CPUScheduler\_V1 (JAVA ver.) Documentation

1. Introduction

CPUScheduler\_V1은 실시간 시스템에서의 CPU 실시간 스케줄링 알고리즘(real-time scheduling)과 실시간성 분석 기법(schedulability analysis)을 구현하고 성능을 확인할 수 있는 JAVA 기반의 시뮬레이터이다. 현재의 소스코드는 uniprocessor에서 동작하는 RM (rate monotonic) 알고리즘이 구현 되어 있으며 실시간성 분석 기법들은 제외되어 있는 상태이다. 수업의 진행을 통해 여러 종류의 알고리즘과 실시간성 분석 기법을 구현하고 이들을 multiprocessor 기반으로 확장하는 것을 목표로 한다.

1. Overall operation

CPUScheduler\_V1은 실험을 위한 1) task set 생성, 2) 실시간 스케줄링 알고리즘 실행, 3) 실시간성 분석 실행(V1에서는 포함되어 있지 않음), 4) 성능 비교 및 검증 5) 그래프 생성(V1에서는 포함되어 있지 않음)의 운영이 순차적으로 진행된다.

1. How to use
   1. task set 생성
2. 본인 task set을 생성하고자 하는 폴더를 만든다. 예를 들어 C:/CPUScheduler/inputFiles 라면, CPUScheduler와 그 하위 폴더인 inputFiles를 만든다.
3. GlobalData.java 파일의 inputFileFolder 변수 값을 생성한 폴더의 경로와 일치시킨다. E.g., public static final String inputFileFolder="C:/CPUScheduler/inputFiles";
4. \_Gen\_Set.java 파일을 실행 시킨다.

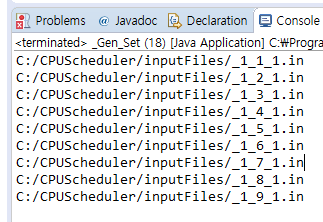
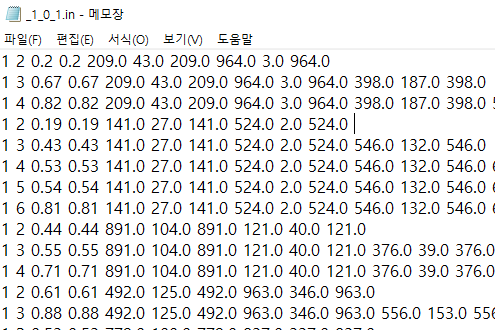
 

Figure 1 Console result Figure 2 resulted input files

* 1. 스케줄링 알고리즘 실행

1. 본인 스케줄링 알고리즘 실행 결과 파일을 생성하고자 하는 폴더를 만든다. 예를 들어 C:/CPUScheduler/outputFiles 라면, CPUScheduler와 그 하위 폴더인 outputFiles를 만든다.
2. GlobalData.java 파일의 outputFileFolder 변수 값을 생성한 폴더의 경로와 일치시킨다. E.g., public static final String outputFileFolder="C:/CPUScheduler/outputFiles";
3. \_Exec\_Set.java 파일을 실행 시킨다.

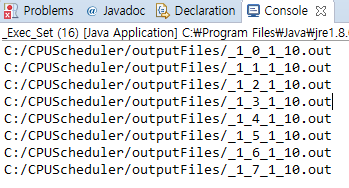
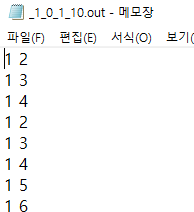
 

Figure 3 console result Figure 4 resulted output files

* 1. 성능 평가

1. \_\_ResultCompare.java 를 실행 시킨다.

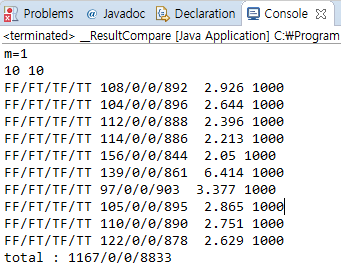


Figure 5 console result

1. 두 알고리즘(혹은 분석기법)의 결과를 비교한다. FF는 둘다 실패(무슨 뜻인지는 수업시간에 설명)한 task set의 개수, FT는 뒷 알고리즘만 성공, TF는 앞 알고리즘만 성공, TT는 둘다 성공한 task set의 개수이다.
2. Source files description
3. \_Gen\_Set.java : task set 을 생성하는 executable 파일. Random seed와 distribution을 이용하여 한 파일당 1000개의 task set을 생성한다. Maximum 100,000개((max seed=10)\*(max distribution=10)\*1000)의 task set이 생성가능하다.
4. \_Exec\_Set.java : 생성된 input 파일의 정보를 읽어 들여 task set 정보를 추출한 후 simulator를 호출한다. 스케줄링 알고리즘 및 실시간성 분석 기법을 호출하는 executable 파일.
5. \_\_ResultCompare : 실행 결과 비교를 통해 성능 분석을 하는 executable 파일. 주어진 task set 중 몇개의 task set이 deadline miss 없이 실행을 끝마쳤는지(혹은 끝마칠 수 있는지) 그 개수를 count 하여 성능을 비교한다.
6. Simulator.java : \_Exec\_Set.java 에 의해 호출되어, 실제로 해당 알고리즘을 0초부터 min(task set들의 period의 최소 공배수, 100,000)초 까지 실행하여 deadline miss 가 없는지 매 초마다 확인한다.
7. GlobalData.java : task set generation시 사용되는 seed의 개수, distribution의 개수, processor의 개수, intput, output folder의 path를 세팅한다. 이 파일에서 세팅된 값은 전체java파일에서 동일하게 사용된다.
8. PlatformInfo.java : 스케줄링에 필요한 platform관련 전반적인 정보를 담고 있다.
9. Task.java : task 의 T, C, D에 대한 정보를 담고 있다.
10. Job.java : task가 T마다 생성하는 하나의 instance(job)에 대한 정보를 담고 있다.
11. Scheduler.java : interface 파일로서 해당 class를 상속받은(implement keyword) class는 반드시 schedule function을 구현해야 한다.
12. Algorithm\_RM.java : scheduler class를 상속받아 schedule 함수를 구현한다. Schedule 함수안에서 Comparators.RMComparator()를 호출하여 readyQueue에 있는 job들을 RM으로 scheduling한다.
13. Comparators.java : 여러 comparator가 구현되어 있고(현재는 RM만 구현되어 있음) scheduler class를 상속받은 class들이 schedule 함수안에서 자신에게 해당되는 comparator()를 호출한다.
14. LCM.java : task들의 T들의 최소 공배수를 만들때 사용된다.