KINERJA ALGORITMA PARALEL UNTUK PENCARIAN KATA DENGAN METODE BOYER-MOORE MENGGUNAKAN PVM

Maria Angela Kartawidjaja[1] Stania Vandika[2]
[1,2]Jurusan Teknik Elektro, Fakultas Teknik, Universitas Katolik Indonesia Atma Jaya
[1]*Email: mariaAKW@gmail.com*

ABSTRACT
Search process is one of important activity in data processing. Searching can take more time if conducted in the huge search space. Therefore, it is needed an efficient search technique. One technique that can be used is parallel computing. This article discusses the search process in parallel using the Boyer-Moore algorithm. Parallel scope is emulated with PVM (Parallel Virtual Machine) software. From the research results, it can be concluded that the performance of parallel computing will increase the word searching compared to the computing performance for its word sequencial search consisting of one letter, and will drop to the word search consisting of two letters or more.

Keywords: *Boyer-Moore, performance, parallel computing, word searching, PVM*

ABSTRAK
Proses pencarian merupakan salah satu kegiatan penting dalam pemrosesan data. Proses pencarian dapat menghabiskan waktu jika ruang pencariannya besar, dan karena itu diperlukan suatu teknik pencarian yang efisien. Salah satu teknik yang dapat digunakan adalah komputasi paralel. Artikel ini membahas proses pencarian kata secara paralel dengan menggunakan algoritma Boyer-Moore. Lingkup paralel diemulasikan menggunakan perangkat lunak PVM Parallel Virtual Machine. Dari hasil penelitian dapat disimpulkan bahwa kinerja komputasi paralel pencarian kata akan meningkat jika dibandingkan dengan kinerja komputasi sekuensial untuk pencarian kata yang terdiri dari satu huruf, dan akan menurun untuk pencarian kata yang terdiri dari dua huruf atau lebih.

Kata Kunci: *Boyer-Moore, kinerja, komputasi paralel, pencarian kata, PVM*

Dengan kemajuan teknologi yang sangat pesat dewasa ini penggunaan komputer untuk membantu manusia dalam menyelesaikan masalah sudah menjadi hal yang biasa. Seiring dengan hal tersebut tuntutan terhadap proses komputasi pun semakin meningkat. Salah satu proses penting dalam komputasi adalah pencarian data. Jika data yang tersedia berjumlah banyak, maka waktu pencarian akan menjadi besar dan hal ini akan mengakibatkan penurunan kinerja komputasi. Oleh karena itu, dibutuhkan suatu teknik yang efisien untuk melakukan pencarian data, misalnya de-ngan menggunakan komputasi paralel.

Artikel ini membahas proses pencarian kata secara paralel dengan menggunakan metode Boyer-Moore. Lingkup paralel diemulasikan dengan menggunakan suatu perangkat lunak yang dinamakan PVM (*Parallel Virtual Machine*) [1].

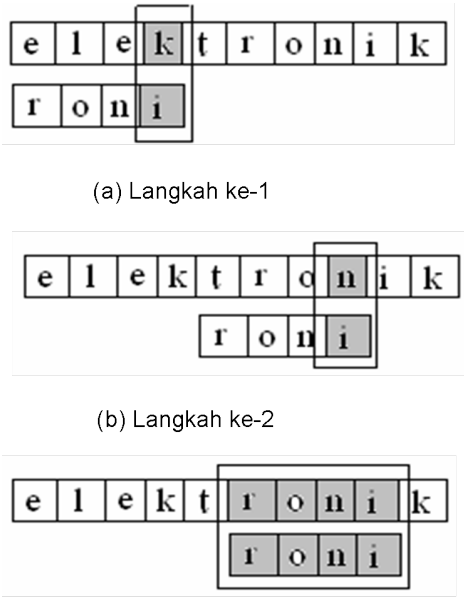
PENCARIAN KATA DENGAN METODE BOYER-MOORE

Pencarian kata bertujuan untuk mencari sebuah kata yang terdiri dari beberapa karakter dalam sekumpulan kata. Kata yang akan dicari disebut sebagai kata kunci atau *keywords*. Prinsip dasar pencarian kata adalah pencocokan karakter-karakter kata kunci dengan karakter-karakter kata yang tersedia. Misalnya suatu kata kunci yang panjangnya m karakter akan dicari di dalam sekumpulan kata yang terdiri dari n karakter dengan ketentuan $n \geq m$[2].

Ada sejumlah algoritma yang dapat digunakan untuk pencarian kata, misalnya algoritma Boyer-Moore, algoritma *Brute Force* dan algoritma Knuth-Morris-Pratt [3, 4]. Dalam artikel ini pencarian kata dilakukan dengan algoritma Boyer-Moore.

Algoritma Boyer-Moore diperkenalkan oleh Bob Boyer dan J.S. Moore pada tahun 1977 [2]. Pada algoritma ini pencocokan kata dimulai dari karakter terakhir kata kunci menuju karakter awalnya. Jika terjadi perbedaan antara karakter terakhir kata kunci dengan kata yang dicocokkan, maka karakter-karakter dalam potongan kata yang dicocokkan tadi akan diperiksa satu per satu. Hal ini dimaksudkan untuk mendeteksi apakah ada karakter dalam potongan kata tersebut yang sama dengan karakter yang ada pada kata kunci. Apabila terdapat kesamaan, maka kata kunci akan digeser sedemikian rupa sehingga posisi karakter yang sama terletak sejajar, dan kemudian dilakukan kembali pencocokan karakter terakhir dari kata kunci. Sebaliknya jika tidak terdapat kesamaan karakter, maka seluruh karakter kata kunci akan bergeser ke kanan sebanyak m karakter, di mana m adalah panjang karakter dari kata kunci. Ilustrasi pencarian kata kunci "roni" pada kata "elektronik" dapat dilihat pada Gambar 1.

Dari Gambar 1(a), dapat dilihat bahwa karakter terakhir dari kata kunci adalah huruf "i" yang dicocokkan dengan huruf "k" pada kata "elektronik". Karena huruf "i" dan huruf "k" berbeda, maka akan dilakukan pencocokan huruf "k" dengan seluruh karakter pada kata kunci. Karena huruf "k" tidak terdapat pada seluruh karakter kata kunci, maka kata kunci bergeser ke kanan sebanyak empat karakter sesuai dengan panjang karakter kata kunci seperti yang tampak pada Gambar 1(b). Setelah dilakukan pergeseran, maka dicocokkan kembali karakter terakhir pada kata kunci yaitu huruf "i" dengan huruf "n". Karena kedua huruf ini berbeda, maka huruf "n" dicocokkan dengan keseluruhan karakter pada kata kunci. Karena pada kata kunci terdapat



Gambar 1: Langkah-langkah pencarian kata dengan algoritma Boyer-Moore

huruf "n", maka kata kunci akan bergeser sedemikian rupa sehingga huruf "n" pada kata kunci memiliki posisi yang sejajar dengan posisi huruf "n" pada kata yang dicocokkan seperti yang ditunjukkan pada Gambar 1(c). Setelah itu dilakukan kembali pencocokan karakter terakhir pada kata kunci, yaitu huruf "i" dengan karakter yang terletak sejajar dengan huruf "i" tersebut. Karena karakter tersebut sama maka dicocokkan kembali karakter berikut yang berada di sebelah kiri huruf "i" sehingga keseluruhan karakter pada kata kunci selesai diperiksa.

Algoritma ini memiliki kompleksitas $\bigcirc(n/m)$ dalam kasus terbaiknya dan $\bigcirc(n)$ dalam kasus terburuknya [5]. Kasus terbaik terjadi jika karakter terakhir pada kata kunci berbeda dengan karakter kata yang dicocokkan tetapi seluruh kata kunci terdapat pada kata yang dicocokkan. Sedangkan kasus terburuk terjadi jika karakter kata kunci yang dicari hanya satu karakter saja dan kata yang dicocokkan terdiri dari beberapa karakter.

KOMPUTASI PARALEL

Komputasi paralel dalam pencarian kata bertujuan untuk mempersingkat waktu pencarian suatu kata kunci di dalam sekumpulan kata yang jika dilakukan pada satu komputer akan membutuhkan waktu yang panjang.

Konsep komputasi paralel sebenarnya sudah ditemukan berpuluh-puluh tahun yang silam seperti yang dikemukakan oleh Wilkinson dan Allen [6]. Artikel yang berkaitan dengan komputasi paralel sudah banyak ditulis oleh para ilmuwan, misalnya Gill [7], Holland [8], Conway [9], Howe dan Moxon [10], Karp [11], Flynn dan Rudd [12].

Pada konsep komputasi paralel ada beberapa ukuran yang umum digunakan dalam mengevaluasi kinerja sistem. Beberapa di antaranya adalah waktu eksekusi suatu program, peningkatan kecepatan (*speedup*), effisiensi prosesor, efisiensi memori dan biaya [5]. Pada artikel ini digunakan waktu eksekusi dan peningkatan kecepatan untuk mengukur kinerja komputasi paralel.

Pada suatu program paralel waktu eksekusi suatu program merupakan gabungan antara waktu komputasi dan waktu komunikasi, sehingga waktu eksekusi pada p proses dapat dinyatakan dengan [6]

$$t_p = t_{komputasi} + t_{komunikasi} \quad (1)$$

dan peningkatan kecepatan proses atau speedup dapat dinyatakan dengan [6]

$$S_p = \frac{t_1}{t_p} \quad (2)$$

dengan: t_1 = waktu eksekusi pada satu proses t_p = waktu eksekusi pada proses p = jumlah proses.

Peningkatan kecepatan proses adalah suatu besaran tak berdimensi yang nilainya berkisar antara 1 dan p, atau secara matematis dinyatakan dengan

$$1 \leq S_p \leq p \quad (3)$$

Peningkatan kecepatan proses dalam Persamaan (2) disebut sebagai peningkatan kinerja relatif [13], yang mengukur seberapa besar peningkatan kinerja komputasi yang dapat diperoleh dengan menggunakan sejumlah prosesor yang bekerja bersama-sama dalam memecahkan suatu masalah.

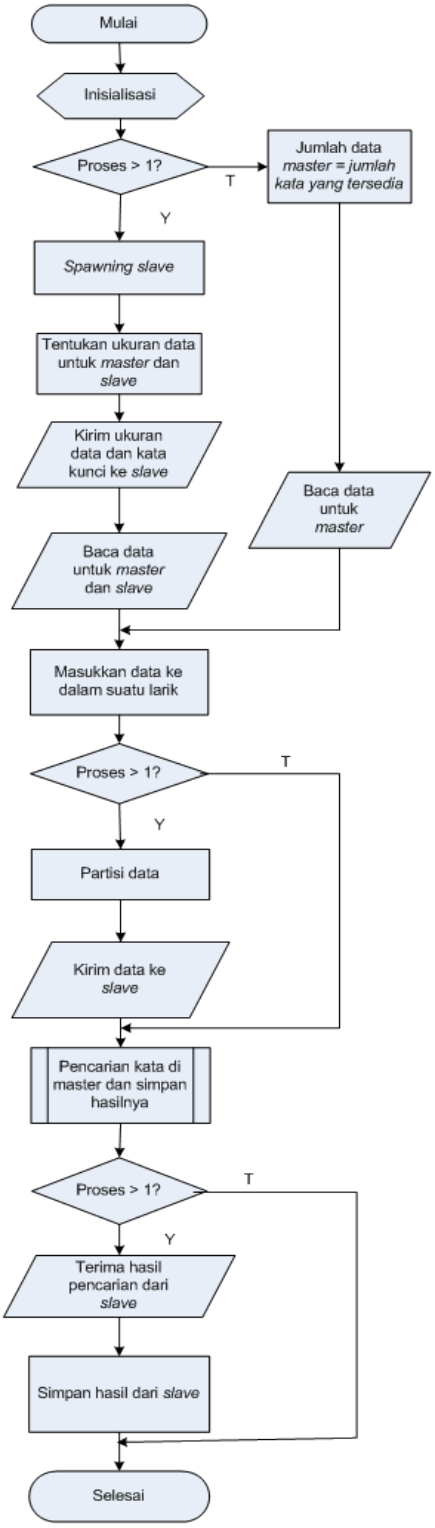
PVM (PARALLEL VIRTUAL MACHINE)

PVM adalah suatu perangkat lunak yang dapat digunakan untuk mengemulasikan suatu lingkup paralel. Perangkat lunak ini dapat digunakan dalam sistem yang homogen maupun heterogen.

Sistem PVM terdiri dari dua bagian, yaitu daemon dan kumpulan pustaka. Daemon terletak pada semua komputer dan membentuk suatu lingkup kerja maya (*virtual*). Kumpulan pustaka dibutuhkan untuk melakukan kerja sama antara proses. Pustaka ini disediakan dalam bahasa pemrograman C dan Fortran.

PVM memungkinkan pembentukan sekumpulan proses yang tidak tergantung pada jumlah prosesor yang digunakan. Masing-masing proses dalam PVM memiliki suatu identifikasi yang unik. Setiap proses akan dipetakan pada prosesor secara otomatis kecuali jika diprogram secara eksplisit oleh pengguna.

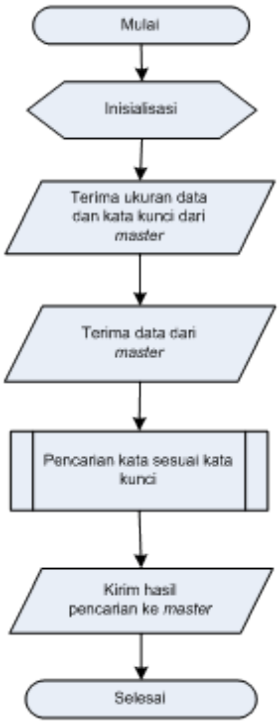
Pemrograman dengan PVM umumnya menggunakan model *master-slave*, di mana pada awalnya hanya ada satu proses yang berjalan yaitu proses *master*, dan kemudian proses *master* ini akan "menciptakan" (*spawn*) proses *slave*. Selain melakukan komputasi, proses *master* bertanggung jawab atas pembentukan proses-proses *slave*, inisialisasi, pengiriman dan pengumpulan data. Sedangkan proses *slave* hanya melakukan komputasi, yang beban kerjanya diatur oleh proses *master*. Hasil komputasi dari proses *slave* kemudian dikirim kembali ke *master*. Komunikasi antara proses *master* dan proses *slave* ini dilakukan dengan cara pengiriman pesan (*message-passing*) [1].



Gambar 2: Diagram alir program pada proses *master*

PERANCANGAN SISTEM

Proses *master* bertugas untuk melakukan seluruh kerja administrasi dan kerja komputasi. Sedangkan proses *slave* hanya bertugas melakukan komputasi. Komunikasi antar proses hanya berlangsung antara proses *master* dan proses *slave*, bukan antara sesama proses *slave*. Diagram alir kerja pada proses *master* ditunjukkan dalam Gambar 2, dan untuk proses *slave* dalam Gambar 3.



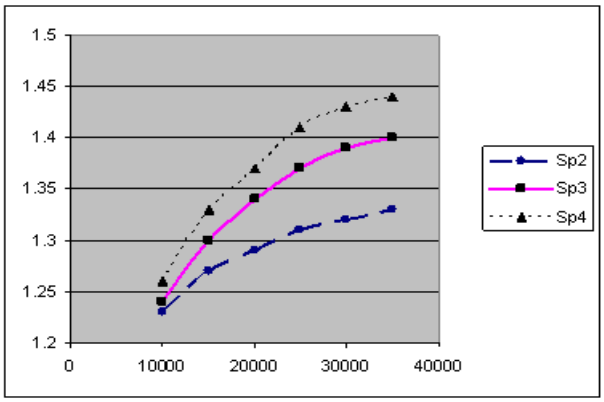
Gambar 3: Diagram alir program paralel pada proses *slave*

Langkah awal yang dilakukan proses *master* adalah inisialisasi semua data yang dibutuhkan seperti jumlah data, banyaknya proses yang digunakan, kata kunci yang dicari serta larik (*array*) untuk menampung kumpulan kata.

Jika proses yang digunakan hanya satu, maka semua kumpulan kata akan digunakan oleh proses *master* untuk melakukan proses pencarian. Kumpulan kata akan dimasukkan ke dalam sebuah larik untuk digunakan dalam proses pencarian kata sesuai kata kunci. Setelah proses pencarian berakhir maka proses *master* akan menyimpan hasil yang diperoleh dalam suatu larik dan komputasi pun berakhir.

Jika terdapat lebih dari satu proses, maka proses *master* melakukan *spawning slave* untuk menciptakan proses *slave* yang dibutuhkan. Kemudian proses *master* menghitung ukuran data yang akan dikirim ke masing-masing proses *slave*. Selanjutnya, proses *master* akan mengirim ukuran data tersebut beserta kata kunci ke proses *slave*. Langkah berikutnya, proses *master* melakukan pembacaan kata dari kumpulan data dan menyimpannya dalam suatu larik. Data tersebut kemudian dipartisi sesuai dengan jumlah proses yang ada. Lalu proses *master* mengirimkan partisi data ke masing-masing proses *slave* sedangkan partisi data miliknya tetap disimpan di dalam larik. Sesudah itu proses master melakukan pencarian kata sesuai dengan kata kunci, serta menyimpan hasil pencarian untuk digabungkan dengan hasil pencarian yang diterimanya dari proses *slave*.

Proses *slave* pada awalnya akan melakukan inisialisasi



Gambar 4: Kinerja komputasi untuk pencarian kata kunci "a"

larik yang digunakan untuk menampung kata kunci dan kumpulan kata yang dikirim oleh proses *master* seperti tampak pada Gambar 3. Sesudah itu proses *slave* melakukan pencarian kata dengan menggunakan kata kunci yang diterima dari proses *master*. Hasil pencarian kata kemudian dikirim ke proses *master*.

HASIL PENGUJIAN DAN ANALISIS

Pengujian sistem untuk pencarian kata dilakukan dengan menggunakan empat buah komputer Intel Pentium 3.0 GHz yang memiliki *Random Access Memory* (RAM) sebesar 512 MB. Komputer-komputer tersebut terhubung dalam jaringan melalui suatu *switch*. Setiap proses akan dipetakan pada setiap prosesor secara otomatis oleh PVM.

Pengujian program sekuensial dilakukan dengan mengeksekusi program pada satu komputer. Pengujian program paralel dilakukan dengan mengeksekusi program pada dua, tiga atau empat komputer yang bekerja bersama-sama. Pada pemrograman paralel salah satu komputer bertindak sebagai proses *master* dan komputer lainnya sebagai proses *slave*. Beban komputasi pada setiap proses diusahakan sama besar agar semua proses menyelesaikan komputasinya dalam waktu yang kurang lebih bersamaan. Dengan demikan, waktu eksekusi program keseluruhan diharapkan akan menurun, yang berarti peningkatan kinerja komputasi.

Waktu eksekusi program pencarian kata menggunakan metode Boyer-Moore pada satu, dua, tiga dan empat prosesor ditunjukkan dalam Tabel 1 untuk pencarian kata menggunakan kata kunci yang terdiri dari satu huruf seperti huruf "a". Pada Tabel 1 tampak adanya penurunan waktu eksekusi seiring dengan bertambahnya jumlah prosesor. Peningkatan kinerja komputasi dihitung dengan menggunakan Persamaan (2), hasilnya ditampilkan pada Tabel 2, dan diilustrasikan pada Gambar 4. Tampak kinerja komputasi yang dinyatakan dengan S_p meningkat seiring dengan bertambahnya jumlah kata dan jumlah prosesor yang digunakan.

Untuk pencarian kata kunci yang terdiri dari dua huruf misalnya "an", waktu eksekusinya ditunjukkan pada Tabel 3 dan kinerja komputasinya pada Tabel 4. Dari Tabel 4 tampak bahwa peningkatan kinerja komputasi dengan

Tabel 1: Waktu eksekusi pencarian kata kunci "a"

Jumlah Data	Waktu eksekusi (ms) 1 p	2 p	3 p	4 p
10000	1645	1334	1327	1301
15000	3728	2925	2861	2798
20000	6644	5133	4942	4864
25000	10438	7941	7596	7428
30000	14978	11342	10784	10469
35000	20454	15332	14578	14184

Tabel 2: Kinerja komputasi untuk pencarian kata kunci "a"

Jumlah Data	*Speedup (Sp)* 2 p	3 p	4 p
10000	1.23	1.24	1.26
15000	1.27	1.30	1.33
20000	1.29	1.34	1.37
25000	1.31	1.37	1.41
30000	1.32	1.39	1.43
35000	1.33	1.4	1.44

Tabel 3: Waktu eksekusi pencarian kata kunci "an"

Jumlah Data	Waktu eksekusi (ms) 1 p	2 p	3 p	4 p
10000	1642	1587	1580	1558
15000	3719	3543	3538	3526
20000	6801	6508	6449	6389
25000	10288	9805	9735	9669
30000	15223	14423	14352	14296
35000	21891	20418	20387	20364

Tabel 4: Kinerja komputasi pencarian kata kunci "an"

Jumlah Data	*Speedup (Sp)* 2 p	3 p	4 p
10000	1.04	1.04	1.05
15000	1.05	1.05	1.05
20000	1.05	1.05	1.06
25000	1.05	1.06	1.06
30000	1.06	1.06	1.06
35000	1.07	1.07	1.07

menggunakan dua, tiga atau empat prosesor tidaklah berarti. Hal yang sama juga terjadi pada pencarian kata kunci yang terdiri dari tiga huruf atau lebih, seperti pencarian kata kunci "ana" dengan waktu eksekusi yang ditunjukkan pada Tabel 5 dan kinerja komputasi pada Tabel 6, serta kata kunci "ning" dengan waktu eksekusi dan kinerja komputasi yang masing-masing ditunjukkan pada Tabel 7 dan Tabel 8. Untuk pencarian kata kunci "ana" dan "ning" tampak adanya suatu penurunan kinerja jika jumlah kata yang tersedia kurang dari 25.000 kata seperti tampak pada Tabel 6 dan Tabel 8.

SIMPULAN

Dari hasil pengujian dapat ditarik simpulan bahwa pencarian kata dengan metode Boyer Moore dengan menggunakan komputasi paralel akan memberikan peningkatan kinerja komputasi untuk kata kunci yang terdiri dari satu huruf saja.

Tabel 5: Waktu eksekusi pencarian kata kunci "ana"

Jumlah Data	Waktu eksekusi (ms)			
	1 p	2 p	3 p	4 p
10000	1539	1649	1762	1834
15000	4021	4183	4259	4372
20000	7464	7517	7583	7659
25000	11573	11518	11732	11689
30000	14936	14572	14663	14719
35000	20962	20275	20387	20578

Tabel 6: Kinerja komputasi pencarian kata kunci "ana"

Jumlah Data	*Speedup (Sp)*		
	2 p	3 p	4 p
10000	0.93	0.87	0.84
15000	0.96	0.94	0.92
20000	0.99	0.98	0.97
25000	1.00	0.99	0.99
30000	1.02	1.02	1.01
35000	1.03	1.03	1.02

Tabel 7: Waktu eksekusi pencarian kata kunci "ning"

Jumlah Data	Waktu eksekusi (ms)			
	1 p	2 p	3 p	4 p
10000	1702	1822	1981	2054
15000	4524	4746	4812	4923
20000	8032	8082	8195	8252
25000	10395	10498	10529	10503
30000	16136	16085	16073	16314
35000	21185	21096	20953	21263

Tabel 8: Kinerja komputasi pencarian kata kunci "ning"

Jumlah Data	*Speedup (Sp)*		
	2 p	3 p	4 p
10000	0.93	0.86	0.83
15000	0.95	0.94	0.92
20000	0.99	0.98	0.97
25000	0.99	0.99	0.99
30000	1.00	1.00	0.99
35000	1.00	1.01	1.00

DAFTAR PUSTAKA

[1] Geist, A.: *PVM: Parallel Virtual Machine A Users Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, England (1994)

[2] Texas, U.: *The Boyer-Moore Fast String Seaching Algorithm*. `http://www.cs.utexas.edu/users/moore/best-ideas/string-searching/index.html` diakses 13 Agustus 2008.

[3] Charras, C., Lecroq, T.: *Exact String Matching Algorithms*. `http://www-igm.univ-mlv.fr/~lecroq/string/` (1997) diakses 13 Agustus 2008.

[4] Graham, A.S.: *String Searching Algorithms*. World Scientific Publishing Co. (1995)

[5] Hartoyo, E.: *Analisis Algoritma Pencarian String (String Matching)*. `http://www.informatika.org/~rinaldi/Stmik/Makalah/MakalahStmik10.pdf` (2006) diakses 13 Agustus 2008.

[6] Wilkinson, B., Allen, M.: *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice-Hall Inc (1999)

[7] Gill, S.: *Parallel Programming*. The Computer Journal 1 (1958)

[8] Holland, J.: *A Universal Computer Capable of Executing an Arbitrary Number of Subprograms Simultaneously*. In: Proc. East Joint Computer Conference 16

[9] Conway, M.: *A Multiprocessor System Design*. In: Proc. AFIPS Fall Joint Computer Conference 4. (1963)

[10] Howe, C., Moxon, B.: *How to Program Parallel Processors*. IEEE Spectrum 24 (1987)

[11] Karp, A.: *Programming for Parallelism*. Computer 20 (1987)

[12] Flynn, M., Rudd, K.: *Parallel Architectures*. ACM Computing Surveys 28 **1** (1996)

[13] Foster, I.: *Designing and Buildng Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Publishing Co (1995)