논문 2012-50-3-9

시간가치와 중요도 기반의 분류 작업 스케쥴링

(Time Value and Importance based Classified Job Scheduling)

심 정 연*

(JeongYon Shim)

요 약

정보가 폭주하고 해야할 일들이 많은 상황에서 바쁜 현대인에게 있어서는 효율적인 에너지와 시간관리가 매우 중요한 일로 여겨지고 있다. 이미 효율적인 시간관리 스케쥴링 방식은 성공적인 삶을 살기 위한 매우 중요한 요소가 되었기 때문에 사건중심의 기존의 단순한 스케쥴링 방식보다 더 정교하게 이러한 기능을 지원해 줄 수 있는 지능적 시스템이 필요하다. 따라서 본 논문에서는 작업을 시간가치와 중요도에 따라 분류하고 이를 효율적으로 시간 프레임에 할당할 수 있는 분류 작업 스케쥴링 방법을 제안하고자 한다. 특히 시간가치가 높은 작업을 우선적으로 처리하고 동시에 시간가치는 높지 않으나 미래의 성공의 열쇠를 쥐고 있는 매우 중요한 일들을 꾸준히 수행할 수 있도록 시간 프레임에 일정부분을 고정 할당하는 새로운 방법을 제안한다.

Abstract

In the information flooded situation having many thing to do, the efficient scheduling of energy and time is regarded as a very important thing for the busy modern people. Because the efficient time scheduling already became a very important key factor for a successful life, the more elaborately designed intelligent system than the previous simple event oriented scheduling system should be supported. Accordingly in this paper we propose Classified Job Scheduling System in which jobs are classified by Time value & Importance and allocated to Time frame efficiently. Especially the new method is proposed with the policy that processes the job with high time value first and concurrently allocates the job of low time vale & high importance to the preempted fixed area of time frame for continuous processing of job which has a key of success in the future.

Keywords : 작업 스케쥴링, 작업 분류, 시간가치와 중요도 기준, FIFO, SJF, Priority

Ⅰ. 서 론

정보 커뮤니케이션 속도가 빨라짐에 따라 현대인들은 정보의 폭주와 처리해야 할 많은 일들 속에서 바쁜 일상을 살아가고 있다. 정보의 흐름이 빠르기 때문에 업무처리 역시 신속함을 요구하는 것이 많으며 적합한 시간에 정확히 처리를 해주어야 한다.

정보화 사회의 중심은 정보이다. 이러한 정보의 운용이 미래의 성패를 좌우하는 경우가 많고 시간 스케쥴링이 잘못되면 귀중한 정보를 잃어버릴 수도 있다. 현시대는 한정된 에너지와 시간 자원 속에서 많은 정보들 중 중요하고 필요한 일들을 순발력 있고 지혜롭게 선택할 수 있는 능력과 이를 효율적으로 스케쥴링할 수 있는 능력을 절실히 필요로 한다.

그동안 시간 스케쥴링할 수 있는 방법이 많이 나와 있으나 일반적으로 업무나 회의등과 같은 사건중심으로 간단하게 스케쥴링하는 것이 대부분이고 시간가치와 중요도를 모두 고려한 정밀한 스케쥴링 방법이 부재한 상황이다. 본 연구에서는 새로운 스케쥴링 방식 연구를 위해서 우리 일상의 상황과는 여러 조건 면에서 차이는 있지만 CPU 스케쥴링 기법을 참고하려고 한다. 큰 차이점이라고 한다면 CPU는 쉬지 않고 끊임없이 일할 수 있다는 것, 시간적 마감시간에 크게 구애받지 않는다는

* 정회원, 강남대학교 교양학부
(Division of General Studies, Kangnam University)
접수일자:2013년1월13일, 수정완료일:2013년2월27일

것, 인터럽트 사건이 많지 않다는 것, 외부 조건이 비교적 단순하다는 것이다. 이에 비해서 인간의 일상 스케쥴링에서는 예기치 못할 상황이 끊임없이 벌어지며 시간의 긴박성, 일의 중요도, 현재 상황, 감정적 요소, 가치관 등 많은 다른 요소들이 큰 영향을 미친다. 이에 따라 일의 우선순위가 완전히 뒤바뀌기도 한다. 보다 성공적인 삶을 위해서는 인간의 시간, 에너지 자원을 효율적으로 활용해야한다. 지속적으로 발생되는 일들을 효율적으로 관리하여 시간을 확보하면 잡다한 일들에 끌려다니는 것이 아니라 확보된 시간 속에서 행복과 열정을 쏟을 수 있는 인간적이고 창조적인 작업에 집중할 수 있다. 만일 우리 일상 작업의 스케쥴링을 도와 줄 수 있는 지능적 시스템이 있다면 많은 도움을 받을 수 있을 것이다.

따라서 본 논문에서는 작업을 시간가치와 중요도에 따라 분류하고 이를 스케쥴링할 수 있는 정밀하고 지능적인 분류 작업 스케쥴링 기법을 제안하고자 한다. 이 시스템의 큰 특징은 시간가치가 큰 작업을 우선적으로 처리하는 것과 시간적으로 급하지는 않지만 미래의 성공의 열쇠를 가지고 있는 중요한 일들을 항상 할 수 있도록 시간 프레임에 IBS(Importance Based Slot)라는 일정한 공간을 확보하여 그 작업을 꾸준히 수행할 수 있도록 하는데 있다.

Ⅱ. 작업 스케쥴링

이 장에서는 일반적으로 CPU 에서 사용되고 있는 작업 스케쥴링 방식을 고찰하려고 한다. 시스템 특성에 따라 여러 가지 방식이 있으나 여기에서는 가장 일반적인 방식인 FIFO, SJF, 우선순위(Priority) 스케쥴링에 대해알아보려고 한다[1].

1. FIFO(First Come First Out)

FIFO는 작업을 도착하는 순서대로 처리하는 비선점형 스케쥴링 알고리즘이며 FCFS(First Come First Service) 스케쥴링이라고도 한다. FIFO큐를 사용하기 때문에 구현하기 쉽고 대부분의 일괄처리 시스템에서는 훌륭한 기법이다. 그러나 새로운 작업이 들어오면 작업의 마지막 위치에 연결되기 때문에 대화식 처리에서는 사용자가 많이 기다리게 되므로 적합하지 않다.

2. SJF(Shortest Job First)

이 기법은 현재 준비상태에 있는 프로세스들 중에서 총 실행시간이 가장 짧은 프로세스부터 스케쥴링하는 기법으로 비선점 정책에 근거하고 있다. 평균 대기시간을 최소화할 수 있으나 실행시간이 긴 프로세스들이 무한정 대기하게 되는 단점을 가지고 있다.

3. 우선순위(Priority) 스케쥴링

비선점형 스케쥴링 기법 중의 하나로 일괄처리시스템에서 중요한 작업에게 우선권을 부여하는 기법이다. 중요도가 높은 작업을 가장 먼저 처리하고 한번 처리가 시작되면 자연대기가 될 때 까지 인터럽트를 받지 않는다.

Ⅲ. 시간과 중요도를 고려한 분류 작업 스케쥴링 메커니즘

1. 작업 분류와 정량화

본 연구에서는 효율적인 작업 스케쥴링을 위해서 시간과 중요도에 따라 작업을 몇 가지 그룹으로 분류하고 이 그룹에 따라 다른 메커니즘을 적용하여 시간 프레임에 작업을 할당하였다. 그림 1에 보인 바와 같이 시간 중요도 공간을 디자인하여 작업을 시간과 중요도 공간에서 4 부분으로 나누고 정밀한 스케쥴링을 위하여 이를 세분화하였다.

시간 중요도 공간에서 1사분면은 시간 가치와 중요도 가치가 높은 영역으로 반드시 처리를 해야만 하는 작업이 이에 해당된다. 이 영역을 두 부분으로 나누었는데 A 영역은 급하게 반드시 처리해야하는 일이다. 시간 내에 처리하지 않았을 경우 큰 차질이 빚어질 수 있는 중요한 일이다.

2사분면은 시간가치는 높으나 중요도는 높지 않은 영역이다. 다시 말하면 중요도는 높지 않으나 시간적으로 급하게 처리해야만 하는 작업을 의미한다. 이 영역 역시 B와 C의 영역으로 나누는데 B는 중요도는 높지 않으나 시간적으로 급하게 처리할 수 밖에 없는 작업들이다. 2사분면의 C 영역은 중요도는 높지 않으나 시간가치가 다소 높은 경우에 해당된다. 3사분면의 D 영역은 시간가치도 낮고 중요도도 낮은 작업이다. 이 영역에 해당되는 작업은 과감하게 폐기한다. 하지 않아도 되는 작업이기 때문에 프로세스에 과부하와 비능률성만 야기할 뿐이다. 4사분면의 영역은 중요도는 높으나 시간가치는 높지 않은 작업들을 의미한다. 급하지는 않으나 장기적으로 볼 때 준비하고 꼭 수행해야 할 아주 중요한 일들이다. 당장 현재에 큰 영향을 미치지는

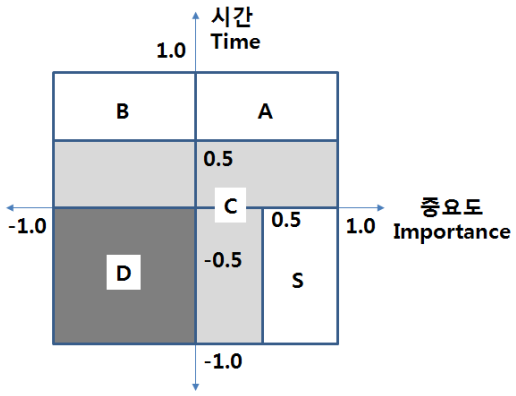

그림 1. 시간과 중요도 공간
Fig. 1. The space of Time and Importance.

표 1. 영역분류
Table 1. Domain classification.

영역	범위
O영역	$T_i = 1.0$
A영역	$0.5 \le T_i < 1.0\ AND\ 0.0 \le I_i \le 1.0$
B영역	$0.5 \le T_i < 1.0\ AND\ -1.0 \le I_i \le 0.0$
C영역	$0.0 \le T_i < 0.5\ AND\ 0.0 \le I_i \le 1.0$, $0.0 \le T_i < 0.5\ AND\ -1.0 \le I_i \le 0.0$ $-1.0 \le T_i < 0.0\ AND\ 0.0 \le I_i \le 0.5$
D영역	$-1.0 \le T_i < 0.0\ AND\ -1.0 \le I_i \le 0.0$
S영역	$-1.0 \le T_i < 0.0\ AND\ 0.5 \le I_i \le 1.0$

않지만 미래 성공의 열쇠를 쥐고 있는 작업들이 해당된다. 4사분면은 S영역과 C 영역으로 구분된다. S 영역의 작업은 미래를 위해서 특별 관리되는 꼭 해야 하는 일이다. C 영역은 중요하지만 시간적으로 여유가 있는 일들이며 1사분면, 2사분면의 C영역과 함께 일괄처리(Batch Processing)한다.

시간과 중요도 공간에서 보인 바와 같이 본 연구에서는 작업을 6가지 영역으로 분류하였다. 이를 명시하면 다음과 같다. 여기서 시간가치는 T_i라 표기하고 [-1.0,1.0]의 값을 갖고 중요도는 I_i라 표기하고 역시 [-1.0, 1.0]의 범위의 값을 갖도록 하였다.

O 영역은 $T_i = 1.0$의 절대적인 시간가치를 가지는 영역으로 절대적 우선순위를 가지고 처리되며 $T_i = 1.0$인 여러 작업이 있는 경우에는 중요도에 따라 정렬하여 처리한다.

A와B 영역에서는 시간 중요도 통합가치를 계산하여 값이 높은 순으로 정렬하여 각각 처리한다.

C영역의 작업들은 서로간의 거리를 계산하여 유사도가 높고 거리가 가까운 유사한 작업끼리 묶어 일괄처리(Batch Processing)하고 D영역의 작업들은 작업 분류 단계에서 사전에 폐기한다. S영역은 시간 프레임(Time Frame)내에 특별 영역을 두어 특별 관리한다.

1. 분류 작업 스케쥴링 시스템 구조

이 시스템의 특성은 모든 작업에 같은 규칙을 적용하여 처리하는 것이 아니라 시간과 중요도에 따라 작업을 영역으로 분류하고 그 영역별 특성에 따라 다른 처리를 한다는데 있다.

다음 그림 2는 작업분류에 따라 시간 프레임을 할당하는 작업 스케쥴링 시스템의 구조를 보이고 있다. 일정한 크기를 가지는 시간 프레임(Time Frame), TF_i이 중심이 되어 시간프레임 단위로 처리된다. I/O인터페이스를 통해 들어온 작업은 작업 분류 모듈에 의해서 6가지 영역으로 분류된다. O,A,B 영역은 T_JOB QUEUE에 정렬되어 TBS(Time Based Slot)에 할당되어 차례로 처리된다. C영역에 해당되는 작업들은 일괄처리하여, 즉 유사도를 측정하여 비슷한 작업끼리 모아 정렬하여 T_JOB QUEUE로 보내 처리된다. S 영역의 작업들은 I_JOB QUEUE로 보내 정렬한 후 IBS (Importance Based Slot)에 할당된다. 그림 3은 원형 시간 프레임 TF_i의 준비 QUEUE를 보이고 있는데 원형으로 구성된 큐이며 큐의 Head와 Tail이 연결되어 있다. 작업의 연결성을 위해 원형으로 디자인하였다.

2. 작업분류에 따른 시간 프레임 할당 방식

이 절에서는 작업분류에 따른 영역별 시간 프레임 방식을 제안한다. 다음은 I/O 인터페이스에 입력되는 작업 JOB_i의 구조를 보이고 있다.

$$(ID_i, T_i, D_i, I_i, W_i, L_{x_i}, L_{y_i}, V_i)$$

ID_i는 작업의 식별자이고 T_i는 시간가치(Time Value)이고, D_i는 마감시간(Deadline), I_i는 중요도 값(Importance degree)을 의미하며 T_i와 I_i는 모두 [-1.0, 1.0]의 값을 갖는다. W_i는 작업량을 나타내며 $W_i \ge 0.0$인 값을 갖는다. L_{x_i}, L_{y_i}는 현재 작업이 진행되고 있는 개념적 위치이며 이 값은 일괄처리 과정에서 유사도 측정에 사용된다. V_i는 시간중요도 통합 가치이다. JOB_i의 구조를 나타내면 다음과 같다.

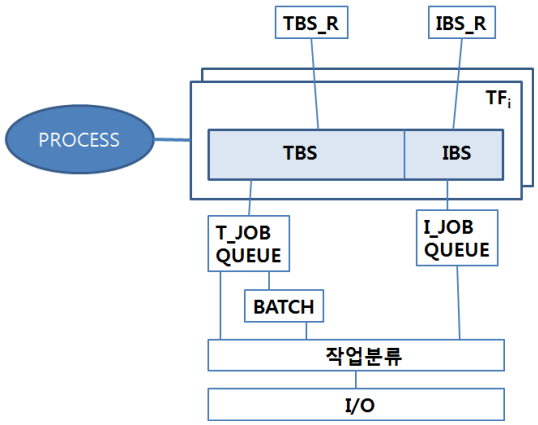

그림 2. 분류 작업 스케쥴링 시스템
Fig. 2. Classified Job Scheduling System.

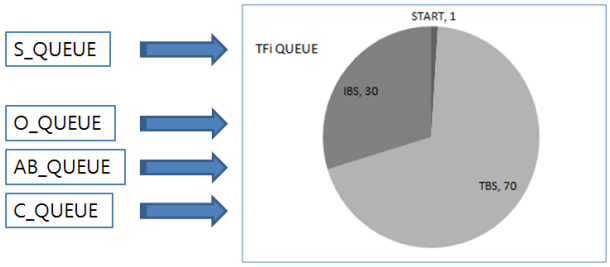

그림 3. 원형 시간 프레임 TF_i와 준비 QUEUE
Fig. 3. Circular Time Frame TF_i and Ready QUEUE.

```
struct Job[i]{
  ID;
  time;
  dline;
  importance;
  work;
  locx;
  locy;
  v;
}
```

본 시스템에서는 시간중요도 통합 가치, V_i, 를 시간과 중요도를 고려한 작업의 가치로 정의하고 이 값을 작업의 할당과 순서를 정하는 척도로 규정하였다. 시간중요도 통합가치는 다음 식 (1)과 같이 영역별 조건과 상황을 고려하여 영역별로 다른 계산법으로 구한다.

시간중요도 통합 가치 V_i 계산

CASE1(O영역) :

$$V_i = I_i$$

CASE2(A영역) :

$$V_i = \frac{T_i + 2.0 * I_i}{3.0}$$

CASE3(B영역)

$$V_i = \frac{2.0 * T_i + |I_i|}{3.0}$$

CASE4(C영역)

$$V_i = \frac{|T_i| + |I_i|}{2.0}$$

CASE5(D영역)

$$V_i = 0.0$$

CASE6(S영역)

$$V_i = \frac{|T_i| + I_i}{2.0} \quad (1)$$

3. 분류 작업 스케쥴링 알고리즘

본 절에서는 주어진 시간 프레임에 작업을 효율적으로 할당할 수 있는 분류 작업 스케쥴링 알고리즘을 제안한다. 분류 작업 스케쥴링 알고리즘은 다음과 같다.

ALGORITHM 1 : 분류 작업 스케쥴링 알고리즘

```
STEP 1 : Initialize, s_TBS=m, s_IBS=k
STEP 2 : While(Not EOF) Do {
STEP 3 : INPUT Job[i];
STEP 4 : If(Job[i]_dline == TF[k]_id)
           Job[i]_time=1.0;
STEP 5 : If((Job[i]_time <= 0.0) AND
           (Job[i]_importance <= 0.0))
         then Job[i]=0.0; /* Discard
         Else{
STEP 6 : CLASSIFY by table(1)
STEP 7 : Calculate Job[i]_v by Eq.(1)}
STEP 8 : MAKE_QUEUE_O
STEP 9 : MAKE_QUEUE_AB
STEP 10 : MAKE_QUEUE_C
           Batch process
STEP 11 : MAKE_QUEUE_S}
STEP 12 : ALLOCATION
STEP 13 : STOP

ALLOCATION
STEP 1 : get ASK;
STEP 2 : If (ASK != C)
```

```
            Then OABS_ALLOCATION;
            Else {
            Allocate Job[i] from QUEUE_C
                  to TBS;
            Allocate Job[i] from QUEUE_S
                  to IBS;}
STEP 3 : RETURN

OABS_ALLOCATION
STEP 1: While ( not EOF AND TF !=FULL) Do {
          Allocate Job[i] from QUEUE_O to TBS;
STEP 2 : If (TBS== FULL)
            Then {get  ASK1;
                  If (ASK1 == 'y')
                   Allocate Job[i] from QUEUE_O
                   to IBS;
                   head_ptr2=tail_ptr2;}
                 Else {head_ptr1=tail_ptr1;
                  Allocate Job[i] from QUEUE_AB
                   to TBS;}
STEP 3 :Allocate Job[i] from QUEUE_S to IBS;}
STEP 4 : RETURN;
```

MAKE_QUEUE_O,MAKE_QUEUE_AB,MAKE_QUEUE_S 모듈에서 큐를 생성하는 방법은 Job[i]의 V_i 값을 기준으로 내림차순으로 배치한다. 그러나 MAKE_QUEUE_C큐의 생성은 위치정보를 이용하여 유사도를 측정하여 비슷한 작업을 묶는 형식으로 진행한다. 현시점을 기준으로 식 (2)와 같은 유사도를 계산하여 가장 유사도가 높은 작업을 선택하여 큐에 차례로 넣는다. 선택되는 현재 작업 i를 기준으로 하였을 때 선택되는 작업 q는 식(3)을 이용하여 구한다.

$$U_{ij} = -\sqrt{(L_{x_i} - L_{x_j})^2 + (L_{y_i} - L_{y_j})^2} \quad (2)$$

$$q = \arg\max_j U_{ij} \quad (3)$$

Ⅳ. 실 험

본 실험에서는 그림 4에서 제시한 바와 같이 20개의 작업을 가지고 분류작업 스케쥴링 기법을 실험하였다. 여기서 Job_Idi는 작업의 식별자이고 Ti는 시간가치, Di는 마감시간 값, Ii는 중요도, Wi는 작업량, Lxi, Lyi는 작업의 개념적 거리를 의미한다. 시간 프레임의 크기는 100으로 놓고 TBS의 크기, 즉 m=70, IBS의 크기 k=30으로 정하였다. Ti=1.0 에 대해서 임계치 I_threshold의 값이 각각 -1.0, 0.0, 0.5, 0.9 일 때 원형 시간 프레임에서 진행되는 작업 큐의 변화를 분석하였다. 임계치 I_threshold는 Ti=1.0인 작업 즉 시간가치가 절대적이어서 꼭 수행하여야하는 작업들을 할당할 때 중요도에 따라 할당 수위를 조정하는 값을 의미한다. 임계치 I_threshold 값이 낮을수록 많은 Ti=1.0인 작업을 수행할 수 있다.

Job_Idi	Ti	Di	Ii	Wi	Lxi	Lyi
1	1.00	1	0.41	20	1	3
2	0.91	30	-0.32	15	1	2
3	-1.00	15	-1.00	5	2	5
4	0.83	30	-0.50	3	-3	2
5	0.32	30	0.43	2	-2	-5
6	0.43	20	-0.14	5	3	4
7	1.00	1	-0.21	30	7	1
8	-0.27	10	0.73	5	5	-3
9	0.73	7	0.94	30	-1	4
10	-0.62	5	0.31	10	-3	-2
11	1.00	1	0.53	10	-2	5
12	-0.74	10	0.92	5	4	8
13	0.85	25	0.87	15	9	3
14	1.00	1	0.94	10	3	-7
15	-1.00	12	1.00	10	-5	-6
16	0.25	20	-0.74	5	6	3
17	-0.34	17	0.27	3	2	4
18	1.00	1	0.85	10	5	3
19	-0.12	5	0.62	2	2	8
20	-0.52	8	-0.42	5	1	7

그림 4. 입력된 작업 Job[i]
Fig. 4. Input Job Job[i].

O_QUEUE

Job_id	14	18	11	1	7
Wi	10	10	10	20	30

AB_QUEUE

Job_id	9	13	4	2
Wi	30	15	3	15

C_QUEUE

Job_id	10	5	17	6	16
Wi	10	2	3	5	5

S_QUEUE

Job_id	15	12	8	19
Wi	10	5	5	2

그림 5. 분류 작업 스케쥴링 알고리즘에 의해 생성된 영역별 QUEUE
Fig. 5. The Domain QUEUE generated by Classified Job Scheduling Algorithm.

그림 5는 제안된 분류 작업 스케쥴링에 의해서 생성된 O영역의 작업들로 구성된 O_QUEUE, A와,B의 영

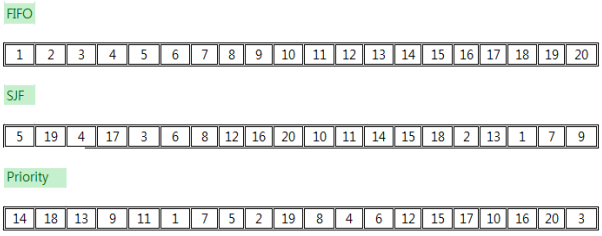

그림 6. FIFO, SJF, Priority 작업스케쥴링에 의해 생성된 QUEUE

Fig. 6. The QUEUE generated by FIFO, SJF, Priority Job Scheduling.

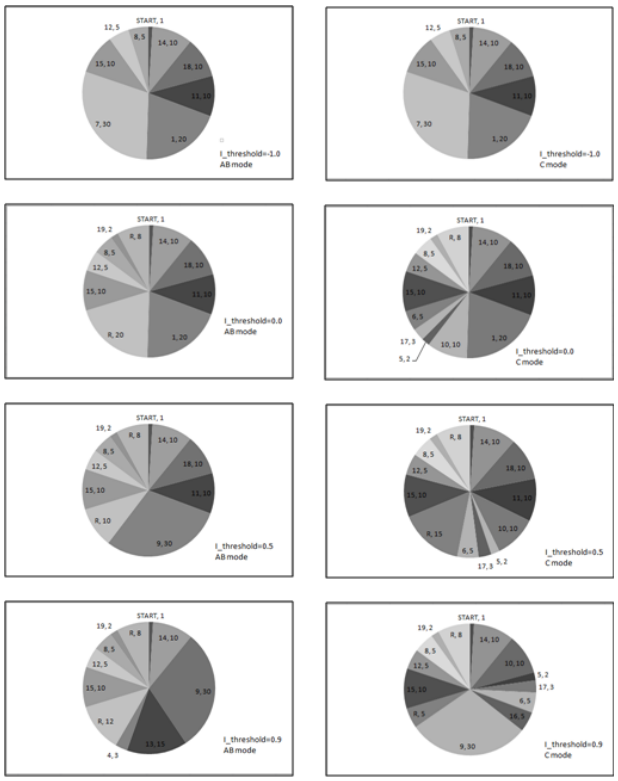

그림 7. I_threshold= -1.0, I_threshold= 0.0, I_threshold= 0.5, I_threshold= 0.9 일 때 각각 AB mode, C mode 에 대한 원형 시간 프레임의 변화

Fig. 7. The change of Circular Time Frame on AB mode and C mode, in the case of I_threshold = -1.0, I_theshold=0.0, I_threshold=0.5, I_threshold = 0.9.

역의 작업들로 구성된 AB_QUEUE, C의 영역의 C_QUEUE, S영역의 S_QUEUE를 나타내고 있다.

다음 그림7은 I_threshold의 값이 각각 -1.0, 0.0, 0.5, 0.9 일 때 원형 시간 프레임에서 진행되는 작업 큐의 변화이다. I_threshold==-1.0인 경우에는 Ti=1.0인 작업 14, 18, 11, 1, 7이 모두 다 할당되었다. 모두 작업량이 80으로 TBS 프레임 70과 IBS 프레임 10이 할당되고 IBS프레임의 나머지부분은 정책대로 S영역의 작업이

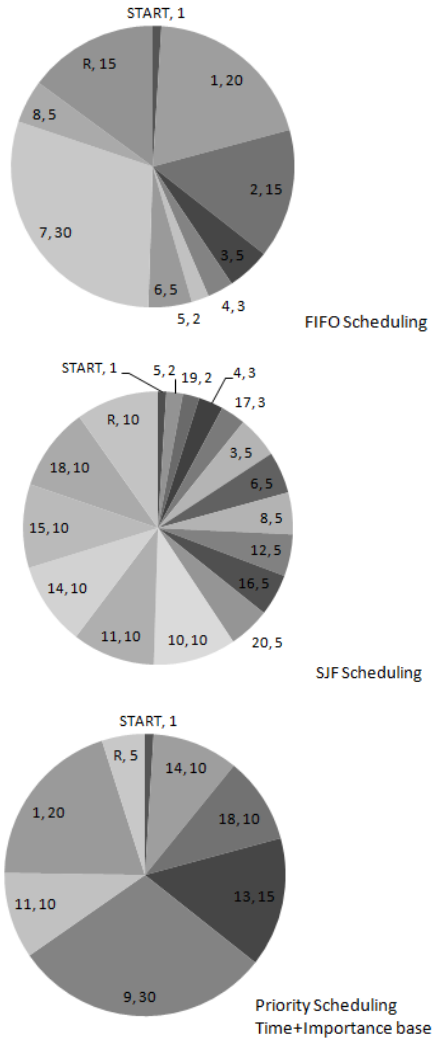

그림 8. FIFO, SJF, Priority 스케쥴링에 대한 원형 시간 프레임의 변화

Fig. 8. The change of Circular Time Frame by FIFO, SJF, Priority Scheduling.

할당되었다. 이 알고리즘에서는 TBS 프레임과 IBS 프레임의 크기를 고정하고 있지만 Ti=1.0인 작업인 경우에만 시간적 가치의 중요성을 고려하여 예외로 하고 있다. 일반적으로 TBS프레임의 할당은 I_threshold에 해당되는 Ti=1.0 인 작업을 우선적으로 할당하고 나머지 영역에 AB 영역의 작업이나 C영역 작업을, IBS 프레임에는 S영역의 작업을 정책에 따라 할당한다. 그림 상에서 표기된 R의 의미는 할당되지 못하고 남은 공간을 의미한다. 실험 결과에서 알 수 있듯이 제안된 분류 작업 스케쥴링 기법은 시간적으로 급하지는 않지만 중요한 일들을 일정 프레임을 확보하여 반드시 수행할 수 있도록 한다는 것과 시간가치가 높은 작업을 우선적으로 처리한다는 것이 특징이다.

그림 6과 그림8은 각각 FIFO 스케쥴링, SJF 스케쥴

링, Priority 스케쥴링의 작업 큐와 할당 결과를 보여주는 것으로 이와 비교해 보면 제안된 방법의 두 가지 특성을 쉽게 확인 할 수 있다. 들어오는 순서대로 처리 하는 FIFO 스케쥴링의 경우 처리되어야 할 작업인 O영역의 11,14,18과 S영역의 12,15,19 작업이 할당되지 못하였고 오히려 폐기되어야 할 작업 3이 할당되었다. SJF 스케쥴링의 경우에는 적은 작업량을 먼저 처리하기 때문에 많은 작업을 처리할 수 있으나 시간적으로 중요한 작업인 O영역의 1과 7의 작업이 할당되지 못하였고 폐기되어야 할 작업 20이 할당되었다. 우선순위(Priority) 스케쥴링에서는 다행스럽게도 폐기되어야 할 작업을 포함하고 있지는 않지만 O영역의 7과 S영역의 19,8,12,15의 작업이 할당받지 못하였다. 실험결과에서 알 수 있듯이 우리 일상의 작업관리는 CPU처리 스케쥴링을 위한 방법들, 즉 FIFO, SJF, 우선순위 스케쥴링보다 유연성과 편리함, 효율성 면에서 더 적합하고 지능적임을 알 수 있다.

V. 결 론

작업을 시간가치와 중요도에 따라 분류하고 이를 효율적으로 시간 프레임에 할당할 수 있는 분류 작업 스케쥴링 방법을 제안하였다. 특히 시간가치가 높은 작업을 우선적으로 처리하고 동시에 시간가치는 높지 않으나 미래의 성공의 열쇠를 쥐고 있는 매우 중요한 일들을 꾸준히 수행할 수 있도록 시간 프레임에 일정부분을 고정 할당하는 새로운 방법을 제안하였다.

20개의 작업을 가지고 제안된 방법을 실험하였으며 FIFO나 SJF, 우선순위 스케쥴링 방식에 비해서 우수한 특성을 나타냄을 입증하였다. 본 논문에서 제시한 시스템은 효율적 시간 관리를 위한 보조 시스템으로서 사용되거나 아니면 자동화된 지능시스템을 구현하는데 많은 기여를 할 수 있으리라 기대한다.

참 고 문 헌

[1] 엄영익,정태명, “컴퓨터운영체제론”, 생능출판사, 1999
[2] JeongYon Shim, Knowledge Network Management System with medicine Self Repairing strategy} ICES2007, LNCS4684, pp119-128, Springer Verlag, Berlin Heidelberg 2007.
[3] Bruce Goldstein, “Sensation and Perception”, Fifth edition, Brooks/Cole Publishing Company, 1999.
[4] Michael A. Arbib, Jeffrey S. Grethe, “ Computing the brain : A guide to Neuroinformatics”, Academic Press, 2001.
{5} Abraham Silberschatz, “Operating System Concepts with JAVA”, Wiley, 2010.
{6} Technical committee on Operating Systems and Operating Environments, “Information Technology porrtable Operating System Interface”, Ins of Elect & Electronic, 1990.
[7] Wei, Andreas, “Operating System Services for Task-Specific Power management-Novel Approach to Energy-Aware Embedded Linux”, VDM Verlag, 2007.
[8] “Job Scheduling Strategies for parallel Processing: 13th International Workshop Jsspp 2007”, Seattle, Wa, USA, June 17, 2007, Springer.
[9] Gawiejnowicz, Stanislew, “Time-Dependent Scheduling”, Springer, 2008.
[10] Neumann.Klaus, Schwindt christoph, Zimmermann Jirgen, “Project Scheduling with Time windows and Scare Resources”, Springer, 2003.
[11] Stankovic John A, Spuri Marco, Ramamritham Krithi, “Deadline Scheduling for Real Time Systems”, Springer, 1998.
[12] 심정연, 개인성을 고려한 지식 감정반응모델의 설계, 대한전자공학회논문지, 47권 1호, pp116-122, 2010.

저 자 소 개

심 정 연(정회원)
1989년 고려대학교 컴퓨터학과 학사 졸업
1991년 고려대학교 컴퓨터학과 석사 졸업
1998년 고려대학교 컴퓨터학과 박사 졸업.
현 강남대학교 부교수
<주관심분야: 인공지능, 기계학습, 감성시스템 ICA, Information Theory, 지식공학시스템>