsung terinstall pada *hardware* dan berfungsi untuk mengatur *hardware* dan *operating system* yang terinstall diatasnya (*virtual machine*)

- **Type 2**
  Berbeda dengan tipe sebelumnya, *hypervisor* terinstall pada *operating system guest* dimana *virtual machine* terinstall layaknya suatu program.

Umumnya *hypervisor* tipe 2 sering digunakan pada komputer - komputer konvensional. Contoh aplikasi ini adalah VMWare[38] maupun VirtualBox [39].

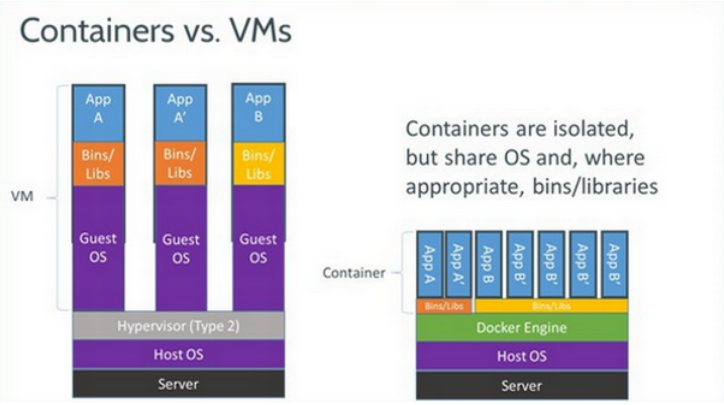
Kebutuhan virtualisasi pada awalnya merupakan suatu cara untuk memberikan kemudahan dalam meng*install operating system* pada mesin - mesin yang sudah lama. Selain itu, dengan virtualisasi kita dapat meng*install* berbagai *operating system* dalam sebuah mesin tanpa harus takut pengaturan tiap - tiap *operating system* akan saling mempengaruhi *operating system* lainnya (prinsip virtualisasi dimana setiap *virtual machine* akan saling tertutup dengan *virtual machine* lainnya). Selain itu, seperti yang digunakan dalam *cloud computing*, virtualisasi mengoptimalkan pemanfaatan *resource* hardware untuk meningkatkan layanan. Sebagai contoh, para developer akan menguji sistem atau aplikasi yang telah dibuat pada *environtment* sebenarnya, sehingga diperlukan suatu wadah uji coba yang sesuai dengan sistem atau aplikasi tersebut. Dengan adanya teknik virtualisasi tersebut, baik dengan *cloud computing* maupun pada komputer sendiri, developer tidak perlu takut sistem atau aplikasi yang dijalankan akan menyebabkan kerusakan pada *environtment* dan berdampak juga pada kerjaan - kerjaan sebelumnya.

Teknik *full virtualization* juga memiliki kekurangan. Untuk mengoptimalkan kinerja *host* dalam meningkatkan kualitas aplikasi yang berjalan pada *virtual machine* (*guest*). Untuk membentuk sebuah *guest*, *hypervisor* akan membutuhkan data memori, prosesor, jaringan yang *fixed*. Dengan begitu, jumlah *virtual machine* yang akan terbentuk akan terikat dengan besarnya *resource* hardware yang dimiliki oleh *guest*. Disatu sisi, ketika suatu *virtual machine* sedang menjalankan proses yang membutuhkan kemampuan komputasi tambahan, *hypervisor* tidak dapat memberikan *resource* tambahan, walaupun *virtual machine* lainnya dalam keadaan *idle* (tidak bekerja).

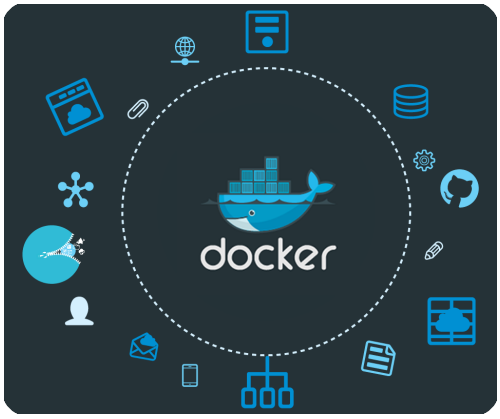
2.9 Docker

Docker, *platform* virtualisasi yang dikembangkan oleh Perusahaan Docker menggunakan teknik berbeda dengan *Full Virtualization*. Docker menggunakan *Docker engine* sebagai ganti dari *hypervisor*.*Docker engine* dikembangkan berdasarkan *Linux containers* (LXC), dimana level virtualisasi terletak pada sistem operasi [28] (bentuk virtualisasi *OS Partitioning*). Dengan begitu suatu server linux dapat menjalankan beberapa sistem linux yang saling terisolasi satu sama lain. Ker-

nel pada ”virtual” server, dalam hal ini disebut *container* yang dibentuk dengan libcontainer [40] akan menggunakan kernel yang sama dengan server fisik. Oleh karena itu, *container* yang dibentuk akan lebih kecil dan padat tanpa perlu menginstall sistem secara keseluruhan dari awal. *container* tidak perlu mengalokasikan sumber daya secara tetap dikarenakan *Docker engine* akan mengalokasikan sumber daya berdasarkan aplikasi yang berjalan pada *container*[26].



Gambar 2.7: Perbandingan virtualisasi pada VM dan container pada Docker[34]



Gambar 2.8: Logo Docker[27]

Walaupun begitu, untuk masing masing virtualisasi, baik itu yang menggunakan *hypervisor* maupun *Docker engine* pada Docker memiliki kelebihan dan kekurangan masing - masing. *Container* yang dibentuk pada Docker harus menggunakan sistem operasi yang sama dengan server fisik. Untuk saat ini, Docker baru mendukung sistem operasi Linux sebagai *container* dan berjalan pada sistem operasi Linux. Walaupun dapat berjalan pada sistem operasi lainnya, namun penggunaan Docker masih dijalankan dalam ”virtual” komputer yang menggunakan sistem operasi Linux pada *Virtual Machine*. Dengan alokasi sumber daya yang tidak tetap (sesuai dengan proses yang berjalan), maka *container* yang terbentuk akan lebih

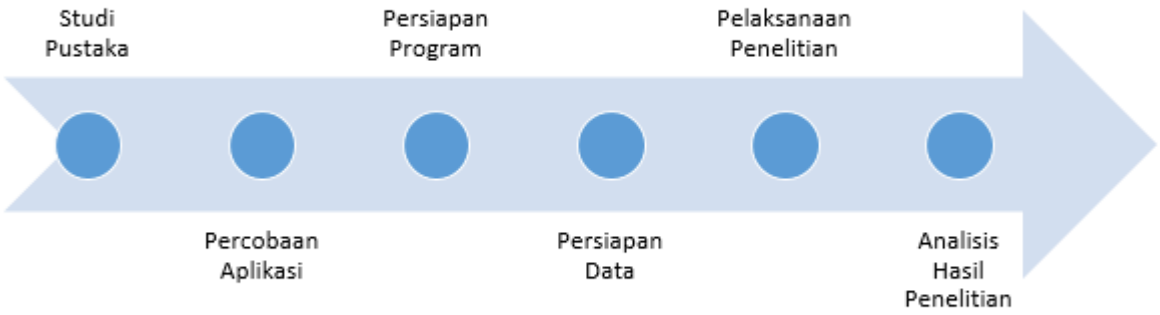
banyak dari ”virtual” server yag dihasilkan dengan menggunakan ”hypervisor” pada umumnya terikat dengan batasan sumber daya. Namun ”virtual” server tersebut lebih beragam dan tidak terikat dengan sistem operasi yang berjalan pada server fisik.

BAB 3
METODOLOGI PENELITIAN

Bab ini akan menjelaskan metodologi penelitian yang dilaksanakan.Bagian ini juga menjelaskan tahapan penelitian yang dilakukan, sistem pendukung penelitian, dan data penelitian

3.1 Tahapan Penelitian

Tahapan dari penelitian yang dilakukan terdiri dari studi pustaka, percobaan aplikasi, persiapan program, persiapan data, pelaksanaan penelitian dan analisis hasil penelitian. Tahapan tersebut dapat dilihat pada gambar berikut ini



Gambar 3.1: Tahapan Penelitian

Pada tahap studi pustaka dilakukan eksplorasi mengenai konsep penemuan obat baik secara tradisional maupun memanfaatkan IT, *cloud computing* dan virtualisasi komputer berdasarkan literatur - literatur tulis yang ada.

Pada tahap percobaan aplikasi penulis mencoba membiasakan diri dengan mengoperasikan aplikasi yang digunakan dalam penelitian, Docker dalam hal virtualisasi maupun Autodock dan Autodock Vina dalam melakukan *virtual screening*.

Pada tahap persiapan sistem dilakukan konfigurasi dan instalasi aplikasi tersebut. Aplikasi yang terlibat dalam penelitian ini diantaranya Autodock versi 4.2,

Autodock Vina versi 1.1, dan Autodock Tools versi 1.5.2 yang akan digunakan untuk preparasi data. Selain itu Docker versi terakhir yang akan digunakan 1.5.0 sebagai platform virtualisasi komputer. Pada tahapan ini , penulis juga membuat beberapa *script* sederhana yang akan digunakan untuk menjalankan aplikasi Autodock dan Autodock Vina secara otomatis dari awal pengerjaan hingga pengumpulan hasil akhir.

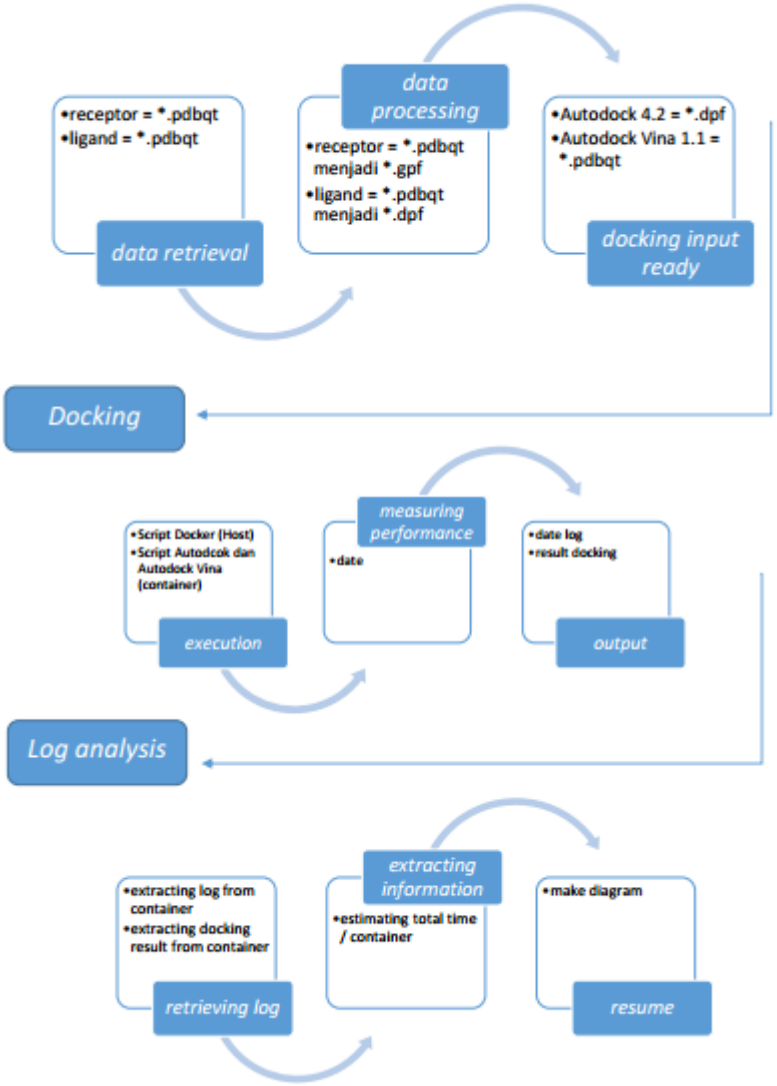
Pada tahap pemrosesan data dilakukan pre-docking atas data mentah penelitian. Proses pre-docking dilakukan sebagai syarat untuk menjalankan aplikasi Autodock 4.2 . Untuk Autodock Vina 1.1 , aplikasi dapat langsung dijalankan tanpa perlu data dipreparasi.

Pelaksanaan penelitian dilaksanakan pada sebuah PC desktop pada Laboratorium Jaringan Komputer lantai 5 gedung B Fasilkom UI. Kedua aplikasi tersebut dijalankan secara bergantian pada *platform* Docker yang telah ter*install*.

Tahap akhir dari penelitian ini adalah analisis hasil penelitian. Analisis hasil penelitian dilakukan atas data keluaran proses virtual screening. Data yang dianalisis adalah data keluaran berupa running time yang diperoleh dengan program date yang tersedia pada sistem operasi berbasis GNU/Linux.

3.1.1 Workflow Penelitian

Alur penelitian ini terbagi menjadi 3, yaitu *pre-docking*, *docking*, dan analisis hasil log *docking*. Proses ini dimulai dari mengunduh *ligand* dan *receptor* hingga mendapatkan hasil akhir berupa log hasil *molecular docking* dari aplikasi Autodock dan Autodock Vina. Berikut merupakan gambar *workflow* :



Gambar 3.2: *Workflow* penelitian

3.2 Sistem Pendukung Penelitian

Penelitian ini menggunakan *platform* Docker yang telah ter*install* pada PC desktop yang terdapat pada Laboratorium Jaringan Komputer Gedung B Lantai 5 Fasilkom UI. Berikut ini merupakan spesifikasi PC desktop yang digunakan :

Spesifikasi Komputer	
Prosesor	Intel i7-2600 CPU @ 3.40GHz X 4
RAM	5.8 GiB
Hardisk	418 GB
Sistem Operasi	debian 8.0
Instruction Set	64-bit

Tabel 3.1: Spesifikasi PC desktop

3.3 Data Penelitian

Data yang digunakan dalam penelitian ini seluruhnya merupakan data diperoleh melalui internet dan valid. Data yang valid merupakan salah satu persyaratan dalam penelitian ini untuk membandingkan hasil performa penelitian ini dengan penelitian terkait yang sebelumnya telah dilakukan.

3.3.1 Pemilihan Data

Data yang digunakan dalam penelitian ini diambil dari herbaldb[29]. herbaldb merupakan situs yang menyediakan data tanaman obat Indonesia yang dimiliki oleh Fakultas Farmasi Universitas Indonesia. Data yang diperoleh telah diolah oleh herbaldb dan jumlah data yang digunakan sebanyak 1406 data *receptor* dan 1 buah data *ligand*

3.3.2 Pemrosesan Data

Data yang telah diperoleh harus menjalani proses *pre-docking* sebelum dapat dilakukan *virtual screening*.Pada kesempatan ini penulis telah memperoleh data *ligand* dan *receptor* dalam ekstensi yang sama, sehingga langkah *pre-processing* hanya dilakukan pada file yang akan digunakan untuk menjalankan aplikasi Autodock versi 4.2 [31].

- **Mempersiapkan AutoGrid Parameter Files untuk receptor**
  AutoGrid Parameter Files pada receptor disiapkan dengan aplikasi Autodock Tools 1.5.2. AutoGrid Parameter Files berisi berkas yang siap dipetakan atas ligand menggunakan autogrid4. Hasil dari tahapan ini adalah berkas receptor dalam format *.gpf

- **Menghitung atomic affinity maps**
  Pemetaan yang sudah dilakukan pada receptor dan menghasilkan berkas re-

ceptor dalam format *.gpf harus dipetakan untuk setiap berkas ligand. Proses pemetaan ini dilakukan dengan cara menghitung atomic affinity maps pada berkas receptor dengan menggunakan aplikasi autogrid4. Tahapan ini akan menghasilkan berkas pemetaan dalam format *.glg, *.map, *maps.xyz dan *.maps.fld.

- **Mempersiapkan docking parameter files untuk setiap ligand**
  Tahapan terakhir adalah menyiapkan docking parameter files untuk setiap ligand dibantu modul phyton dari Autodock Tools 1.5.2 yang bernama . Docking parameter files ini berisi informasi terkait ligand seperti jumlah atom, koordinat atom dan jumlah torsi aktif yang dimiliki yang sudah dipetakan ke dalam informasi pada receptor. Hasil dari tahapan ini adalah berkas docking parameter files dalam format *.dpf.

Sedangkan tahapan *pre-docking* pada aplikasi Autodock Vina versi 1.1 tidak perlu dilakukan lagi dikarenakan ekstensi file telah memenuhi persyaratan dalam menjalankan aplikasi tersebut.

Berikut merupakan perbandingan tahapan *pre-docking* antara Autodock versi 4.2 dan Autodocok versi 1.1

Autodock 4.2	Autodock Vina 1.1
1. Mempersiapkan AutoGrid parameterfiles untuk receptor	1.Tahap pre-docking selesai
2. Menghitung atomic affinity maps	Transparansi proses pre-docking Autodock Vina 1.1
3. Mempersiapkan docking parameter files untuk setiap ligand	
4. Tahap pre-docking selesai	

Tabel 3.2: Preparasi data pada Autodock dan Autodock Vina

3.4 Skenario Eksperimen Penelitian

Dalam penelitian ini, penulis akan melaksanakan 10 skenario eksperimen dengan 5 skenario melibatkan Autodock dan 5 skenario melibatkan Autodock Vina. Tiap - tiap skenario pada Autodock maupun Autodock vina adalah sama, yaitu menjalankan aplikasi tersebut pada *platform* Docker dengan data yang telah dipreparasi. Perbedaan mendasar untuk setiap skenario adalah jumlah *container* yang digunakan dan persebaran data yang ditampung oleh tiap - tiap container. Persebaran data dapat diestimasi dengan total data *receptor* yang digunakan dibagi dengan jumlah *container* untuk setiap skenario.

Eksperimen	Jumlah container	Jumlah data receptor per container
1	50	28
2	100	14
3	150	9
4	200	7
5	250	5

Tabel 3.3: Pembagian *container* dan data setiap *container*

3.5 Pengukuran Kinerja

Dalam mengukur kinerja penelitian ini , penulis menggunakan 1 parameter yaitu *running time*. *Running time* digunakan program *date* yang tersedia dalam sistem operasi debian. Penulis akan mengumpulkan setiap *running time* untuk masing masing *container* yang dijalankan pada setiap eksperimen dan akan diolah (proses terlama, tercepat dan rata rata waktu proses).

BAB 4
HASIL PENELITIAN DAN ANALISIS

Bab ini akan membahas hasil penelitian yang telah dilakukan dan analisis terhadap hasil penelitian tersebut.

4.1 Hasil Penelitian

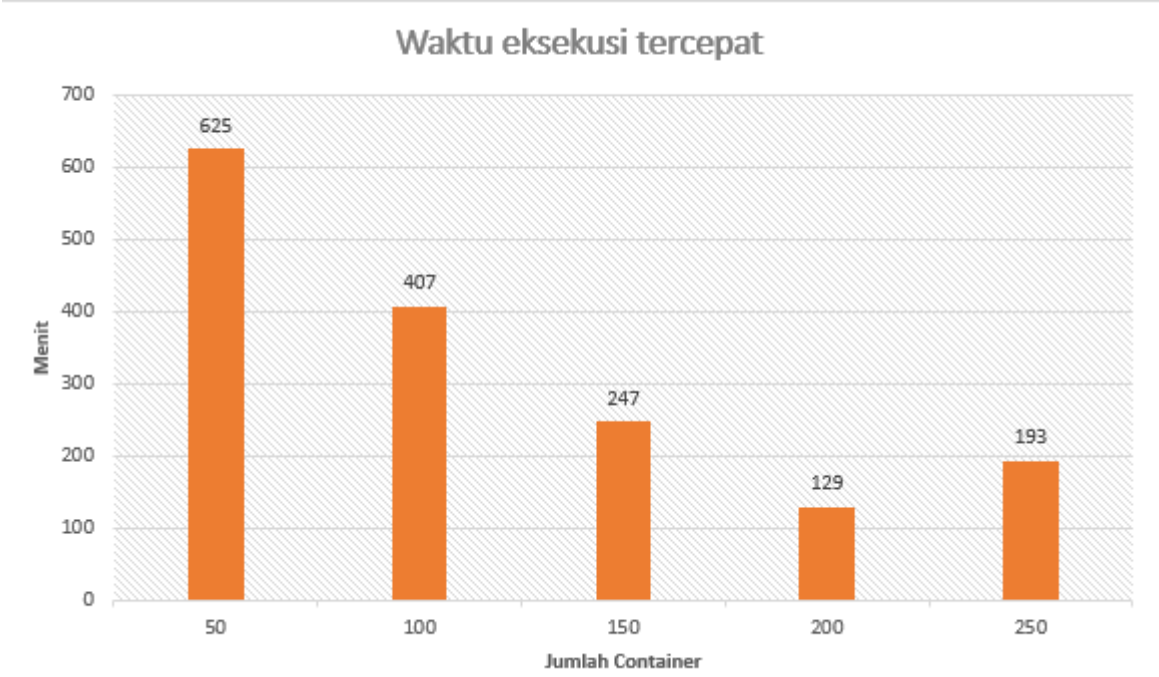
Berdasarkan kerangka penelitian yang telah dijelaskan pada bab sebelumnya, terdapat 10 skenario percobaan dimana masing masing 5 skenario pada aplikasi Autodock dan 5 skenario pada aplikasi Autodock Vina. Secara garis besar, skenario tersebut adalah menjalankan aplikasi yang telah ter*install* pada container Docker yang telah dibuat. Untuk setiap skenario jumlah data pada masing masing container akan dibagi berdasarkan total data (1406 data *receptor*) dibagi dengan jumlah container yang terbentuk. Setelah data terbagi, masing - masing container akan menjalankan aplikasi (Autodock ataupun Autodock Vina) dan waktu akan dicatat dari awal eksekusi data pertama hingga waktu akhir ketika aplikasi selesai mengeksekusi data terakhir.

Berikut merupakan data yang diperoleh dari 5 skenario aplikasi Autodock Vina :

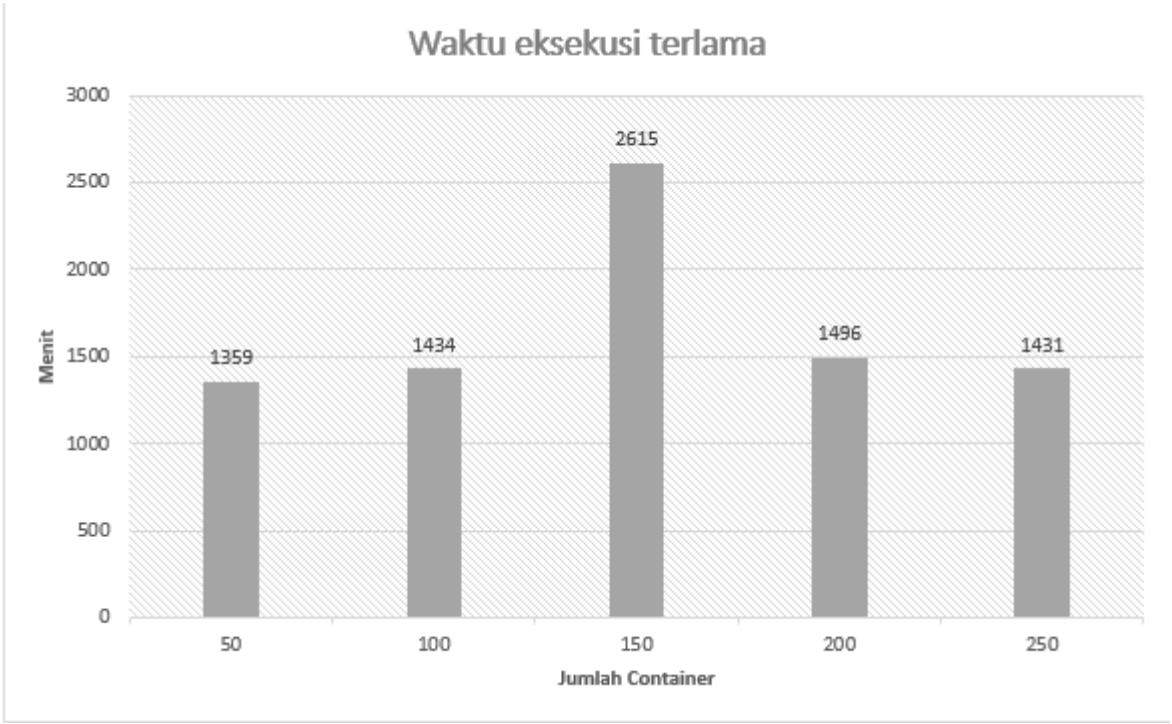
Jumlah container	Waktu eksekusi (menit)		
	tercepat	terlama	Rata-rata
50	625	1359	968
100	407	1434	891
150	247	2615	745
200	129	1496	705
250	193	1431	720

Tabel 4.1: Hasil skenario eksperimen pada aplikasi Autodock Vina

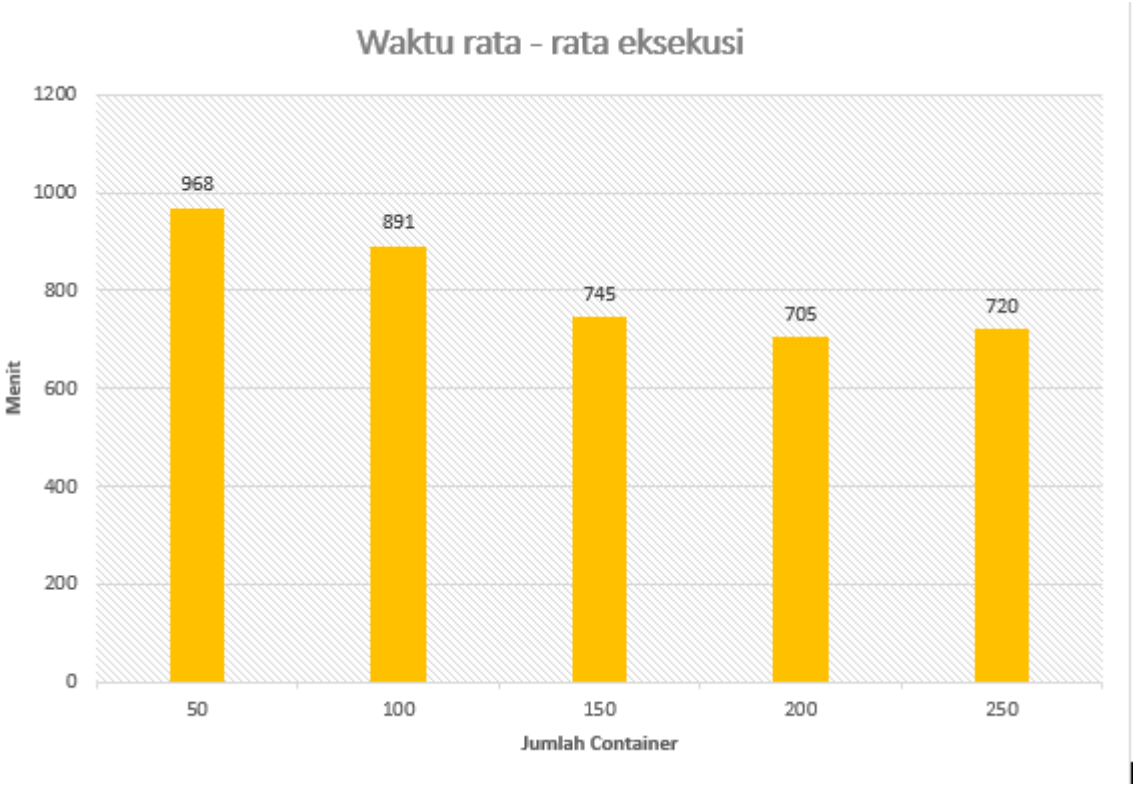
Untuk masing masing data waktu eksekusi tersebut dapat disajikan menjadi tabel grafik :



Gambar 4.1: Waktu eksekusi tercepat untuk masing - masing skenario



Gambar 4.2: Waktu eksekusi terlama untuk masing - masing skenario



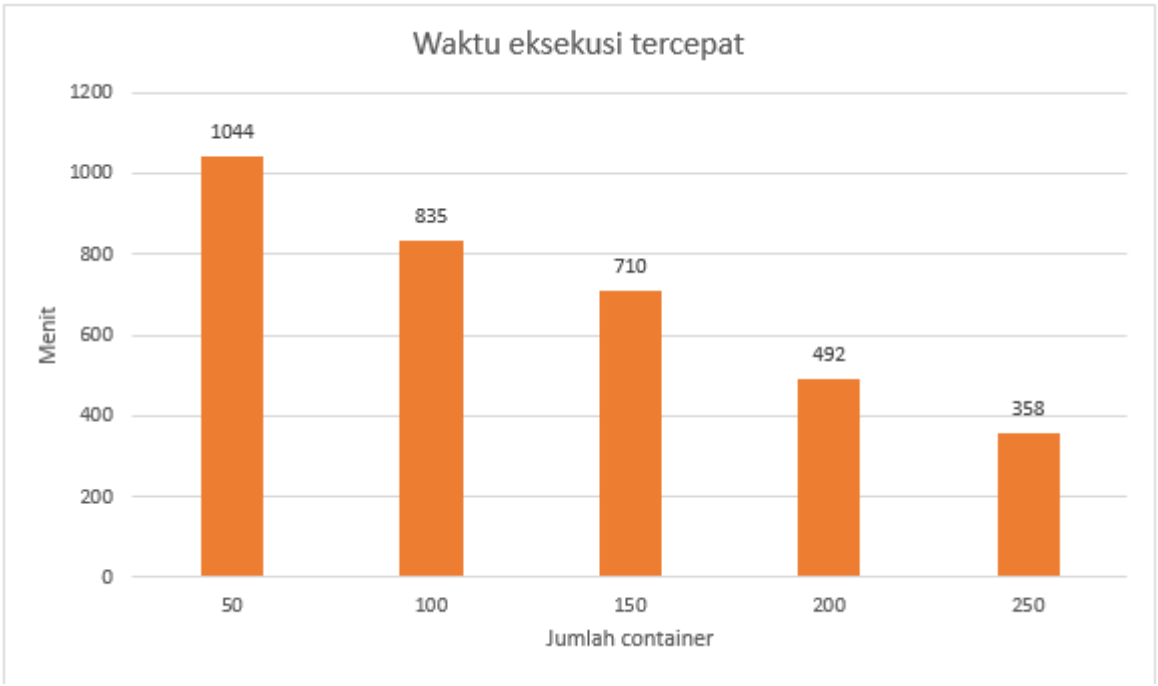
Gambar 4.3: Grafik dari rata - rata waktu eksekusi masing - masing skenario

Berikut merupakan data yang diperoleh dari 5 skenario aplikasi Autodock :

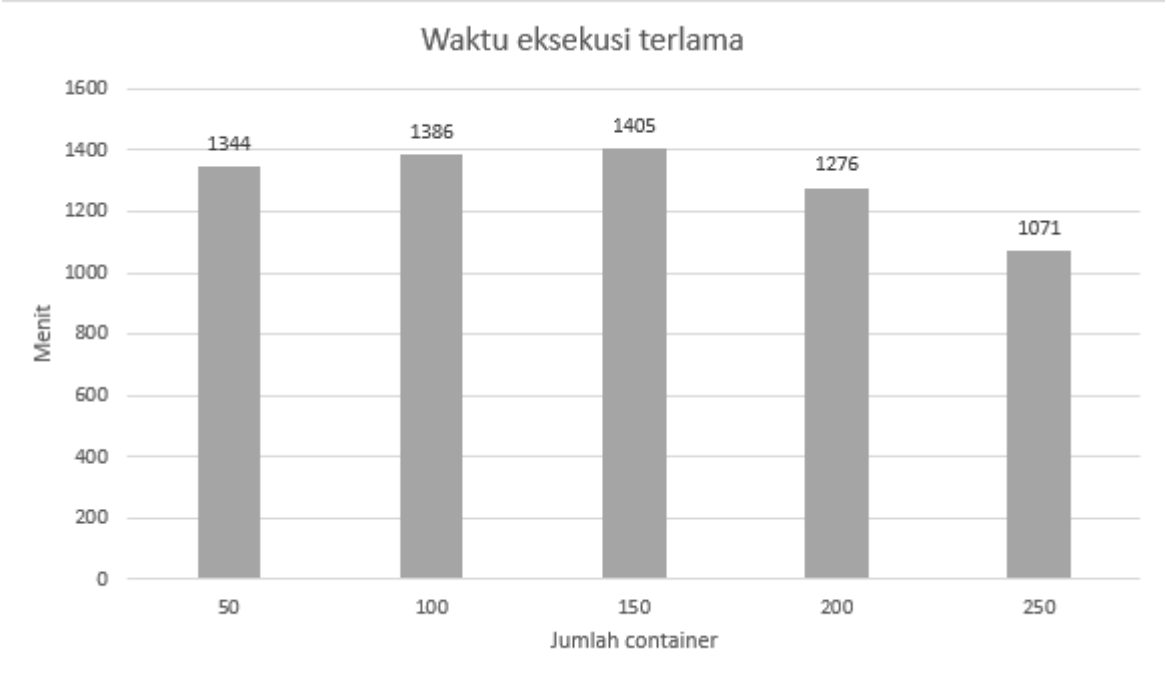
Jumlah container	Waktu eksekusi (menit)		
	tercepat	terlama	Rata-rata
50	1044	1344	1255
100	835	1386	1259
150	710	1405	1216
200	0	1276	1007
250	0	1071	811

Tabel 4.2: Hasil skenario pada eksperimen Autodock

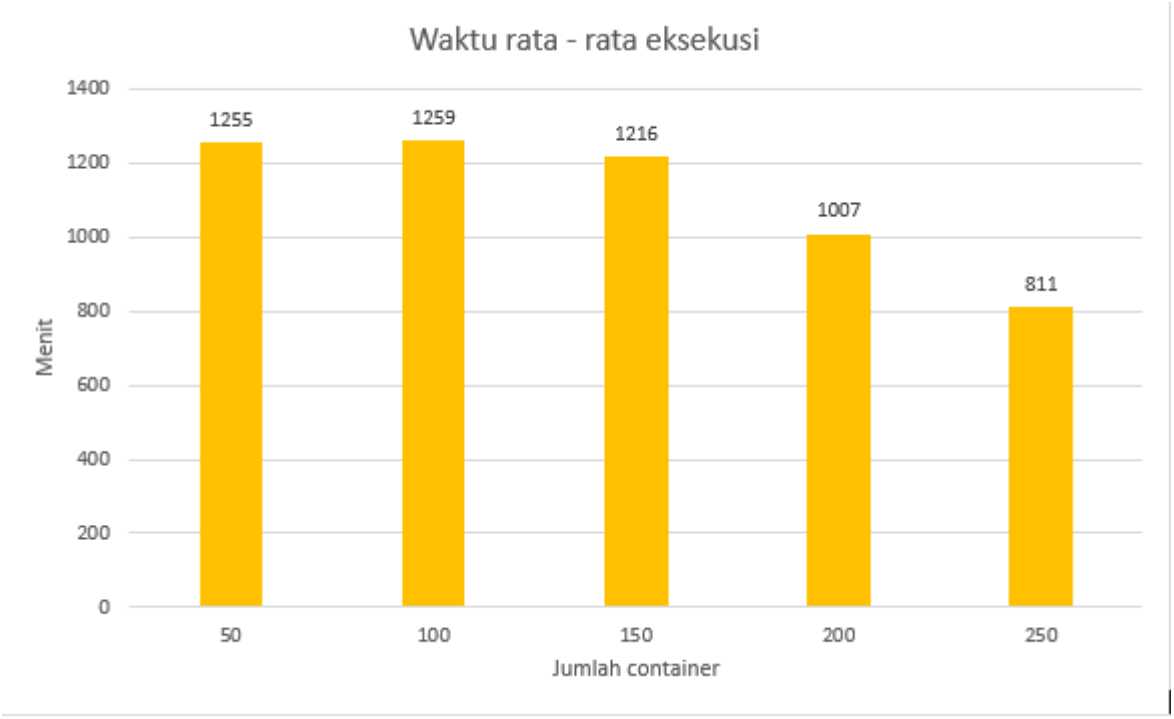
Untuk masing - masing data waktu eksekusi tersebut dapat disajikan menjadi tabel grafik :



Gambar 4.4: Waktu eksekusi tercepat untuk masing - masing skenario



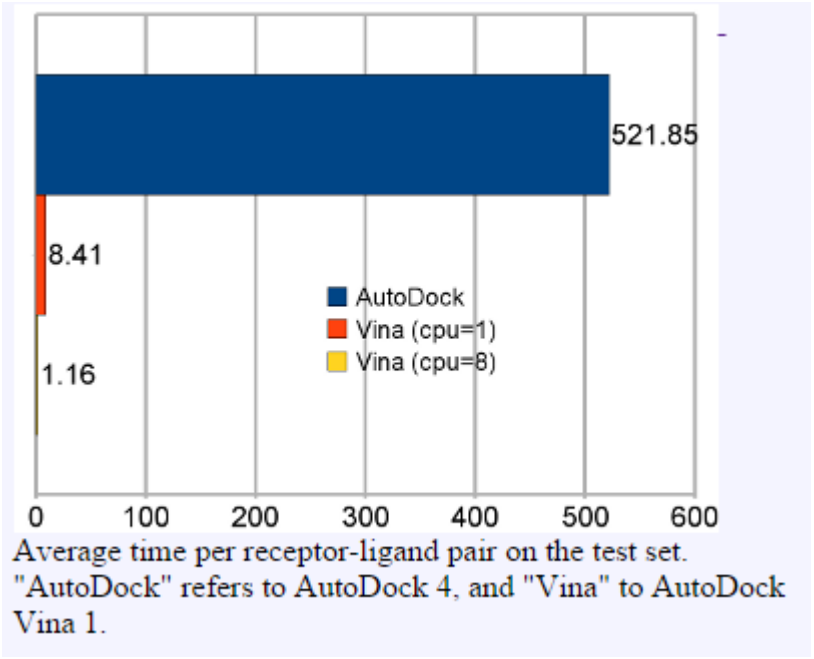
Gambar 4.5: Waktu eksekusi terlama untuk masing - masing skenario



Gambar 4.6: Grafik rata - rata waktu eksekusi masing - masing skenario

4.2 Analisis

Pada situs resmi Autodock telah ditampilkan perbandingan performa antara Autodock dan Autodock Vina [35] seperti berikut :



Gambar 4.7: Perbandingan waktu *running* pada situs resmi Autodock [14]

Eksekusi pada Autodock Vina jauh lebih cepat dibandingkan dengan Autodock, hal ini dikarenakan pada aplikasi Autodock Vina memanfaatkan *multithreading* sehingga dapat mengoptimalkan kinerja dari banyak CPU [14]. Sehingga memberikan gambaran awal bahwa dalam skenario penelitian yang dilakukan, Autodock Vina akan memberikan waktu eksekusi yang lebih cepat dibandingkan dengan Autodock. Untuk setiap rata - rata waktu eksekusi pada skenario Autodock Vina dibandingkan dengan rata - rata waktu eksekusi pada skenario Autodock memberikan hasil yang sama dengan grafik yang diberikan pada situs resmi Autodock.

Pada tabel waktu rata - rata eksekusi untuk aplikasi Autodock dan Autodock Vina dapat dilihat bahwa grafik dari setiap skenario memberikan *trend* menurun. Dengan begitu dapat dilihat bahwa jumlah container dan jumlah data menjadi faktor yang mempengaruhi waktu proses tersebut. Dapat disimpulkan bahwa, semakin banyak container yang dibentuk maka semakin sedikit data yang digunakan pada aplikasi Autodock maupun Autodock Vina pada masing - container. Semakin banyak container juga memperbanyak eksekusi aplikasi yang dilakukan secara paralel. Pada tabel grafik diatas terlihat tidak menurun secara mulus. Penulis memperkirakan bahwa kompleksitas molekul juga mengambil andil dalam lamanya waktu yang dibutuhkan oleh aplikasi Autodock maupun Autodock Vina.

Dari skenario yang telah dilakukan, penulis juga mengamati kinerja CPU dan memori yang digunakan oleh Docker. Berdasarkan penjelasan Marek Goldmann [36] , setiap container yang dibentuk akan membagi sama rata *resource* CPU. Pembagian tersebut hanyalah *relative weight*, dimana setiap container dapat menggunakan *shared* CPU, bukan kecepatan kasar CPU yang digunakan oleh container. Container yang terbentuk awalnya akan memiliki 1024 *shared* CPU dari CPU fisik, dan akan terbagi sama rata selama container baru akan terbentuk. Hasil dari penelitian ini baik menjalankan aplikasi Autodock maupun Autodock Vina dapat dilihat sebagai berikut :

Relative weight CPU (%)		
Jumlah container	Autodock Vina	Autodock
50	15,9 - 16,9	15,6 -19,6
100	7,9 - 12,3	8 - 8,6
150	5 - 5,6	6-7,6
200	3,3 - 3,9	4,6 - 5
250	0,7 - 3	3,6 - 4

Tabel 4.3: Persentasi *relative weight* pemanfaatan CPU

Walaupun dari tabel diatas tidak semulus perhitungan yang dikatakan oleh

Marek Goldmann, angka - angka tersebut tetap mendekati pembagian secara merata. Penelitian yang dilakukan menggunakan *desktop* CPU dengan prosesor Intel i-7 , dimana prosesor secara fisik memiliki 4 core namun dengan adanya *Hyper Threading*, terdapat logical core untuk masing masing core sehingga terbaca menjadi 8 core (800 %). Dalam kenyataannya, nilai *relative weight* dapat berubah mengikuti besarnya *task* yang dikerjakan oleh container tersebut dan nilai *relative weight* container lain. Nilai *relative wieght* container dalam keadaan *idle* dapat terbagi sama rata untuk container lain yang masih mengerjakan *task* (menyesuaikan dengan *task* dalam container). Dalam hal pembagian memori, setiap container dapat menggunakan memori pada *host* hingga kapasitas maksimum memori pada *host*. Pada eksperimen yang dijalankan, terlihat bahwa untuk aplikasi Autodock Vina maupun Autodock yang dijalankan hanya menggunakan 0,1 % - 0,9 % dari memori pada komputer *host*

Dalam penelitian ini penulis juga membandingkan performa yang diberikan oleh Docker dengan penelitian yang telah dilakukan oleh Muhammad H. Hilman [37]. Beliau menggunakan aplikasi Autodock Vina dalam *virtual screening*. Berikut merupakan gambar tabel waktu yang dibutuhkan dalam memproses 1 molekul yang dapat dikerjakan dalam *virtual cluster* dalam Amazon EC2 :

Parameter	SCluster	LCluster	XLCluster
Memory	1,7 GB	7,5GB	1,5 GB
Compute Unit / Processors	1,2 GHz X 1	1,2 GHz X 4	1,2 GHz X 8
Platform	32-bit	64-bit	64-bit
Harga/jam	$0,085	$0,34	$0,68

Tabel 4.4: Spesifikasi *virtual cluster* [37]

Parameter	SCluster	LCluster	XLCluster
Total ExecTime(s)	375.624	108.179	48.421
AvgExecTime(s)	267.000	77.000	34.000

Tabel 4.5: Hasil pengukuran *virtual screening* [37]

Pemrosesan 1 Molekul dalam detik (s)			
Parameter	SCluster	LCluster	XLCluster
Total ExecTime(s)	267,15	76,94	34,43
AvgExecTime(s)	189,9	54,76	24,18

Tabel 4.6: Waktu yang dibutuhkan dalam memproses 1 buah Molekul berdasarkan tabel [37] diatas

Hasil yang diperoleh dari eksperimen penelitian ini :

Pemrosesan 1 Molekul dalam detik (s)		
Jumlah container	Autodock Vina	Autodock
50	41,30	53,55
100	38,02	54,01
150	32,19	52,29
200	30,08	43,37
250	31,02	35

Tabel 4.7: Waktu yang dibutuhkan dalam memproses 1 buah Molekul

Dapat dilihat bahwa hasil yang diberikan oleh Docker pada sebuah *desktop* PC tidak jauh berbeda dengan yang diberikan oleh Amazon EC2 . Dari tabel tersebut dapat dilihat dalam eksekusi pada Docker dengan container terkecil dapat memberikan hasil yang lebih cepat dibandingkan dengan LCluster. Waktu tercepat masih dipegang oleh XLCluster, namun hasil yang diberikan oleh Docker dapat mendekati *virtual cluster* yang digunakan oleh Amazon EC2.

DAFTAR REFERENSI

[1] Jeff Clark. (n.d). *Introduction to LaTeX*. 26 Januari 2010. http://frodo.elon.edu/tutorial/tutorial/node3.html.

[2] Pharma *Drug Discovery and Development : UNDERSTANDING THE R and D PROCESS*.February 2007.http://www.phrma.org/sites/default/files/pdf/rd_brochure_022307.pdf.

[3] Myers S, Baker A. *Drug discovery - an operating model for a new era*.Nat Biotechnol 2001; 19: 72730

[4] Matthew Segall.*Can we really do computer-aided drug design?*.J Comput Aided Mol Des (2012) 26:121124

[5] Si-sheng OU-YANG, Jun-yan LU, Xiang-qian KONG, Zhong-jie LIANG, Cheng LUO, Hualiang JIANG.*Review Computational drug discovery*.Acta Pharmacologica Sinica (2012) 33: 11311140

[6] Rajamani R, Good AC.2007. *"Ranking poses in structure-based lead discovery and optimization: current trends in scoring function development"*.Current Opinion in Drug Discovery and Development 10 (3): 30815.

[7] Bachwani Mukesh, Kumar Rakesh.*Molecular Docking : A Review*.IJRAP (2011) 2 (6) 1746 - 1751.

[8] *Drug design*. http://strbio.biochem.nchu.edu.tw/classes/special_topics_biochem/course_ppts/rational_drug_design-2014.pdf.diakses pada 3 Juni 2015 pukul 04:49

[9] Schneider G, Fechner U.August 2005. *"Computer-based de novo design of drug-like molecules"*. Nat Rev Drug Discov 4 (8): 64963.

[10] Kitchen DB, Decornez H, Furr JR, Bajorath J.2004.*"Docking and scoring in virtual screening for drug discovery: methods and applications"*.Nature reviews. Drug discovery 3 (11): 93549.

[11] Wei BQ, Weaver LH, Ferrari AM, Matthews BW, Shoichet BK.2004. *"Testing a flexible-receptor docking algorithm in a model binding site"*. J. Mol. Biol. 337 (5): 116182.

[12] Morris.2007,May 04.*"AutoDock is the most cited docking software"* http://autodock.scripps.edu/news/autodock-is-the-most-cited-docking-software diakses 3 Juni 2015 pukul 06:59

[13] Garrett M. Morris et al.2009.*"AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility"*.J Comput Chem. 2009 December ; 30(16): 27852791. doi:10.1002/jcc.21256

[14] Oleg Trott and Arthur J. Olson.2010.*"AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading"*.J Comput Chem. 2010 January 30; 31(2): 455461. doi:10.1002/jcc.21334.

[15] Logo Autodock. http://www.kdm.wcss.wroc.pl/w/images/thumb/Autodock_logo.gif/200px-Autodock_logo.gif diakses 3 Juni 2015 pukul 07:49.

[16] Layanan cloud. http://upload.wikimedia.org/wikipedia/commons/thumb/b/b5/Cloud_computing.svg/2000px-Cloud_computing.svg.png diakses 3 Juni 2015 pukul 20.18

[17] Contoh model *pharmacophore*.http://en.wikipedia.org/wiki/Pharmacophore#/media/File:PharmacophoreModel_example.svg diakses 3 Juni 2015 pukul 18:09

[18] Rester, U.July 2008. *"From virtuality to reality - Virtual screening in lead discovery and lead optimization: A medicinal chemistry perspective"*. Curr Opin Drug Discov Devel 11 (4): 55968.

[19] Top 500 List. http://www.top500.org/lists/2014/11/ diakses 3 Juni 2015 pukul 18:11

[20] Muhammad H. Hilman.Juni 2010.*"Evaluasi Kinerja Autodock 4.2 dan Autodock Vina 1.1 dalam Proses Molecular Docking dan Virtual Screening di Lingkungan Cluster Hastinapura"*.Fakultas Ilmu Komputer,Universitas Indonesia.

[21] Ian Foster.Juli 2002.*"Lecture Note : What is the Grid? A Three Point Checklist* .http://dlib.cs.odu.edu/WhatIsTheGrid.pdf . diakses 3 Juni 2015 19.00

[22] Shuang Zhang, Shufen Zhang, Xuebin Chen, Xiuzhen Huo.2010.*"Cloud Computing Research and Development Trend"*.2010 Second International Conference on Future Networks.

[23] Fatima A. Alali,Chia-Lun Yeh.2012.*"Cloud Computing: Overview and Risk Analysis"*.JOURNAL OF INFORMATION SYSTEMS Vol. 26,No. 2 Fall 2012 pp.1333

[24] IBM Global Education White Paper.October 2007. *"Virtualization in Education"*.http://www-07.ibm.com/solutions/in/education/download/Virtualization_in_Education.pdf

[25] Dinesh. *"Disdvantages of Virtualization,What's Your Opinion"*.http://www.sysprobs.com/disadvantages-virtualization-opinion

[26] Chris Wolf. August 2014.*"VMware and Docker - Better Together"*.http://blogs.vmware.com/cto/vmware-docker-better-together/ diakses 3 Juni 2015 pukul 20.35

[27] Logo Docker. http://getcloudify.org/img/docker.png

[28] Tobby Banerjee.November 2014.*"LXC vs LXD vs Docker - Making sense of the rapidly evolving container ecosystem"*.https://www.flockport.com/lxc-vs-lxd-vs-docker-making-sense-of-the-rapidly-evolving-container-ecosystem/. diakses 3 Juni 2015 pukul 20.45

[29] Arry Yanuar, Abdul Munim , Akma Bertha Aprima Lagho, Rezi Riadhi Syahdi, Marjuqi Rahmat, and Heru Suhartanto.2011. *Medicinal Plants Database and Three Dimensional Structure of the Chemical Compounds from Medicinal Plants in Indonesia*.IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 1, September 2011

[30] Website resmi The Scripps Research Institute, http://www.scripps.edu/about/index.html diakses 6 Juni 2015 pukul 14.39

[31] The Scripps Research Institute, Tutorial Autodock 4.2, http://autodock.scripps.edu/faqs-help/tutorial diakses 6 Juni 2015 pukul 10.29

[32] The Scripps Research Institute, Tutorial Autodock Vina 1.1 , http://vina.scripps.edu/tutorial.html diakses 6 Juni 2015 pukul 10.29

[33] Amir Ali Semnanian.May 2013.*"A STUDY ON VIRTUALIZATION TECHNOLOGY AND ITS IMPACT ON COMPUTER HARDWARE"*Department of Computer Engineering and Computer Science California State University, Long Beach.

[34] Virtual machine vs docker. http://www.rightscale.com/blog/sites/default/files/docker-containers-vms.png

[35] Gambar tabel perbandingan Autodock dan Autodock Vina. http://vina.scripps.edu/.

[36] Marek Goldmann.11 Sept 2014.*"Resource management in Docker"*. https://goldmann.pl/blog/2014/09/11/resource-management-in-docker/. diakses pada 12 Juni 2015 pukul 16.08

[37] Muhammad H. Hilman.Januari 2012.*"Analisis Teknik Data Mining dan Kinerja Infrastruktur Komputasi Cloud Sebagai Bagian dari Sistem Perancangan Obat Terintegrasi"*.Fakultas Ilmu Komputer,Universitas Indonesia.

[38] Situs resmi VMWare. http://www.vmware.com/ap. diakses 12 Juni 2015 pukul 22.49

[39] Situs resmi VirtualBox.https://www.virtualbox.org/. diakses 12 Juni 2015 pukul 22.55

[40] Steven J. Vaughan-Nichols.11 Juni 2014.*"Docker libcontainer unifies Linux container powers"*.http://www.zdnet.com/article/docker-libcontainer-unifies-linux-container-powers/. diakses 12 Juni 2015 pukul 23.00