Metode Divide and Conquer Parallel dan Parallel-Reduce Pada Cilk for Untuk Aplikasi E-Voting Berbasis Sistem Prosesor Multicore

Adnan
Jurusan Teknik Elektro
Universitas Hasanuddin
Makassar, Indonesia
Email: adnan@unhas.ac.id

***Abstract*—Paper ini mengusulkan penerapan algoritma *divide-and-conquer* parallel dan teknik *work-stealing* pada aplikasi electronic voting. Dalam paper ini kami menggunakan metode yang telah disebutkan, sehingga setiap thread mengambil email dari antrian secara interleave. Masing-masing thread mengakumulasi jumlah suara parsial yang diekstrak dari email tadi. Kami juga mengusulkan penggunaan metode *parallel-reduce* untuk mengatasi masalah kondisi-balapan ketika menghitung hasil perolehan akhir penghitungan suara.**

Keywords—multicore; e-voting ; work-stealing ; parallel-reduce ; kondisi-balapan;

I. PENDAHULUAN

Beberapa tahun terakhir ini, arsitektur komputer-komputer cenderung berevolusi menjadi komputer parallel. Sistem multiprosesor moderen yang dikenal sebagai *multicore* sedang menjadi trend. Prosesor-prosesor *multicore* sedang mendominasi pasar. Prosesor-prosesor *multicore* dengan mudah dijumpai pada komputer-komputer laptop, dekstop, dan juga server-server. Selain itu, prosesor-prosesor *multicore* juga mengisi segment pasar cluster-cluster superkomputer.

Diadopsinya prosesor-prosesor *multicore* pada kebanyakan sistem-sistem komputer berdampak pada bagaimana perangkat-perangkat lunak harus dikembangkan. Bahwasanya, perangkat-perangkat lunak yang nantinya berjalan pada komputer yang mutakhir dianggap harus mampu memanfaatkan sistem multiprosesor. Artinya, perangkat lunak harus berjalan secara *multithreading*. Meskipun sistem operasi moderen mendukung *multithreading*, perangkat lunak harus dikembangkan menggunakan teknik pemrograman *multithreading*.

Dengan pemrograman *multithreading* pada sistem komputer berbasis prosesor *multicore* diharapkan program yang dikembangkan dapat meningkatkan kinerja komputasinya. Pemrograman parallel ini sangat signifikan manfaatnya untuk mempersingkat waktu eksekusi algoritma yang memiliki kompleksitas tinggi. Waktu eksekusi dapat dikurangi secara signifikan oleh komputasi berkinerja tinggi pada sistem komputer berbasis *multicore*. Komputasi yang berkinerja tinggi dapat diperoleh jika inti-inti (cores) pada prosesor tersebut dimanfaatkan secara efisien untuk komputasi algoritma yang benar-benar diperlukan. Ini artinya, pemrograman harus dapat menekan biaya komputasi tambahan (overhead). Jadi teknik pemrograman parallel yang efisien harus diterapkan.

Teknik *multithreading* harus dapat pula mengatasi masalah ketidak-seimbangan beban. Beban komputasi harus dibagi secara merata kepada seluruh thread-thread yang berjalan pada inti-inti prosesor. Jika beban komputasi tidak merata diantara inti-inti prosesor, maka akan terdapat beberapa inti yang berbeban lebih dan ada beberapa inti yang berbeban kurang. Inti-inti yang berbeban kurang memerlukan waktu yang lebih pendek daripada inti-inti yang berbeban lebih. Pada akhirnya waktu eksekusi sebuah algoritma ditentukan oleh inti yang mengeksekusi tugas-tugas lebih lama. Sebaliknya jika beban komputasi terbagi secara merata maka waktu eksekusi oleh semua inti-inti prosesor akan sama. Pada saat demikian, waktu eksekusi yang tercepat diperoleh.

Salah satu aplikasi yang potensial untuk memanfaatkan prosesor-prosesor multicore adalah aplikasi e-voting untuk keperluan pemilihan kepala daerah (PILKADA). Pada aplikasi e-voting seperti yang dikembangkan oleh BPPT, setiap komputer TPS tempat pemungutan suara (TPS) mengirimkan hasil perhitungan suara ke server KPU menggunakan protokol surat elektronik. Untuk keperluan otentifikasi, setiap email harus diberi *digital signature* oleh komputer TPS. Pada sisi komputer KPU, setiap email yang diterima diverifikasi keaslian pengirim dengan cara memverifikasi *digital signature* tadi. Hanya email dari komputer TPS yang asli yang akan diekstrak datanya. Kemudian komputer KPU mengakumulasi jumlah peroleh suara masing-masing calon kepala daerah. Email-email yang lain harus diabaikan.

Pada kasus aplikasi e-voting, komputer KPU harus melakukan komputasi algoritma *digital-signature* secara berulang-ulang sebanyak jumlah email yang diterimanya. Algoritma-algoritma *digital-signature* menggunakan *public-key infrastructure* seperti RSA[1] dan DSS[2] adalah algoritma yang memiliki kompleksitas-waktu yang tinggi. Jika komputer KPU memanfaatkan hanya sebuah prosesor untuk memverifikasi keaslian ribuan email dan kemudian mengakumulasi jumlah perolehan suara dari setiap komputer TPS, komputer KPU akan membutuhkan waktu yang lama untuk keperluan tersebut. Disini, prosesor-prosesor *multicore* dengan menggunakan teknik pemrograman *multithreading* dapat dimanfaatkan untuk mengurangi waktu eksekusi program e-voting.

II. PENELITIAN TERKAIT

Cilk[3], [4] adalah perluasan bahasa C untuk pemrograman *multithreading*. Dengan perluasan, kata tersebut bermaksud bahwa Cilk menambahkan sejumlah kata kunci pemrograman C sehingga pemrogram dapat membuat program *multithreaded*. Cilk juga merupakan sistem *runtime* yang secara internal terdapat penjadwal tugas-tugas (*task scheduler*). Cilk juga memiliki sejumlah struktur data internal untuk keperluan sinkronisasi *tugas-tugas*. Penjadwal tugas Cilk menggunakan thread-thread level sisem operasi yang disebut pekerja atau *worker*. Pekerja-pekerja Cilk membuat tugas-tugas baru dengan metode *lazy-task creation*[5]. Saat pekerja membuat tugas baru (anak/child), pekerja tersebut mengalokasikan *stack-frame* baru dan menyisipkan tugas induk *task parent* ke antrian berakhir ganda miliknya (*double-ended queue*). Antrian Pekerja menyisipkan tugas-tugas pada ekor (tail) antrian. Jika pekerja kehabisan tugas baru, pekerja mengambil tugas lama dari ekor antrian miliknya sendiri. Dengan demikian pekerja menjadwalkan tugas secara *LIFO*. Jika antrian pekerja menjadi kosong, pekerja mencuri, yang kemudian disebut pencuri, tugas-tugas dari kepala ((head)) antrian milik pekerja yang lain.

III. SISTEM E-VOTING

Bagian ini mendiskusikan rancangan sistem e-voting dengan teknik pemrograman sekuensial dan juga dengan teknik pemrograman parallel *multithreading*. Termasuk yang didiskusikan adalah algoritma komputasi sistem e-voting.

A. Perhitungan Suara pada sisi TPS

Seperti yang telah di ulas pada bagian pendahuluan, sistem e-voting memungkinkan pemilih untuk memberikan suara secara terkomputerisasi. Pada sistem e-voting, para pemilih tidak lagi memberi tanda lubang pada surat suara tetapi mereka dapat langsung menginput *vote* melalui perangkat masukan layar sentuh komputer yang berada di setiap TPS. Komputer-komputer di TPS (dua atau tiga komputer) secara terpisah menghitung perolehan suara masing-masing calon kepala daerah berdasarkan pilihan para pemilih. Pemilih hanya diberikan satu kali kesempatan memilih melalui layar sentuh komputer TPS. Secara independen, komputer TPS melakukan perhitungan perolehan suara parsial. Setelah pemilihan melewati batas waktu yang telah ditentukan, proses perhitungan suara pada komputer TPS harus berhenti dengan sendirinya. Hasil perhitungan suara parsial ditabulasi ke dalam pesan elektronik. Dengan demikian inti isi pesan elektronik adalah vektor $\bar{T}$ yang berdimensi n sama dengan jumlah calon kepala daerah. Pesan elektronik yang memuat peroleh suara masing-masing calon kemudian diberi tanda-tangan digital menggunakan kunci privat milik masing-masing TPS. Setelah pesan ditanda-tangani secara digital, pesan elektronik dikirim melalui infrastruktur internet. Gambar 1 memberikan ilustrasi konektifitas komputer TPS dan komputer KPU melalui jaringan internet.

B. Pemrosesan surat suara di KPU

Di sini diasumsikan, pasangan kunci publik-privat telah dibangkitkan dan didistribusikan secara aman oleh KPU. Pihak KPU dapat membangkitkan pasangan kunci publik-privat dan mendistribusikan kunci privat secara aman melalui media yang

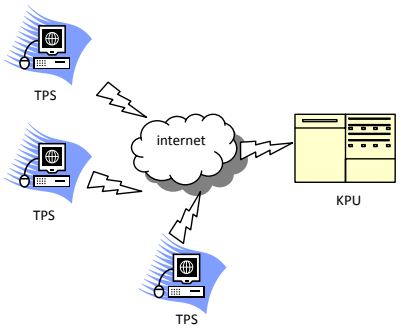

Fig. 1. Arsitektur sistem e-voting

aman seperti disk optik. Adapun kunci publik masing-masing komputer TPS telah disimpan pada database kunci komputer KPU.

Konsep utama dari pemrosesan surat suara pada komputer KPU adalah melakukan penjumlahan vektor suara secara terakumulasi. Dengan demikian, hasil tabulasi seluruh peroleh suara dapat dinyatakan sebagai array $T[t]$, dimana t adalah jumlah total pesan dari seluruh TPS. Sehinggal tabulasi total suara $\bar{K}$ di KPU diberikan pada persamaan 1.

$$K = \sum_{i=0}^{t-1} \bar{T}(i) \tag{1}$$

Ditunjukkan pada gambar 2, komputer KPU sebuah prosesor mengambil pesan elektronik satu-persatu. Prosesor tersebut kemudian mengambil kunci publik yang bersesuai dengan pengirim pesan dari database. Kunci publik tersebut digunakan untuk memverifikasi keaslian pengirim pesan agar pesan yang diterima benar-benar surat suara dari TPS yang sah. Pesan dapat disimpan pada suatu bentuk struktur data *list* yang memungkinkan prosesor mengambil pesan berikutnya setelah selesai memproses pesan sebelumnya. Disebabkan hanya ada akses tunggal oleh hanya sebuah prosesor, akses ke struktur data seperti demikian menggunakan sebuah prosesor tidak perlu menimbulkan overhead komputasi yang besar. Adapun algoritma pemrosesan ini ditunjukkan pada gambar 3

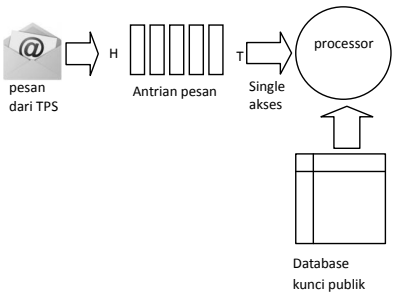

Fig. 2. Model Antrian Pesan dan Pemrosesan Pada Komputer KPU sistem e-voting

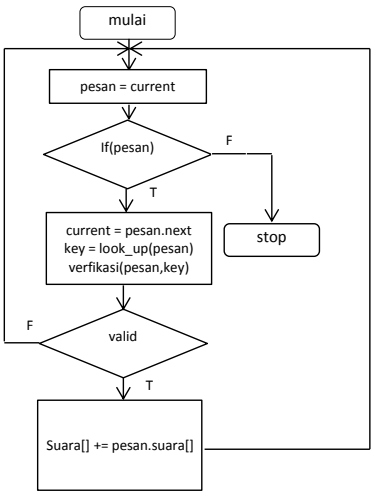

Fig. 3. Flowchart perhitungan Suara dengan metode akumulasi vektor. Dimensi vektor sesuai dengan jumlah calon kepala daerah

IV. PEMROGRAMAN PARALLEL SISTEM E-VOTING

Pada bagian ini didiskusikan desain sistem e-voting dengan teknik pemrograman parallel yang ditujukan untuk sistem komputer parallel berbasis prosesor multicore. Gambar 4 memperlihatkan model komputasi sistem e-voting menggunakan prosesor multicore. Teknik pemrograman parallel untuk prosesor multicore yang cocok adalah teknik *multithreading*.

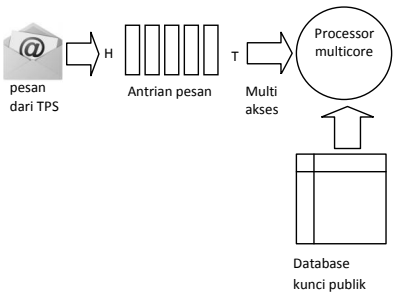

Fig. 4. Model Antrian Pesan dan Pemrosesan Pada Komputer KPU sistem E-Voting dengan menggunakan prosesor multicore

Dengan teknik *multithreading*, thread-thread dapat dianggap sebagai prosesor-prosesor logika yang berbagi memori satu sama lain. Dengan urutan yang tidak tentu, thread-thread mengakses antrian pesan. Jika antrian diimplementasikan dengan struktur senarai, thread-thread harus mengakses antrian pesan secara *mutual-exclusive*. Kabar buruk *mutual-exclusive* pada prosesor multicore yaitu berkontribusi pada overhead. Overhead tersebut akan semakin besar dengan bertambahnya jumlah inti-inti prosesor yang digunakan. Pada akhirnya eksekusi perangkat lunak tidak efisien dan mengurangi skalabilitasnya. Overhead yang besar dapat dihindari jika antrian pesan-pesan diimplementasikan menggunakan array. Index dapat diberikan pada pesan-pesan sebelum pesan-pesan diproses lebih lanjut.

```
N = jumlahCalon;
cilk_for(i=0;i<t;i++)
   K += pesan[i].suara
```

Fig. 5. Penggunaan statemen cilk_for

A. Algoritma Divide and Conquer Parallel

Dengan adanya indeks pada antrian pesan, pemrosesan surat suara dapat dilakukan dengan teknik data parallelism, seperti menggunakan *cilk_for*. Pada sistem evoting, thread-thread dapat memproses pesan-pesan dari komputer TPS ditulis dengan statemen cilk_for seperti pada gambar 5.

Potongan kode program pada gambar 5 sangat mungkin terjadi kesalahan mengakumulasi data suara. Kesalahan tersebut akibat kondisi balapan (race condition) terjadi pada vektor $\vec{K}$. Masalah ini dapat diatasi dengan menggunakan variabel reduksi (reduction variable).

Pada sistem runtime, cilk_for menerapkan algoritma *divide and conquer*. Algoritma ini membagi ruang iterasi $[a : b]$ menjadi dua bagian sub ruang iterasi $[a : m]$ dan $[m : b]$, dengan $m = (int)(a + b)/2$. Karena dilakukan secara rekursif, algoritma *divide-and-conquer* membentuk pohon biner seperti yang diilustrasikan pada gambar 6. Rekursi *divide-and-conquer* tidak dilanjutkan jika rata-rata parallelism (*average parallelism*) *A* sudah cukup lebih besar dari jumlah inti-inti prosesor yang tersedia.

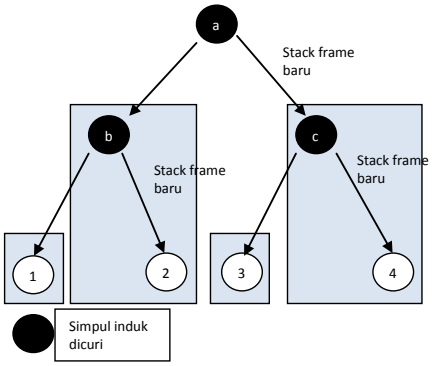

Fig. 6. Pohon eksekusi algoritma divide and conquer parallel oleh penjadwalan workstealing

Sub iterasi simpul-simpul terminal dijadwalkan ke thread-thread dengan teknik *work-stealing*. Pada bentuk pohon biner atau bentuk pohon seimbang yang serupa, thread-thread lebih baik mencuri simpul induk. Seperti pada ilustrasi gambar 7 sebuah thread mencuri simpul a Kemudian prosesor tersebut mengalokasikan stack-frame baru dan mengeksekusi simpul c dan kemudian 3. Sedangkan sebuah thread lain dapat mencuri simpul b lalu kemudian mengeksekusi simpul 2 dan demikian seterusnya untuk thread yang lainnya. Thread-thread mengeksekusi loop pada simpul-simpul terminal 1, 2, 3, 4 secara parallel. Iterasi-iterasi dijadwalkan pada simpul-simpul terminal tersebut. Setiap thread dapat mengerjakan loop untuk blok iterasi yang berurutan, sehingga ada thread-thread mengeksekusi badan prosedur C yang dipanggil dari simpul-simpul anak kiri. Badan prosedur C tersebut berisi loop seperti yang diperlihatkan pada gambar 8(a). Sedangkan untuk simpul

anak (simpul 2 dan 4) sebelah kanan, terdapat badan prosedur C yang berisi loop seperti yang diperlihatkan pada gambar 8(b).

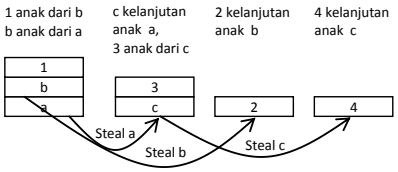

Fig. 7. Ilustrasi stack-frame eksekusi thread-thread pada algoritma divide and conquer dan penjadwalan workstealing

Cara penjadwalan parallel loop yang lain yakni pada gambar 9 dengan memberikan nomor identitas id kepada node-node terminal sebagai nomor `thread_id`. Thread_id ini digunakan sebagai indeks relatif lokal untuk mengakses array pesan.

B. Variabel Reduksi pada algoritma Divide and Conquer

Kesalahan dalam mengakumulasi array $suara[][]$ secara paralel dapat terjadi akibat kondisi balapan *race condition* pada variable global vektor K seperti pada gambar 5. Kondisi balapan terhadap variable global dapat diatasi dengan menggunakan variable reduksi atau reducer[6], sehingga potongan kode yang menghilangkan race condition dengan menggunakan variabel reduction ditunjukkan pada gambar 10

Variabel reduksi adalah salinan lokal dari variabel global yang dilihat oleh sebuah thread. Thread-thread tidak langsung mengakumulasi data suara ke variabel global, tetapi ke variabel reduksi. Hasil total perhitungan suara diperoleh dengan mereduksi array $T[p]$ menjadi sebuah vektor K berdimensi n dengan operasi jumlah (`sum_reduction`) seperti yang ditunjukkan pada persamaan 2. Pada persamaan 2 r adalah jumlah banyak salinan variabel reduksi pada mana thread-thread mengakumulasi perolehan suara.

$$\bar{K} \leftarrow sum\_red(\bar{T}[r]) \tag{2}$$

Operasi reduksi pada persamaan 2 dapat dituliskan secara iteratif sebagai persamaan 3. Ide yang naif diimplementasikan untuk reduksi adalah dengan menggunakan statemen kontrol program loop yang mengakumulasi semua elemen pada index i. Namun cara ini memerluan agar semua thread-thread selesai dengan iterasi yang dikerjakan mereka sebelumnya.

$$\bar{K} = \sum_{i=0}^{p-1} \bar{T}[i] \tag{3}$$

Cara yang lebih baik yaitu mereduksi melalui pohon biner yang dibentuk oleh algoritma *divide-and-conquer*. Dengan kalimat yang lebih spesifik, reduksi dilakukan terhadap hasil

```
for(l=a;l<m;l++){//badan loop}
```
(a) Loop untuk sub iterasi pada simpul anak kiri

```
for(r=m;r<b;r++){//badan loop}
```
(b) Loop untuk sub iterasi pada simpul anak kanan

Fig. 8. Loop paralel yang dikerjakan oleh thread-thread dari simpul kiri dan kanan pohon biner.

```
for(i= a+thread_id; i<N; i+=NNode)
{
  //badan loop
}
```

Fig. 9. Penjadwalan loop setiap simpul. Pada gambar, NNode adalah jumlah simpul terbawah pohon yang tidak memiliki simpul anak (terminal)

perhitungan per dua simpul anak sub pohon biner. Pada gambar 11 diilustrasikan bahwa A adalah variabel global vektor suara. Variabel B, C, D, E, F, G, H adalah variabel reduksi yang dialokasikan oleh thread-thread paralel. Dengan demikian terdapat 8 thread yang mengakumulasi vektor suara $\bar{T}$ ke variabel-variabel tersebut. Pada pohon biner 8 thread saling berpasangan. Demikian pula berarti variabel-variabel yang dikerjakan mereka. Kemudian variabel reduksi simpul kanan direduksi ke variabel simpul kiri. Yakni secara formal operasi reduksi pada pohon *divide-and-conquer* dituliskan sebagai $A \leftarrow A+B$. Demikian seterusnya hingga yang diperoleh hanya sebuah simpul A yang menampung seluruh akumulasi vektor suara $\bar{T}[t]$ (baris 4).

V. RINGKASAN DAN PENELITIAN SELANJUTNYA

Bagian ini merangkum pembahasan pada bagian-bagian sebelumnya. Pada bagian ini juga dibahas penelitian terkait selanjutnya dimasa yang akan datang.

Prosesor-prosesor multicore telah menjadi umum digunakan sehingga pemrograman parallel dengan teknik multithreading telah menjadi hal yang lumrah. Artikel ini mengajukan usulan penerapan pemrograman paralel-multithreading pada aplikasi e-voting. Pemrograman dengan teknik multithreading ini di usulkan untuk menangani komputasi tanda-tangan digital yang intensif pada aplikasi e-voting tersebut.

Untuk menghindari overhead komputasi, pesan-pesan yang berisi akumulasi suara setiap TPS, diindeks lalu diproses oleh

```
N = jumlahCalon;
sum_reducer<VecInt[N]> K(zeroes)
cilk_for (i=0;i<t;i++)
   K += pesan[i].suara
```

Fig. 10. Penggunaan variable reduksi dan cilk_for

1. (A) (B) (C) (D) (E) (F) (G) (H) → (A+B) (C+D) (E+F) (G+H)

2. (A) (C) (E) (G) → ((A+C)) ((E+G))

3. (A) (E) →(A+E)

4. (A)

Fig. 11. Proses reduksi 8 simpul (A B C D E F G H) menjadi sebuah simpul A = (A+B+C+D+E+F+G+H) pada bohon biner divide-and-conquer.

prosesor multicore. Thread-thread memproses pesan-pesan tersebut secara paralel sesuai dengan indeks yang dimiliki thread-thread tersebut. Sehingga model pemrograman yang tepat untuk aplikasi e-voting pada prosesor multicore adalah model *data-parallel*. Dalam artikel ini digunakan cilk_for yang merupakan bentuk dari pemrograman data paralel, yang menggunakan metoda *divide-and-conquer*.

Aplikasi E-voting menggunakan model *data-parallel* harus menggunakan variabel reduksi. Menghitung akumulasi vektor $suara[n]$ dapat dilakukan secara parsial oleh sebanyak p thread sehingga terdapat p vektor $suaran$ yang dapat dituliskan sebagai array $suara[n][p]$. Hasil perolehan suara yang akhir dapat diperoleh dengan mereduksi $suara[n][p]$ menjadi suatu vektor $suara[n]$.

Intel Many Integrated Core (MIC) akan segera masuk ke pasar. Untuk mengekploitasi teknologi masa depan ini, nested paralelism merupakan hal yang menjanjikan. Nested paralelism dapat menolong meningkatkan kinerja program parallel ketika paralelism pada level terluar sedikit, namun besar pada level dalam. Bentuk lain dari Aplikasi E-voting memiliki karakteristik seperti demikian, sehingga nested paralelism pada E-Voting dapat memanfaatkan teknologi manycore.

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosytems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] National Institute of Standards and Technology, *FIPS PUB 186-2: Digital Signature Standard (DSS)*. pub-NIST:adr: National Institute for Standards and Technology, Jan. 2000. [Online]. Available: http://www.itl.nist.gov/fipspubs/fip186-2.pdf

[3] R. Blumofe *et al.*, "Cilk: An efficient multithreaded runtime system," *Journal of Parallel and Distributed Computing*, vol. 37, no. 1, pp. 55–69, 1996.

[4] M. Frigo, C. E. Leiserson, and K. H. Randall, "The implementation of the Cilk-5 multithreaded language," *ACM SIGPLAN Notices*, vol. 33, no. 5, pp. 212–223, May 1998. [Online]. Available: http://www.acm.org:80/pubs/citations/proceedings/pldi/277650/p212-frigo/

[5] E. Mohr, D. A. Kranz, and R. H. Halstead, Jr., "Lazy task creation: A technique for increasing the granularity of parallel programs," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. PDS-2, no. 3, pp. 264–280, Jul. 1991.

[6] M. Frigo, P. Halpern, C. E. Leiserson, and S. Lewin-Berlin, "Reducers and other cilk++ hyperobjects," in *Proceedings of the 21st Annual ACM Symposium on Parallel Algorithms and Architectures (21st SPAA'09)*. Calgary, Alberta, Canada: ACM SIGACT & ACM SIGARCH, Aug. 2009, pp. 79–90, cilk Arts.