ANALISIS KINERJA SISTEM CLUSTER TERHADAP APLIKASI SIMULASI DINAMIKA MOLEKULAR NAMD MEMANFAATKAN PUSTAKA CHARM++

Muhammad Nazir Akbar, A.Benny Mutiara MQN,Chandra Yulianto

Universitas Gunadarma
Jl. Margonda Raya 100, Depok, 16424
e-mail : amutiara@staff.gunadarma.ac.id,chandra@staff.gunadarma.ac.id

Abstrak

Program NAMD mampu untuk memenuhi semua kriteria yang diinginkan. Program ini dirancang dengan mengimplementasikan pustaka Charm++ untuk pembagian tugas perhitungan secara paralel. NAMD memiliki sistem automatic load balancing secara periodik yang cerdas, sehingga dapat memaksimalkan penggunaan kemampuan mesin yang tersedia. Program ini juga dirancang secara moduler, sehingga dapat dimodifikasi dan ditambah dengan sangat mudah.NAMD menggunakan banyak kombinasi algoritma perhitungan dan tehnik-tehnik numerik lainnya dalam melakukan tugasnya. NAMD 2.5 mengimplementasikan semua tehnik dan persamaan perhitungan yang digunakan dalam dunia simulasi dinamika molekular saat ini. NAMD dapat berjalan diatas berbagai mesin paralel termasuk arsitektur cluster.

Kata Kunci : Atom, Charm++, Cluster, Linux, Molekular, NAMD, Paralel, Simulasi, VMD.

1. Pendahuluan

Proses simulasi dinamika molekular menghitung kedudukan atom dengan memecahkan persamaan-persamaan dari pergerakannya secara numerik. Proses perhitungan ini dibantu oleh rumus medan energi empiris yang memperkirakan energi atom dalam sistem biopolimer secara aktual.

Awalnya, program simulasi dinamika molekular dibangun diatas mesin-mesin serial. Namun untuk menghitung proses simulasi molekular yang lebih besar, mutlak diperlukan kekuatan komputasi yang juga lebih besar. Salah satu cara untuk dapat menjalankan simulasi semacam ini adalah dengan menggunakan komputer paralel. Selain faktor kecepatan, komputer paralel juga lebih unggul dari segi biaya.

NAMD merupakan program paralel pada UNIX yang dirancang khusus untuk simulasi dinamika molekular struktur biologi. NAMD dirancang untuk berjalan dengan efisien diatas mesin-mesin paralel. Software NAMD merupakan properti intelektual dari The Board of Trustees of the University of Illinois, mengatasnamakan The Theoretical Biophysics Group pada Beckman Institute. Pada saat penulisan ini dibuat, NAMD telah mencapai versi rilis 2.5, yang juga digunakan dalam pengujian ini nantinya.

Selain NAMD, beberapa program komputer telah dikembangkan dengan tujuan yang sama. Dua diantaranya yang paling populer adalah X-PLOR dan CHARMM. Dibanding program lain yang sejenis, NAMD memiliki kelebihan tersendiri, yaitu:

1. Kompatibilitas rumus medan energi.

   Baik NAMD, CHARMM, maupun X-PLOR menggunakan rumus medan energi yang sama. Rumus medan energi ini mencakup perhitungan kondisi interaksi lokal, yaitu interaksi keterikatan (antara 2, 3, dan 4 atom), juga interaksi pairwise (elektrostatis dan energi van der Waals).
2. Efisiensi algoritma elektrostatis.

   NAMD menerapkan algoritma PME (Particle Mesh Ewald) yang berarti mengikutsertakan interaksi elektrostatis secara penuh pada simulasi dinamika molekular. Algoritma ini berguna untuk mengurangi kompleksitas perhitungan evaluasi energi elektrostatis dari O (N^2) menjadi O (N log N).
3. Skema *multiple time step*.

NAMD menggunakan metode integrasi *velocity* Verlet untuk mengetahui posisi dan kecepatan atom-atom pada waktu tertentu. Untuk mengurangi waktu evaluasi energi elektrostatis jangka panjang, NAMD menerapkan skema multiple time step. Interaksi lokal yang terjadi (bonded, van der Waals, dan interaksi elektrostatis dalam jarak tertentu) dihitung pada tiap-tiap langkah waktu tertentu. Interaksi-interaksi lain yang memiliki jangkauan lebih panjang (interaksi elektrostatis diluar jarak tertentu) hanya dihitung beberapa kali saja. Penjelasan lebih lengkap dapat dibaca pada NAMD 2.5 User Guide (terlampir).

4. Kompatibilitas format input/output.

   Format data masukan dan keluaran yang digunakan NAMD identik dengan format data yang digunakan oleh CHARMM 11 dan X-PLOR. Format data input meliputi file koordinat PDB, file struktur PSF, dan file parameter medan energi. Sedangkan format data output meliputi file koordinat PDB dan file *trajectory* DCD. Hal ini memastikan data input/output dari NAMD juga dapat dibaca oleh CHARMM atau X-PLOR dan sebaliknya.

5. Kemudahan dalam pemodifikasian dan penambahan.

   Tujuan perancangan NAMD yang utama adalah dalam hal ekstensibilitas. NAMD dirancang dalam gaya berorientasi objek menggunakan bahasa C++. Karena simulasi dinamika molekular merupakan bidang yang relatif baru, tehnik-tehnik dan algoritma baru masih akan terus ditemukan. Untuk itu modul NAMD dirancang agar siapapun dapat menambahkan dan mencoba algoritma simulasi molekular baru dengan mudah.

6. Simulasi interaktif.

   NAMD memperbolehkan suatu sistem yang sedang dalam proses simulasi untuk dapat dilihat secara visual 3 dimensi dan diubah-ubah menggunakan aplikasi luar VMD (Visual Molecular Dynamics). Aplikasi ini datang dalam paket yang berbeda dengan NAMD, namun dikembangkan oleh pihak yang sama. Untuk informasi lebih lanjut dapat dilihat pada alamat http://www.ks.uiuc.edu/Research/vmd/

7. Keseimbangan beban kerja.

   Salah satu faktor penting dalam aplikasi paralel adalah pendistribusian beban perhitungan yang sama pada setiap prosesor. Simulasi molekular paralel mendistribusikan beban dalam bagian-bagian yang disebut *spatial decomposition*. Namun pembagian beban ini menyebabkan bagian yang telah di-*mapping* ke setiap prosesor menjadi tidak beraturan, khususnya pada proses evaluasi macam-macam tipe energi yang berbeda.

   NAMD mengatasi masalah ini dengan menggunakan *spatial decomposition* yang simpel dan seragam, dimana keseluruhan model dibagi-bagi menjadi ruang berbentuk kubus yang disebut *patch*. Selanjutnya, algoritma *load balancer* NAMD akan mengatur proses pembagian dan perhitungan seseimbang mungkin. Selama proses simulasi, NAMD akan memonitor beban dan melakukan pengaturan yang diperlukan secara otomatis.

2. Dasar-Dasar Simulasi NAMD

Untuk menjalankan sebuah simulasi, NAMD memerlukan 4 (empat) macam file masukan, yaitu:

1. File PDB (Protein Data Bank)
2. File PSF (Protein Structure File)
3. File parameter medan energi (force field parameter)
4. File konfigurasi NAMD

2.1. File PDB (Protein Data Bank)

File PDB menyimpan data koordinat atom dan atau kecepatan dari suatu sistem molekular. File ini menyimpan keseluruhan informasi mengenai nama, jenis, dan juga jaringan molekul tersebut.

Lebih lengkapnya, dalam file ini tersimpan data pengarang, catatan revisi, catatan jurnal, referensi, sekuen asam amino, *stoichiometry*, lokasi struktur sekunder, *cyrstal lattice*, kelompok simetri, serta catatan ATOM dan HET-ATM. Catatan ATOM dan HET-ATM inilah yang menyimpan koordinat-koordinat dari protein-protein, air-air, ion-ion, dan macam-macam atom heterogen kristal lainnya. Arsitektur file PDB memungkinkan penyimpanan lebih dari satu set koordinat atom-atom.

File PDB digunakan sebagai format data masukan dan keluaran pada simulasi NAMD. PDB merupakan standar format data untuk banyak program simulasi dinamika molekular lainnya, termasuk CHARMM dan X-PLOR.

2.2. File PSF (Protein Structure File)

File PSF menyimpan informasi struktural dari suatu protein secara spesifik. Informasi ini terbagi atas lima bagian utama yaitu mengenai macam-macam atom, *bond*, *angle*, *dihedral*, dan *improper*. Kesemuanya diperlukan untuk pengaplikasian medan energi tertentu pada simulasi nantinya.

Format file PSF ini mengikuti standar format dari CHARMM dan X-PLOR. Walaupun terdapat perbedaan cara penyimpanan pada kedua format ini, NAMD dapat membaca keduanya dengan sempurna. NAMD juga menyediakan utiliti **`psfgen`** yang dapat menghasilkan file PSF berdasarkan file PDB yang diinginkan.

2.3. File parameter medan energi (force field parameter)

Pada simulasi dinamika molekular, yang dimaksud dengan medan energi (force field) adalah persamaan matematika dari kondisi potensial yang dirasakan atom-atom dalam sebuah sistem. NAMD sendiri mampu membaca keempat macam tipe medan energi, yaitu CHARMM, X-PLOR, AMBER, dan GROMACS. Secara default, NAMD menggunakan parameter tipe X-PLOR yang lebih fleksibel.

File parameter ini berisikan konstanta numerik yang diperlukan untuk mengevaluasi gaya dan energi pada struktur dan koordinat atom-atom (disediakan oleh file PSF dan PDB). Parameter ini berguna antara lain untuk mengatur panjang *equilibrium* dan kekuatan ikatan antar atom.

2.4. File konfigurasi NAMD

File ini berisikan semua konfigurasi dan pilihan yang dibutuhkan NAMD untuk menjalankan sebuah simulasi. Dengan kata lain, file ini memberitahu NAMD bagaimana sebuah simulasi harus dijalankan. Sebuah simulasi dinamika molekular membutuhkan kontrol masukan seperti temperatur, langkah waktu (time step), lama simulasi, pengaktifan fasilitas tertentu, nama file input, nama file output, dan berbagai parameter spesifik lainnya.

3. Charm++ Sebagai Pustaka Paralel Pada NAMD

Charm++ merupakan sebuah *runtime library* yang mengizinkan komunikasi antar objek-objek C++ dengan sangat efisien. Charm++ lebih merupakan ekstensi paralel untuk bahasa C++, yang dikembangkan oleh PPL (Parallel Programming Laboratory) dalam beberapa tahun terakhir. Charm++ menggunakan model pemrograman SPMD (Single Program Multiple Data) yang dipopulerkan oleh MPI. Model pemrograman ini bisa dibilang sangat mirip dengan CORBA, Java RMI, atau RPC, namun lebih difokuskan kepada mesin-mesin paralel kinerja tinggi.

Charm++ merupakan bahasa yang bersifat *data driven* dan secara mutlak mendukung OOP (Object-Oriented Programming). Bahasa ini mendukung tiga jenis objek paralel, yaitu: chares (individual), chare groups, dan chare arrays. Charm++ hanya digunakan untuk menulis aplikasi-aplikasi khusus paralel. Lain halnya dengan MPI, pustaka ini tidak memiliki *compiler* sendiri, sehingga tidak mendukung otomatisasi kerja paralel. Lagipula, otomatisasi kerja paralel hanya berguna bagi beberapa aplikasi numerik biasa.

Agar dapat bekerja secara paralel, sebuah algoritma terlebih dahulu haruslah diubah menjadi paralel pula. Pada kebanyakan kasus, hanya sedikit bagian program serial yang harus diubah agar dapat bekerja secara paralel. Charm++ mendukung penggunaan chare arrays, yang menjadikannya

lebih ekspresif dan mudah digunakan dibanding bahasa paralel lainnya. Konversi program serial ke paralel akan menjadi lebih mudah dengan menggunakan kerangka kerja Charm++.

Charm++ memiliki fasilitas lebih yang tidak dimiliki oleh pustaka paralel lainnya. Pustaka ini mendukung migrasi objek-objek yang bersifat application-independent, yang hampir tidak mungkin dilakukan pada MPI. Kode paralel MPI seringkali memiliki masalah skalabilitas yang disebabkan oleh ketidakseimbangan beban kerja dan ongkos komunikasi. Dengan mengubah kode MPI kedalam Charm++, kinerja aplikasi akan meningkat secara dramatis. Charm++ bahkan dapat menjalankan kode MPI secara langsung, tanpa harus diubah terlebih dulu, cukup dengan menggunakan fasilitas AMPI (Adaptive MPI)

Charm++ dapat berjalan pada banyak mesin, baik pada arsitektur *shared-memory* atau *distributed-memory*, dan SMP (Symmetric Multi Processor) atau non-SMP. Secara khusus, pustaka ini dapat berjalan pada mesin-mesin serial, Windows, Linux, Ethernet Cluster, Myrinet Cluster, SGI Origin 2000, IBM SP3, Cray T3E, Intel Paragon, ASCI Red, dan mesin apapun yang dapat menjalankan MPI atau SHMEM diatasnya. Beberapa aplikasi besar yang memanfaatkan librari paralel Charm++, antara lain aplikasi simulasi dinamika molekular NAMD dan aplikasi CSAR (Center for Simulation of Advanced Rockets).

Charm++ diciptakan oleh Prof. L. V. Kale, dari Departemen Ilmu Komputer, University of Illinois, Urbana-Champaign, dan grup riset PPL (Parallel Programming Laboratory). Lebih dari 100 orang telah memberikan kontribusi pada pustaka ini, selama kurang lebih 15 tahun. Charm++ bersifat open-source dan bebas untuk pemakaian yang bertujuan akademik, edukasi, dan riset.

Charm++ bekerja secara asinkron dan bekerja menggunakan metode *Asynchronous Remote Method Invocation*. Artinya, pada algoritma paralel, *caller* berlangung terus, tanpa diblok sampai kembalinya *method*, secepatnya. Objek-objek pemrograman juga bersifat *remote*, yang bisa berada di mesin berbeda, dipisahkan oleh suatu jaringan. *Method Invocation* berarti pustaka ini menggunakan *method* untuk memanggil (menggunakan *caller*) kelas-kelas C++ satu sama lain.

Kelebihan metode pemrograman pada Charm++ adalah tidak diperlukannya proses *block*, saat proses *invoke* atau invokasi remote method berlangsung. Hal ini merupakan perbedaan terbesar dari Charm++ dengan algoritma CORBA, Java RMI, atau RPC. Proses invokasi sebuah *asynchronous method* mempunyai implementasi dan semantik yang sama dengan proses pengiriman pesan tunggal biasa.

Dibandingkan metode lain, cara asinkron memiliki kelebihan dari segi efisiensi waktu. Invokasi dapat diimplementasikan seperti pengiriman pesan tunggal biasa. Tidak seperti metode sinkron, proses *thread blocking*, *unblocking*, dan *return message* tidak diperlukan pada cara ini.

Kelebihan lain dari metode asinkron adalah kemudahan dalam pelaksanaan tugas secara paralel. Pada dua eksekusi perintah yang berbeda, tidak ada keharusan bagi perintah kedua untuk menunggu diselesaikannya perintah pertama. Keduanya dapat berjalan bersama, sehingga unggul dari segi waktu. Pada metode sinkron, *caller* akan mengalami *block*, menggunakan *sync*, yang mana memerlukan penggunaan *threads.*

4. Aplikasi VMD

VMD (Visual Molecular Dynamics) merupakan aplikasi yang dirancang untuk memvisualisasikan dan menganalisa sistem biopolimer (protein, asam nukleid, lipid, dan membran. VMD berjalan diatas mayoritas sistem UNIX, Apple MacOS X, dan Microsoft Windows. Seperti halnya NAMD, VMD juga dikembangakan oleh Theoritical Biophysics Group, di University of Illinois and Beckman Institute, Urbana. VMD memang dikembangkan sebagai aplikasi visual bagi aplikasi simulasi dinamika molekular NAMD.

5. Simulasi Pengujian Pada Mesin Cluster

Simulasi dengan beberapa data molekular berbeda akan dicoba pada mesin cluster untuk mendapatkan hasil yang diinginkan. Data hasil simulasi ini kemudian akan diolah lebih lanjut untuk mengetahui efektifitas dan efisiensi NAMD 2.5 pada arsitektur cluster.

Simulasi akan dilakukan memakai data molekular **decalanin** (66 atom, tiny), **ER-GRE** (36.573 atom, spherical) dan **ApoA1** (92.224 atom, periodic). Masing-masing jenis molekul ini memiliki jumlah atom dan parameter perhitungan yang berbeda. Jumlah atom dan tingkat kerumitan parameter inilah yang akan menentukan, pada pengujian mana data molekular tersebut digunakan. Lebih jelasnya dapat dilihat pada file konfigurasi NAMD masing-masing data molekular (terlampir).

5.1. Metode Pengujian

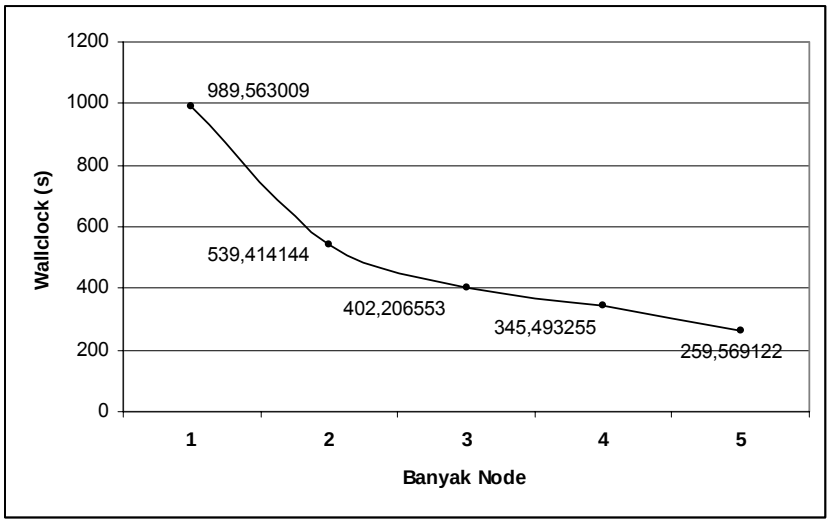
Pengujian dilakukan sebanyak 3 (tiga) kali untuk masing-masing simulasi dan atau konfigurasi mesin yang berbeda. Tiap-tiap simulasi dilakukan dalam kondisi *fresh* setelah me-*restart* mesin terlebih dahulu. Setelah 3 kali pengujian, mesin akan di-restart kembali untuk memulihkan kondisi memori sebelum simulasi berikutnya dilakukan. Nilai rata-rata kemudian diambil untuk diolah lebih lanjut sebagai perbandingan dengan nilai hasil simulasi lainnya.

Pengujian ini sendiri dilakukan dengan berbagai metode, dalam kaitannya pada pembuktian efektifitas dan efisiensi NAMD diatas mesin cluster. Sebagai tambahan, akan dilakukan semacam simulasi *benchmark* untuk mencoba mengukur kinerja maksimal dan batas ketahanan mesin cluster.

5.2. Pengukuran Speedup Mesin Cluster

Tidak ada cara yang lebih tepat untuk menilai efektifitas suatu mesin paralel selain dengan mengukur *speedup* sistem tersebut. Nilai ini didapat dengan membandingkan hasil kerja sekuensial dengan hasil paralel untuk menilai tingkat percepatan yang dicapai mesin paralel.

Simulasi pengujian ini dilakukan dengan menggunakan data molekular ER-GRE (36.573 atom, spherical). Simulasi ini terdiri dari 500 numsteps, dengan temperatur sebesar 300^{o} Kelvin, dan menggunakan sampai empat file parameter medan energi.



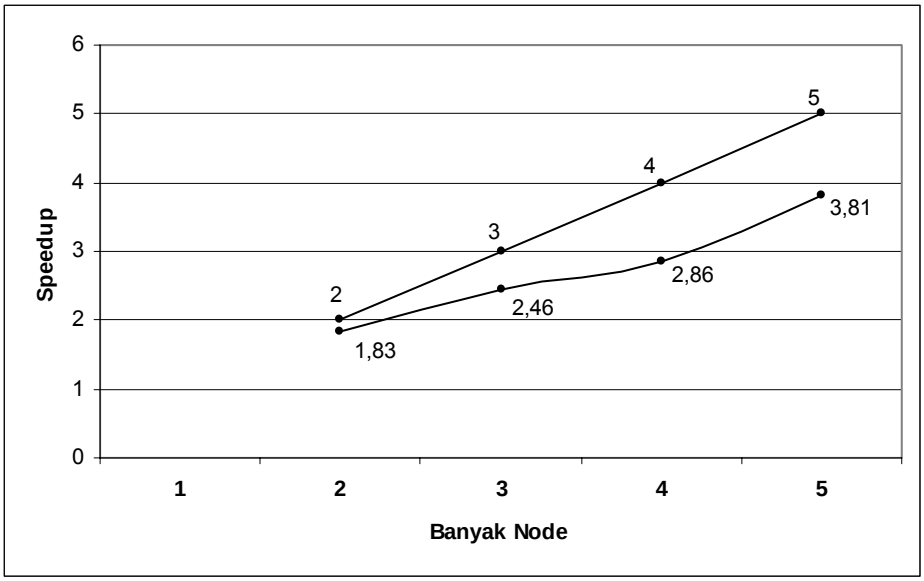
Gambar 1. Grafik nilai rata-rata lama waktu simulasi ER-GRE.

Menggunakan hasil diatas, kemudian dapat dihitung nilai speedup mesin cluster. Speedup diperoleh dengan membagi waktu sekuensial dengan waktu paralel tiap-tiap pengujian. Persamaannya sebagai berikut:

$$\text{Speedup} = \frac{\text{Waktu Sekuensial}}{\text{Waktu Paralel}} = \frac{Ts}{Tp}$$

Tabel 1. Nilai speedup simulasi ER-GRE pada mesin cluster.

Banyak Node	Rata-rata WallClock (s)	Speedup	Speedup Ideal
1	989,563009	-	-
2	539,414144	1,83	2 kali
3	402,206553	2,46	3 kali
4	345,493255	2,86	4 kali
5	259,569122	3,81	5 kali



Gambar 2. Grafik speedup simulasi ER-GRE pada mesin cluster.

Dari angka-angka diatas dapat terlihat bahwa tingkat percepatan yang dicapai oleh sistem cluster ini masih jauh dari ideal. Jika dilihat pada pemakaian 2 buah prosesor (2 node), speedup yang diperoleh mencapai tingkat 1,83 kali lebih cepat. Peningkatan ini sangat berarti karena dapat menghemat waktu sampai 450 detik. Pada pemakaian 3 buah node peningkatan ini semakin menurun, dan pada pemakaian 4 node, speedup yang dicapai bahkan masih belum melampaui nilai 3.

5.3. Tingkat Efisiensi Paralel Mesin Cluster

Pengujian tingkat efisiensi sebuah mesin paralel merupakan sebuah tolak ukur yang bersifat objektif. Besar objek atau data simulasi yang menjadi bahan pengujian akan menentukan tingkat efisiensi sebuah mesin paralel. Oleh karena itu, untuk mesin dan besar data yang berbeda, akan diperoleh tingkat efisiensi yang berbeda pula.

Pengujian kali ini dilakukan masih dengan menggunakan data molekular ER-GRE (36.573 atom, spherical). Oleh karena itu hasil pengujian sebelumnya masih bisa digunakan untuk mencari efisiensi paralel mesin cluster. Efisiensi paralel didefinisikan dengan persamaan berikut:

$$\text{Efisiensi Paralel } (\eta) = \frac{\text{Waktu Sekuensial}}{\text{P x Waktu Paralel}} \text{ x } 100\%$$

Tabel 2. Tingkat efisiensi paralel simulasi ER-GRE pada mesin cluster.

Banyak Node	Rata-rata WallClock (s)	Jumlah Pemroses x Waktu Paralel (s)	Efisiensi Paralel (%)
1	989,563009	-	-
2	539,414144	1.078,828288	91,73%
3	402,206553	1.206,619659	82,01%
4	345,493255	1.381,973020	71,61%
5	259,569122	1.297,845610	76,25%

Tabel diatas memperlihatkan bahwa tingkat efisiensi paralel mesin cluster dalam menjalankan simulasi ER-GRE sangat memuaskan. Meski terlihat penurunan tingkat efisiensi pada setiap penambahan jumlah node, penurunan ini masih dalam batas yang wajar.

Ada fenomena yang sedikit menarik perhatian pada pengujian ini. Pada pemakaian 5 buah node, tingkat efisiensi terlihat mengalami kenaikan dari 71,61% menjadi 76,25%. Kira-kira apa yang akan terjadi apabila dilakukan pengujian pada 6 buah node? Ada kemungkinan, tingkat efisiensi paralel ini akan mengalami kenaikan lagi.

Walaupun hal ini tidak dapat dibuktikan, karena keterbatasan mesin pengujian dan lain hal, fakta ini tetap menunjukkan satu hal: bahwa dalam simulasi tersebut, pemakaian 5 mesin masih lebih efisien dibandingkan pemakaian 4 mesin. Untuk saat ini, fenomena tersebut hanya dapat dianggap sebagai faktor acak (random factor) yang tidak terduga. Faktor acak ini sangat erat kaitannya dengan apa yang disebut dengan ongkos komunikasi atau *communication cost.*

5.4. Ongkos Komunikasi

Pada pemecahan suatu masalah dengan cara paralel, biasanya tingkat efisiensi akan menurun saat jumlah pemroses dinaikkan. Kenapa hal ini bisa terjadi? Hal ini disebabkan tidak lain oleh terjadinya ongkos komunikasi. Ongkos komunikasi merupakan faktor yang sangat dominan terjadi pada proses komputasi paralel. Ongkos komunikasi adalah lama waktu yang digunakan untuk proses pembuatan, pengiriman, dan penerimaan data. Semakin banyak jumlah pemroses, dalam hal ini jumlah node yang digunakan, maka akan semakin besar nilai ongkos komunikasi tersebut. Angka ini akan menjadi semakin besar jika tingkat masalah yang harus dipecahkan semakin kecil, karena waktu yang dibutuhkan untuk memecahkan masalah malah lebih singkat ketimbang waktu komunikasi data itu sendiri.

Sebagai pembuktian, akan dilakukan simulasi pengujian dengan menggunakan data molekular decalanin (66 atom, tiny). Simulasi ini terdiri dari 1000 numsteps, dengan temperatur sebesar 300^{o} K. Lebih jelasnya dapat dilihat pada file konfigurasi untuk simulasi decalanin (terlampir). Pada tabel berikut, dapat dilihat lama waktu simulasi pengujian pada mesin cluster:

Tabel 3. Tingkat efisiensi paralel simulasi decalanin pada mesin cluster.

Banyak Node	WallClock (s)	Jumlah Pemroses x Waktu Paralel (s)	Efisiensi Paralel (%)
1	6,792329	-	-
2	17,772844	35,545688	19,11%
3	18,340601	55,021803	12,34%

Pada tabel diatas dapat dilihat bahwa lama waktu simulasi semakin bertambah seiring kenaikan jumlah pemroses. Kasus ini menjadi semacam antiklimaks bagi komputasi paralel. Simulasi ini dapat menunjukkan secara sempurna, untuk tingkat permasalahan yang relatif kecil, sangat tidak efisien jika diselesaikan secara paralel. Waktu komunikasi disini terbukti jauh lebih lama dibandingkan waktu pemrosesan itu sendiri.

5.5. Simulasi Benchmark

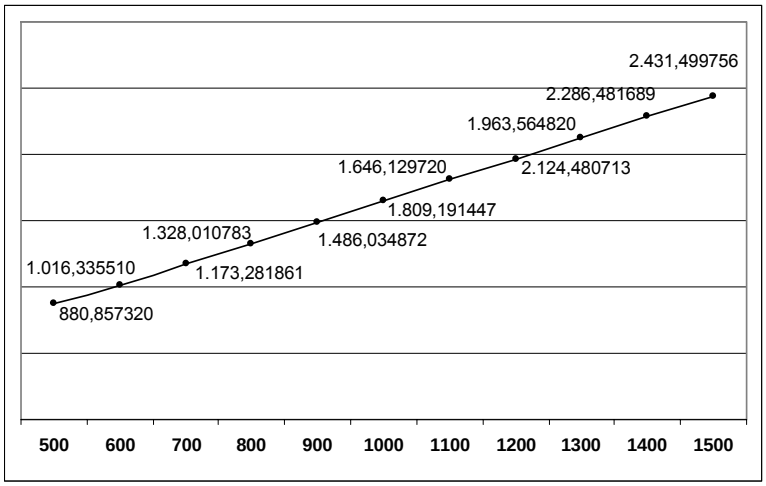
Pengujian berikut ini lebih bertujuan untuk mengetahui batas kemampuan mesin cluster dalam menjalankan simulasi skala besar. Untuk simulasi pengujian kali ini, data molekular yang akan dipakai adalah ApoA1 (92.224 atom, periodic).

Simulasi ini tidak akan dijalankan secara paralel bertahap, melainkan langsung memakai seluruh kemampuan mesin cluster yang ada (5 mesin). Sebagai catatan, simulasi ApoA1 yang hanya dijalankan diatas 1 mesin (cluster1) saja, memakan waktu kurang lebih 4 hari (99,88403 jam). Waktu ini lebih berupa harga estimasi, karena simulasi tadi sebenarnya tidak pernah terselesaikan, karena keterbatasan waktu dan lain hal. Kemungkinan besar, akan dibutuhkan waktu yang lebih lama lagi untuk menyelesaikan simulasi tadi secara benar.

Simulasi ini terhitung berat karena menyertakan perhitungan elektrostatis penuh, PME (Particle Mesh Ewald), ditambah penggunaan 2 (dua) file parameter medan energi. Simulasi pengujian akan dilakukan sebanyak 11 (sebelas) kali, masing-masing dengan besar numsteps yang berbeda.

Tabel 4. Nilai rata-rata lama waktu simulasi ApoA1.

Jumlah *numsteps*	Rata-rata WallClock (s)	Rata-Rata CPUTime (s)	Rata-rata Memory Used (kB)
500	880,857320	846,403341	43.886
600	1.016,335510	1.001,873352	42.570
700	1.173,281861	1.156,503337	42.571
800	1.328,010783	1.309,586670	42.571
900	1.486,034872	1.463,863363	42.572
1000	1.646,129720	1.622,039998	42.571
1100	1.809,191447	1.763,896688	42.569
1200	1.963,564820	1.924,103312	42.571
1300	2.124,480713	2.077,383301	42.571
1400	2.286,481689	2.232,583333	42.571
1500	2.431,499756	2.383,686605	42.571



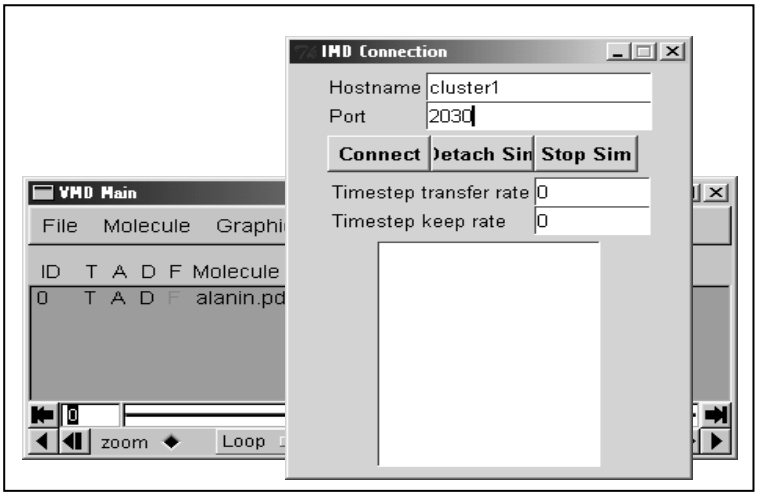
Gambar 3. Grafik nilai rata-rata lama waktu simulasi ApoA1.

6. Simulasi Dinamika Molekular Interaktif

Mengetahui mekanisme dari penggabungan dan pemecahan makromolekul sangatlah penting dalam mempelajari proses dan struktur-struktur biologis. Beberapa contoh yang umum seperti proses pengikatan dan pelepasan substrat-substrat dalam suatu reaksi enzim, juga proses pengenalan sekuen DNA oleh domain ikatan DNA dalam protein. Proses-proses semacam ini, seperti halnya pelepasan ikatan molekul avidin-biotin complex, pemecahan retinal dari bacteriorhodopsin, peregangan DNA, dapat dipelajari secara lebih mendalam dengan bantuan rekayasa SMD (Steered Molecular Dynamics).

Aplikasi simulasi dinamika molekular NAMD, memiliki kemampuan melakukan beberapa tehnik SMD berbeda, seperti rotasi, translasi, dan pengaplikasian energi pada satu atau lebih atom. Dikombinasikan dengan aplikasi VMD, yang datang dengan dukungan IMD (Interactive Molecular Dynamics), SMD bahkan dapat dilakukan secara *real-time*.

Apa kehebatan simulasi interaktif ini sebenarnya? Sebagai contoh ringan, IMD memperbolehkan kita untuk menarik molekul gula dalam sebuah simulasi saluran gliserol GlpF. Kita dapat menarik, mendorong, dan mengaplikasikan energi tertentu pada atom atau molekul tadi sesuai keinginan. Kita bahkan dapat merasakannya di tangan kita seolah-olah atom atau molekul itu berada di tangan kita sendiri (dengan bantuan hardware mouse/tracker khusus). Dengan semua kelebihan ini, pemahaman-pemahaman baru mengenai cara kerja dan fungsi suatu sistem biopolimer akan lebih mudah diperoleh.



Gambar 4. Fasilitas IMD pada aplikasi VMD.

Pada prinsipnya, setiap simulasi NAMD dapat dimodifikasi agar dapat dijalankan secara interaktif. Tentu saja simulasi yang dijalankan dengan cara ini membutuhkan mesin dengan tenaga yang cukup besar. Mesin cluster pengujian terbukti mampu menangani IMD secara real-time tanpa masalah berarti.

Simulasi pengujian dilakukan dengan menggunakan data molekular decalanin (66 atom, tiny). Setelah mengubah file konfigurasi decalanin agar dapat dijalankan secara interaktif, simulasi IMD dapat dijalankan dengan mulus pada mesin cluster menggunakan 5 (lima) buah node.

7. Kesimpulan

Melalui hasil pengujian dan analisa yang diperoleh sebelumnya, dapat diambil kesimpulan bahwa:

1. Pengaturan konfigurasi hardware memiliki kontribusi besar dalam kinerja keseluruhan mesin cluster. Terbukti bahwa kesalahan kecil dalam pengaturan BIOS, misalnya besar *shared-memory* untuk VGA, dapat membawa perubahan besar dalam hal kecepatan kerja.
2. NAMD terbukti memiliki algoritma paralel dengan skalabilitas tinggi, karena saat jumlah pemroses dinaikkan, tingkat efisiensi tetap dapat dijaga dengan memperbesar ukuran masalah.
3. Mesin cluster pengujian dapat dikatakan memiliki tingkat efisiensi dan efektifitas yang cukup tinggi. Terbukti dengan banyaknya waktu yang dapat dihemat saat menjalankan simulasi-simulasi besar. Penurunan tingkat efisiensi lebih disebabkan oleh ukuran permasalahan yang tidak lagi cukup besar untuk layak diproses pada banyak node.
4. Binari NAMD yang dikompilasi secara lengkap (menyertakan TCL, FFTW, VMD plugins molfile) akan menjalankan simulasi-simulasi tertentu dengan lebih cepat. Pada penyelesaian simulasi ApoA1 yang menyertakan perhitungan PME (Particle Mesh Ewald), waktu yang dapat dihemat cukup signifikan.
5. Mesin cluster terbukti pula mampu menjalankan simulasi interaktif secara *real-time* tanpa masalah berarti. IMD (Interactive Molecular Dynamics) akan menjadi standar teknologi dalam dunia simulasi dinamika molekular di masa mendatang.

8. Daftar Pustaka

[1] Ted G. Lewis, Hesham El-Rewini, In-Kyu Kim, *Introduction to Parallel Computing*, Prentice-Hall International, Englewood Cliffs, New Jersey, 1992.

[2] Theoritical and Computational Biophysics Group, *NAMD – Scalable Molecular Dynamics*, http://www.ks.uiuc.edu/Research/namd.
M. Bhandarkar, R. Brunner, C. Chipot, A. Dalke, S. Dixit, P. Grayson,
J. Gullingsrud, A. Gursoy, D. Hardy, W. Humphrey, D. Hurwitz, N. Krawetz,

[3] M. Nelson, J. Phillips, A. Shinozaki, G. Zheng, F. Zhu, *NAMD User's Guide Version 2.5*, Theoritical Biophysics Group, University of Illinois and Beckman Institute, Urbana, Illinois, 2003.

[4] Timothy Isgro, James Phillips, Marcos Sotomayor, Elizabeth Villa, *NAMD TUTORIAL*, Theoritical and Computational Biophysics Group, University of Illinois and Beckman Institute, Illinois, 2003.
Tamar Schlick, Robert D. Skeel, Axel T. Brunger, Laxmikant V. Kale,

[5] John A. Board, Jr., Jan Hermans, Klaus Schulten, *Algorithmic Challenges in Computational Molecular Biophysics*, Journal of Computational Physics 151, 1999.

[6] Laxmikant Kale, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Aritomo Shinozaki, Khrisnan Varadarajan, Klaus Schulten, *NAMD 2: Greater Scalability for Parallel Molecular Dynamics*, Journal of Computational Physics 151, 1999.

[7] James C. Phillips, Gengbin Zheng, Sameer Kumar, Laxmikant V. Kale, *NAMD: Biomolecular Simulation on Thousands of Processors*, IEEE, 2002

[8] Laxmikant Kale, *The Charm++ Programming Language Manual*, Parallel Programming Laboratory, University of Illinois, Urbana-Champaign, Illinois, 2003.
[9] L. V. Kale and S. Krishnan, *Charm++: Parallel Programming with Message-Driven Objects*, Parallel Programming using C++, Edisi Gregory V. Wilson and Paul Lu, hal. 175-213, MIT Press, 1996.
[10] E. Caddigan, J. Cohen, J. Gullingsrud, J. Stone, *VMD User's Guide*, Theoritical Biophysics Group, University of Illinois and Beckman Institute, Urbana, Illinois, 2003.