Pemodelan Penjadwalan Multilevel Feedback Queue Menggunakan Dynamic Time Quantum Pada Kasus Pemesanan Makanan di Restoran

Tri Wahyu Prasetyo
Jurusan Informatika
Universitas Sebelas Maret
Jl. Ir. Sutami No. 36 A Surakarta
triwahyuprasetyo@gmail.com

Wiharto
Jurusan Informatika
Universitas Sebelas Maret
Jl. Ir. Sutami No. 36 A Surakarta
wi_harto@yahoo.com

Afrizal Doewes
Jurusan Informatika
Universitas Sebelas Maret
Jl. Ir. Sutami No. 36 A Surakarta
afrizal.doewes@gmail.com

ABSTRAK

Dalam hal pelayanan di restoran, lamanya waktu tunggu pesanan disajikan, kesalahan urutan memasak dan tertukarnya pesanan adalah masalah yang membuat pelayanan menjadi tidak maksimal. Untuk menghindari hal tersebut perlu diterapkan sistem penjadwalan memasak yang efisien. Pada penelitian ini metode penjadwalan yang diterapkan adalah Multilevel Feedback Queue menggunakan Dynamic Time Quantum (MLFQ DTQ). Algoritma MLFQ terdiri dari beberapa queue, yang masing-masing memiliki time quantum dan algoritma sendiri (Round Robin atau FCFS). Algoritma ini menjadwalkan pesanan dengan cara mengelompokkan pesanan berdasarkan lama waktu memasaknya ke dalam queue sesuai dengan besar kecilnya time quantum queue. Kemudian dilakukan penjadwalan sesuai algoritma pada masing-masing queue secara berurutan mulai dari queue dengan prioritas tertinggi ke prioritas yang lebih rendah.

Hasil dari penelitian ini adalah, dari 19 percobaan seluruhnya menyatakan bahwa rata-rata response time MLFQ lebih cepat dibanding FCFS, baik pada parameter koki berjumlah 3, 4, maupun 5. Dari 19 percobaan yang sama, pada koki berjumlah 3, 4 dan 5 masing-masing terdapat 15, 14 dan 17 percobaan yang menunjukkan rata-rata response time MLFQ DTQ lebih cepat dibanding MLFQ. Sehingga dapat disimpulkan bahwa algoritma MLFQ DTQ dominan memiliki rata-rata response time yang lebih cepat daripada algoritma MLFQ.

Kata kunci : ***Dynamic Time Quantum, Multilevel Feedback Queue, Penjadwalan, Queue.***

1. PENDAHLUAN

Persaingan dalam bisnis kuliner meningkat seiring dengan bertambahnya restoran-restoran besar di kota-kota di Indonesia. Setiap pengusaha restoran berlomba-lomba dalam mendapatkan costumer sebanyak-banyaknya dengan meningkatkan kualitas produk dan pelayanan. Dalam menghadapi persaingan, setiap restoran harus mempunyai keunggulan dibanding dengan restoran lain seperti dari segi suasana, cita rasa, fasilitas, pelayanan dan kecepatan penanganan masalah yang dihadapi oleh konsumen sehingga konsumen merasa puas, dihargai, menjadi loyal terhadap restoran tersebut [1].

Namun dalam pelaksanaannya, kadang muncul beberapa masalah yang membuat pelayanan menjadi tidak maksimal. Seperti waktu menunggu pesanan yang lama, terjadinya penumpukan pesanan di dapur yang berakibat kesalahan urutan memasak dan tertukarnya pesanan. Untuk menghindari penumpukan pemesanan makanan di dapur/koki, salah satu solusinya adalah dengan menerapkan sistem antrian ataupun penjadwalan untuk semua pesanan yang masuk [2].

Pada umumnya penjadwalan pada restoran dilakukan berdasarkan aturan yang datang pertama dilayani pertama atau *First Come First Serve* (FCFS). Namun algoritma FCFS memiliki kelemahan yaitu jika proses yang sedang dikerjakan memakan waktu yang lama maka akan berakibat meningkatnya *response time* dan *waiting time* proses selanjutnya. Selain FCFS terdapat algoritma penjadwalan lain yaitu *Multilevel Feedback Queue* (MLFQ). Algoritma MLFQ menjadwalkan dengan cara memasukkan proses ke beberapa queue. Masing-masing queue memiliki algoritma sendiri seperti *Round Robin* atau FCFS. Algoritma MLFQ biasa digunakan pada penjadwalan CPU.

Terdapat beberapa penelitian terkait implementasi MLFQ pada penjadwalan CPU. Penelitian [3] yang membahas tentang perbedaan *Multilevel Queue* (MLQ) dan MLFQ pada salah satu kesimpulannya menyebutkan bahwa MLFQ memberikan *performance* yang bagus seperti pada *Shortest Job First* (SJF) untuk proses pendek yang interaktif dan fair bagi proses panjang yang membutuhkan kerja CPU secara intensif. Penelitian [4] yang membahas tentang perbaikan algoritma *Multilevel Feedback Queue* pada kasus penjadwalan CPU dengan menambahkan *sorting* pada *queue Round Robin*, disimpulkan bahwa cara yang diusulkan memberikan hasil yang lebih baik dibandingkan cara MLFQ tradisional.

Penelitian [5] tentang teknik penjadwalan MLFQ menggunakan M/M/c *queue* pada *grid computing*, menyimpulkan cara yang diusulkan bekerja lebih baik pada kebanyakan *scenario* ketika dibandingkan dengan algoritma FCFS dan PBS_PRO. Penelitian [6] tentang optimalisasi MLFQ pada penjadwalan CPU dengan mengimplementasikan *Dynamic Time Quantum* disimpulkan bahwa *starvation* pada proses yang panjang dapat dikurangi serta berkurangnya *turnaround time* dan *waiting time* yang berdampak pada eksekusi yang lebih cepat. Beberapa penelitian tersebut menunjukkkan bahwa penggunaan algoritma MLFQ pada kasus penjadwalan CPU telah terbukti memiliki keunggulan.

Penjadwalan CPU bersifat *preemtive* yaitu proses dapat diinterupsi bila sudah mencapai batas *time quantum*. Sedangkan pada kasus penjadwalan memasak adalah *non preemptive* yaitu proses yang sedang berjalan tidak dapat diinterupsi. Selain itu penjadwalan CPU tidak mengenal *fairness* sedangkan pada penjadwalan memasak, *fairness* menjadi salah satu hal yang diperhatikan. Yang dimaksud *fairness* disini adalah tingkat kecocokan antara urutan mulai

memasak maupun urutan selesai memasak yang dihasilkan algoritma terhadap urutan yang seharusnya(urutan yang dihasilkan FCFS).

Pada penelitian [2] tentang implementasi algoritma *Multilevel Feedback Queue* pada kasus pemesanan makanan di restoran, disimpulkan pada algoritma MLFQ terdapat perbaikan dari segi total waktu jika dibanding dengan algoritma FCFS yang biasa diterapkan di restoran. Namun pembahasan yang dilakukan pada penelitian ini tidak mengulas tentang *response time* dan *fairness*. Sehingga penelitian tersebut belum bisa menjawab bagaimana performa *response time* dan *fairness* algoritma MLFQ pada kasus penjadwalan memasak.

Atas dasar hal tersebut, penulis bermaksud memodelkan penjadwalan *Multilevel Feedback Queue* menggunakan *Dynamic Time Quantum* pada kasus penjadwalan memasak di restoran. Alasan digunakannya DTQ pada MLFQ adalah karena telah dibuktikan pada penelitian sebelumnya bahwa dengan digunakannya DTQ berdampak pada eksekusi yang lebih cepat pada penjadwalan CPU. Selain itu penelitian [7] menyimpulkan bahwa *Round Robin* dengan *dynamic quantum* lebih baik daripada *static quantum*.

2. DASAR TEORI

2.1 Sistem Antrian

Antrian terjadi disebabkan oleh jumlah kebutuhan akan layanan yang melebihi kapasitas fasilitas layanan, sehingga pengguna fasilitas layanan yang tiba tidak bisa segera mendapat layanan dikarenakan kesibukan layanan [2]. Disiplin antrian adalah konsep yang membahas mengenai kebijakan dalam menentukan pelanggan mana yang dipilih dari antrian untuk dilayani, berdasarkan urutan kedatangan pelanggan. Ada empat bentuk disiplin pelayanan yang biasa digunakan dalam praktek yaitu [8]:

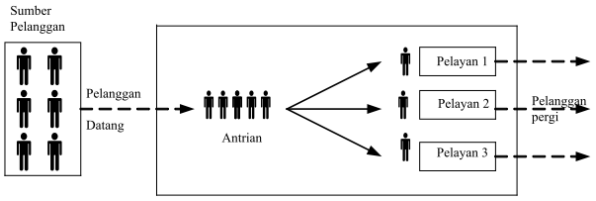
1. FIFO
   First In First Out, siapa yang datang awal dilayani awal, siapa yang datang akhir dilayanai akhir.
2. LIFO
   Last In First Out, siapa yang datang terakhir dilayani lebih dahulu.
3. RS
   Random Service, pelanggan yang dilayani dipilih secara random.
4. *Priority*
   Pelanggan yang dilayani dipilih berdasarkan prioritas.

Dalam sistem antrian, ada dua tipe prioritas, yaitu prioritas *non-preemptive* dan prioritas *preemtive*, berikut adalah penjelasannya [9] :

1. Prioritas *Non-preemptive*
   Proses dengan prioritas rendah yang sedang dilayani tidak dapat diinterupsi oleh proses lain meskipun proses tersebut prioritasnya lebih tinggi. Sehingga setiap proses yang akan dijalankan harus menunggu hingga proses yang sedang dilayani selesai.
2. Prioritas *Preemtive*
   Proses yang sedang dilayani dapat diinterupsi oleh proses yang memiliki prioritas lebih tinggi. Kemudian proses dengan prioritas rendah dapat dilayani kembali setelah proses dengan prioritas tinggi selesai yang dimulai dari titik saat dimana terjadinya interupsi.

2.2 Model Antrian

Berdasarkan sifat pelayanannya, model antrian dapat diklasifikasikan sebagai fasilitas-fasilitas pelayanan dalam susunan saluran dan *phase* yang akan membentuk suatu struktur antrian yang berbeda-beda. Istilah saluran menunjukkan jumlah jalur untuk memasuki sistem pelayanan. Sedangkan istilah *phase* berarti jumlah stasiun-stasiun pelayanan, dimana para langganan harus melaluinya sebelum pelayanan dinyatakan lengkap [8]. Model antrian pada kasus penjadwalan memasak adalah *Multi Chanel Single Phase*. *Multi Chanel Single Phase* terjadi jika dua atau lebih fasilitas pelayanan dialiri oleh suatu antrian tunggal [2]. Contoh lain adalah antrian pada loket di kantor pos dan bank, seperti yang ditunjukkan pada Gambar 1.



Gambar 1. Contoh *Multi chanel single phase* [8].

2.3 Kriteria Penjadwalan

Terdapat beberapa kriteria penjadwalan yang mempengaruhi efisiensi penjadwalan. Kriteria penjadwalan yang menjadi fokus dalam penelitian ini adalah *response time* dan *fairness*. *Response time* adalah jumlah waktu yang dihabiskan mulai dari suatu proses masuk hingga pertama kali proses mendapat respond [3]. *Response time* memberikan pengaruh terhadap lamanya waktu tunggu suatu proses mulai dikerjakan. Jika *response time* dapat diminimalkan maka lamanya waktu tunggu dapat dikurangi sehingga waktu yang diperlukan untuk pengerjaan suatu proses akan menjadi lebih singkat.

Fairness adalah tingkat kesesuaian antara urutan pengerjaan proses yang dihasilkan suatu algoritma dengan urutan pengerjaan proses yang seharusnya (urutan FCFS). Kriteria *fairness* dalam kasus penjadwalan memasak menjadi sangat penting karena berdampak langsung terhadap kepuasan pelanggan. Dalam hal ini akan menjadi tidak adil apabila pelanggan yang datang akhir mendapat layanan lebih awal daripada pelanggan yang telah datang sebelumnya. Dengan memaksimalkan *fairness*, maka kepuasan pelanggan dapat dijaga, sehingga kriteria penjadwalan yang baik dapat tercapai.

2.4 Algoritma Penjadwalan

Algoritma penjadwalan yang baik adalah yang dapat mengoptimalkan performance-nya. Algoritma penjadwalan yang digunakan pada penelitian ini diantaranya adalah :

2.4.1 First Come First Served (FCFS)

FCFS adalah aturan dimana pelaggan yang dilayani terlebih dahulu adalah pelanggan yang datang lebih awal. FCFS tidak cocok bila diterapkan pada kondisi yang real time. Ini dikarenakan suatu proses dapat memonopoli dengan eksekusi yang panjang/lama sehingga menghalangi proses pendek untuk dieksekusi. Selain itu, jika proses yang sedang dikerjakan memakan waktu yang lama maka akan berakibat meningkatnya *response time* dan *waiting time* proses selanjutnya.

2.4.2 Round robin (RR)

Round Robin scheduling adalah versi *preemptive* dari *First Come First Served scheduling*. Pada RR proses dimasukkan pada *first in first out sequence* tetapi setiap proses diijinkan untuk berjalan dalam waktu yang terbatas. Interval waktu ini biasa disebut *time slice* atau *quantum*. Jika suatu proses tidak selesai dieksekusi hingga *time slice* habis, maka proses tersebut *preempted* atau berhenti hingga mendapat giliran dieksekusi kembali. Proses yang *preempted*

ditempatkan pada *ready list* paling belakang dimana proses tersebut harus menunggu proses yang ada pada list tersebut tereksekusi dahulu. Mengalokasikan *time quantum* terlalu kecil dan terlalu besar, dapat mengakibatkan penurunan *performance* yang disebabkan oleh meningkatnya *context switching* yang berlebihan.

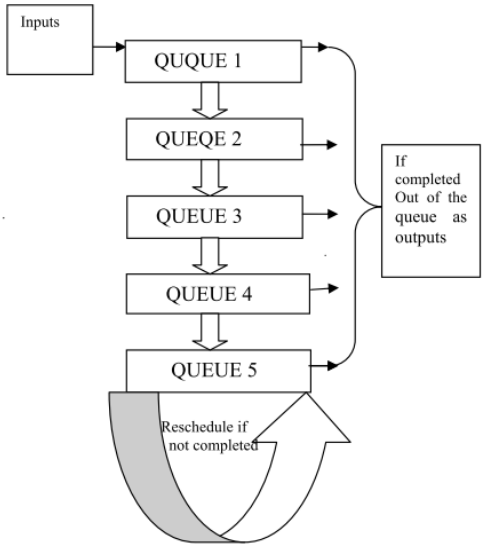
Pada *Round Robin*, jika *time quantum* terlalu besar, mengakibatkan *response time* dari proces terlalu tinggi/lama yang mungkin tidak ditoleransi pada lingkungan yang interaktif. Jika *time quantum* terlalu kecil, maka mengakibatkan *context switch* yang berlebihan sehingga menghasilkan *throughput* yang minimal [10]. Tidak ada cara standar untuk menentukan *quantum*. *Quantum* yang besar akan memaksimalkan *waiting time* dan *quantum* yang kecil dapat meningkatkan *context switching* yang berlebihan antar proses. *Quantum* pada *Round Robin* dapat ditentukan secara statis maupun dinamis. Untuk meningkatkan *performance*, *quantum* dialokasikan secara dinamis pada semua iterasi. *Performance* dari *Round Robin* dengan *Dynamic Quantum* lebih baik dibanding Round Robin dengan *Static Quantum*[7].

2.4.3 Multilevel Feedback Queue (MLFQ)

Multilevel Feedback Queue menjadwalkan proses dengan cara memasukkan proses-proses ke dalam beberapa *queue* berdasarkan besar *time quantum*. Masing-masing *queue* memiliki algoritma sendiri seperti *Round Robin* atau FCFS. Jika masih terdapat proses yang belum selesai dikerjakan maka proses tersebut dapat berpindah dari *queue* satu ke *queue* lain atau berpindah ke *queue* dengan prioritas lebih rendah. Setiap *queue* memiliki *unique time slice* yang berangsur-angsur meningkat dari *queue level* atas ke *queue* yang *levelnya* lebih rendah.

Salah satu penelitian tentang pengembangan algoritma MLFQ yang telah dilakukan adalah penggunaan *Dynamic Time Quantum* untuk meningkatkan *performance* MLFQ. Pada algoritma *Multilevel Feedback Queue* menggunakan *Dynamic Time Quantum, time quantum* dihitung secara dinamis dan berbeda-beda setiap *queue*. Jumlah level dibuat berdasarkan *time quantum* dan *burst time* dari proses [6].

Langkah awal pengerjaan algoritma *Multilevel Feedback Queue* menggunakan *Dynamic Time Quantum* adalah menentukan jumlah *queue*. Pada contoh arsitektur ini dipakai 5 *queues* (Q1 ,Q2 ,Q3 ,Q4 dan Q5) seperti yang ditunjukkan pada Gambar 2. Semua proses yang jadwalkan akan dimasukkan ke Q1. Proses dikerjakan berurutan berdasarkan level prioritas paling tinggi atau Q1. Ketika *time quantum* habis, dan proses belum selesai maka proses tersebut masuk ke level berikutnya. Langkah ini diulangi hingga sampai pada Q5. Ketika proses mencapai *queue* paling rendah, kemudian sisa *burst time* akan dikirim ke *queue* yang sesuai nilainya untuk dijadwalkan ulang sehingga eksekusi proses dapat diselesaikan [6].



Gambar 2. Alur eksekusi pada level *queue* yang berbeda [6].

Untuk mendapatkan nilai *time quantum* yang optimal, langkah awal adalah menghitung *mean* dan *median* dari semua *burst time*. *Mean* dan *median* dihitung menggunakan rumus dibawah ini [6].

$$Mean = \sum_{i=1}^{n} \frac{CPU\ burst\ time\ of\ all\ processes}{no.\ of\ processes}$$

$$Median = \begin{cases} Y_{(n+1)/2}, & \text{if n=odd} \\ 1/2(Y_{n/2} + Y_{(1+n/2)}), & \text{if n=even} \end{cases}$$

Setelah didapat *mean* dan *median*, *time quantum* pertama adalah nilai mutlak dari pengurangan *mean* dan *median,* seperti pada rumus dibawah. Sedangkan nilai *time quantum* selanjutnya adalah dua kali *time quantum* sebelumnya. Penghitungan *time quantum* dilakukan berulang hingga nilai *time quantum* yang dihasilkan adalah lebih besar atau sama dengan *burst time* proses yang terbesar.

$$t_{q1} = |Mean - Median| \qquad t_{q3} = 2 * t_{q2}$$
$$t_{q2} = 2 * t_{q1} \qquad t_{q4} = 2 * t_{q3}$$

3. METODOLOGI

3.1 Studi literatur dan Pemahaman

Studi literatur dan pemahaman ini dilakukan dengan *browsing* internet, pencarian melalui buku, artikel dan jurnal yang relevan dengan objek yang dikaji sehingga diperoleh ketepatan langkah dalam melakukan penelitian. Literasi yang didapat merupakan bahan materi yang berhubungan dengan topik penelitian, diantaranya yaitu mengenai beberapa algoritma penjadwalan seperti *First Come First Served, Round Robin, Multilevel Feedback Queue* dan *Multilevel Feedback Queue dengan Dynamic Time Quantum*.

3.2 Pengumpulan Data

Data yang digunakan dalam penelitian adalah data tentang masakan yang meliputi nama item dan lama waktu pengerjaan. Pengumpulan data dilakukan dengan cara mengamati daftar menu kemudian mencari informasi lama waktu masak. Metode yang digunakan dalam menentukan lama waktu pengerjaan setiap item adalah dengan mengukur waktu secara langsung ketika koki sedang memasak. Waktu pengerjaan diukur dengan satuan detik. Data diambil dari 4 rumah makan yang berbeda. Data waktu yang diperoleh kemudian diambil nilai rata-rata pada setiap item yang sama.

3.3 Penyeleksian Data

Tahap ini dilakukan untuk mendapatkan data makanan dan minuman yang bisa dikerjakan bersamaan apabila terdapat lebih dari satu porsi yang sama dalam sekali pengerjaan. Setiap sekali pengerjaan bersama, jumlah porsi dibatasi maksimal sebanyak 3 porsi. Pembatasan sebanyak 3 porsi dikarenakan berdasarkan keterangan dari koki, jika dalam sekali pengerjaan melebihi 3 porsi akan mengurangi kualitas rasa dari masakan.

3.4 Implementasi

Aplikasi yang dibuat adalah simulasi untuk penjadwalan memasak dengan 3 algoritma pembanding yaitu *First Come First Served, Multilevel Feedback Queue* dan *Multilevel Feedback Queue* menggunakan *Dynamic Time Quantum*.

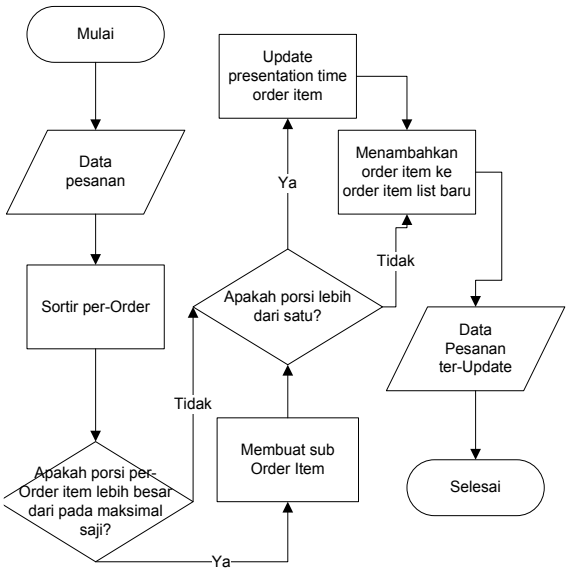
3.4.1 Membuat data pesanan

Pada penelitian ini, data pesanan dibuat secara random. Data pesanan yang dibuat dibagi menjadi dua yaitu makanan dan minuman. Proses pembuatan data pesanan serta penjadwalannya dilakukan secara terpisah antara makanan dan minuman. Parameter dalam membuat data pesanan adalah :

a. Jumlah *order*
   Order adalah satu kelompok pesanan atau kumpulan dari beberapa *order item.*
b. Jumlah *order item* dalam satu *order*
   Order item adalah satu pesanan dengan satu jenis *menu item* yang terdiri dari satu porsi atau lebih. *Menu item* adalah satu jenis makanan atau minuman.
c. Jumlah porsi dalam satu *order item*
d. *Menu id*
   Menu id adalah *id* dari *menu item* yang dipesan.
e. *Time interval*
   Time interval adalah selisih waktu kedatangan antara suatu *order* dengan *order* berikutnya.

3.4.2 Pengerjaan pesanan

Sebelum *order* dijadwalkan, data makanan dan minuman dilakukan proses sortir seperti yang ditunjukkan pada Gambar 3. Proses sortir diterapkan pada setiap *order*. Sortir dilakukan untuk menyatukan *item-item* pesanan yang memiliki kesamaan menu *id* ke dalam satu *order item*. Selanjutnya *order item* diperiksa jumlah porsinya, jika jumlah porsi lebih besar dari pada maksimal saji (tiga porsi) maka *order item* dipecah menjadi beberapa *sub order item* dengan maksimal jumlah porsi adalah tiga porsi.

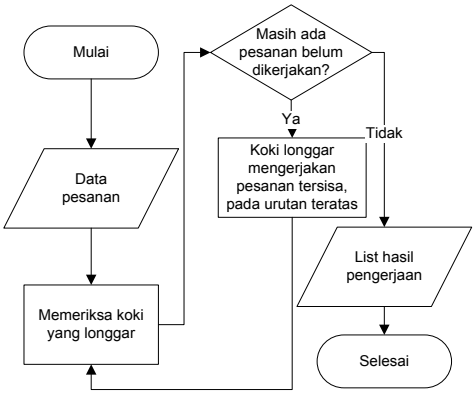


Gambar 3. Proses sortir order

Jika dalam suatu *order item* terdapat lebih dari satu porsi, maka *presentation time*-nya akan dilakukan penambahan sebesar 10% dari *presentation time* untuk setiap pertambahan 1 porsi. Sebagai contoh bila suatu *menu item* dimasak dengan waktu 100 detik dan terdapat 3 porsi pada suatu *order item*, maka *presentation time* yang dihasilkan adalah 100 detik + (20% * 100 detik) yaitu 120 detik. Angka 10% yang digunakan adalah hasil dari rata-rata penambahan total *presentation time* pada jumlah porsi berbeda sesuai data penelitian. Dan berikut adalah alur pengerjaan sesuai algoritma :

3.4.2.1 FCFS

Prinsip First Come First Serve adalah datang awal dikerjakan awal. Pada algoritma ini, koki secara bergantian mengerjakan pesanan sesuai dengan urutan kedatangan. Pesanan yang berada pada urutan berikutnya akan dikerjakan oleh koki yang selesai lebih awal. Alur pengerjaan algoritma First Come First Serve dapat diilustrasikan seperti Gambar 4.



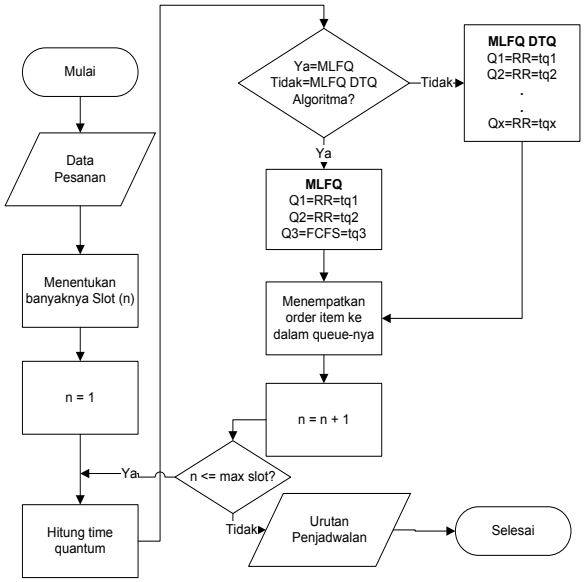
Gambar 4. Alur pengerjaan FCFS

3.4.3.2 MLFQ dan MLFQ DTQ

Pada algoritma ini variabel yang ditentukan diawal adalah jumlah koki dan besar *slot time. Slot time* adalah jarak atau ukuran waktu setiap *slot*. *Slot* digunakan untuk mengelompokkan pesanan berdasarkan waktu kedatangan. Sebagai contoh bila ditentukan *limit time* adalah 60, waktu kedatangan order paling awal 08:29:48, waktu kedatangan order paling akhir 08:34:50, maka pembagian *slot*nya adalah

1. *Slot* 1 → 08:29:48 - 08:30:48
2. *Slot* 2 → 08:30:49 - 08:31:49
3. *Slot* 3 → 08:31:50 - 08:32:50
4. *Slot* 4 → 08:32:51 - 08:33:51
5. *Slot* 5 → 08:33:52 - 08:34:52

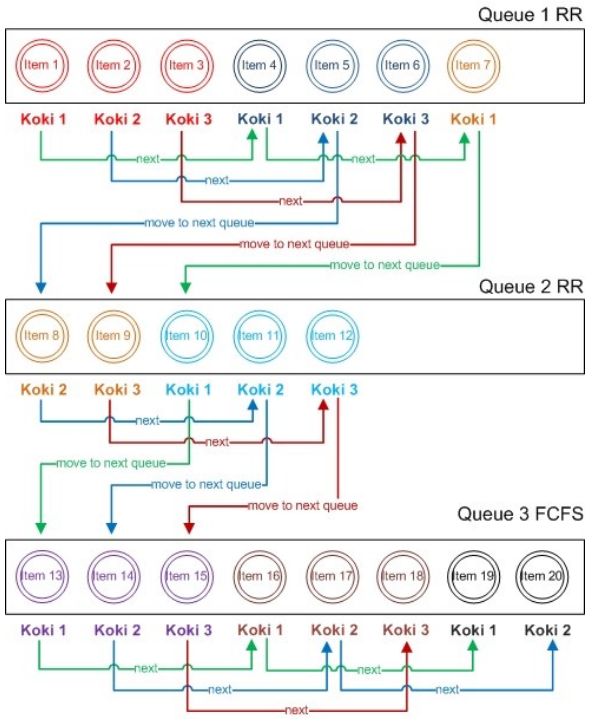
Pengerjaan dimulai dari *slot* yang paling awal. Untuk setiap *slot* langkah awal pengerjaan adalah menghitung *time quantum*(tq). Batas maksimal *time quantum* adalah lebih dari atau sama dengan *presentation time* pesanan terlama dalam *slot* tersebut. Untuk algoritma MLFQ, akan dibuat maksimal 3 *queue*(Q) yaitu Q1 dengan algoritma *Round Robin* dan tq1, Q2 dengan algoritma *Round Robin* tq2 dan Q3 dengan algoritma FCFS. Sedangkan algoritma MLFQ DTQ akan dibuat *queue* sejumlah *time quantum* yang ada. Gambar 5 mengilustrasikan pengerjaan pada MLFQ & MLFQ DTQ.



Gambar 5. Alur pengerjaan MLFQ & MLFQ DTQ

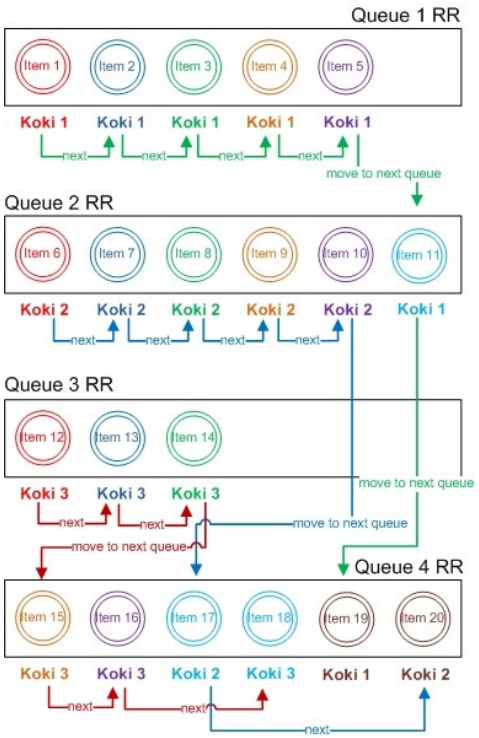
Langkah selanjutnya adalah menempatkan setiap pesanan ke dalam *queue* yang telah ditentukan *time quantum*nya. Setiap pesanan yang akan masuk ke dalam *queue* diurutkan berdasarkan urutan kedatangannya. Setiap pesanan akan masuk dan diperiksa mulai dari *queue* pertama. Apabila *presentation time* pesanan lebih kecil atau sama dengan *time quantum* pada queue pertama maka pesanan ditempatkan pada *queue* pertama. Jika *presentation time* pesanan lebih besar dari pada *time quantum queue* pertama, maka pesanan akan berpindah ke *queue* berikutnya dan diperiksa kembali kesesuaian antara besar *presentation time* pesanan dengan *time quantum queue*. Proses tersebut berulang-ulang hingga semua pesanan menempati *queue*nya.

Pada penelitian ini diusulkan urutan pengerjaan yang berbeda. Cara pengerjaan yang telah digunakan sebelumnya pada MLFQ dapat disebut sebagai urutan secara horizontal. Urutan pengerjaan yang diusulkan pada penelitian ini adalah urutan secara vertikal. Berikut adalah penjelasan cara pengerjaan urutan memasak secara horizontal dan vertikal. Gambar 6 adalah ilustrasi alur pengerjaan secara horizontal. Warna yang sama menandakan pengerjaan *item* dalam waktu hampir bersamaan oleh koki yang beda.



Gambar 6. Contoh pengerjaan MLFQ secara horizontal

Pengerjaan secara horizontal adalah mengerjakan pesanan mulai dari *queue* pertama dan setiap koki mengerjakan sesuai urutan pesanan dalam queue yang telah dijadwalkan. Satu queue dapat dikerjakan oleh lebih dari satu koki. Koki yang selesai paling awal kemudian mengerjakan pesanan yang menempati urutan teratas dalam *queue*. Jika pesanan dalam suatu *queue* telah selesai dikerjakan semua, maka koki bisa pindah ke *queue* selanjutnya yang masih terdapat pesanan yang belum dikerjakan. Urutan yang dikerjakan menurut ilustrasi gambar adalah (K1:I1, K2:I2, K3:I3), (K1:I4, K2:I5, K3:I6), (K1:I7, K2:I8, K3:I9), (K1:I10, K2:I11, K3:I12), (K1:I13, K2:I14, K3:I15), (K1:I16, K2:I17, K3:I18), (K1:I19, K2:I20). *Item* dalam tanda kurung dikerjakan dalam waktu yang bersamaan, K adalah koki dan I adalah *item*.



Gambar 7. Contoh pengerjaan MLFQ secara vertikal

Gambar 7 adalah ilustrasi alur pengerjaan secara vertikal. Pengerjaan secara vertikal adalah pertama-tama setiap koki diinisialisasi mengerjakan pesanan hanya pada suatu *queue*. Jika jumlah *queue* lebih banyak dari pada jumlah koki, maka setiap koki diinisialisasi mengerjakan satu *queue*, kemudian koki yang telah selesai mengerjakan pesanan pada *queue*-nya dapat berpindah ke *queue* baru yang belum mendapat koki. Jika tidak terdapat *queue* sisa, maka koki dapat berpindah ke *queue* berikutnya untuk membantu koki lain menyelesaikan pesanan pada *queue*-nya. Sedangkan jika jumlah *queue* lebih sedikit dari pada jumlah koki, maka setiap *queue* diinisialisasi dikerjakan oleh satu koki, kemudian koki yang belum mendapat jatah *queue* akan ditempatkan pada *queue-queue* tersebut dengan cara menambahkan satu koki pada setiap *queue* dimulai dari *queue* teratas, secara berulang hingga semua koki mendapat jatah *queue*. Urutan yang dikerjakan menurut ilustrasi gambar adalah (K1:I1, K2:I6, K3:I12), (K1:I2, K2:I7, K3:I13), (K1:I3, K2:I8, K3:I14), (K1:I4, K2:I9, K3:I15), (K1:I5, K2:I10, K3:I16), (K1:I11, K2:I17, K3:I18), (K1:I19, K2:I20).

3.5 Analisa Pengujian

Spesifikasi komputer yang digunakan dalam simulasi ini adalah labtop dengan *prosesor core* i5, ram 2 GB, vga Nvidia 512 MB dan hardisk 500 GB. Simulasi dilakukan dengan melakukan penjadwalan terhadap beberapa pesanan menggunakan algoritma yang berbeda yaitu *First Come First Served, Multilevel Feedback Queue* dan *Multilevel Feedback Queue* menggunakan *Dynamic Time Quantum*. Parameter yang digunakan untuk pengujian adalah jumlah koki dan besar *slot time*. Hasil *output* dari penjadwalan akan dianalisa mengenai rata-rata *response time* pengerjaan dan *fairness* (apakah urutan memasak yang dihasilkan sesuai dengan urutan FCFS) dari segi *start time* dan *finish time*.

Algoritma FCFS digunakan sebagai pembanding dalam pengujian *fairness* karena urutan yang dihasilkan baik *start time* dan *finish time* dianggap merepresentasikan *fairness* dalam kasus penjadwalan memasak. Konsep *fairness* yang dimaksud dalam penelitian ini adalah sama dengan konsep algoritma FCFS yaitu yang datang lebih awal akan mendapat prioritas dikerjakan lebih awal dari pada yang datang setelahnya. Selain itu, urutan memasak yang digunakan di restoran pada umumnya adalah menggunakan konsep dari algoritma FCFS.

Response time dihitung dari waktu pesanan mulai dimasak dikurangi waktu kedatangan pesanan. *Response time* dihitung per-*item* yang dikerjakan. Penilaian *response time* pada penelitian ini adalah berdasarkan nilai rata-rata yang dapat dihitung dari total *response time* seluruh item dibagi banyaknya *item*. *Start time* adalah waktu atau kapan suatu pesanan mulai dikerjakan. *Finish time* adalah waktu atau kapan suatu pesanan selesai dikerjakan. Rata-rata *response time* dihitung dengan rumus dibawah ini.

$$\text{Response time} = \text{Start time item} - \text{Arrival time it}$$

$$\text{Rata} - \text{rata response time} = \frac{\text{Total seluruh response time}}{\text{Banyaknya item}}$$

Pengujian *fairness* dilakukan pada algoritma MLFQ dan MLFQ DTQ dengan pembanding adalah FCFS. Pengujian dilakukan dengan parameter "sebelum", "sesudah" dan "*index* sama". Misal terdapat hasil urutan *finish time* pesanan seperti pada Tabel 1.

Tabel 1. Contoh finish time untuk parameter "sebelum"

Algoritma	Urutan Finish Time											
FCFS	2	1	3	4	5	6	7	8	9	10	11	12
MLFQ	2	1	3	4	5	6	8	9	7	10	11	12

Parameter "sebelum" digunakan untuk membandingkan kesesuaian antara MLFQ terhadap FCFS dalam hal suatu order harus selesai dikerjakan sebelum order-order tertentu. Contoh parameter "sebelum" berdasarkan Tabel 1 adalah order 8 pada FCFS selesai dikerjakan sebelum order 9, 10, 11 dan 12 maka order 8 pada MLFQ minimal harus dikerjakan sebelum order 9, 10, 11 dan 12. Karena order 8 pada MLFQ selesai sebelum order 9, 7, 10, 11 dan 12 maka order 8 pada MLFQ sesuai dengan order 8 pada FCFS. Contoh lain, order 7 pada FCFS selesai sebelum order 8, 9, 10, 11 dan 12. Sedangkan order 7 pada MLFQ selesai sebelum order 10, 11 dan 12 maka order 7 pada MLFQ tidak sesuai dengan order 7 pada FCFS.

Tabel 2. Contoh finish time untuk parameter "sesudah"

Algoritma	Urutan Finish Time											
FCFS	2	1	3	4	5	6	7	8	9	10	11	12
MLFQ	2	1	3	4	5	6	8	9	7	10	11	12

Parameter "sesudah" digunakan untuk membandingkan kesesuaian antara MLFQ terhadap FCFS dalam hal suatu order dikerjakan sesudah order-order tertentu. Contoh parameter "sesudah" berdasarkan Tabel 2 adalah order 6 pada FCFS dikerjakan sesudah order 2, 1, 3, 4 dan 5 maka order 6 pada MLFQ minimal harus dikerjakan sesudah order 2, 1, 3, 4 dan 5. Karena order 6 pada MLFQ dikerjakan sesudah order 2, 1, 3, 4 dan 5 maka order 6 pada MLFQ sesuai dengan order 6 pada FCFS. Contoh lain, order 8 pada FCFS dikerjakan sesudah order 2, 1, 3, 4, 5, 6 dan 7. Sedangkan order 8 pada MLFQ dikerjakan sesudah order 2, 1, 3, 4, 5 dan 6 maka order 8 pada MLFQ tidak sesuai dengan order 8 pada FCFS.

Parameter "index sama" digunakan untuk membandingkan kesesuaian posisi atau urutan suatu order pada MLFQ terhadap posisi atau urutan order yang sama pada FCFS. Sebagai contoh, berdasarkan Tabel 3 order 3 pada FCFS berada di urutan ke 3 sedangkan order 3 pada MLFQ berada di urutan ke 3 maka index order 3 pada MLFQ

dan FCFS sesuai. Contoh lain, order 7 pada FCFS berada di urutan ke 7 sedangkan order 7 pada MLFQ berada di urutan ke 9 maka index order 7 pada MLFQ dan FCFS tidak sesuai.

Tabel 3. Contoh urutan finish time parameter "index sama"

Algoritma	Urutan Finish Time											
FCFS	2	1	3	4	5	6	7	8	9	10	11	12
MLFQ	2	1	3	4	5	6	8	9	7	10	11	12

Penilaian *fairness start time* dan *finish time* didapat dari hasil rata-rata skor parameter "sebelum", "sesudah" dan "*index* sama" seperti rumus di bawah ini.

$$\text{Rata} - \text{rata fairness} = \frac{\text{skor "sebelum"} + \text{skor "sesudah"} + \text{skor "index sama"}}{3 * \text{total order}} * 100\%$$

Hasil perhitungan *fairness finish time* dari Tabel 1 adalah parameter "sebelum" terdapat 11 *order* yang sesuai, parameter "sesudah" terdapat 10 *order* yang sesuai dan parameter "*index* sama" terdapat 9 *order* yang sesuai. Sehingga rata-rata *fairness finish time* yang didapat adalah ((11+10+9)/(3*12)) x 100% = 83%.

4. HASIL DAN PEMBAHASAN

Data pesanan dibuat secara random. Tabel 4 berisi *renge* minimal dan maksimal dari parameter yang digunakan dalam simulasi.

Tabel 4. Parameter Create Order

Parameter	Min	Max
Menu id	1	30
Jumlah *order* item dalam 1 *order*	1	5
Jumlah porsi dalam 1 *order item*	1	5
Jumlah *order*	10	20
Time Interval	10 detik	20 detik

Pada penelitian ini sampel data pesanan yang digunakan adalah sebanyak 15 *order parent*. Pada proses penjadwalan, parameter yang digunakan seperti pada Tabel 5.

Tabel 5. Parameter Penjadwalan

Parameter	Value
Total Koki	3, 4, 5
First Slot Time	10 detik
Increment Slot Time	5 detik
Max Slot Time	100 detik

Max Slot Time adalah setengah dari selisih kedatangan orderParent pertama dengan orderParent terakhir dalam satu kasus, nilai tersebut dipilih agar dalam satu kasus minimal terdapat 2 slot penjadwalan.

4.1 Perbandingan Rata-rata Response time

a. Perbandingan percobaan FCFS dan MLFQ

Tabel 4. Perbandingan FCFS dan MLFQ

Koki	FCFS	Equal	MLFQ
3	0	0	19
4	0	0	19
5	0	0	19

b. Perbandingan percobaan MLFQ dan MLFQ DTQ

Tabel 5. Perbandingan MLFQ dan MLFQ DTQ

Koki	MLFQ	Equal	MLFQ DTQ
3	1	3	15
4	2	3	14
5	0	2	17

c. Perbandingan percobaan FCFS dan MLFQ V

Tabel 6. Perbandingan FCFS dan MLFQ V

Koki	FCFS	Equal	MLFQ V
3	0	0	19
4	0	0	19
5	0	0	19

d. Perbandingan percobaan MLFQ V dan MLFQ DTQ V

Tabel 7. Perbandingan MLFQ V dan MLFQ DTQ V

Koki	MLFQ	Equal	MLFQ DTQ
3	7	1	11
4	9	3	7
5	7	0	12

e. Perbandingan percobaan MLFQ dengan MLFQ V

Tabel 8. Perbandingan MLFQ dan MLFQ V

Koki	MLFQ	Equal	MLFQ V
3	19	0	0
4	19	0	0
5	19	0	0

f. Perbandingan percobaan MLFQ DTQ dan MLFQ DTQ V

Tabel 9. Perbandingan MLFQ DTQ dan MLFQ DTQ V

Koki	MLFQ DTQ	Equal	MLFQ DTQ V
3	19	0	0
4	19	0	0
5	19	0	0

Hasil dari 19 percobaan, pada Tabel 4 menunjukkan rata-rata *response time* MLFQ lebih cepat dibanding FCFS, baik pada parameter koki berjumlah 3, 4, maupun 5. Tabel 5 rata-rata *response time* MLFQ DTQ dominan lebih cepat dibandingkan MLFQ, dengan perbandingan 15 percobaan pada koki berjumlah 3 orang, 14 percobaan pada koki berjumlah 4 orang, dan 17 percobaan pada koki berjumlah 5 orang. Tabel 6 rata-rata *response time* MLFQ vertikal lebih cepat dibandingkan dengan FCFS, baik pada parameter koki berjumlah 3, 4, maupun 5.

Hasil perbandingan pada Tabel 7 didapat rata-rata *response time* MLFQ vertikal lebih cepat dibandingkan dengan MLFQ DTQ vertikal, baik pada parameter koki berjumlah 3, 4 maupun 5. Tabel 8 rata-rata *response time* MLFQ lebih cepat dibandingkan dengan MLFQ vertikal, baik pada parameter koki berjumlah 3, 4, maupun 5. Tabel 9 rata-rata *response time* MLFQ DTQ lebih cepat dibandingkan dengan MLFQ DTQ vertikal, baik pada parameter koki berjumlah 3, 4, maupun 5.

4.2 Perbandingan Rata-rata Response Slot 30, 45 dan 60

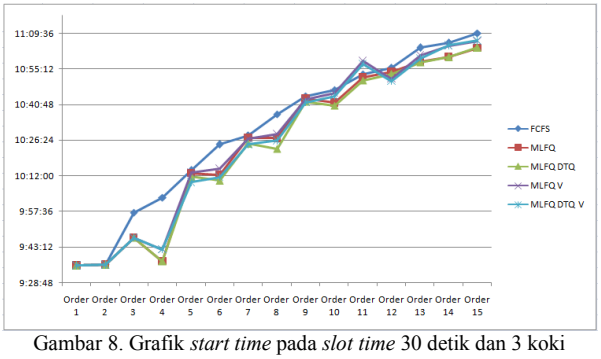
Tabel 10. Perbandingan rata-rata Response Time per-Slot

Slot Time	Koki	Rata-rata Response				
		FCFS	MLFQ	MLFQ DTQ	MLFQ V	MLFQ DTQ V
30	3	2856.64	2690.06	2645.76	2728.02	2711.74
30	4	2073.85	1960.39	1942.16	1980.20	1991.5
30	5	1609.48	1499.65	1487.92	1540.35	1551.30
45	3	2856.64	2602.28	2596.55	2668.43	2664.45
45	4	2073.84	1904.93	1902.10	1959.70	1949.08
45	5	1609.48	1504.33	1493.94	1520.51	1516.93
60	3	2856.64	2520.59	2520.59	2611.12	2611.12
60	4	2073.85	1856.95	1856.95	1900.80	1990.80
60	5	1609.48	1436.47	1422.23	1483.91	1479.14

Tabel 10 adalah hasil dari perbandingan rata-rata response time algoritma FCFS, MLFQ, MLFQ DTQ, MLFQ V dan MLFQ DTQ V dengan slot time 30, 45 dan 60 detik pada uji coba menggunakan koki berjumlah 3, 4, dan 5. Berdasarkan Tabel 12, jika diamati rata-rata response time algoritma MLFQ DTQ adalah yang paling cepat dibandingkan dengan FCFS, MLFQ, MLFQ vertikal dan MLFQ DTQ vertikal.

4.3 Perbandingan Start Time dan Finish Time

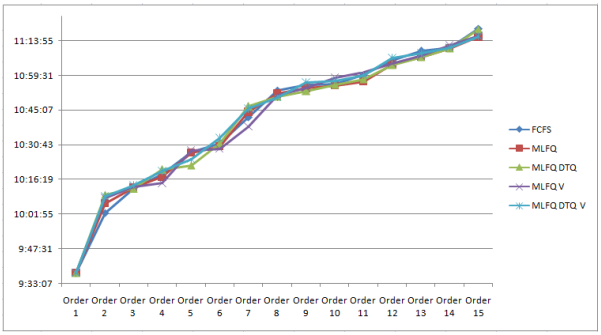
4.4.1 Start Time



Gambar 8. Grafik *start time* pada *slot time* 30 detik dan 3 koki

Grafik pada Gambar 8 menggambarkan perbandingan antara algoritma FCFS, MLFQ, MLFQ DTQ, MLFQ V dan MLFQ DTQ V dalam hal *start time order* dengan parameter *slot time* 30 detik dan koki 3 orang. Dapat dilihat pada *order* 3, 4, 6, 8, 13 dan 14, FCFS memiliki *start time* yang lebih lama dibandingkan dengan algoritma lain. Pada algoritma MLFQ dan MLFQ DTQ, semua *order* dikerjakan lebih awal jika dibandingkan dengan *order* pada FCFS. Sedangkan pada algoritma MLFQ V dan MLFQ DTQ V, semua *order* kecuali *order* 11 dikerjakan lebih awal jika dibandingkan dengan *order* pada FCFS.

4.4.2 Finish Time



Gambar 9. Grafik *finish time* pada *slot time* 30 detik dan 3 koki

Grafik pada Gambar 8 menggambarkan perbandingan antara algoritma FCFS, MLFQ, MLFQ DTQ, MLFQ V dan MLFQ DTQ V dalam hal *finish time order* dengan parameter *slot time* 30 detik dan koki 3 orang. Berdasarkan grafik dalam Gambar 9, algoritma MLFQ memiliki *finish time* lebih cepat jika dibandingkan algoritma lain pada *order* 6, 10, 11, 14 dan 15. Sedangkan algoritma MLFQ DTQ memiliki *finish time* lebih cepat jika dibandingkan algoritma lain pada *order* 3, 5, 9, 12 dan 13.

4.4 Perbandingan Fairness

Tabel 11. Perbandingan Fairness Start Time

Slot Time	Koki	Fairness Start Time			
		MLFQ	MLFQ DTQ	MLFQ V	MLFQ DTQ V
30	3	64 %	64 %	82 %	82 %
30	4	64 %	64 %	73 %	73 %
30	5	73 %	73 %	82 %	73 %
45	3	71 %	71 %	82 %	73 %
45	4	71 %	73 %	78 %	62 %
45	5	58 %	58 %	73 %	64 %
60	3	64 %	64 %	78 %	78 %
60	4	64 %	64 %	73 %	73 %
60	5	64 %	64 %	69 %	69 %

Tabel 11 adalah hasil dari perbandingan *fairness start time* algoritma MLFQ, MLFQ DTQ, MLFQ V dan MLFQ DTQ V terhadap FCFS dengan *slot time* 30, 45 dan 60 detik pada uji coba menggunakan koki berjumlah 3, 4, dan 5. Tabel 12 adalah hasil dari perbandingan *fairness finish time* algoritma MLFQ, MLFQ DTQ, MLFQ V dan MLFQ DTQ V terhadap FCFS dengan *slot time* 30, 45 dan 60 detik pada uji coba menggunakan koki berjumlah 3, 4, dan 5.

Tabel 12. Perbandingan Fairness Finish Time

Slot Time	Koki	Fairness Finish Time			
		MLFQ	MLFQ DTQ	MLFQ V	MLFQ DTQ V
30	3	91 %	82 %	87 %	78 %
30	4	82 %	73 %	69 %	78 %
30	5	69 %	69 %	73 %	69 %
45	3	69 %	60 %	78 %	78 %
45	4	73 %	69 %	91 %	82 %
45	5	64 %	64 %	78 %	73 %
60	3	51 %	51 %	51 %	51 %
60	4	51 %	51 %	47 %	47 %
60	5	42 %	42 %	47 %	47 %

Pada kasus penjadwalan pemesanan makanan di restoran, faktor *fairness* sangat penting karena berkaitan dengan keadilan pelayanan antara pengunjung. Sehingga fairness pada penelitian ini dianggap lebih penting daripada keunggulan *response time*. Dari data Tabel 11 dapat disimpulkan pengerjaan secara vertikal memiliki *fairness start time* yang lebih baik dari pada pengerjaan secara biasa/horizontal. Sedangkan pada Tabel 14 perbandingan *fairness finish time* pengerjaan vertikal sedikit lebih unggul daripada pengerjaan secara horizontal. Dan pada pengerjaan vertikal terlihat MLFQ Vertikal lebih unggul daripada MLFQ DTQ Vertikal baik dari *fairness start time* maupun *finish time*.

Dari semua perbandingan yang telah dilakukan, algoritma MLFQ Vertikal dianggap paling unggul dari algoritma pembanding lain. Karena pada MLFQ Vertikal memiliki *fairness* paling unggul meskipun *response time*-nya tidak lebih bagus daripada MLFQ DTQ, namun *response time* MLFQ Vertikal masih lebih baik jika dibandingkan FCFS.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pada penelitian, hasil dan pembahasan yang telah dilakukan dan dipaparkan di atas, maka dapat ditarik kesimpulan :

- Hasil dari 19 percobaan dapat diperoleh, rata-rata *response time* MLFQ lebih cepat dibanding FCFS, baik pada parameter koki berjumlah 3, 4, maupun 5. Pada percobaan yang sama disimpulkan bahwa rata-rata *response time* MLFQ DTQ dominan lebih cepat dibandingkan MLFQ, dengan perbandingan 15 percobaan pada koki berjumlah 3 orang, 14 percobaan pada koki berjumlah 4 orang, dan 17 percobaan pada koki berjumlah 5 orang.
- Hasil dari 19 percobaan dapat diperoleh, rata-rata *response time* MLFQ vertikal lebih cepat dibandingkan dengan FCFS. Pada percobaan yang sama disimpulkan bahwa rata-rata *response time* MLFQ lebih cepat dibandingkan dengan MLFQ vertikal dan disimpulkan pula rata-rata *response time* MLFQ DTQ lebih cepat dibandingkan dengan MLFQ DTQ vertikal.
- Pengerjaan secara horizontal atau biasa memiliki keunggulan pada segi *response time*, sedangkan pengerjaan secara vertikal memiliki keunggulan pada segi *fairness*.

- MLFQ vertikal dianggap paling bagus dari pada algoritma pembanding yang lain karena paling unggul dari segi *fairness* dan memiliki rata-rata *response time* yang lebih baik jika dibandingkan dengan FCFS.

5.2 Saran

Untuk pengembangan penelitian ini selanjutnya dapat dilakukan pada cara penentuan besar *slot time* yang menghasilkan rata-rata *response time, fairness, start time* dan *finish time* paling optimal.

6. DAFTAR PUSTAKA

[1] B. H. Salim, "Studi Deskriptif Custumer Complaint di Restoran Hotel Meritus Surabaya," *Calyptra : Jurnal Ilmiah Mahasiswa Universitas Surabaya*, vol. 2, no. 2, 2013.

[2] D. A. Ginting, S. S. Aristoteles, and O. D. Endah, "Implementation of Multilevel Feedback Queue Algorithm in Restaurant Order Food Application Development for Android and iOS Platforms," *International Journal of Computer Applications*, vol. 80, no. 13, pp. 24–30, 2013.

[3] K. K. Bharathi, "Studying Main Differences between Multilevel Queue (MLQ) and Multilevel Feedback Queue (MLFQ),"*International Journal of Computer Applications in Engineering Sciences*, vol. 3, no. 1, p. 29, 2013.

[4] R. K. Yadav and A. Upadhyay, "A Fresh Loom for Multilevel Feedback Queue Scheduling Algorithm,"*International Journal of Advances in Engineering Sciences*, vol. 2, no. 3, pp. 21–23, 2012.

[5] D. Chouhan, S. D. Kumar, and J. A. Ajay, "A MLFQ Scheduling Technique Using M/M/c Queues for Grid Computing,"*International Journal of Computer Science Issues*, vol. 10, no. 2, 2013.

[6] H. S. Behera, R. K. Naik, and S. Parida, "Improved Multilevel Feedback Queue Scheduling Using Dynamic Time Quantum and Its Performance Analysis,"*International Journal of Computer Science and Information Technology*, vol. 3, pp. 3801–3807, 2012.

[7] M. R. A. Dhumal, M. T. A. Maktum, and M. L. Ragha, "Dynamic Quantum based Genetic Round Robin Algorithm,"*International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 03, 2014.

[8] L. F. Napitu, "Analisis Antrian Pada Pt. Bank Rakyat Indonesia Cabang Pematang Siantar Unit Pasar Horas," *Universitas Sumatera Utara*, 2008.

[9] H. Ilyas, *Rekayasa Trafik Telekomunikasi*. UNJANI, 2013.

[10] S. M. Mostafa, S. Z. Rida, and S. H. Hamad, "Finding time quantum of round robin CPU scheduling algorithm in general computing systems using integer programming,"*International Journal of Research and Reviews in Applied Sciences*, vol. 5, no. 1, pp. 64–71, 2010.