한국사물인터넷학회논문지 Vol. 3, No. 1, pp. 21-24, 2017 https://doi.org/10.20465/KIOTS.2017.3.1.021

최적의 RR 스케줄링의 최대 할당 시간 결정

한경현[1], Hoang Thi Huyen Trang[1], 황성운[2*]
[1]홍익대학교 전자전산공학과, [2]홍익대학교 컴퓨터정보통신공학과

Determination of maximum allocation time for optimal RR scheduling

KyungHyun Han[1], Hoang Thi Huyen Trang[1], Seong Oun Hwang[2*]
[1]Department of Electronics and Computer Engineering, Graduate School of Hongik University
[2]Department of Computer and Information Communications Engineering, Hongik University

요 약 현대의 컴퓨터는 여러 프로세스를 처리해야 한다. 운영체제에서는 소수의 CPU로 많은 프로세스를 처리하기 위해서 스케줄링을 이용한다. 스케줄링의 종류에는 FCFS, SJF, RR이 있다. 이 중 RR은 최대 할당 시간을 정해야 한다. 본 논문에서는 최적의 최대 할당 시간을 찾기 위해 특정 샘플에 대해 GLM 알고리즘으로 분석하였다. 이 분석 방법을 통해 원하는 조건에 따른 최대 할당 시간을 지정할 수 있다.

주제어 : 컴퓨터 시스템, 통계

Abstract Modern computers have to deal with multiple processes. The operating system uses scheduling to handle many processes with a small number of CPUs. Types of scheduling include FCFS, SJF, and RR. Of these, RR shall determine the maximum allocation time. In this paper, we analyzed the GLM algorithm for specific samples to find the optimal maximum allocation time. This analysis method allows us to specify the maximum allocation time according to the desired conditions.

Key Words : Computer Systems, Statistics

1. 서론

현대의 컴퓨터는 항상 여러 프로세스를 처리해야 한다. 예를 들어 인터넷에 접속하는 경우, 웹 브라우저 프로세스를 처리하며 백신 프로세스로 모니터링 하고, 운영체제에서 기본적으로 항시 사용되는 프로세스도 계속 처리해야 한다. 보통 수십 개의 프로세스가 처리를 위한 CPU 작업을 요청한다. 하지만 우리가 컴퓨터를 구성할 때 사용하는 CPU는 현재 보통 4코어 형태이다. 운영체제에서는 소수의 CPU로 많은 프로세스를 처리하기 위해서 스케줄링을 이용한다. 스케줄링의 종류에는 FCFS, SJF, RR이 있다. FCFS 는 먼저 CPU를 요구하는 프로세스부터 처리하고, SJF는 빨리 끝나는 프로세스부터 처리하고, RR은 먼저 CPU를 요구하는 프로세스부터 처리하되 1회당 최대 CPU 점유 시간을 제한한다.

이 중 RR은 각 프로세스가 1회 당 CPU를 최대한 점유할 수 있는 시간인 '최대 할당 시간'을 정해야 한다. 이 시간이 작을수록 CPU가 처리 할 프로세스를 자주 바꾸기 때문에 사용자가 요구하는 프로세스를 처리하여 사용자 입력에 반응하는 속도가 빠르지만, 프로세스를 교체

*교신저자 : 황성운(sohwang@hongik.ac.kr)
접수일 2017년 2월 28일 심사완료일 2017년 3월 20일

하는 것에도 시간이 소요되기 때문에 프로세스의 평균 대기시간과 종료시간이 증가하게 된다. 반대로 이 시간이 길수록 프로세스 교체 횟수가 줄어들어 평균 대기시간과 종료시간은 감소하지만, 사용자가 요구하는 프로세스를 처리하기 전에 다른 프로세스를 처리하는 시간이 늘어나므로 사용자 입력에 반응하는 속도가 느려지게 된다. 따라서 평균 대기 시간과 종료시간이 너무 길지 않고 사용자 입력에 반응하는 시간이 짧은 최적의 할당 시간을 찾아야 한다.

2. 분석 기법

최적의 할당 시간을 찾기 위해서는 할당 시간에 따른 각 프로세스의 평균 대기시간과 모든 프로세스가 종료된 시간을 비교 분석하여 평균 대기 시간과 종료시간이 너무 길어지지 않는 범위에서 가장 짧은 할당 시간을 찾아야 한다. 따라서 할당 시간에 변화를 주면서 프로세스 세트 샘플을 시뮬레이트해서 평균 대기시간과 종료시간을 수집해야 한다. 프로세스 세트 샘플이 많을수록 좋지만 여기서는 1개의 샘플만 시뮬레이트 한 대신, 사용하는 시뮬레이터에서 프로세스가 한 번에 CPU 사용을 요청하는 시간을 랜덤하게 하고 10번씩 수행하였다. 사용한 프로세스 세트 샘플은 다음과 같다.

〈Table 1〉 Sample of a process set

	생성 시간	총 요구 시간	1회당 요구 시간
Process 1	0	200	3
Process 2	0	500	9
Process 3	0	500	20
Process 4	100	100	1
Process 5	100	500	100

이 샘플이 가지는 속성은 다음과 같다.

〈Table 2〉 Attributes of data

알고리즘	RR의 최대 할당 시간에 따라 a~k까지 표시했다. a=1, b~k는 10, 20, 30, …, 100이다. 참고용으로 FCFS와 SJF도 있다.
평균 대기시간	각 프로세스가 종료되기 위해 CPU 할당을 기다린 시간의 평균이다.
종료시간	모든 프로세스가 종료된 시간에서 모든 프로세스의 총 CPU 요구 시간을 뺀 값이다. 이것은 이상적인 경우 모든 프로세스의 총 CPU 요구 시간의 합과 같은 값이기 때문에 알고리즘에 따른 변화를 판단하기 위해서이다.

위 데이터에서 할당 시간에 따른 평균 대기시간과 종료시간의 변화를 보기 위해서 알고리즘을 X, 평균 대기시간과 종료시간을 각각 Y1, Y2로 하여 GLM으로 분석했다. 분석에 사용한 sas code는 다음과 같다.

```
proc glm data=cpu;
class algorigm;
model WT=algorigm /solution;
lsmeans algorigm /pdiff cl adjust=tukey;
means algorigm / waller;
run;
```

[Fig. 1] Code to analyze algorithm and average waiting time

```
proc glm data=cpu;
class algorigm;
model FT=algorigm /solution;
lsmeans algorigm /pdiff cl adjust=tukey;
means algorigm / waller;
run;
```

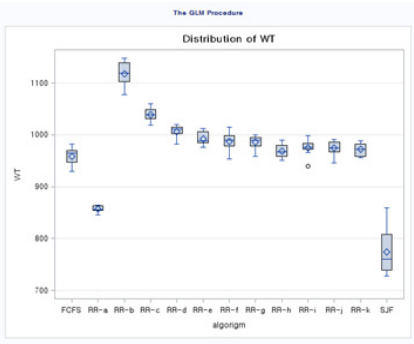
[Fig. 2] Code to analyze algorithm and finish time

3. 결과

분석한 결과는 평균 대기시간과 종료시간으로 나누어서 설명한다. 단, 각 설명에서 최대 할당 시간이 1인 것은 제외했다. 그 이유는 최대 할당 시간이 1일 경우 프로세스 교체가 매우 빈번히 일어나는데 반해 총 프로세스 교체 시간이 길어지게 되지만, 이 시뮬레이터는 프로세스 교체 시간을 계산에 넣지 않기 때문에 값을 신뢰할 수 없기 때문이다.

3.1 평균 대기시간

최대 할당 시간에 따른 평균 대기시간의 변화는 다음 그림3과 같이 나타났다. 이 그래프에서 최대 할당 시간이 길수록 평균 대기시간이 짧아지는 것을 확인할 수 있다. 하지만 사용자 입력에 대한 반응 시간을 생각하면 최대 할당 시간을 무조건 늘릴 수는 없다. 위 그래프에서 평균 대기시간은 최대 할당 시간이 짧을수록 평균 대기시간이 큰 폭으로 증가하는 것을 확인 할 수 있다. 즉, 최대 할당 시간이 증가하더라도 평균 대기시간이 비슷한 구간을 찾아 그 구간에서 최대 할당 시간이 제일 작은 것을 선택하면 된다.



[Fig. 3] Graph of average waiting time change with maximum allocation time

다음 그림4는 평균 대기시간이 비슷한 것으로 묶은 것이다. 이 표에서 평균 대기시간이 가장 짧은 것은 최대 할당 시간이 70과 100인 것이다. 하지만 비슷한 것끼리 묶었을 때 양 쪽으로 포함되는 값이 있다. 이러한 것을 합쳐서 구간을 만들면 최대 할당 시간이 30~100인 것은 평균 대기시간이 비슷하다고 할 수 있다. 따라서 최대 할당 시간을 결정하는 경우 30~100 사이에서 정하는 것이 좋고, 평균 대기시간만을 기준으로 할 경우 70으로 정하면 좋다.

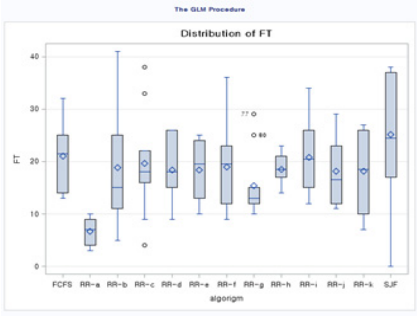
Means with the same letter are not significantly different.

Waller Grouping			Mean	N	algorigm
	A		1117.340	10	RR-b
	B		1038.840	10	RR-c
	C		1006.140	10	RR-d
	C				
D	C		992.000	10	RR-e
D					
D	E		986.620	10	RR-f
D	E				
D	E	F	985.940	10	RR-g
	E	F			
G	E	F	975.080	10	RR-i
G	E	F			
G	E	F	974.440	10	RR-j
G		F			
G	H	F	971.920	10	RR-k
G	H				
G	H		958.700	10	RR-h
	H				
	H		958.880	10	FCFS
	I		856.940	10	RR-a
	J		774.440	10	SJF

[Fig. 4] A set of maximum allocation times with similar average waiting times

3.2 종료시간

최대 할당 시간에 따른 종료시간의 변화는 다음 그림4와 같이 나타났다. 이 그래프에서 최대 할당 시간이 길수록 종료시간은 60인 경우를 제외하면 큰 변화는 없는 것을 확인할 수 있다.



[Fig. 5] Graph of average finish time change with maximum allocation time

Means with the same letter are not significantly different.

Waller Grouping		Mean	N	algorigm
	A	25.100	10	SJF
	A			
B	A	21.000	10	FCFS
B	A			
B	A	20.800	10	RR-i
B	A			
B	A	19.600	10	RR-c
B	A			
B	A	19.000	10	RR-f
B	A			
B	A	18.800	10	RR-b
B	A			
B	A	18.500	10	RR-h
B	A			
B	A	18.400	10	RR-e
B	A			
B	A	18.400	10	RR-d
B	A			
B	A	18.200	10	RR-k
B	A			
B	A	18.100	10	RR-j
B				
B		15.400	10	RR-g
	C	6.700	10	RR-a

[Fig. 6] A set of maximum allocation times with similar average finish times

다음 표4는 종료시간이 비슷한 것끼리 묶은 것이다. 이 표에서 종료시간이 가장 짧은 것은 60이긴 하지만, 모든 범위에서 종료시간이 비슷하다고 할 수 있다. 따라서 최대 할당 시간을 결정하는 경우 모든 범위에서 정할 수 있고, 종료시간만을 기준으로 할 경우 60으로 정하면 좋다.

3.3 종합

위 두 조건을 종합해보는 경우 최대 할당 시간으로 정할 수 있는 범위는 30 이상이다. 이 범위에서 평균 대기시간이 중요하면 70, 종료시간이 중요하면 60, 사용자 입력 반응 시간이 중요하면 30으로 정하면 된다. 단, 다른 조건에서의 요구치가 있다면 그에 맞는 범위를 다시 찾아서 정해야 한다.

4. 결론

이 레포트에서는 RR의 최대 할당 시간을 결정하기 위해서, 프로세스의 평균 대기시간, 종료시간, 사용자 입력 반응 시간을 고려하였다. 분석 결과 모든 관점에서 좋은 성능을 보여주는 최대 할당 시간이 없기 때문에 무엇이 가장 중요한지 생각해서 결정해야 한다.

주의할 점은 이 분석에서 사용한 프로세스 세트가 하나라는 것이다. 이 결과를 신용하기 위해서는 더 많은 샘플을 시뮬레이트한 데이터를 분석해야 한다.

ACKNOWLEDGMENTS

이 논문은 2017년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구이다(No. 2017R1A2B4001801).

REFERENCES

[1] http://terms.naver.com/entry.nhn?docId=2270386&cid=51173&categoryId=51173

[2] https://en.wikipedia.org/wiki/Generalized_linear_model

한 경 현(KyungHyun Han) [비회원]

- 2015년 2월 : 홍익대학교 컴퓨터정보통신학과 (학사)
- 2017년 2월 : 홍익대학교 전자전산공학과 (석사)
- 2017년 3월 ~ 현재 : 홍익대학교 전자전산공학과 (박사)

<관심분야>
사이버보안

Hoang Thi Huyen Trang [비회원]

- 2014년 4월 : 베트남 PTIT 전자통신학과(학사)
- 2017년 2월 : 홍익대학교 전자전산공학과 (석사)

<관심분야>
암호

황 성 운(Seong Oun Hwang) [비회원]

- 1998년 2월 : 포항공과대학교 정보통신학과 (석사)
- 2004년 8월 : 한국과학기술원 전자전산학과 (박사)
- 2006년 1월 ~ 2006년 12월 : University of Michigan 박사후 연구원
- 2008년 3월 ~ 현재 : 홍익대학교 컴퓨터정보통신학과 부교수

<관심분야>
정보보호, 암호