목차

1. **Implement Multiprocessor Scheduling Algorithms with single-queue approach, and multi-queue approach. (You can copy & paste the codes)**

주어진 코드에서 미구현된 내용과 작동하지 않는 부분이 많아, 해당 부분을 전부 수정하고, 미구현 내용을 구현하였다.

Struct - Burst

```
// 데이터 구조체
typedef struct burst {
    int pid;  // 프로세스 ID
    int burst_length;  // burst 길이
    double arrival_time;  // 도착 시간
    double start_time;  // 시작 시간
    double real_start_time;  // 실제 시작 시간
    double remaining_time;  // 남은 시간
    double finish_time;  // 완료 시간
    double turnaround_time;  // 반환 시간
    double waiting_time;  // 대기 시간
    double response_time;  // 응답 시간
    int cpu_id;  // 실행된 CPU ID
    char finished;  // 완료 여부
    struct burst *next;  // 다음 burst를 가리키는 포인터
} Burst;
```

기존의 int 형태이던 arrival_time, remaining_time, finish_time, turnaround_time 변수를 double 로 변경하였고, 반환 시간, 대기 시간, 응답 시간을 집계하기 위해 부수적으로 사용되는 start_time, real_start_time 변수를 추가하였다.

Logic - processor_thread, dequeue

```
void *processor_thread(void *arg) {
    int cpu_id = (int) (long) arg;  // 스레드에 할당된 CPU ID를 가져옴
    MutexQueue *mutex_queue = &mutex_queues[cpu_id];  // 해당 CPU ID에 해당하는 큐를 가져옴

    while (true) {
        Burst *burst = NULL;
        double current_time = (get_wall_clock_time() - simulation_start_time) * 1000;  // 현재 시뮬레이션 시간 계산

        // 해당 큐가 비어있지 않은 경우
        if (!is_queue_empty(&mutex_queue->queue)) {
            // 스케줄링 알고리즘에 따라 다음 실행할 burst를 가져옴
            if (strcmp(alg, "RR") == 0) {  // 라운드 로빈 알고리즘
                burst = dequeue_valid_burst(&mutex_queue->queue, current_time);
            } else if (strcmp(alg, "SJF") == 0) {  // 최단 작업 우선 알고리즘
                burst = find_shortest_job(&mutex_queue->queue, current_time);
            } else if (strcmp(alg, "FCFS") == 0) {  // 선입 선출 알고리즘
                burst = dequeue_valid_burst(&mutex_queue->queue, current_time);
            }
        }

        // 실행할 burst가 존재하는 경우
        if (burst != NULL) {
            if (strcmp(alg, "RR") == 0) {  // 라운드 로빈 스케줄링
                burst->cpu_id = cpu_id;  // CPU ID 설정
                Burst *copy_burst = (Burst *) malloc(sizeof(Burst));  // burst 사본 생성
                *copy_burst = *burst;
                copy_burst->start_time = (get_wall_clock_time() - simulation_start_time) * 1000;  // 시작 시간 설정
                burst->real_start_time = copy_burst->real_start_time =  // 실제 시작 시간 설정
                        copy_burst->real_start_time == 0 ? copy_burst->start_time : MIN(copy_burst->real_start_time,
                                                                                        copy_burst->start_time);

                // 타임 퀀텀보다 남은 시간이 긴 경우
                if (copy_burst->remaining_time > Q) {
                    usleep(Q * 1000);  // 타임 퀀텀만큼 대기
```

```
                copy_burst->remaining_time = burst->remaining_time -= Q;  // 남은 시간 감소
                copy_burst->finish_time = (get_wall_clock_time() - simulation_start_time) * 1000;  // 완료 시간 갱신
                enqueue(&mutex_queue->queue, burst);  // burst 를 다시 큐에 추가
            } else {
                // 타임 퀀텀 이내에 완료될 경우
                usleep((int) burst->remaining_time * 1000);  // 남은 시간만큼 대기
                copy_burst->remaining_time = burst->remaining_time = 0;  // 남은 시간을 0 으로 설정
                copy_burst->finish_time = (get_wall_clock_time() - simulation_start_time) * 1000;  // 완료 시간 설정
                copy_burst->turnaround_time = copy_burst->finish_time - burst->arrival_time;  // 반환 시간 계산
                copy_burst->waiting_time = copy_burst->turnaround_time - copy_burst->burst_length;  // 대기 시간 계산
                copy_burst->response_time = copy_burst->real_start_time - copy_burst->arrival_time;  // 응답 시간 계산
                copy_burst->finished = 'Y';  // 완료 플래그 설정
                completed_processes++;  // 완료된 프로세스 수 증가
            }
            enqueue(&finished_bursts, copy_burst);  // 결과 큐에 burst 복사본 추가
        } else {
            // SJF 또는 FCFS 스케줄링의 경우
            burst->cpu_id = cpu_id;
            burst->start_time = burst->real_start_time = (get_wall_clock_time() - simulation_start_time) * 1000;
            usleep(burst->remaining_time * 1000);  // burst 의 남은 시간만큼 대기
            burst->finish_time = (get_wall_clock_time() - simulation_start_time) * 1000;  // 완료 시간 설정
            burst->turnaround_time = burst->finish_time - burst->arrival_time;  // 반환 시간 계산
            burst->waiting_time = burst->turnaround_time - burst->burst_length;  // 대기 시간 계산
            burst->response_time = burst->real_start_time - burst->arrival_time;  // 응답 시간 계산
            burst->finished = 'Y';  // 완료 플래그 설정
            enqueue(&finished_bursts, burst);  // 결과 큐에 burst 추가
            completed_processes++;  // 완료된 프로세스 수 증가
        }
    }

    // 최대 시뮬레이션 시간을 초과한 경우 반복문 종료
    if (get_wall_clock_time() - simulation_start_time > max_simulation_time) {
        break;
    }
}

return NULL;
}
```

```
// (도착 시간이) 유효한 burst 를 꺼내는 함수
Burst *dequeue_valid_burst(Queue *queue, double current_time) {
    int count = 0;
    // 큐를 순회하며 현재 시간 이전에 도착한 burst 탐색
    for (Burst *burst = queue->head; burst != NULL; burst = burst->next, count++) {
        if (burst->arrival_time <= current_time) {
            // 도착 시간이 현재 시간보다 이전인 경우, 해당 burst 를 제거
            for (int i = 0; i < count; i++) {
                enqueue(queue, dequeue(queue));
            }
            return dequeue(queue);
        }
    }
    return NULL;  // 해당하는 burst 가 없는 경우 NULL 반환
}
```

- RR 스케줄링이 FCFS와 같은 방식으로 구현되어 있는 것을 고치고, RR과 FCFS 스케줄링에서 도착 시간을 고려하지 않고 스케줄링 하여, 시간 지표들이 음수로 나오는 것을 수정하였다.
- finish_time을 계산할 때, 프로세스의 sleep 시간과 단위가 통일되어 있지 않아, 시간이 제대로 계산되지 않던 문제를 수정하였다.

Logic - find_shortest_job

```
// 최단 작업 찾기 함수
Burst *find_shortest_job(Queue *queue, double current_time) {
    Burst *prev = NULL;
    Burst *shortest_prev = NULL;
    Burst *shortest = NULL;
    double min_remaining_time = DBL_MAX;
```

```
    // 큐의 각 burst를 순회하며 최단 작업을 찾음
    for (Burst *burst = queue->head; burst != NULL; prev = burst, burst = burst->next) {
        // 현재 시간 이전에 도착한 burst 중 남은 시간이 가장 짧은 burst를 찾음
        if (burst->arrival_time <= current_time && burst->remaining_time < min_remaining_time) {
            min_remaining_time = burst->remaining_time;
            shortest = burst;
            shortest_prev = prev;
        }
    }

    // 찾은 최단 burst를 큐에서 제거
    if (shortest_prev != NULL) {
        shortest_prev->next = shortest->next;
    } else if (shortest != NULL) {
        queue->head = shortest->next;
    }

    if (shortest == queue->tail) {
        queue->tail = shortest_prev;
    }

    return shortest;
}
```

도착 시간을 고려하지 않고 최단 작업을 검색하여, 시간 지표가 제대로 나오지 않던 문제를 수정하였다.

Logic – parse_arguments

```
void parse_arguments(int argc, char *argv[]) {
    int opt;  // 옵션을 저장할 변수


    // 명령줄 옵션을 정의한 구조체 배열
    static struct option long_options[] = {
            {"n_processors",        required_argument, 0, 'n'},
            {"assignment_policy",    required_argument, 0, 'a'},
            {"scheduling_algorithm", required_argument, 0, 's'},
            {"infile",              required_argument, 0, 'i'},
            {"outmode",             required_argument, 0, 'm'},
            {"outfile",             required_argument, 0, 'o'},
            {"random",              required_argument, 0, 'r'},
            {"time",                required_argument, 0, 't'},
            {0, 0,                                     0, 0}
    };

    // getopt_long 함수를 이용하여 명령줄 인자를 순차적으로 파싱
    while ((opt = getopt_long(argc, argv, "n:a:s:i:m:o:r:t:", long_options, NULL)) != -1) {
        switch (opt) {
            case 'n':  // 프로세서 수 설정
                n_processors = atoi(optarg);
                break;
            case 'a':  // 할당 정책 설정
                sscanf(optarg, "%c %2s", &sap, qs);
                qs[sizeof(qs) - 1] = '\0';
                break;
            case 's':  // 스케줄링 알고리즘 설정
                sscanf(optarg, "%4s %d", alg, &Q);
                alg[sizeof(alg) - 1] = '\0';
                break;
            case 'i':  // 입력 파일 이름 설정
                strncpy(infile, optarg, sizeof(infile) - 1);
                infile[sizeof(infile) - 1] = '\0';
                break;
            case 'm':  // 출력 모드 설정
                outmode = atoi(optarg);
                break;
            case 'o':  // 출력 파일 이름 설정
                strncpy(outfile, optarg, sizeof(outfile) - 1);
                outfile[sizeof(outfile) - 1] = '\0';
                break;
            case 'r':  // 랜덤 모드 설정
                sscanf(optarg, "%d %d %d %d %d %d %d", &T, &T1, &T2, &L, &L1, &L2, &PC);
                random_flag = true;
                break;
            case 't':  // 최대 시뮬레이션 시간 설정
                max_simulation_time = atoi(optarg);
                break;
            default:  // 알 수 없는 옵션에 대한 처리
                printf("Usage: %s [-n N] [-a \"SAP QS\"] [-s \"ALG Q\"] [-i INFILE] [-m OUTMODE] [-o OUTFILE] [-r \"T
T1 T2 L L1 L2 PC\"]\n",
                       argv[0]);
                exit(1);
        }
    }
}
```

- s flag에서 알고리즘(alg)과 타임 퀀텀(Q)을 초기화할 때, alg 배열의 길이가 4로 되어있어, FCFS가 인자로 들어온 경우 제대로 처리되지 않던 문제를 수정하였다.
- getopt_long 함수는 공백 기준으로 인자를 파싱하기 때문에, 한 플래그에 여러 인자를 사용하는 경우 " (큰 따옴표)로 묶어주지 않으면, 제대로 파싱이 이루어지지 않는다. Usage 안내 메시지를 출력하는 부분에 해당 내용이 나와있지 않아 큰 따옴표를 추가하였다.
- 매크로로 정의되어 있던 최대 시뮬레이션 시간을, t 플래그를 통해 사용자가 입력받아 조절할 수 있게 수정하였다.

Logic – output_simulation_results

```
void output_simulation_results() {
    FILE *output_file = NULL;
    // 출력 모드에 따라 파일을 열거나 표준 출력을 사용
    if (outmode != 1) {
        output_file = fopen(outfile, "w");
        if (output_file == NULL) {
            perror("Error opening output file");
            exit(1);
        }
    }

    // 평균 대기 시간, 평균 응답 시간, 평균 반환 시간 계산
    double average_turnaround_time = 0.0;
    double average_waiting_time = 0.0;
    double average_response_time = 0.0;
    int burst_count = 0;
    for (Burst *burst = finished_bursts.head; burst != NULL; burst = burst->next) {
        if (burst->finished == 'Y') {
            average_turnaround_time += burst->turnaround_time;
            average_waiting_time += burst->waiting_time;
            average_response_time += burst->response_time;
            burst_count++;
        }
    }
    average_turnaround_time /= burst_count;
    average_waiting_time /= burst_count;
    average_response_time /= burst_count;

    // 계산된 평균값을 출력
    if (outmode == 1 || outmode == 3) {
        printf("Average_Turnaround_Time: %lf, Average_Waiting_Time: %lf, Average_Response_Time: %lf\n",
               average_turnaround_time, average_waiting_time, average_response_time);
        // 테이블 헤더 출력
        printf("%-10s\t%-10s\t%-15s\t%-15s\t%-20s\t%-15s\t%-15s\t%-20s\t%-20s\t%-20s\t%-10s\n",
               "Burst_ID", "CPU_ID", "Burst_Length", "Arrival_Time", "Real_Start_Time",
               "Start_Time", "Finish_Time", "Turnaround_Time", "Waiting_Time", "Response_Time", "Finished");
    }
    if (outmode == 2 || outmode == 3) {
        // 파일에 결과를 출력
        fprintf(output_file, "Average_Turnaround_Time: %lf, Average_Waiting_Time: %lf, Average_Response_Time: %lf\n",
                average_turnaround_time, average_waiting_time, average_response_time);
        fprintf(output_file, "%-10s\t%-10s\t%-15s\t%-15s\t%-20s\t%-15s\t%-15s\t%-20s\t%-20s\t%-20s\t%-10s\n",
                "Burst_ID", "CPU_ID", "Burst_Length", "Arrival_Time", "Real_Start_Time",
                "Start_Time", "Finish_Time", "Turnaround_Time", "Waiting_Time", "Response_Time", "Finished");
    }

    // 각 burst 에 대한 상세 정보를 출력
    for (Burst *burst = finished_bursts.head; burst != NULL; burst = burst->next) {
        if (outmode == 1 || outmode == 3) {
            printf("%-10d\t%-10d\t%-15d\t%-15.2lf\t%-20.2lf\t%-15.2lf\t%-15.2lf\t%-20.2lf\t%-20.2lf\t%-20.2lf\t%-
10c\n",
                   burst->pid,
                   burst->cpu_id,
                   burst->burst_length,
                   burst->arrival_time,
                   burst->real_start_time,
                   burst->start_time,
                   burst->finish_time,
                   burst->turnaround_time,
                   burst->waiting_time,
                   burst->response_time,
                   burst->finished);
        }
        if (outmode == 2 || outmode == 3) {
            fprintf(output_file,
                    "%-10d\t%-10d\t%-15d\t%-15.2lf\t%-20.2lf\t%-15.2lf\t%-15.2lf\t%-20.2lf\t%-20.2lf\t%-20.2lf\t%-
10c\n",
```

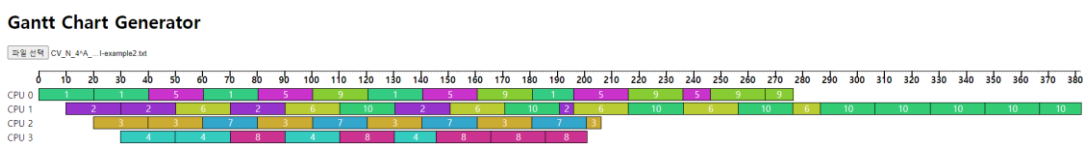
```
                burst->pid,
                burst->cpu_id,
                burst->burst_length,
                burst->arrival_time,
                burst->real_start_time,
                burst->start_time,
                burst->finish_time,
                burst->turnaround_time,
                burst->waiting_time,
                burst->response_time,
                burst->finished);
        }
    }

    // 파일이 열려있으면 닫음
    if (output_file != NULL) {
        fclose(output_file);
    }
}
```

평균 반환 시간, 평균 대기 시간, 평균 응답 시간을 계산하고, 간격에 맞게 지표들을 출력할 수 있도록 수정하였다.

Logic (mps_cv.c) – mutex

- finished_bursts를 MutexQueue 유형으로 변경하여, finished_bursts_lock을 사용하지 않고, enqueue의 mutex를 이용하여 처리할 수 있게 변경하였다.
- mps_cv.c에서 미구현된 기능들을 구현하고, 기타 버그를 수정하고 기능을 개선하였다.

Gantt Chart Generator




```
function parseCSV(text) {
    const lines = text.split('\n').filter(line => line);
    let startIndex = lines.findIndex(l => l.startsWith('Burst_ID'));

    let keys = lines[startIndex].split(/\s+/).filter(token => token);
    return lines.slice(startIndex + 1, lines.length).map(line => {
        const tokens = line.split(/\s+/).filter(token => token);
        return tokens.reduce((a, c, i) => ({...a, [keys[i]]: c}), {});
    });
}

function renderGanttChart(data) {
    const container = document.getElementById('ganttChartContainer');
    const cpuLabels = document.querySelector('.cpu-labels');
    container.innerHTML = '';
    cpuLabels.innerHTML = '';

    const maxTime = Math.max(...data.map(d => d.Finish_Time));
    const cpuCount = Math.max(...data.map(d => parseInt(d.CPU_ID))) + 1;
    const scaleInterval = 10;

    for (let i = 0; i < cpuCount; i++) {
        const label = document.createElement('div');
        label.textContent = `CPU ${i}`;
        label.style.height = '25px';
        label.style.lineHeight = '25px';
        cpuLabels.appendChild(label);
    }

    // Render time scales and labels
    for (let i = 0; i <= maxTime; i++) {
        const scale = document.createElement('div');
        scale.classList.add('time-scale');
        scale.style.left = `${(i / maxTime) * 100}%`;

        if (i % scaleInterval === 0) {
            scale.classList.add('major');
            const label = document.createElement('div');
            label.textContent = i;
            label.classList.add('time-scale-label');
            label.style.left = `${(i / maxTime) * 100}%`;
            container.appendChild(label);
        }

        container.appendChild(scale);
```

```
    }

    data.forEach(d => {
        const burst = document.createElement('div');
        burst.classList.add('burst');
        burst.innerHTML = `<span>${d.Burst_ID}</span>`;
        burst.style.left = `${(d.Start_Time / maxTime) * 100}%`;
        burst.style.width = `${((d.Finish_Time - d.Start_Time) / maxTime) * 100}%`;
        burst.style.top = `${parseInt(d.CPU_ID) * 25}px`;
        burst.style.backgroundColor = stringToColor(d.Burst_ID + d.CPU_ID);
        container.appendChild(burst);
    });
}

function stringToColor(str) {
    let hash = 0;
    for (let i = 0; i < str.length; i++) {
        hash = str.charCodeAt(i) + ((hash << 7) - hash);
        hash = hash & hash;
        hash = Math.abs(hash);
    }
    const hue = hash % 360;
    return `hsl(${hue}, 60%, 50%)`;
}
```

출력 결과를 분석하기 위해서, HTML과 JavaScript를 이용하여 출력 파일을 선택하거나, 붙여넣기하면, 간트 차트를 그려주는 WebApp을 제작하였다. (`gantt-chart-generator/index.html`)

2. Test them and compare the performance. (Test results)

Test Input & Setting

Burst_ID	Burst_Length	Arrival_Time
1	95	0
2	85	10
3	105	20
4	75	30
5	90	40
6	110	50
7	80	60
8	95	70
9	90	80
10	175	90

위와 같은 입력을 이용하여 테스트를 진행하였다. (tests/example3.txt 파일 참조)

- Arrival_Time을 10씩 늘리고, 다양한 Burst_Length에 대해서 테스트 할 수 있게 10개의 프로세스 목록을 구성
- 프로세서 수는 4로 고정
- assignment_policy 인자는 각각 S (Single Queue), M(Multi Queue) – RM, M – LM에 대해서 테스트를 진행
- 스케줄링 알고리즘 인자는 FCFS, RR(Q:20), SJF에 대해서 테스트를 진행

- mps.c와 mps_cv.c의 테스트를 분석한 결과, 각 프로세스의 스레드를 직접 sleep 상태로 만들어, 스케줄링을 시뮬레이션하는 본 코드의 특성 상, Queue에 mutex를 적용해도, 프로세서의 대다수가 sleep 상태에 머무르게 되고, 각 프로세서 식별자(CPU_ID)당 프로세스 큐가 별도로 할당되기 때문에 별다른 결과의 차이를 관찰할 수 없었다. 따라서 본 결과 분석에서는 mps.c의 출력 결과만을 정리하였다. (mps_cv.c의 결과는 results/CV_* 파일에서 확인할 수 있다)

Single Queue + FCFS

Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	0	60	0	0.06	0.06	60.13	60.13	0.13	0.06	Y
2	0	85	20	60.14	60.14	145.22	125.22	40.22	40.14	Y
3	0	105	20	145.22	145.22	250.29	230.29	125.29	125.22	Y
4	0	75	30	250.29	250.29	325.42	295.42	220.42	220.29	Y
5	0	90	30	325.42	325.42	415.5	385.5	295.5	295.42	Y
6	0	45	50	415.5	415.5	460.57	410.57	365.57	365.5	Y
7	0	35	50	460.58	460.58	495.66	445.66	410.66	410.58	Y
8	0	30	50	495.66	495.66	525.73	475.73	445.73	445.66	Y
9	0	80	50	525.73	525.73	605.81	555.81	475.81	475.73	Y
10	0	95	50	605.81	605.81	700.89	650.89	555.89	555.81	Y
11	0	100	60	700.89	700.89	800.97	740.97	640.97	640.89	Y
12	0	175	60	800.98	800.98	976.06	916.06	741.06	740.98	Y
13	0	45	70	976.06	976.06	1021.19	951.19	906.19	906.06	Y
14	0	65	80	1021.19	1021.19	1086.27	1006.27	941.27	941.19	Y
15	0	55	80	1086.27	1086.27	1141.34	1061.34	1006.34	1006.27	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
554.070174	478.070174	477.98529

Single Queue + RR (Q: 20)

Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	0	60	0	0.08	0.08	20.15	0	0	0	N
2	0	85	20	20.15	20.15	40.23	0	0	0	N
3	0	105	20	40.24	40.24	60.31	0	0	0	N
4	0	75	30	60.31	60.31	80.37	0	0	0	N
5	0	90	30	80.37	80.37	100.44	0	0	0	N
6	0	45	50	100.44	100.44	120.51	0	0	0	N
7	0	35	50	120.51	120.51	140.59	0	0	0	N
8	0	30	50	140.59	140.59	160.66	0	0	0	N
9	0	80	50	160.66	160.66	180.72	0	0	0	N
10	0	95	50	180.73	180.73	200.79	0	0	0	N
11	0	100	60	200.79	200.79	220.91	0	0	0	N
12	0	175	60	220.91	220.91	240.98	0	0	0	N
13	0	45	70	240.99	240.99	261.1	0	0	0	N
14	0	65	80	261.1	261.1	281.21	0	0	0	N
15	0	55	80	281.21	281.21	301.29	0	0	0	N
1	0	60	0	0.08	301.29	321.41	0	0	0	N
2	0	85	20	20.15	321.41	341.49	0	0	0	N
3	0	105	20	40.24	341.49	361.61	0	0	0	N
4	0	75	30	60.31	361.61	381.73	0	0	0	N
5	0	90	30	80.37	381.73	401.79	0	0	0	N
6	0	45	50	100.44	401.8	421.86	0	0	0	N
7	0	35	50	120.51	421.86	436.93	386.93	351.93	70.51	Y
8	0	30	50	140.59	436.93	447	397	367	90.59	Y
9	0	80	50	160.66	447	467.08	0	0	0	N
10	0	95	50	180.73	467.08	487.21	0	0	0	N
11	0	100	60	200.79	487.21	507.33	0	0	0	N
12	0	175	60	220.91	507.33	527.44	0	0	0	N
13	0	45	70	240.99	527.44	547.55	0	0	0	N
14	0	65	80	261.1	547.55	567.72	0	0	0	N
15	0	55	80	281.21	567.72	587.79	0	0	0	N
1	0	60	0	0.08	587.79	607.86	607.86	547.86	0.08	Y
2	0	85	20	20.15	607.86	627.93	0	0	0	N
3	0	105	20	40.24	627.93	648	0	0	0	N
4	0	75	30	60.31	648.01	668.07	0	0	0	N
5	0	90	30	80.37	668.07	688.14	0	0	0	N
6	0	45	50	100.44	688.14	693.21	643.21	598.21	50.44	Y
9	0	80	50	160.66	693.21	713.28	0	0	0	N
10	0	95	50	180.73	713.28	733.35	0	0	0	N
11	0	100	60	200.79	733.35	753.41	0	0	0	N
12	0	175	60	220.91	753.41	773.49	0	0	0	N
13	0	45	70	240.99	773.49	778.56	708.56	663.56	170.99	Y
14	0	65	80	261.1	778.56	798.63	0	0	0	N
15	0	55	80	281.21	798.63	813.69	733.69	678.69	201.21	Y
2	0	85	20	20.15	813.69	833.77	0	0	0	N
3	0	105	20	40.24	833.78	853.9	0	0	0	N
4	0	75	30	60.31	853.9	868.98	838.98	763.98	30.31	Y
5	0	90	30	80.37	868.98	889.09	0	0	0	N
9	0	80	50	160.66	889.09	909.17	859.17	779.17	110.66	Y
10	0	95	50	180.73	909.17	929.24	0	0	0	N
11	0	100	60	200.79	929.24	949.31	0	0	0	N
12	0	175	60	220.91	949.31	969.48	0	0	0	N
14	0	65	80	261.1	969.48	974.64	894.64	829.64	181.1	Y
2	0	85	20	20.15	974.64	979.71	959.71	874.71	0.15	Y
3	0	105	20	40.24	979.71	999.78	0	0	0	N
5	0	90	30	80.37	999.79	1009.86	979.86	889.86	50.37	Y
10	0	95	50	180.73	1009.86	1024.98	974.98	879.98	130.73	Y
11	0	100	60	200.79	1024.98	1045.09	985.09	885.09	140.79	Y
12	0	175	60	220.91	1045.09	1065.2	0	0	0	N
3	0	105	20	40.24	1065.2	1070.31	1050.31	945.31	20.24	Y
12	0	175	60	220.91	1070.31	1090.42	0	0	0	N
12	0	175	60	220.91	1090.42	1110.59	0	0	0	N
12	0	175	60	220.91	1110.59	1130.66	0	0	0	N
12	0	175	60	220.91	1130.66	1145.73	1085.73	910.73	160.91	Y

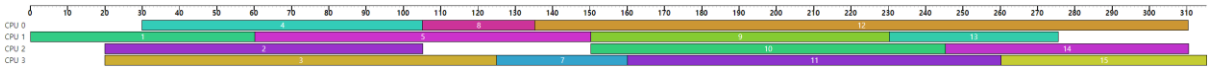
Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
807.0475	731.0475	93.93905

Single Queue + SJF

Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	0	60	0	0.05	0.05	60.12	60.12	0.12	0.05	Y
8	0	30	50	60.12	60.12	90.19	40.19	10.19	10.12	Y
7	0	35	50	90.19	90.19	125.27	75.27	40.27	40.19	Y
6	0	45	50	125.27	125.27	170.39	120.39	75.39	75.27	Y
13	0	45	70	170.39	170.39	215.47	145.47	100.47	100.39	Y
15	0	55	80	215.47	215.47	270.55	190.55	135.55	135.47	Y
14	0	65	80	270.55	270.55	335.62	255.62	190.62	190.55	Y
4	0	75	30	335.62	335.62	410.7	380.7	305.7	305.62	Y
9	0	80	50	410.71	410.71	490.78	440.78	360.78	360.71	Y
2	0	85	20	490.78	490.78	575.87	555.87	470.87	470.78	Y
5	0	90	30	575.87	575.87	665.95	635.95	545.95	545.87	Y
10	0	95	50	665.95	665.95	761.08	711.08	616.08	615.95	Y
11	0	100	60	761.08	761.08	861.24	801.24	701.24	701.08	Y
3	0	105	20	861.25	861.25	966.33	946.33	841.33	841.25	Y
12	0	175	60	966.33	966.33	1141.41	1081.41	906.41	906.33	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
429.3973	353.3973	353.3073

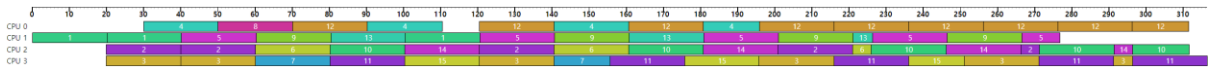
Multi Queue & RM + FCFS



Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	1	60	0	0.09	0.09	60.17	60.17	0.17	0.09	Y
4	0	75	30	30	30	105.16	75.16	0.16	0	Y
2	2	85	20	20	20	105.17	85.17	0.17	0	Y
3	3	105	20	20	20	125.1	105.1	0.1	0	Y
8	0	30	50	105.16	105.16	135.27	85.27	55.27	55.16	Y
5	1	90	30	60.17	60.17	150.25	120.25	30.25	30.17	Y
7	3	35	50	125.1	125.1	160.18	110.18	75.18	75.1	Y
9	1	80	50	150.25	150.25	230.34	180.34	100.34	100.25	Y
10	2	95	50	150.25	150.25	245.34	195.34	100.34	100.25	Y
11	3	100	60	160.18	160.18	260.28	200.28	100.28	100.18	Y
13	1	45	70	230.34	230.34	275.43	205.43	160.43	160.34	Y
12	0	175	60	135.27	135.27	310.35	250.35	75.35	75.27	Y
14	2	65	80	245.34	245.34	310.42	230.42	165.42	165.34	Y
15	3	55	80	260.28	260.28	315.36	235.36	180.36	180.28	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
152.7714	74.55716	74.45811

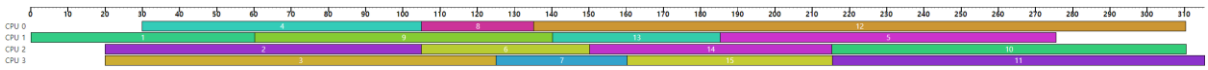
Multi Queue & RM + RR (Q: 20)



Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	1	60	0	0.09	0.09	20.17	0	0	0	N
2	2	85	20	20.02	20.02	40.09	0	0	0	N
3	3	105	20	20.02	20.02	40.1	0	0	0	N
1	1	60	0	0.09	20.17	40.24	0	0	0	N
4	0	75	30	30.02	30.02	50.08	0	0	0	N
3	3	105	20	20.02	40.1	60.16	0	0	0	N
2	2	85	20	20.02	40.1	60.17	0	0	0	N
5	1	90	30	40.24	40.24	60.38	0	0	0	N
8	0	30	50	50.08	50.08	70.16	0	0	0	N
6	2	45	50	60.17	60.17	80.25	0	0	0	N
7	3	35	50	60.17	60.17	80.29	0	0	0	N
9	1	80	50	60.38	60.38	80.5	0	0	0	N
12	0	175	60	70.16	70.16	90.27	0	0	0	N
10	2	95	50	80.25	80.25	100.34	0	0	0	N
11	3	100	60	80.29	80.29	100.37	0	0	0	N
13	1	45	70	80.5	80.5	100.57	0	0	0	N
4	0	75	30	30.02	90.27	110.38	0	0	0	N
15	3	55	80	100.37	100.37	120.47	0	0	0	N
14	2	65	80	100.34	100.34	120.47	0	0	0	N
1	1	60	0	0.09	100.57	120.64	120.64	60.64	0.09	Y
3	3	105	20	20.02	120.47	140.57	0	0	0	N
2	2	85	20	20.02	120.47	140.58	0	0	0	N
12	0	175	60	70.16	120.47	140.58	0	0	0	N
5	1	90	30	40.24	120.64	140.73	0	0	0	N
7	3	35	50	60.17	140.57	155.66	105.66	70.66	10.17	Y
6	2	45	50	60.17	140.58	160.67	0	0	0	N
4	0	75	30	30.02	140.58	160.67	0	0	0	N
9	1	80	50	60.38	140.73	160.81	0	0	0	N
11	3	100	60	80.29	155.67	175.76	0	0	0	N
12	0	175	60	70.16	160.67	180.77	0	0	0	N
10	2	95	50	80.25	160.67	180.78	0	0	0	N
13	1	45	70	80.5	160.81	180.9	0	0	0	N
15	3	55	80	100.37	175.76	195.84	0	0	0	N
4	0	75	30	30.02	180.77	195.86	165.86	90.86	0.02	Y
14	2	65	80	100.34	180.78	200.89	0	0	0	N
5	1	90	30	40.24	180.9	201.01	0	0	0	N
3	3	105	20	20.02	195.84	215.96	0	0	0	N
12	0	175	60	70.16	195.86	215.97	0	0	0	N
2	2	85	20	20.02	200.89	220.99	0	0	0	N
9	1	80	50	60.38	201.02	221.19	0	0	0	N
6	2	45	50	60.17	220.99	226.06	176.06	131.06	10.17	Y
13	1	45	70	80.5	221.19	226.34	156.34	111.34	10.5	Y
12	0	175	60	70.16	215.97	236.07	0	0	0	N
11	3	100	60	80.29	215.96	236.07	0	0	0	N
10	2	95	50	80.25	226.06	246.16	0	0	0	N
5	1	90	30	40.24	226.37	246.56	0	0	0	N
15	3	55	80	100.37	236.07	251.16	171.16	116.16	20.37	Y
12	0	175	60	70.16	236.07	256.16	0	0	0	N
14	2	65	80	100.34	246.18	266.29	0	0	0	N
9	1	80	50	60.38	246.56	266.65	216.65	136.65	10.38	Y
3	3	105	20	20.02	251.17	271.26	0	0	0	N
2	2	85	20	20.02	266.29	271.36	251.36	166.36	0.02	Y
12	0	175	60	70.16	256.21	276.29	0	0	0	N
5	1	90	30	40.24	266.65	276.75	246.75	156.75	10.24	Y
11	3	100	60	80.29	271.26	291.33	0	0	0	N
10	2	95	50	80.25	271.36	291.42	0	0	0	N
12	0	175	60	70.16	276.29	296.36	0	0	0	N
3	3	105	20	20.02	291.34	296.4	276.4	171.4	0.02	Y
14	2	65	80	100.34	291.42	296.49	216.49	151.49	20.34	Y
12	0	175	60	70.16	296.36	311.44	251.44	76.44	10.16	Y
10	2	95	50	80.25	296.5	311.6	261.6	166.6	30.25	Y
11	3	100	60	80.29	296.4	316.48	256.48	156.48	20.29	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
205.208	125.9223	10.92985

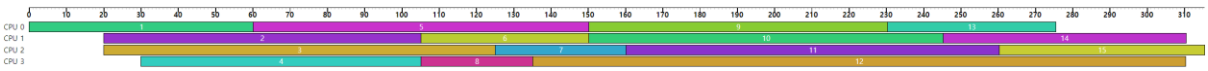
Multi Queue & RM + SJF



Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	1	60	0	0.07	0.07	60.2	60.2	0.2	0.07	Y
2	2	85	20	20	20	105.1	85.1	0.1	0	Y
4	0	75	30	30	30	105.1	75.1	0.1	0	Y
3	3	105	20	20	20	125.14	105.14	0.14	0	Y
8	0	30	50	105.1	105.1	135.21	85.21	55.21	55.1	Y
9	1	80	50	60.21	60.21	140.3	90.3	10.3	10.21	Y
6	2	45	50	105.1	105.1	150.2	100.2	55.2	55.1	Y
7	3	35	50	125.14	125.14	160.3	110.3	75.3	75.14	Y
13	1	45	70	140.3	140.3	185.4	115.4	70.4	70.3	Y
14	2	65	80	150.2	150.2	215.32	135.32	70.32	70.2	Y
15	3	55	80	160.3	160.3	215.4	135.4	80.4	80.3	Y
5	1	90	30	185.4	185.4	275.51	245.51	155.51	155.4	Y
12	0	175	60	135.21	135.21	310.33	250.33	75.33	75.21	Y
10	2	95	50	215.32	215.32	310.44	260.44	165.44	165.32	Y
11	3	100	60	215.4	215.4	315.47	255.47	155.47	155.4	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
140.6271	64.62707	64.51545

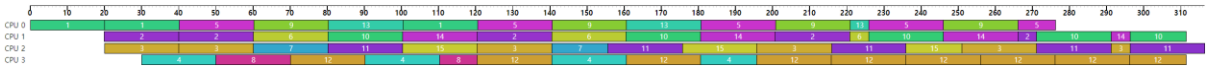
Multi Queue & LM + FCFS



Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	0	60	0	0.04	0.04	60.14	60.14	0.14	0.04	Y
2	1	85	20	20	20	105.08	85.08	0.08	0	Y
4	3	75	30	30	30	105.1	75.1	0.1	0	Y
3	2	105	20	20	20	125.08	105.08	0.08	0	Y
8	3	30	50	105.1	105.1	135.17	85.17	55.17	55.1	Y
6	1	45	50	105.09	105.09	150.16	100.16	55.16	55.09	Y
5	0	90	30	60.14	60.14	150.22	120.22	30.22	30.14	Y
7	2	35	50	125.08	125.08	160.22	110.22	75.22	75.08	Y
9	0	80	50	150.22	150.22	230.33	180.33	100.33	100.22	Y
10	1	95	50	150.16	150.16	245.27	195.27	100.27	100.16	Y
11	2	100	60	160.22	160.22	260.33	200.33	100.33	100.22	Y
13	0	45	70	230.33	230.33	275.42	205.42	160.42	160.33	Y
12	3	175	60	135.17	135.17	310.26	250.26	75.26	75.17	Y
14	1	65	80	245.27	245.27	310.35	230.35	165.35	165.27	Y
15	2	55	80	260.33	260.33	315.41	235.41	180.41	180.33	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
149.236	73.23598	73.14347

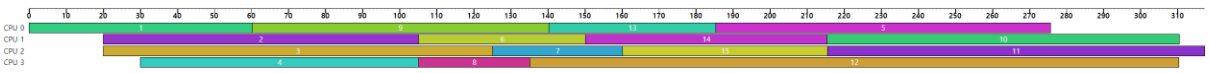
Multi Queue & LM + RR (Q: 20)



Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	0	60	0	0.06	0.06	20.14	0	0	0	N
2	1	85	20	20.02	20.02	40.09	0	0	0	N
3	2	105	20	20.05	20.05	40.13	0	0	0	N
1	0	60	0	0.06	20.14	40.21	0	0	0	N
4	3	75	30	30.02	30.02	50.09	0	0	0	N
2	1	85	20	20.02	40.1	60.16	0	0	0	N
3	2	105	20	20.05	40.13	60.2	0	0	0	N
5	0	90	30	40.21	40.21	60.28	0	0	0	N
8	3	30	50	50.09	50.09	70.16	0	0	0	N
6	1	45	50	60.16	60.16	80.23	0	0	0	N
7	2	35	50	60.2	60.2	80.27	0	0	0	N
9	0	80	50	60.28	60.28	80.36	0	0	0	N
12	3	175	60	70.16	70.16	90.23	0	0	0	N
10	1	95	50	80.23	80.23	100.31	0	0	0	N
11	2	100	60	80.27	80.27	100.35	0	0	0	N
13	0	45	70	80.36	80.36	100.45	0	0	0	N
4	3	75	30	30.02	90.23	110.35	0	0	0	N
14	1	65	80	100.31	100.31	120.41	0	0	0	N
8	3	30	50	50.09	110.35	120.47	70.47	40.47	0.09	Y
15	2	55	80	100.35	100.35	120.48	0	0	0	N
1	0	60	0	0.06	100.46	120.55	120.55	60.55	0.06	Y
2	1	85	20	20.02	120.41	140.52	0	0	0	N
12	3	175	60	70.16	120.47	140.56	0	0	0	N
3	2	105	20	20.05	120.48	140.58	0	0	0	N
5	0	90	30	40.21	120.55	140.65	0	0	0	N
7	2	35	50	60.2	140.58	155.66	105.66	70.66	10.2	Y
6	1	45	50	60.16	140.52	160.59	0	0	0	N
4	3	75	30	30.02	140.56	160.63	0	0	0	N
9	0	80	50	60.28	140.65	160.71	0	0	0	N
11	2	100	60	80.27	155.66	175.73	0	0	0	N
10	1	95	50	80.23	160.59	180.67	0	0	0	N
12	3	175	60	70.16	160.63	180.7	0	0	0	N
13	0	45	70	80.36	160.71	180.77	0	0	0	N
4	3	75	30	30.02	180.7	195.78	165.78	90.78	0.02	Y
15	2	55	80	100.35	175.73	195.82	0	0	0	N
14	1	65	80	100.31	180.67	200.75	0	0	0	N
5	0	90	30	40.21	180.77	200.85	0	0	0	N
12	3	175	60	70.16	195.79	215.92	0	0	0	N
3	2	105	20	20.05	195.82	215.92	0	0	0	N
2	1	85	20	20.02	200.75	220.84	0	0	0	N
9	0	80	50	60.28	200.85	220.93	0	0	0	N
6	1	45	50	60.16	220.84	225.93	175.93	130.93	10.16	Y
13	0	45	70	80.36	220.93	226	156	111	10.36	Y
12	3	175	60	70.16	215.92	236.01	0	0	0	N
11	2	100	60	80.27	215.93	236.01	0	0	0	N
10	1	95	50	80.23	225.94	246.02	0	0	0	N
5	0	90	30	40.21	226.01	246.08	0	0	0	N
15	2	55	80	100.35	236.01	251.08	171.08	116.08	20.35	Y
12	3	175	60	70.16	236.01	256.08	0	0	0	N
14	1	65	80	100.31	246.02	266.13	0	0	0	N
9	0	80	50	60.28	246.08	266.16	216.16	136.16	10.28	Y
3	2	105	20	20.05	251.09	271.17	0	0	0	N
2	1	85	20	20.02	266.13	271.21	251.21	166.21	0.02	Y
12	3	175	60	70.16	256.1	276.16	0	0	0	N
5	0	90	30	40.21	266.16	276.22	246.22	156.22	10.21	Y
10	1	95	50	80.23	271.21	291.28	0	0	0	N
11	2	100	60	80.27	271.17	291.3	0	0	0	N
12	3	175	60	70.16	276.16	296.23	0	0	0	N
14	1	65	80	100.31	291.28	296.34	216.34	151.34	20.31	Y
3	2	105	20	20.05	291.3	296.36	276.36	171.36	0.05	Y
12	3	175	60	70.16	296.23	311.31	251.31	76.31	10.16	Y
10	1	95	50	80.23	296.34	311.4	261.4	166.4	30.23	Y
11	2	100	60	80.27	296.36	316.43	256.43	156.43	20.27	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
196.0602	120.0602	10.18498

Multi Queue & LM + SJF



Burst_ID	CPU_ID	Burst_Length	Arrival_Time	Real_Start_Time	Start_Time	Finish_Time	Turnaround_Time	Waiting_Time	Response_Time	Finished
1	0	60	0	0.05	0.05	60.15	60.15	0.15	0.05	Y
2	1	85	20	20	20	105.09	85.09	0.09	0	Y
4	3	75	30	30	30	105.1	75.1	0.1	0	Y
3	2	105	20	20	20	125.09	105.09	0.09	0	Y
8	3	30	50	105.1	105.1	135.18	85.18	55.18	55.1	Y
9	0	80	50	60.15	60.15	140.24	90.24	10.24	10.15	Y
6	1	45	50	105.09	105.09	150.2	100.2	55.2	55.09	Y
7	2	35	50	125.09	125.09	160.21	110.21	75.21	75.09	Y
13	0	45	70	140.24	140.24	185.36	115.36	70.36	70.24	Y
14	1	65	80	150.2	150.2	215.32	135.32	70.32	70.2	Y
15	2	55	80	160.21	160.21	215.45	135.45	80.45	80.21	Y
5	0	90	30	185.36	185.36	275.62	245.62	155.62	155.36	Y
12	3	175	60	135.18	135.18	310.26	250.26	75.26	75.18	Y
10	1	95	50	215.32	215.32	310.4	260.4	165.4	165.32	Y
11	2	100	60	215.45	215.45	317.27	257.27	157.27	155.45	Y

Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
140.7297	64.72967	64.4969

Average Times

Type	Average_Turnaround_Time	Average_Waiting_Time	Average_Response_Time
Single Queue + FCFS	554.070174	478.070174	477.98529
Single Queue + RR (Q: 20)	807.047482	731.047482	93.939053
Single Queue + SJF	429.397284	353.397284	353.30725
Multi Queue & RM + FCFS	152.771446	74.55716	74.458112
Multi Queue & RM + RR (Q: 20)	205.207968	125.922253	10.929847
Multi Queue & RM + SJF	140.627069	64.627069	64.515451
Multi Queue & LM + FCFS	**149.235983**	**73.235983**	**73.143469**
Multi Queue & LM + RR (Q: 20)	**196.060247**	**120.060247**	**10.184975**
Multi Queue & LM + SJF	**140.729669**	**64.729669**	**64.496902**

Analysis

Single Queue

FCFS: 평균 반환 시간과 평균 반응 시간이 높게 나타난다. FCFS는 도착 순서대로 작업을 처리하므로, 긴 작업이 앞에 있을 경우 뒤에 있는 짧은 작업들도 오래 기다려야 하는 상황이 발생한다. 이로 인해 평균 반환 시간과 반응 시간이 증가하는 경향을 보였다.

RR: 평균 반환 시간과 평균 반응 시간이 FCFS보다 높지만, 평균 반응 시간은 상대적으로 낮게 나타났다. RR은 각 작업에 고정된 시간(타임 퀀텀) 동안 CPU를 할당하고, 시간이 지나면 다음 작업으로 넘어가는 방식을 취한다. 이러한 방식은 작업들이 빠르게 CPU에 접근할 수 있도록 하여 반응 시간을 줄이지만, 자주 발생하는 컨텍스트 스위칭으로 인해 반환 시간이 길어질 수 있다.

SJF: 평균 반환 시간과 평균 반응 시간이 가장 낮게 나타났다. SJF는 가장 짧은 작업을 먼저 처리한다. 짧은 작업이 우선적으로 처리되기 때문에 평균적으로 반환 시간과 반응 시

간이 줄어드는 경향을 보였다.

Multi Queue

FCFS: 다중 큐에서 FCFS를 적용할 경우, 각 큐는 독립적으로 도착 순서대로 작업을 처하게 된다. 이를 통해 작업 분산이 잘 이루어지면서 전체적인 반환 시간과 반응 시간을 줄여주는 효과를 보였다.

RR: 다중 큐에서 RR을 사용하면 각 큐에서 타임 퀀텀을 적용하여 작업을 처리한다. 이를 통해 작업들이 더 빠른 응답을 받을 수 있게 하면서도, 전체적인 반환 시간을 상당히 줄여주는 효과를 보였다.

SJF: 다중 큐에서 SJF를 사용하면 각 큐가 독립적으로 가장 짧은 작업을 우선적으로 처리하게 된다. 이를 통해 평균 반환 시간과 반응 시간을 크게 줄여주는 효과를 보였다.

RM & LM

RM: RM은 작업을 순환적으로 분배한다. 이 방법은 각 큐가 비슷한 수준의 부하를 가지도록 하여 고르게 작업을 처리할 수 있게 한다. 이는 전반적으로 균형 잡힌 스케줄링을 가능하게 하여 평균 반환 시간과 반응 시간을 낮추는 효과를 보였다.

LM: LM은 현재 가장 적은 부하를 가진 큐에 새로운 작업을 할당한다. 이 방식은 시스템의 전체 부하를 효율적으로 분산시키면서도 각 큐의 작업 처리를 최적화한다. 결과적으로 평균 반환 시간과 반응 시간이 낮아지는 경향을 보였다.

Conclusion

단일 큐 방식보다, 다중 큐 방식이 부하 분산을 통해 대체로 더 좋은 성능을 보였고, SJF가, FCFS보다 대체로 더 좋은 성능을 보여주었다. RR은 반환 시간과 대기 시간 측면에서는 가장 떨어지는 성능을 보였지만, 모든 경우에서 가장 낮은 응답 시간을 보여주었다. RM과 LM의 경우 단순히 순환으로 프로세스를 분배한 RM에 비해, LM이 프로세스들이 더 고르게 분포되는 경향을 보였고, 실험 결과에서도 RM과 비교해서 근소한 성능 향상을 보여주었다. 더 복잡한 스케줄링 시나리오에서 더 큰 성능 향상이 있을 것으로 보인다.

3. Impressions

이번 과제에서는 FCFS, RR, SJF의 스케줄링 방식과 단일/다중 큐, LM/RM 방식에 대한 구현과

분석을 진행하였다. 이 과정에서 기존 코드에 존재했던 미구현된 부분들과 작동하지 않는 부분들을 수정하고, 필요한 기능들을 추가하여 코드의 안정성과 정확성을 높였다. 또한, 다양한 스케줄링 방식과 그에 따른 성능 변화를 분석함으로써, 효율적인 프로세스 관리 방법에 대해 더 깊이 고민해볼 수 있는 기회가 되었다.