CPU Scheduling 알고리즘 비교와 개선된 Round Robin Scheduling

오주은(22112055)

*Student, Dept. of Computer Engineering, Yeungnam University, Longshoreman, Korea

[요 약]

컴퓨터가 점차 보편화되면서 컴퓨터 교육이 중요해지고 있다. 이러한 환경에서 학생들에게 컴퓨팅 원리를 이해시키기 위한 많은 연구가 진행되고 있다. 본 프로젝트에서는 학생들이 어려워하는 CPU Scheduling에 대해 여러 알고리즘을 실행하고 그 결과를 출력해주는 프로그램을 제안한다. 본 프로젝트를 통해 손으로 직접 계산하기 번거로웠던 각 정책의 평균 대기 시간, 평균 반환 시간 등을 쉽게 구할 수 있고 여러 가지 정책들의 성능을 비교할 수 있다. 또한 PBRR이라는 새로운 Scheduling 정책을 제안함으로써 CPU Scheduling 연구에 기여하고자 한다. 이하, 1장 서론에 이어 본 프로젝트 보고서의 구성은 다음과 같다. 2장에서는 배경지식과 본 프로젝트와 관련된 연구를 기술한다. 3장에서는 본 프로젝트에 대한 전체적인 접근을 설명한다. 다음 4장에서는 실험 설계 및 실험 결과에 관해 서술한다. 마지막으로 5장에서는 본 프로젝트의 결과를 요약하며 결론 및 향후 개선 방향에 관해 서술한다.

▸키워드 : CPU Scheduling, FCFS, SJF, Prioirity Scheduling, Round Robin Scheduling

I. 서론

기술의 급속한 발전과 함께 컴퓨터 교육은 점점 더 중요해지고 있다. 모든 것이 디지털화되고 사물 간의 정보 교환이 가능한 사물인터넷 시대로 접어들면서 컴퓨팅의 원리를 이해하는 것이 중요하다.

계속해서 증가하는 컴퓨터 시스템의 복잡성은 운영 체제 설계 및 기능과 같은 기본 개념에 대한 강력한 이해를 요구한다. 운영 체제는 응용 프로그램과 소프트웨어가 실행되는 기반을 형성하므로 이를 이해하는 것은 강력하고 효율적인 시스템을 구축하는 데 필수적 요소이다. 운영 체제는 컴퓨터 시스템의 중추이며 리소스의 효율적인 사용은 최적의 성능을 위해 필수적이다. CPU는 이러한 시스템에서 중요한 리소스이며 CPU Scheduling은 다른 프로세스에 CPU 시간을 할당하는 데 중요한 역할을 한다. 점차 보편화되고 있는 멀티코어 프로세서용 운영체제에서는 효율적인 CPU Scheduling 알고리즘을 제공해야 한다. 이를 위해서는 CPU Scheduling 알고리즘에 대한 정확한 이해가 필수적이다.

본 프로젝트에서는 FCFS(First-Come-First-Serve), 비선점형 SJF(Shortest Job First), RR(Round Robin) 및 비선점형 우선순위 Scheduling의 네 가지 일반적인 CPU Scheduling 정책을 구현하고 평가한다. 또한 기존에 프로세스의 우선 순위를 고려하지 않았던 라운드 로빈 정책을 보완하는 PBRR(Priority Based Round Robin)이라는 새로운 정책을 제안한다.

본 프로젝트의 주요 목표는 서로 다른 CPU Scheduling 정책의 성능을 연구하고 비교하는 것이다. 구체적으로, 본 프로젝트에서는 CPU Scheduling의 효율성과 효과를 측정하는 평가 기준들을 기준으로 정책을 평가하는 것을 목표로 한다. 다양한 정책을 구현하고 평가함으로써 각 정책의 강점과 약점에 대한 통찰력을 제공하고 학생들이 나아가 일정 정책을 선택할 때 정확한 정보에 입각한 결정을 내릴 수 있도록 한다.

본 프로젝트는 CPU Scheduling 정책 연구에 대한 실습 접근 방식을 제공한다. 정책을 구현하고 평가함으로써 학생들은 다양한 정책이 작동하는 방식과 시스템 성능에 미치는 영향을 더 깊이 이해할 수 있다. 또한 새로운 정책인 PBRR을 제안하여 CPU Scheduling 분야에 기여하고 실제 시나리오에서 Scheduling 문제에 대한 잠재적 솔루션을 제공하기를 희망한다.

II. 배경지식 및 관련 기술

2.1. 배경지식

본 장에서는 본 프로젝트를 진행하기 위한 사람과 프로젝트 문서를 참고하는 사람이 기본적으로 알고 있어야 하는 배경지식과 이전에 제안된 Round Robing Scheduling 개선 정책들을 소개한다.

CPU Utilization: 시간당 CPU를 사용한 시간의 비율을 뜻한다. CPU 사용률은 컴퓨터 시스템의 성능 분석 및 최적화를 위한 중요한 평가 기준이다.

처리량: 주어진 시간 동안 완료된 프로세스 수를 측정한 것이다. CPU Scheduling의 맥락에서 처리량은 단위 시간당 완료되는 프로세스 수라 할 수 있다. 높은 처리량은 시스템이 많은 수의 프로세스를 효율적으로 처리하고 있음을 나타낸다.

반환 시간: 프로세스 실행 시작에서 완료까지 걸리는 시간이다. 즉, 프로세스의 대기 시간과 버스트 시간의 합이다. 반환 시간은 프로세스의 전체 응답 시간을 반영하므로 Scheduling 알고리즘의 성능을 평가하는 중요한 평가 기준이다.

대기 시간: 프로세스가 실행을 시작하기 전에 Ready Queue에서 기다리는 시간이다. 대기 시간은 프로세스의 응답 시간에 직접적인 영향을 미치기 때문에 Scheduling 알고리즘의 효율성을 평가하는 중요한 평가 기준이다.

응답 시간: 프로세스가 Ready Queue에 들어가게 된 순간부터 실행을 시작하는 데 걸리는 시간이다. 응답 시간은 프로세스가 실행을 시작하는 데 걸리는 사용자 인식 시간을 반영하기 때문에 Scheduling 알고리즘의 성능을 평가하는 중요한 평가 기준이다.

Time Quantum: 라운드 로빈 Scheduling 알고리즘에서 프로세스가 지속적으로 실행될 수 있는 최대 시간을 말한다. TIme Quantum이 만료된 후 CPU는 실행 중인 프로세스를 중단하고 Ready Queue의 다음 프로세스로 전환한다.

우선순위: Scheduling 알고리즘에서 프로세스의 상대

적 중요도를 결정하는 프로세스에 할당된 값이다. 우선순위는 정적이거나 동적일 수 있으며 프로세스 중요도, 기한 또는 리소스 요구 사항과 같은 요소를 기반으로 할 수 있다.

문맥 교환: 운영 체제 커널이 현재 실행 중인 프로세스의 문맥을 저장하고 다른 프로세스의 문맥을 복원한 후 새 프로세스를 실행하는 것이다.

2.2. 관련 기술

본 장에서는 본 프로젝트에서 제안하는 신규정책과 관련된 Round Robin Scheduling을 개선한 선행 연구에 대해 소개하고자 한다. 기존 Round Robin Scheduling 알고리즘을 개선하기 위해 동적 Time Quantum을 사용하는 여러 연구들이 제안되었다.

Manish Kumar Mishra와Dr. Faizur Rashid[1]의 연구에서 모든 프로세스에 대해 고정된 Time Quantum을 사용하는 기존의 RR 알고리즘을 개선한다. 새로운 알고리즘에서 Time Quantum은 프로세스의 우선순위에 따라 달라진다. 우선 순위가 높은 프로세스에는 더 짧은 Time Quantum이 할당되고 우선 순위가 낮은 프로세스에는 더 긴 Time Quantum이 할당된다. 이를 통해 우선순위가 높은 프로세스가 자주 실행되어 응답시간이 향상되는 결과를 보였다.

Harshal Bharatkumar Parekh[2]연구에서 프로세스는 처음에 CPU Burst time을 기준으로 정렬되며 Burst time이 더 짧은 작업에 더 높은 우선순위가 부여된다. 우선 순위가 높은 프로세스가 먼저 실행되도록 선택되며 둘 이상의 프로세스가 동일한 우선 순위를 가질 때 RR Scheduling을 사용하여 각 프로세스에게 CPU 시간을 동일하게 할당하여 실행시키는 알고리즘을 제안했다. 제안된 알고리즘은 기존 RR 알고리즘과 비교했을 때 throughput 20% 향상, 대기시간 30%감소, 처리 시간 40% 감소된 결과를 보여주었다. 또한 처리량과 대기 시간 측면에서 SJF와 priority Scheduling 알고리즘보다 더 나은 성능을 보임을 증명하였다.

Abdulaziz A. Alsulami[3]의 연구에서 제안된 알고리즘은 피드백 메커니즘을 사용한다. CPU가 idle하거나 짧은 작업만이 ready Queue에서 대기하고 있을 때 Time Quantum을 동적으로 늘려 문맥 교환의 수를 줄여 응답 시간을 개선한다. 제안된 알고리즘은 기존 RR 알고리즘과 비교했을 때 평균 대기 시간, 평균 응답 시간, CPU uitlization 측면에서 더 나은 성능을 보임을 증명하였다.

III. CPU Scheduling

3.1. 주요 주제의 개요

(그림 1) 프로젝트 구조도

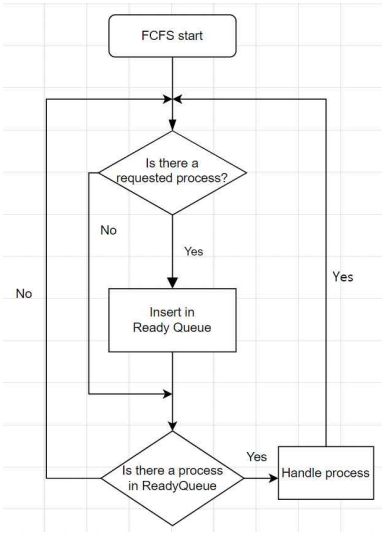
본 프로젝트는 여러 가지 CPU Scheduling을 구현하고 각 알고리즘들을 비교한다. 이를 위해 데이터 셋을 무작위로 추출하고 만들어진 데이터를 통해 각 Scheduling 알고리즘을 실행한다. 얻어진 결과들을 분석하여 결과를 도출한다.

3.2. 핵심 알고리즘 및 기능

본 장에서는 프로그램에서 구현한 기존 CPU Scheduling 기법에 대한 동작을 약설하고 새롭게 제안하는 Scheduling에 대해 상세한다. 각 정책들의 순서도를 포함하여 알고리즘을 구체화한다.

2.1. First Come First Served Scheduling

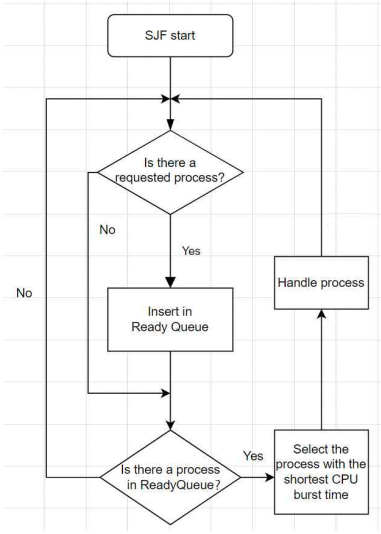
FCFS Scheduling은 Reday Queue에 삽입된 순서대로 프로세스들을 처리하는 비선점형 Scheduling 알고리즘이다.



(그림 2) FCFS Scheduling Flow Chart

2.2. Shortest Job First Scheduling

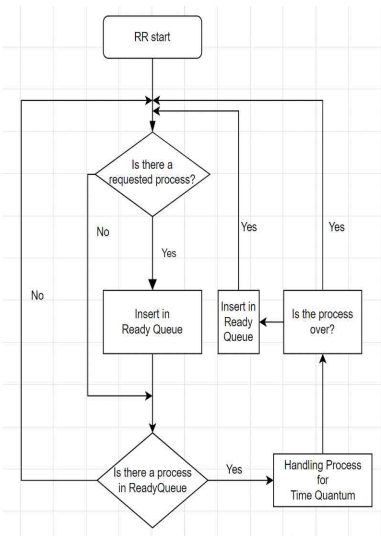
SJF는 비선점형 Scheduling으로 Reday Queue에 삽입된 프로세스들 중 CPU 이용 시간의 길이가 가장 짧은 프로세스부터 실행하는 Scheduling 알고리즘이다.



(그림 3) Shortest Remaining Time First Scheduling Flow Chart

2.3. Round Robin Scheduling

Round Robin Scheduling은 시분할 시스템을 위해 설계된 선점형 Scheduling의 하나로서, 프로세스들 사이에 우선순위를 두지 않고, 순서대로 Time Quantum만큼의 시간 동안 돌아가며 CPU를 할당하는 Scheduling 알고리즘이다.

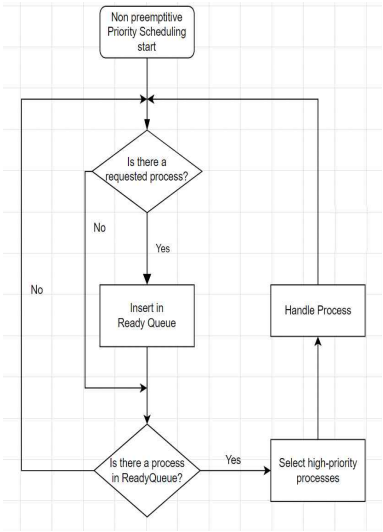


(그림 4) Round Robin Scheduling Flow Chart

2.4. Non Preemptive Priority Scheduling

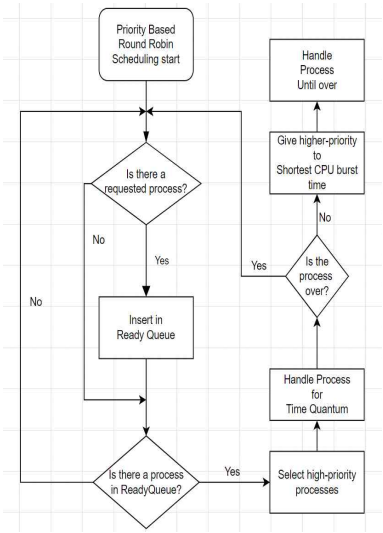
Non Preemptive Priority Scheduling은 요청한 프로세스들 중 가장 높은 우선순위를 가진 프로세스부터 실행하는 방식의 비선점형 Scheduling 알고리즘이다.

본 프로젝트에서는 우선순위를 0~9까지 고려하고 우선순위 값이 작을수록 높은 우선순위를 가진다고 가정한다.



(그림 5) Non Preemptive Priority Scheduling Flow Chart

2.5. Priority Based Round Robin Scheduling



(그림 6) Priority Based Round Robin Scheduling Flow Chart

본 프로젝트에서 제안하는 PBRR Scheduling은 Round Robin과 priority scheduling 알고리즘의 통합을 기반으로 한다.

모든 프로세스는 처음에 주어진 우선순위에 따라 Round Robin 방식으로 CPU를 한 번 할당 받는다. 이

후 두 번째 단계에서 CPU는 Ready Queue에 남아있는 프로세스들의 CPU Burst time 순서대로 정렬된다. 프로세스의 남은 Burst time에 따라 새 우선순위가 할당되고 남아있는 CPU Burst time이 가장 짧은 프로세스가 가장 높은 우선순위를 새롭게 할당받는다. 우선순위에 따라 프로세스는 Burst time이 끝날 때까지 선점되지 않고 실행된다.

기존 RR 알고리즘에서는 우선순위를 고려하지 않는 반면 PBRR 알고리즘에서는 우선순위를 고려하며 실행된다. 제안된 알고리즘에서는 두 번째 CPU를 할당받을 때 CPU Burst time이 짧은 프로세스에게 높은 우선순위를 부여하기 때문에 기존 RR 알고리즘에 비해 평균 대기 시간 감소를 기대할 수 있다.

3.3. 알고리즘 동작 사례

본 장에서는 각 알고리즘의 동작 사례를 보여주고 Gantt Chart를 활용하여 동작 절차를 소개하며 각 알고리즘의 특징을 설명한다.

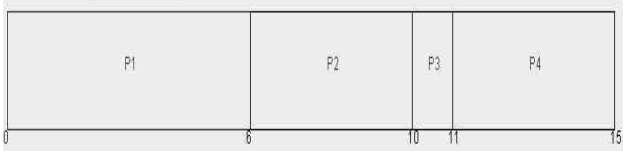
Process	Arrival Time	Burst Time	Priority
1	0	6	3
2	1	4	2
3	4	1	5
4	5	4	1

(표 1) 데이터 셋 예시

표 1에 있는 데이터 셋에 대해 각 알고리즘을 동작한 결과를 통해 본 장에서 동작 사례를 보여준다.

3.1. First Come First Served Scheduling

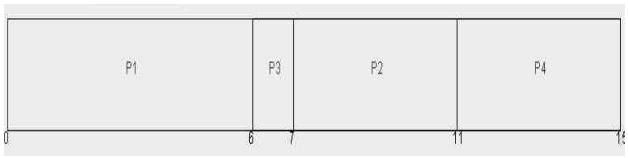
FCFS는 비선점형 Scheduling으로 arrival time이 빠른 순서대로 동작한다. 문맥 교환이 적게 발생하는 장점을 가지고 있다. 하지만 호위 효과를 발생시킬 수 있는 Scheduling 알고리즘이다. 이는 4장의 성능 평가에서 추가 데이터 셋을 통해 확인한다.



(그림 7) FCFS 동작 사례

3.2. Shortest Job First Scheduling

SJF는 비선점형 Scheduling 방식으로 CPU Burst time이 짧은 프로세스에게 CPU를 먼저 할당하는 방식이다. 이를 통해 Scheduling 방식 중 최소 평균 대기 시간을 보장한다.



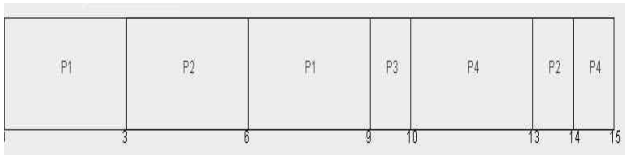
(그림 8) SJF 동작 사례

3.3. Round Robin Scheduling

Round Robin Scheduling은 선점형 Scheduling 방식으로 프로세스가 들어온 순서대로 정해진 Time Quantum만큼 CPU를 돌아가며 할당받는 방식이다. Time Quantum이 끝나면 다른 프로세스를 실행해야 하므로 문맥 교환이 많아지는 단점이 있다.

표 1의 데이터 셋에 대해서 Time Quantum을 3으로 설정하고 실행하였다.

그림 9를 보면 Time Quantum만큼 실행 후 다른 프로세스들을 실행하고 있어 다른 Scheduling 알고리즘에 비해 문맥 교환이 많이 발생하고 있음을 확인할 수 있다.

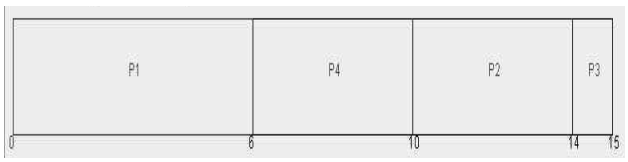


(그림 9) Round Robin Scheduling 동작 사례

3.4. Non Preemptive Priority Scheduling

Non Preemptive Priority Scheduling은 비선점형 Scheduling 방식으로 우선순위가 높은 프로세스에게 CPU를 먼저 할당하는 방식이다. 하지만 비선점형 Scheduling 방식이기 때문에 높은 우선순위를 가진 프로세스보다 낮은 우선순위의 프로세스가 먼저 실행되는 우선순위 역전 현상이 발생할 수 있다.

그림 10을 보면 P1이 끝난 후에 우선순위가 높은 P4가 실행되는 것을 확인할 수 있다.



(그림 10) Non Preemptive Priority Scheduling 동작 사례

3.5. Priority Based Round Robin Scheduling

3.5.1. 표 1의 데이터셋에 대한 동작 결과

그림 11은 Time Quantum을 3으로 설정하고 표 1의 데이터셋을 PBRR 알고리즘을 이용해 동작한 결과이다. PBRR의 동작 사례에 대해서는 추가 데이터셋을 사용하여 설명한다.

(그림 11) PRBB 동작 사례

3.5.2. 추가 데이터셋에 대한 동작 사례

Process	Arrival Time	Burst Time	Priority
1	0	22	4
2	0	18	2
3	0	9	1
4	0	10	3
5	0	4	5

(표 2) Priority Based Round Robin 추가 데이터 셋

주어진 추가 데이터 셋에 대해서는 TIme Quantum을 5로 가정한다. 또한 모든 프로세스의 Arrival Time을 동일하다 가정한다.

해당 데이터 셋에 대한 동작 사례는 그림 16을 통해 확인할 수 있다. PBRR 알고리즘은 두 라운드로 구성된다. 먼저 우선순위가 높은 프로세스부터 Time Quantum만큼 돌아가며 실행된다.

Process	executed Burst	Priority
3	5	1
2	5	2
4	5	3
1	5	4
5	4	5

(표 3) 첫 라운드 때 프로세스 실행 순서와 실행 시간

첫 라운드를 실행하고 남은 CPU Burst time에 따라 프로세스의 변경된 우선순위는 표 4와 같다.

Process	Remaining Burst	Priority
3	4	1
4	5	2
2	13	3
1	17	4

(표 4) 첫 라운드 실행 후 변경된 우선순위

각 프로세스는 실행이 완료될 때까지 CPU를 할당 받는다.

IV. 성능 평가

4.1. 실험 환경

본 프로젝트는 java를 통해 프로그램을 구현하고 IntelIJ 개발 환경을 사용하였다. JDK 1.8.0을 사용하여 실행하였다. 데이터 셋은 Abraham Silberschatz의 운영체제 10판에서 사용된 예제와 chat GPT를 통해 생성한 데이터 셋을 사용하였다.

본 프로젝트는 0~9까지의 우선순위를 가정하고 값이 작을수록 높은 우선순위를 가진다고 가정한다. 또한 문맥 교환 overhead는 없음을 가정한다.

1.1. 기본 데이터 셋

본 프로젝트에서는 구현한 알고리즘을 비교하기 위해 랜덤 데이터 셋을 사용한다. 테스트에 쓰일 프로세스를 랜덤으로 10개 생성하였다. CPU Burst Time은 1~25의 값으로 생성하고 우선순위는 0~9로 선정하였다.

랜덤 데이터 셋은 chat GPT를 통해 생성하여 테스트에 사용하였다.

Process	Arrival Time	Burst Time	Priority
1	1	5	6
2	2	8	2
3	3	2	8
4	4	9	4
5	5	3	5
6	10	2	3
7	12	1	5
8	15	7	2
9	18	6	7
10	25	4	4

(표 5) 랜덤 데이터 셋

표 5의 데이터 셋을 각각의 알고리즘을 통해 실행하고 10개의 프로세스에 대한 평균 반환 시간, 평균 대기 시간, 평균 응답 시간을 통해 성능을 비교하였다.

1.2. FCFS 추가 데이터 셋

추가적인 FCFS 성능 평가를 위해 두가지 데이터 셋을 사용한다. FCFS는 프로세스의 arrival time이 빠른 순서대로 실행되므로 CPU Burst Time이 오름차순인 케이스

와 내림차순인 케이스를 비교하여 평균 대기시간을 비교하였다.

Process	Arrive Time	Burst Time	Priority
1	0	24	1
2	1	4	1
3	2	5	1

(표 6) FCFS 추가 데이터 셋 1

Process	Arrive Time	Burst Time	Priority
1	3	24	1
2	0	4	1
3	2	5	1

(표 7) FCFS 추가 데이터 셋 1

1.3. SJF 추가 데이터 셋

SJF는 CPU Burst time이 짧은 프로세스에게 높은 우선순위를 주는 알고리즘으로 계속해서 Burst Time이 짧은 프로세스가 들어오게 된다면 긴 Burst time를 가진 프로세스는 실행되지 못하는 기아 현상이 나타날 수 있다. 표 8의 추가 데이터 셋을 통해 이 현상을 확인한다.

Process	Arrive Time	Burst Time	Priority
1	0	2	1
2	1	11	1
3	2	3	1
4	2	3	1
5	2	1	1
6	4	3	1
7	5	2	1
8	6	4	1

(표 8) SJF 추가 데이터 셋

1.4. Non preemptive priority Scheduling 추가 데이터 셋

Process	Arrive Time	Burst Time	Priority
1	0	4	5
2	1	3	1
3	2	2	2
4	3	4	1

(표 9) Non preemptive priority Scheduling 추가 데이터 셋

non preemptive priority Scheduling은 비선점형 Scheduling으로 프로세스가 실행되면 우선순위가 더 높은 프로세스가 들어오더라도 선점하지 못한다. 그렇기 때문에 우선순위가 역전되는 Priority Inversion 현상이 발생할 수 있다. 표 9의 추가 데이터 셋을 통해 이 현상을 확인한다.

1.5 Round Robin과 PBRR 추가 데이터 셋

Round Robin과 PBRR 두 Scheduling 모두 TIme Quantum을 사용한다. 따라서 두 Scheduling을 비교하기 위해 표 2에서 제공한 데이터 셋에 대해 Time Quantum을 2에서 6으로 변경해가며 문맥 교환 횟수를 비교한다.

4.2. 실험 결과 및 분석

2.1. 랜덤 데이터 셋에 대한 결과

표 5의 랜덤 데이터 셋에 대한 각 알고리즘의 결과는 다음과 같다.

평균 반환시간	평균 대기 시간	평균 응답 시간
18.5	13.8	13.8

(표 10) 랜덤 데이터 셋 FCFS 동작 결과

평균 반환시간	평균 대기 시간	평균 응답 시간
12.6	7.9	7.9

(표 11) 랜덤 데이터 셋 SJF 동작 결과

평균 반환시간	평균 대기 시간	평균 응답 시간
20.5	15.8	15.8

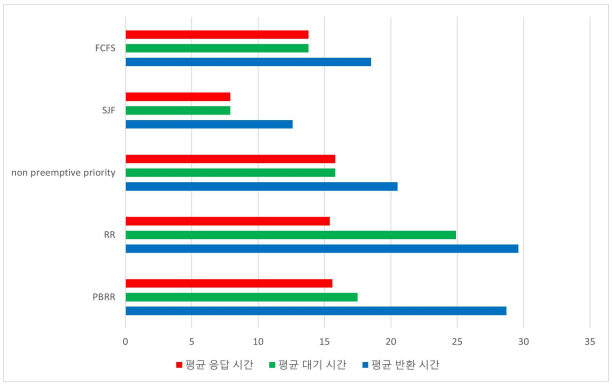
(표 12) 랜덤 데이터 셋 Non Preemptive Priority 동작 결과

평균 반환시간	평균 대기 시간	평균 응답 시간
29.6	24.9	15.4

(표 13) 랜덤 데이터 셋 Round Robin 동작 결과

평균 반환시간	평균 대기 시간	평균 응답 시간
28.7	17.5	15.6

(표 14) 랜덤 데이터 셋 PBRR 동작 결과

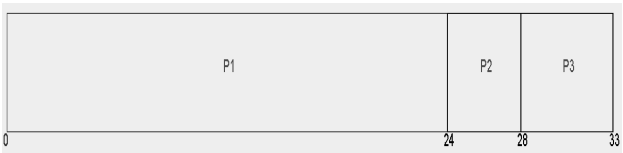


(그림 12) 랜덤 데이터 셋 결과 그래프

랜덤 데이터 셋에 대한 전체적인 결과를 그림 12를 통해 확인할 수 있다. SJF에서 평균 응답 시간, 평균 대기 시간, 평균 반환 시간이 전반적으로 짧음을 확인하였다. 또한 FCFS에서 타 알고리즘에 비해 평균 반환 시간이 높게 나왔음을 확인하였다. 이는 CPU Burst Time을 고려하지 않고 Arrival Time만을 고려하기 때문에 나타난 결과로 보인다.

랜덤 데이터 셋에서 RR에 비해 PBRR의 성능이 전반적으로 더 좋음을 확인할 수 있었다. 특히 대기 시간 측면에서 RR에 비해 PBRR의 성능이 많이 향상됨을 확인할 수 있었다.

2.2. FCFS 추가 데이터 셋에 대한 결과



(그림 14) 표 6의 데이터 셋에 대한 FCFS 동작결과

(그림 15) 표 7의 데이터 셋에 대한 FCFS 동작결과

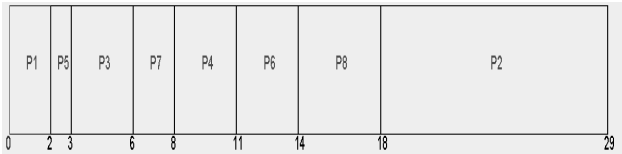
	평균 반환 시간	평균 대기 시간	평균 응답 시간
표 6의 데이터 셋	27.3	16.3	16.3
표 7의 데이터 셋	13.7	2.7	2.7

(표 15) FCFS 추가 데이터 셋 동작 결과

FCFS 성능 평가를 위해 추가적인 데이터 셋으로 실행한 결과 두 데이터 셋 사이의 평균 반환 시간, 평균 대기 시간, 평균 응답 시간이 크게 차이 나는 것을 확인할 수 있다. 두 데이터 셋의 평균 대기 시간과 평균 응답 시간은 표 15에서 확인할 수 있다. 이를 통해 FCFS는 프로세스의 입력 순서에 따라 성능 차이가 크게 나는 것을 알 수 있다.

FCFS는 그림 14와 같이 먼저 도착한 프로세스가 CPU Burst time이 매우 길면, 이후에 도착한 프로세스가 첫 번째 프로세스가 끝날 때까지 매우 긴 시간을 기다리게 된다. 이러한 효과를 호위 효과라 한다. FCFS는 호위 효과가 발생할 수 있는 정책임을 확인하였다.

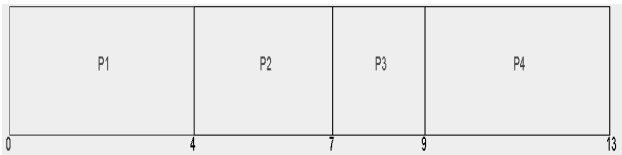
2.3. SJF 추가 데이터 셋에 대한 결과



(그림 16) 표 8의 데이터 셋에 대한 SJF 동작 결과

그림 16을 통해 프로세스 2는 1초에 도착했음에도 불구하고 CPU burst Time이 길어 18초가 되어서야 실행됨을 확인할 수 있다. 계속해서 CPU Burst time이 프로세스 2보다 짧은 프로세스가 들어오게 된다면 프로세스 2는 실행되지 못하는 기아 현상이 발생할 것이다.

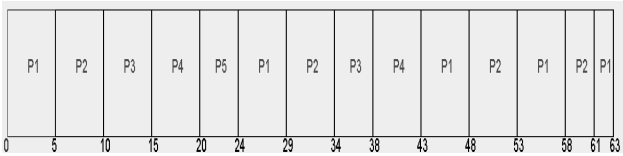
2.4. Non preemptive priority Scheduling 추가 데이터 셋에 대한 결과



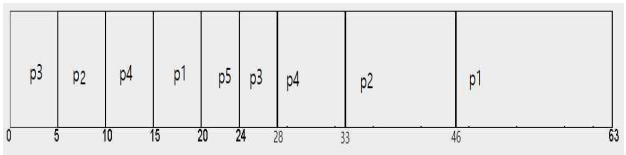
(그림 17) 표 9의 데이터 셋에 대한 Non preemptive priority Scheduling 동작 결과

그림 17을 통해 표 9에 대한 Non preemptive priority Scheduling 결과를 확인할 수 있다. 1초에 우선순위 1을 가진 프로세스 2가 들어왔지만 선점하지 못하고 프로세스 1이 다 실행되고 나서야 실행됨을 확인하였다. Non preemptive priority Scheduling에서 Priority Inversion이 발생함을 확인할 수 있었다.

2.5. Round Robin과 PBRR 추가 데이터 셋에 대한 결과



(그림 18) 표 2 데이터 셋에 대한 RR 동작 결과



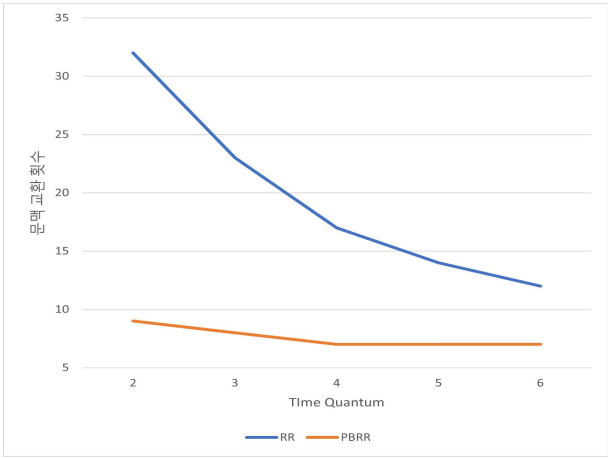
(그림 19) 표 2 데이터 셋에 대한 PBRR 동작 결과

	평균 반환 시간	평균 대기 시간	문맥 교환 횟수
RR	45.8	33.2	13
PBRR	38.8	26.2	8

(표 16) 표 2 데이터 셋에 대한 RR과 PRBB 성능 결과

그림 18과 19를 통해 추가 데이터 셋에 대한 각 알고리즘의 동작 결과를 확인할 수 있다.

표 16을 통해 본 프로젝트에서 제안하고 있는 PBRR 알고리즘에서 평균 반환 시간과 평균 대기 시간 모두에서 더 나은 성능을 보이고 있음을 확인할 수 있었다.



(그림 20) Time Quantum에 따른 각 알고리즘의 문맥교환 횟수

또한 그림 20에서 확인할 수 있듯 Time Quantum에 따른 문맥 교환 횟수가 PBRR 알고리즘에서가 RR 알고리즘보다 적음을 확인할 수 있었다.

해당 데이터 셋을 통한 비교를 통해 본 프로젝트에서 제안하고 있는 PBRR 알고리즘이 기존의 RR알 고리즘에 비해 평균 대기 시간, 평균 반환 시간, 문맥 교환 수가 적음을 확인하였다. 해당 알고리즘은 남아있는 CPU Burst Time이 적은 프로세스에 높은 우선순위가 할당되어 두 번째 라운드에서 먼저 실행되기 때문에 평균 대기 시간을 줄일 수 있었다.

V. 결론

본 프로젝트는 입력된 데이터 셋에 대해 여러 가지 Scheduling 알고리즘으로 실행하고 분석해주는 프로그램을 소개한다. 또한 기존 RR Scheduling 정책에서 프로세스의 우선순위를 고려한 새로운 CPU Scheduling 정책인 PBRR을 제안하고 그 성능을 기존 RR Scheduling 알고리즘과 비교하였다. RR 알고리즘과 비교 결과 PBRR이 평균 대기 시간 및 반환 시간 측면에서 더 나은 성능을 제공함을 입증하였다.

더불어 우선순위가 더 높은 프로세스에 CPU 시간을 공평하게 할당함으로써 실제 시나리오에서 Scheduling 문제에 대한 잠재적 솔루션을 제공할 수 있음을 시사한다. RR 알고리즘에서 발생할 수 있는 문제점인 잦은 문맥 교환을 본 알고리즘에서는 두 번째 라운드에서 남은 CPU Burst time으로 우선순위를 정하여 CPU Burst time이 끝날 때까지 실행하여 문맥 교환 횟수를 줄이도록 하였다.

또한 본 프로젝트에서 제공하는 프로그램이 CPU Scheduling 학습에 편의를 제공하고 보다 효율적이고 효과적인 Scheduling 정책 개발에 기여할 수 있기를 바란다.

본 프로젝트는 문맥 교환 overhead가 없음을 가정하고 진행되었다. 하지만 CPU scheduling은 문맥 교환 overhead의 영향을 받고 있다. 따라서 향후 진행할 프로젝트에서는 이를 반영하여 발전하는 방식으로 연구를 전개해 나갈 계획이다.

참고 자료

1) Manish Kumar Mishra1, Dr. Faizur Rashid2.2014 An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum *International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.4, No.4, August 2014*

2) Harshal Bharatkumar Parekh. Improved Round Robin CPU scheduling algorithm: Round Robin, Shortest Job First and priority algorithm coupled to increase throughput and decrease waiting time and turnaround time. 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)

3) Abdulaziz A. Alsulami. Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling. *2019 SoutheastCon*