BAB II
DASAR TEORI

2.1 Komputasi Paralel

Komputasi adalah proses yang dijalankan prosesor dalam sebuah rentang satuan waktu yang meliputi:

1) operasi baca: menerima masukan berupa data dengan ukuran tertentu,
2) operasi penghitungan: melakukan operasi aritmatik atau logik terhadap masukan,
3) operasi tulis: mengembalikan hasil penghitungan dengan ukuran tertentu.

Komputasi paralel adalah eksekusi proses pengoperasian aritmatik atau logik yang sejenis secara bersamaan.[2]

Sebuah masalah komputasi yang kompleks dibagi menjadi bagian-bagian yang lebih kecil, sederhana, dan dijalankan di beberapa prosesor secara bersamaan sehingga keseluruhan proses dapat diselesaikan dengan lebih cepat.

Pada komputasi sekuensial (serial) masukan diterima dan keluaran dikirim melalui memori internal atau perangkat antarmuka. Sementara pada komputasi paralel sebuah prosesor dapat menerima masukan atau mengembalikan keluarannya melalui *shared memory* atau prosesor lain melalui bus interkoneksi antar prosesor atau jaringan komputer.

Tujuan utama komputasi paralel adalah peningkatan kecepatan komputasi. Hukum Amdahl (Gene Amdahl, 1967) dan Gustafon (John Gustafon, 1988) menjelaskan peningkatan kecepatan komputasi paralel pada sejumlah prosesor identik.

Hukum Amdahl menjelaskan bahwa peningkatan kecepatan komputasi paralel adalah:

$$S = \frac{1}{\frac{1-\alpha}{P}+\alpha} \qquad (2\text{-}1)$$

dengan

α = rasio bagian rutin program yang tidak dapat diparalelisasi terhadap keseluruhan rutin program

P = cacah prosesor

Hukum Gustafon menjelaskan peningkatan kecepatan komputasi beberapa prosesor dengan persamaan yang lebih sederhana:

$$S = \alpha + P\,(1-\alpha) \qquad (2\text{-}2)$$

Barry Wilkinson dan Michael Allen di dalam bukunya menjelaskan bahwa fokus utama dalam merencanakan solusi menggunakan multiprosesor adalah perkiraan akan seberapa besar peningkatan kecepatan multiprosesor dalam menyelesaikan suatu problem. Perbandingan tersebut dapat dilakukan menggunakan solusi terbaik yang dapat dilakukan oleh prosesor tunggal, yakni algoritma sekuensial terbaik pada sistem prosesor tunggal dibandingkan dengan algoritma paralel pada multiprosesor yang sedang diteliti. Dengan persamaan sebagai berikut:[4]

$$S(p) = \frac{ts}{tp} \qquad (2\text{-}3)$$

dengan

S(p) = peningkatan kecepatan jika menggunakan multiprosesor

t_s = waktu proses menggunakan sistem prosesor tunggal

t_p = waktu proses menggunakan multiprosesor

Peningkatan kecepatan komputasi paralel secara teori dibatasi kendala sekuensial bahwa sebuah rutin tidak dapat dipecah menjadi unit yang lebih kecil dan harus dijalankan secara sekuensial di satu prosesor. Sementara secara praktek peningkatan kecepatan komputasi paralel dibatasi performa perangkat keras, penjadwalan proses sistem operasi, konfigurasi lingkungan komputasi dan *overhead* rutin program karena komunikasi dan sinkronisasi antar *proses*.

Program yang dibuat untuk komputasi paralel memiliki kerumitan tersendiri dibanding program sekuensial karena memunculkan kendala baru karena kondisi pacu dan dependensi data antar *proses*. Kondisi pacu dapat ditangani dengan mekanisme lokalisasi dan *locking* data, sementara dependensi data ditangani dengan mekanisme komunikasi *message* antar *proses*. Kedua mekanisme tersebut membutuhkan sinkronisasi antar *proses* yang memunculkan tambahan rutin komputasi yang memperlambat dan menambah kerumitan program.

2.2 *Cluster*

Cluster adalah kumpulan elemen yang dapat beroperasi secara mandiri, yang disatukan dengan media untuk menjalankan tindakan yang terkoordinasi dan kooperatif.

Clustering adalah konsep dan teknik yang digunakan untuk meningkatkan kemampuan dari sistem yang ada melalui agregasi dan sintesis dari elemen-elemen yang lebih sederhana sehingga menciptakan kompleksitas dan bentuk fungsional sistem yang baru.

Cluster (klaster komputer) adalah kumpulan komputer yang dapat beroperasi secara mandiri, yang disatukan dengan jaringan komunikasi data dan mendukung perangkat lunak yang memungkinkan pengaturan rutin beban komputasi secara bersamaan yang bertujuan untuk mengerjakan satu rutin komputasi yang lebih besar.

Di bidang rekayasa komputer, *clustering* diterapkan untuk membuat struktur sistem baru dari elemen-elemen komputasi yang telah ada untuk mendapatkan peningkatan kemampuan komputasi yang jika dilakukan dengan metode konvensional (mengganti perangkat keras dengan performansi yang lebih tinggi) memiliki harga ekonomis yang sangat mahal.

Clustering termasuk teknik arsitektur sistem komputer yang digunakan untuk meningkatkan performa, kegegasan pengguna, dan reliabilitas.

Cluster komoditas adalah kumpulan *node* komputasi dari komputer *general purpose* dalam jaringan lokal yang perangkat kerasnya dijual secara bebas dan perangkat lunaknya dapat diperoleh dengan bebas. Sehingga *cluster* komoditas memiliki fleksibilitas konfigurasi, *upgrade,* dan kemudahan penggunaan.

Cluster (klaster komputer) dibagi ke dalam beberapa kategori, sebagai berikut:

- Klaster untuk ketersediaan yang tinggi (*High-availability clusters*)

  High-availability cluster, yang juga sering disebut sebagai *Failover Cluster* pada umumnya diimplementasikan untuk tujuan meningkatkan ketersediaan layanan yang disediakan oleh klaster tersebut. Elemen klaster akan bekerja dengan memiliki *node-node* redundan, yang kemudian digunakan untuk menyediakan layanan saat salah satu elemen kluster mengalami kegagalan. Ukuran yang paling umum dari kategori ini adalah dua *node*, yang merupakan syarat minimum untuk melakukan redundansi. Implementasi klaster jenis ini akan mencoba untuk menggunakan redundansi komponen klaster untuk menghilangkan kegagalan di satu titik (*Single Point of Failure*).

  Ada beberapa implementasi komersial dari sistem klaster kategori ini, dalam beberapa sistem operasi. Meski demikian, proyek Linux-HA adalah salah satu paket yang paling umum digunakan untuk sistem operasi GNU/Linux.

Dalam keluarga sistem operasi Microsoft Windows NT, sebuah layanan yang disebut dengan *Microsoft Cluster Service* (MSCS) dapat digunakan untuk menyediakan klaster kategori ini. MSCS ini diperbarui lagi dan telah diintegrasikan dalam Windows 2000 Advanced Server dan Windows 2000 Datacenter Server, dengan nama ***Microsoft Clustering Service***. Dalam Windows Server 2003, Microsoft Clustering Service ini ditingkatkan lagi kinerjanya.

- Klaster untuk pemerataan beban komputasi (*Load-balancing clusters*)
  Klaster kategori ini beroperasi dengan mendistribusikan beban pekerjaan secara merata melalui beberapa *node* yang bekerja di belakang (*back-end node*). Umumnya klaster ini akan dikonfigurasikan sedemikian rupa dengan beberapa *front-end load-balancing* redundan. Karena setiap elemen dalam sebuah klaster *load-balancing* menawarkan layanan penuh, maka dapat dikatakan bahwa komponen klaster tersebut merupakan sebuah klaster aktif/klaster HA aktif, yang bisa menerima semua permintaan yang diajukan oleh client.

- Klaster hanya untuk komputasi (*Compute clusters*)
  Seringnya, penggunaan utama klaster komputer adalah untuk tujuan komputasi, ketimbang penanganan operasi yang berorientasi I/O seperti layanan Web atau basis data. Sebagai contoh, sebuah klaster mungkin mendukung simulasi komputasional untuk perubahan cuaca atau tabrakan kendaraan. Perbedaan utama untuk kategori ini dengan kategori lainnya adalah seberapa eratkah penggabungan antar *node*-nya. Sebagai contoh, sebuah tugas komputasi mungkin membutuhkan komunikasi yang sering antar *node*--ini berarti bahwa klaster tersebut menggunakan sebuah jaringan terdedikasi yang sama, yang terletak di lokasi yang sangat berdekatan, dan mungkin juga merupakan *node-node* yang bersifat homogen. Desain klaster seperti ini, umumnya disebut juga sebagai ***Beowulf Cluster***. Ada juga desain yang lain, yakni saat sebuah tugas komputasi hanya menggunakan satu atau beberapa *node* saja, dan membutuhkan komunikasi antar-*node* yang sangat sedikit atau tidak ada sama sekali. Desain klaster ini, sering disebut sebagai *"Grid"*. Beberapa *compute cluster* yang dihubungkan secara erat yang

didesain sedemikian rupa, umumnya disebut dengan "*Supercomputing*". Beberapa perangkat lunak *Middleware* seperti MPI atau *Parallel Virtual Machine* (PVM) mengizinkan program *compute clustering* agar dapat dijalankan di dalam klaster-klaster tersebut.

- *Grid computing*

  Grid pada umumnya adalah *compute cluster*, tapi difokuskan pada throughput seperti utilitas perhitungan ketimbang menjalankan pekerjaan-pekerjaan yang sangat erat yang biasanya dilakukan oleh *Supercomputer*. Seringnya, grid memasukkan sekumpulan komputer, yang bisa saja didistribusikan secara geografis, dan kadang diurus oleh organisasi yang tidak saling berkaitan.

  Grid computing dioptimalkan untuk beban pekerjaan yang mencakup banyak pekerjaan independen atau paket-paket pekerjaan, yang tidak harus berbagi data yang sama antar pekerjaan selama proses komputasi dilakukan. *Grid* bertindak untuk mengatur alokasi pekerjaan kepada komputer-komputer yang akan melakukan tugas tersebut secara independen. Sumber daya, seperti halnya media penyimpanan, mungkin bisa saja digunakan bersama-sama dengan komputer lainnya, tapi hasil sementara dari sebuah tugas tertentu tidak akan memengaruhi pekerjaan lainnya yang sedang berlangsung dalam komputer lainnya.

  Sebagai contoh grid yang sangat luas digunakan adalah proyek **Folding@home**, yang menganalisis data yang akan digunakan oleh para peneliti untuk menemukan obat untuk beberapa penyakit seperti *Alzheimer* dan juga kanker. Proyek lainnya, adalah **SETI@home**, yang merupakan proyek grid terdistribusi yang paling besar hingga saat ini. Proyek **SETI@home** ini menggunakan paling tidak 3 juta komputer rumahan yang berada di dalam komputer rumahan untuk menganalisis data dari teleskop radio observatorium Arecibo (*Arecibo Observatory radiotelescope*), mencari bukti-bukti keberadaan makhluk luar angkasa. Dalam dua kasus tersebut, tidak ada komunikasi antar node atau media penyimpanan yang digunakan bersama-sama.

2.3 *Cluster* Beowulf

Cluster Beowulf adalah *cluster* komoditas yang menggunakan komponen komputer *consumer-grade* dengan harga yang murah (contoh: komponen komputer bekas pakai), menggunakan perangkat lunak gratis dengan lisensi yang bersifat FOSS (*free open source software)* untuk membangun keseluruhan sistem, dan mengakomodasi keperluan komputasi paralel. *Cluster* Beowulf memiliki keuntungan karena stuktur dan metode implementasinya, antara lain: skalabilitas, konvergensi, fleksibilitas konfigurasi, *failover,* kemudahan, kecepatan, dan harga implementasi yang murah dibandingkan sistem *cluster* konvensional.

Komputasi dengan *cluster* Beowulf melibatkan 4 hal dalam pertimbangan sistem:[3]

1. sistem perangkat keras,
2. manajemen sumber daya komputasi,
3. pustaka pemrograman paralel,
4. algoritma paralel.

Sistem perangkat keras meliputi aspek komponen perangkat keras tiap *node* komputasi, antarmuka dan topologi jaringan. Manajemen sumber daya komputasi adalah serangkaian perangkat lunak yang digunakan untuk instalasi, konfigurasi dan inisialisasi, administasi, dan pemantauan aktivitas, dan perawatan *cluster.* Pustaka pemrograman paralel adalah *framework* yang digunakan untuk membuat program komputasi paralel. Algoritma paralel memberikan model dan pendekatan paralelisasi rutin komputasi.

2.4 Permainan Catur

Kata catur diambil dari bahasa Sanskerta yang berarti "empat". Namun kata ini sebenarnya merupakan singkatan dari *caturangga* yang berarti empat sudut. Di India kuno permainan catur memang dimainkan oleh empat peserta yang berada di empat sudut yang berbeda. Hal ini lain dari permain catur modern di mana pesertanya hanya dua orang saja. Kemudian kata *caturangga* ini diserap dalam bahasa Persia menjadi *shatranj*. Kata chess dalam bahasa Inggris diambil dari bahasa Persia *shah*.

Gambar 2.1 Permainan Catur

Catur adalah sebuah permainan yang dimainkan oleh dua orang. Sebelum bertanding, pecatur memilih biji catur yang akan ia mainkan. Terdapat dua warna yang membedakan bidak atau biji catur, yaitu hitam dan putih. Pemegang buah putih memulai langkah pertama, yang selanjutnya diikuti oleh pemegang buah hitam secara bergantian sampai permainan selesai.

2.4.1 Ketentuan Catur

Permainan dilangsungkan di atas papan yang terdiri dari 8 lajur dan 8 baris kotak/petak berwarna hitam dan putih (atau terang dan gelap) secara berselang seling. Permainan dimulai dengan 16 buah pada masing-masing pihak, yang disusun berbaris secara khusus pada masing-masing sisi papan catur secara berhadap-hadapan. Satu buah hanya bisa menempati satu petak. Pada bagian terdepan masing-masing barisan terdapat 8 pion, diikuti di belakangnya dua benteng (*rook*), dua kuda atau ksatria (*knight*), dua gajah (*bishop*), satu menteri atau ratu atau ster (*queen*), serta satu raja (*king*).

2.4.2 Gerakan

Sebelum bertanding, pecatur memilih warna buah yang akan ia mainkan. Pemegang buah putih memulai langkah pertama, yang selanjutnya diikuti oleh pemegang buah hitam secara bergantian. Setiap langkah hanya boleh menggerakkan satu bidak saja (kecuali untuk rokade di mana ada dua bidak yang digerakkan). Bidak dipindahkan ke petak kosong, ataupun yang ditempati oleh bidak lawan, yang berarti menangkapnya dan memindahkan bidak lawan dari permainan. Ada pengecualian, yaitu untuk gerakan *en passant*.

Setiap bidak catur memiliki gerakan yang unik sebagai berikut:

- Raja dapat bergerak satu petak ke segala arah. Raja juga memiliki gerakan khusus yang disebut rokade yang turut melibatkan sebuah benteng.

- Benteng dapat bergerak sepanjang petak horizontal maupun vertikal, tetapi tidak dapat melompati bidak lain. Seperti yang telah di atas, benteng terlibat dalam gerakan rokade.
- Gajah dapat bergerak sepanjang petak secara diagonal, tetapi tidak dapat melompati bidak lain.
- Ratu memiliki gerakan kombinasi dari Benteng dan Gajah.
- Kuda memiliki gerakan mirip huruf L, yaitu memanjang dua petak dan melebar satu petak. Kudalah satu-satunya bidak yang dapat melompati bidak-bidak lain.
- Pion dapat bergerak maju (arah lawan) satu petak ke petak yang tidak ditempati. Pada gerakan awal, pion dapat bergerak maju dua petak. Pion juga dapat menangkap bidak lawan secara diagonal, apabila bidak lawan tersebut berada satu petak di diagonal depannya. Pion memiliki dua gerakan khusus, yaitu gerakan menangkap *en passant* dan promosi.

2.4.3 Akhir Permainan

Tujuan permainan adalah mencapai posisi skak mat (checkmate). Hal ini bisa terjadi bila Raja terancam dan tidak bisa menyelamatkan diri ke petak lain. Tidak selalu permainan berakhir dengan kekalahan, karena bisa terjadi pula peristiwa seri atau remis di mana kedua belah pihak tidak mampu lagi meneruskan pertandingan karena tidak bisa mencapai skak mat. Peristiwa remis ini bisa terjadi berdasarkan kesepakatan maupun tidak. Salah satu contoh remis yang tidak berdasarkan kesepakatan - tetapi terjadi adalah pada keadaan remis abadi. Keadaan remis yang lain adalah keadaan pat, dimana yang giliran melangkah tidak bisa melangkahkan buah apapun termasuk Raja, tetapi tidak dalam keadaan terancam skak. Dalam pertandingan catur pihak yang menang biasanya mendapatkan nilai 1, yang kalah 0, sedang draw 0.5.

2.4.4 Istilah Dalam Catur

Rokade

Rokade (dalam bahasa Inggris, *castling*) merupakan gerakan khusus dalam catur di mana Raja bergerak dua petak menuju Benteng di baris pertamanya, kemudian meletakkan Benteng pada petak terakhir yang dilalui Raja. Persyaratan rokade adalah sebagai berikut:

- Bidak Raja dan Benteng yang akan dilibatkan dalam rokade harus belum pernah bergerak.
- Tidak ada bidak lain di antara Raja dan Benteng.
- Raja tidak sedang di-skak, dan petak-petak yang dilalui Raja tidak sedang diserang oleh bidak lawan.

En passant

Ketika pion bergerak dua petak maju dan ada pion lawan yang berada satu petak dalam baris tujuan, maka pion lawan dapat menangkap dan menempati petak yang baru saja dilalui pion tersebut (seolah-olah pion tersebut bergerak satu petak maju). Namun demikian, gerakan ini hanya dapat dilakukan sesaat setelah gerakan pion maju dua petak, atau hak lawan untuk melakukan gerakan en passant ini hilang.

Promosi

Ketika pion telah maju hingga menempati baris paling akhir, berbarengan dengan gerakan maju tersebut, pion dipromosikan dan harus ditukar dengan bidak berdasarkan keinginan pemain, yaitu Ratu, Benteng, Gajah, ataupun Kuda dengan warna yang sama. Pada umumnya, pion dipromosikan menjadi Ratu. Tidak ada peraturan yang membatasi bidak yang dipilih sebagai promosi, jadi dimungkinkan memiliki bidak yang melebihi jumlahnya waktu awal permainan (semisal, dua Ratu).

Skak

Ketika Raja sedang diserang oleh satu atau lebih bidak lawan, keadaan ini disebut dengan skak (check). Pemain yang Rajanya diskak harus menggerakkan Rajanya supaya tidak terserang. Hal ini dapat dilakukan dengan menangkap bidak lawan yang menyerang, menutup serangan lawan dengan menempatkan sebuah bidak di antaranya (apabila yang menyerang Ratu, Benteng, atau Gajah dan ada petak kosong di antara Raja dan bidak lawan), atau memindahkan Raja ke petak yang tidak sedang diserang. Rokade tidak diijinkan apabila Raja sedang diskak.

2.5 Teori Permainan Catur

Setiap permainan mempunyai teori dan peraturan. Sebuah game terdiri dari satu set pemain, satu set gerakan atau strategi untuk pemain tersebut, dan beberapa spesifikasi akibat dijalankannya gerakan tersebut.

Definisi dari teori permainan untuk catur adalah:

Definisi 1.

Game (Permainan) adalah deskripsi formal dari suatu situasi strategis.

Definisi 2.

Player (Pemain) adalah pelaku/subjek yang membuat keputusan dalam permainan.

Definisi 3.

Payoff adalah sebuah angka, atau alat bantu yang merefleksikan hasil tujuan untuk seorang pemain, untuk alasan apapun.

Definisi 4.

Zero-sum game adalah permainan dimana untuk seluruh hasil, total dari payoff seluruh pemain adalah nol. Dalam zero-sum game untuk dua orang, satu perolehan pemain adalah satu kehilangan untuk pemain lawan, jadi keinginan mereka berdua berlawanan.

Definisi 5.

Perfect information – Informasi sempurna, permainan ini memiliki kandungan informasi yang sempurna saat pada waktu kapanpun seorang pemain memperoleh giliran, dan mengetahui keadaan permainan saat itu dan juga seluruh langkah-langkah yang sudah dibuat sejak awal permainan.

Definisi 6.

Extensive game (permainan yang ekstensif) menjelaskan dengan pohon informasi bagaimana permainan tersebut dimainkan. Hal tersebut menjelaskan pemain mana yang melakukan gerakan, dan seluruh informasi yang dipunyai pemain pada setiap titik keputusan.

Definisi 7.

Strategi adalah seluruh kemungkinan gerakan yang ada pada seorang pemain. Dalam sebuah permainan ekstensif, strategi adalah pilihan-pilihan yang lengkap, satu untuk setiap titik keputusan pemain tersebut.

Definisi 8.

Rationality – Rasionalitas, yaitu sifat yang dimiliki oleh semua pemain yang membuat pemain tersebut bermain untuk memperoleh payoff sebesar mungkin.

Definisi 9.

Finite – Terbatas, permainan ini hanya membolehkan setiap pemain jumlah gerakan yang terbatas dan jumlah pilihan yang terbatas pada setiap gerakan.

Menggunakan istilah-istilah diatas, sebuah permainan catur bisa dilihat sebagai "**two-player zero-sum game of perfect information"**.

- Ada dua pemain (lebih sering disebut Putih dan Hitam).
- Ada tepat tiga kemungkinan akhir dari permainan: Putih menang-Hitam kalah, Hitam menang-Putih kalah, dan seri.
- Pada setiap giliran, antara putih atau hitam, seluruh informasi tentang permainan tersebut dari awal dimulainya permainan hingga keadaan saat ini diketahui, hal itu merepresentasikan **"perfect information".**

Permainan catur bersifat **terbatas,** sebagaimana gerakan sebuah bidak terbatas pada peraturan yang berlaku.

Permainan catur bisa digambarkan sebagai pohon (*tree*) yang mempunyai posisi awal (*root*), cabang merepresentasikan seluruh gerakan yang legal dan node merepresentasikan posisi bidak. Hal inilah yang merupakan **bentuk ekstensif**.

Diasumsikan bahwa kedua pemain bersifat **rasional,** yaitu selalu memilih strategi yang mendapatkan *payoff* sebesar mungkin.

2.6 Algoritma Shannon

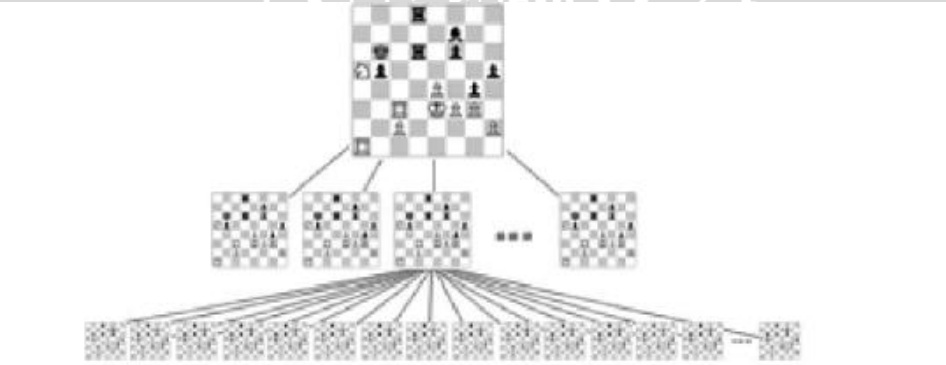
Karena dengan teknologi saat ini membuat sistem evaluasi untuk seluruh gerakan yang mungkin pada catur *tidaklah* mungkin, program catur harus memiliki suatu fungsi pencarian agar bisa bermain dengan baik.

Tingkat Kompleksivitas Algoritma untuk Catur menurut Shannon adalah 10^{120} atau disebut juga sebagai *Shannon Number*. Jumlah tersebut sangatlah besar (hingga melebihi jumlah partikel di alam semesta) dan tidak mungkin dihitung menggunakan teknologi saat ini.

Pencarian adalah metode untuk melihat kedepan pada setiap gerakan dan mengevaluasi posisi setelah melakukan gerakan tersebut.

Claude Shannon mengkategorikan pencarian menjadi dua tipe:

- **Type A** – sebuah *brute-force search* yang melihat seluruh kemungkinan dengan kedalaman yang sudah ditentukan. Strategi Type-A dikemukakan oleh Claude Shannon pada publikasinya yang sangat terkenal: *Programming a Computer for Playing Chess,* sebagai strategi *brute-force*, yang disebutkan Shannon bahwa strategi ini terlalu lambat dan lemah, dikarenakan oleh rata-rata factor percabangan dan ledakan eksponensial yang sangat besar. Karena itulah pada saat itu dia memilih strategi Type-B. Tetapi dengan penemuan algoritma alpha-beta dan seluruh ekstensinya, *brute-force* menjadi sangat sukses dari tahun 70an hingga sekarang, karena prosedur untuk mengklasifikasikan dan membuang gerakan yang "jelek" pada Type-B sangatlah sulit dan rawan dengan kesalahan.



Gambar 2.2 Shannon Type-A

Algoritma Type-A cukuplah sederhana, namun membutuhkan komputasi yang teramat besar (tergantung jumlah kedalaman pencarian). Seperti pada gambar 2.2, Algoritma Shannon Type-A untuk empat tingkat kedalaman dapat dijelaskan seperti berikut:

```
Max   Min    Max    Min    f(M_ijkl M_ijk M_ij M_i P)
M_i   M_ij   M_ijk  M_ijkl                          . . . . (1)
```

1) Max adalah pemain yang mendapat giliran saat ini.
2) Max melakukan gerakan **pertama** dari **seluruh** kemungkinan gerakan yang ada. (M_i).
3) Max berpikir: jika saya melakukan M_i bagaimanakah kira-kira langkah yang akan diambil **lawan** yang **paling** menguntungkan untuknya (M_ij).
4) Pencarian berlangsung hingga empat tingkat kedalaman diselesaikan, yaitu: M_ijkl.

- **Type B** - sebuah pencarian selektif yang hanya melihat pada cabang yang "penting". Type-B, seperti telah dijelaskan diatas adalah pendekatan selektif untuk mencari *minimax trees* yang hanya terdiri dari beberapa gerakan yang dianggap bagus yang berlawanan dengan strategi *brute-force* Type-A. Cara ini bekerja dengan berkonsentrasi pada pemilahan gerakan berdasarkan pengetahuan luar tentang permainan catur itu sendiri, contohnya saat pergerakan awal, orang lebih banyak memilih menggerakan pion di tengah daripada pion di tepi. Cara ini sangat tergantung pada ketrampilan bermain sang *programmer* dan kualitas dari pemilahan tersebut, sehingga hasil untuk tiap orang bisa sangat berbeda drastis.

Terinspirasi oleh eksprimen dari Adriann de Groot, Shannon dan para programmer pada masa itu lebih memilih strategi Type-B. Tipe ini menggunakan semacam heuristik statis dengan tujuan hanya untuk melihat cabang yang penting, dengan kekurangan banyak strategi-strategi yang terlewat karena jumlahnya yang sangat banyak sekali.

Type-B merupakan algoritma yang popular hingga tahun 1970, dimana Type-A memiliki processing power yang lebih besar dan algoritma brute-forcenya menjadi lebih kuat.

Kebanyakan program sekarang lebih dekat pada Type-A, tetapi mempunyai beberapa karakteristik Type-B (hal inilah yang membedakan banyak program-program catur yang ada sekarang).

2.7 Open MPI

Komputasi paralel di sebuah *cluster* Beowulf dilakukan dengan membagi proses komputasi menjadi bangian-bagian dan menggunakan beberapa *proses* yang masing-

masing dijalankan di prosesor berbeda. Umumnya sebuah program yang sama dapat digunakan di semua *proses*, namun dengan masukan dan parameter yang berbeda.

Pada permasalahan yang sederhana, program dapat digunakan beberapa *proses* berbeda hanya dengan masukan dan parameter yang berbeda tanpa komunikasi antar proses. Namun jika permasalahan menjadi kompleks, komunikasi antar *proses* diperlukan dan perlu didesain dengan mangkus dan sangkil pada tahap implementasi.

Salah satu pendekatan komunikasi antar proses adalah mengoordinasikan aktivitas *proses-proses* dengan mekanisme pengiriman dan penerimaan *message* berupa deretan *byte* dengan struktur data tertentu antar *proses.* Spefisikasi mekanisme komputasi antar proses dengan *message* ini disebut MPI.

MPI (*message-passing interface)* adalah spesifikasi putsaka *message-passing.* Terdapat 3 bagian utama dari deskripsi spesifikasi MPI:

- MPI menjabarkan model *message-passing* komputasi paralel yang mana tiap proses memiliki lingkup alamatnya masing-masing dan melakukan sinkronisasi antar proses dengan menerima dan mengirimkan *message*,
- MPI menjabarkan spesifikasi pustaka: kumpulan sub-rutin dan argumennya. MPI bukan merupakan pustaka dalam bahasa pemrograman tertentu, namun rutin MPI diimpelemtasikan melalui bahasa pemrograman konvensional antara lain: C, C++, dan Fortran,
- MPI adalah spesifikasi, bukan merupakan implementasi pemrograman. Spesifikasi MPI dibuat oleh MPI Forum. Terdapat tiga spesifikasi umum MPI: MPI-1 (1994-1998), MPI-2 (1998-2009), dan MPI-3 (2012).

Dalam implementasinya, Open MPI menggunakan *suite* SSH (Secure Shell) untuk menjalankan program secara *remote* di komputer lain.

2.8 Paralelisasi Algoritma Minimax

Algoritma pencarian *game-tree* Minimax ini termasuk ke dalam *Brute-force tree search* yang *embarrassingly* mudah untuk di implementasi secara paralel. Setelah membagi petak *board* pada tingkat kedalaman pertama untuk setiap prosesor, lalu setiap prosesor akan menjalankan tugasnya masing-masing. Setiap prosesor akan melakukan proses iterasi untuk mencari dan menghitung nilai gerakan tiap bidak pada giliran pemain saat ini.