IBM LoadLeveler의 CPU 수에 따른 작업 우선 순위 부여

이영주, 최재민, 김성준, 우준
한국과학기술정보연구원
e-mail:yjlee@kisti.re.kr

Job Priority Give Account to CPU Numbers in IBM LoadLeveler

Young-Joo Lee, Jae-Min Choi, Sung-Jun Kim, Joon-Woo
Korea Institute of Science Technology Information

요 약

하나의 시스템을 다수의 사용자가 사용하는 경우 한정된 시스템의 자원을 효율적으로 배분하기 위하여 작업관리 시스템을 사용한다. 이러한 작업관리 시스템은 여러가지 종류가 있으며 시스템의 특성과 사용의 작업 패턴을 고려하여 적당한 작업관리 시스템을 선택하여 사용한다. IBM 시스템은 자체로 제공하는 작업관리 시스템인 LoadLeveler 주로 사용하고 있으며, 최근에는 몇가지 기능이 향상된 LSF를 이용하는 추세이기도 하다.

작업관리 시스템의 LoadLeveler에서는 CPU 수에 따라서 작업 우선 순위를 부여하고자 할 때 이러한 기능을 할 수 있는 환경 변수가 제공되지 않는데, 이러한 LoadLeveler의 환경에서 시스템의 환경파일에 스크립트 프로그램으로 이러한 기능을 구현하여 CPU 수에 따른 작업 우선순위를 적용함으로서 시스템 전체의 처리 효율을 향상 할 수 있게 하였다.

1. 서론

작업관리 시스템은 한정된 전체 시스템의 자원을 다수의 사용자가 요구한 자원에 따라서 효율적으로 배분하고 관리할 수 있는 시스템이다. 이러한 작업관리 시스템은 전체 시스템 자원을 하나의 사용자가 독점 사용하는 것을 막고 동시에 다수의 사용자가 각각의 프로그램에서 처리할 자원을 요구할 때 각각의 요구자원을 배분한다. 이러한 작업관리 시스템은 여러 종류가 있으며 시스템의 종류와 작업의 특성에 따라서 적당한 작업관리 시스템을 사용되고 있다. KISTI에 설치된 IBM p595 시스템에서는 LoadLeveler, SUN 시스템에서는 SunGridEngine, 리눅스 클러스터 시스템은 PBS 등이 사용되고 있다. 이러한 작업관리 시스템은 전체 작업의 흐름을 원활하게 함으로서 시스템의 전체 작업처리에 효율에 많은 영향을 준다. 작업관리 시스템을 설계하고 관리하는 데 여러가지 환경변수가 있지만 그 중에서도 CPU와 메모리가 가장 큰 영향을 주는 환경 변수이다.

본 논문에서는 작업관리 시스템 LoadLeveler에서 CPU 수에 따른 작업 우선순위를 부여하여 사용자들이 프로그램 작업 시 많은 CPU를 사용하게 함으로서 시스템의 전체 작업 처리 효율을 높이고자 한다.

2. 관련 연구

2.1 LoadLeveler

LoadLeveler는 분산컴퓨터를 위한 작업로드 관리 시스템으로서 사용자가 여러 노드 또는 시스템을 하나의 컴퓨터처럼 사용할 수 있게 한다. 또한 여러 시스템의 작업 부하를 균형 있게 관리하며, 순차 프로그램과 병렬 프로그램을 모두 지원한다. 이 작업관리 시스템은 원래는 위스콘신 대학교의 Condor Job Scheduler를 기초로 한 것으로 IBM에 의해 사용자 기반 우선 순위, NFS/AFS 지원, GUI 등의 기능이 추가되었다.

2.2 LoadLeveler 구조

LoadLeveler는 사용자의 프로그램을 실행하기 위해 명시한 요구자원을 시스템 자원으로부터 할당받아 처리한다. 사용자가 LoadLeveler를 통하여 작업을 제출하면 작업은 일단 LoadLeveler의 큐에 들어가 대기하다가 해당 작업의 처리 조건이 가능한 큐를 찾으면 해당 큐에서 작업을 실

행한다.

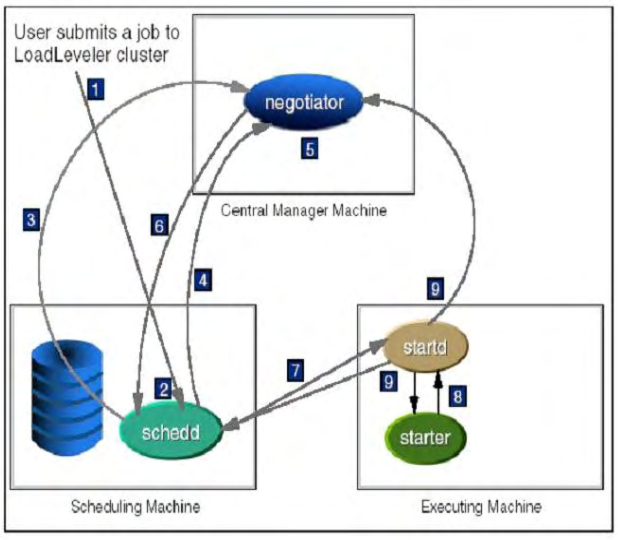
2.3 작업 진행 단계

작업관리 시스템에 작업을 제출하면 일반적으로 <표 1>과 같이 5 단계의 과정을 거쳐 실행된다.

<표 1> 작업 실행 순서

제출 → 대기 → 시작 → 수행 → 종료

(그림 1)은 LoadLeveler의 구조를 나타낸 것이다. 작업을 제출하면 LoadLeveler의 관리자가 작업정보와 노드정보를 분석하여 적당한 자원을 찾아서 해당 큐에 작업을 할당한다.



(그림 1) LoadLeveler 구성

2.4 작업 처리

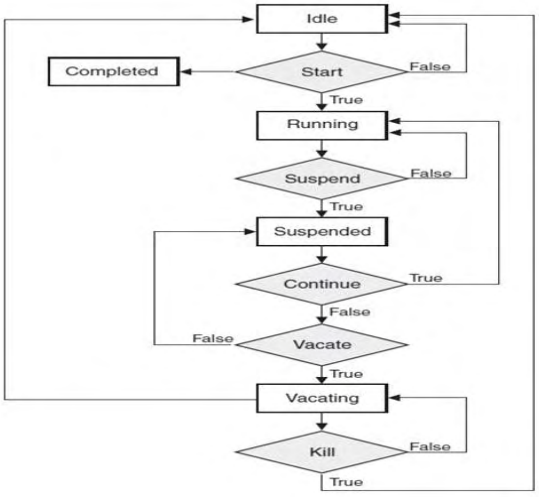
작업 처리에 있어서 처리소요시간은 작업의 대기시간+실행시간이므로 실행시간은 컴퓨터의 성능에 따라 이미 정해지기 때문에 작업의 대기시간에 따라 작업수행 효율이 결정되기 때문에 이에 대한 최적화가 필요하다.

2.5 Scheduling

LoadLeveler job scheduling에는 크게 두 가지가 있다.

□ Backfill Scheduler

일단 Run Queue에 들어가 수행을 시작하면 종료될 때까지 요청된 CPU를 점유한다. 즉 태스크 당 CPU 사용 우선순위는 클래스 구분 없이 동등한 것으로 간주한다. 따라서 CPU 사용효율은 좋으나 수행중인 작업의 클래스별 서비스 레벨 적용은 불가능하다.



(그림 2) 작업처리 흐름도

<표 2> 큐의 작업 실행 순서

	P1	P2	P3	P4	P5	P6	P7	P8	P9
Run Queue	A	A	A	A	A	B	B	B	B

□ Gang Scheduler

Job Matrix를 구성하여 정해진 순서에 따라 일정한 Time Slice 동안 CPU를 번갈아 점유한다. 이때 Time Slice 할당 개수를 execution_factor라고 하며 1, 2, 3이 가능하다.

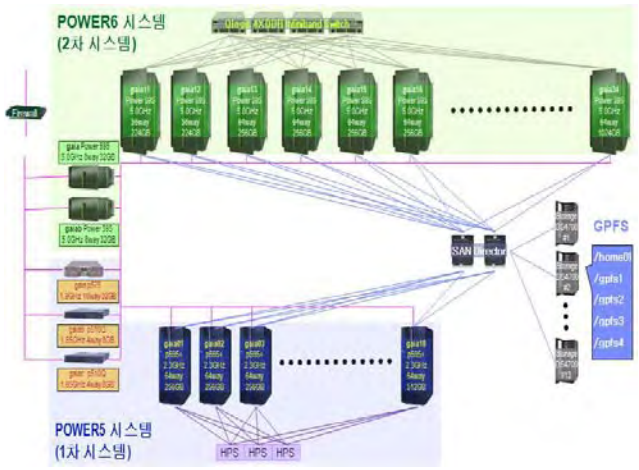
<표 3> 작업의 CPU 할당

Time slice	P1	P2	P3	P4	P5	P6	P7	P8	P9
T1	A	A	A	A	A	A	B	B	B
T2	A	A	A	A	A	A	C	C	C
T3	A	A	A	A	A	A	D	D	D

3. 시스템 구현

3.1 시스템 구성

gaia 2차 시스템은 p595 시스템으로서 (그림 3)와 같이 24개의 노드로 구성되어 있으며 시스템의 전체 외부 디스크는 GPFS를 이용하여 공유하고 있으며 홈디렉토리는 사용자의 편의성을 고려하여 gaia 1차 시스템의 홈디렉토리를 공유하고 있고 사용자의 큰 파일 작업을 위하여 별도의 스크래치 디스크가 4개 GPFS로 연결되어 있다. 1, 2차 시스템 모두 각각의 노드에는 별도의 로컬 디스크가 연결되어 있다.



(그림 3) IBM 2차 시스템 구성도

3.2 클래스 구성

IBM 시스템은 큐를 클래스라 부른다. 이러한 클래스의 구성은 시스템 및 작업의 특성에 따라 여러 가지로 구성할 수 있는데 여기서는 단순하게 전체의 노드를 하나의 큐로만 구성하였다.

<표 4> Class table

Queue name	가용 노드수	가용 CPU수	Wall_clock_limit
large	24	1~1536	5일

3.3 LoadLeveler 환경 구성

IBM LoadLeveler는 CPU 관련한 환경 구성 기능을 제공하지 않기 때문에 시스템 환경의 Load_Admin 파일과 Load_config 파일에 별도의 스크립트를 작성하여 CPU 수의 변수값을 구하여 이를 CPU 수 환경 변수에 적용하여 우선순위 기능을 할 수 있게 하였다.

4. 실행결과

(그림4)는 LoadLeveler의 시스템 환경 변수를 응용하여 LoadLeveler의 환경변수를 재설정하여 보낸 작업들을 모니터링 한 것이다. (그림4)를 보면 CPU 수에 따른 작업들에 대한 우선순위가 적용된 것을 알 수 있다.

LoadLeveler에서의 작업 우선순위는 기본적으로 작업이 먼저 제출하는 순서에 따라서 우선순위가 결정된다. (그림4)에서 맨위의 두작업을 살펴보면 10:55:32에 보낸 작업은 우선순위가 -7131522이고 10:41:42에 보낸 작업은 우선순위가 -7130692이다 따라서 제출된 작업의 작업 우선순위는 CPU 수가 같을 경우 제출 순서에 따라서 순서적으로 적용되는 것을 알 수 있다. 본 사이트에서는 정책 상 사용자 작업이 64CPU 이상을 사용하는 작업에 대하여 우선순위를 적용하고자 하였다. 이것은 작업에서 많은 CPU를 사용하면 그만큼 전체 시스템을 효율적으로 사용할 수 있고 그만큼 전체 효율이 향상되기 때문에 사용자 작업에 많은 CPU 사용을 권장하고 유도하기 위해서이다. (그림4)의 사용자 작업 중에서 64CPU를 사용하는 작업을 살펴보면 00:27:35에 보낸 작업의 우선순위가 -6971445이고 08:06:06에 보낸 작업은 -6905356 인 것을 알 수 있다. 이는 작업을 나중에 제출했어도 CPU를 많이 사용하는 작업이 먼저 제출된 작업보다 우선순위가 높게 적용된 것을 확인할 수 있다.

```
Class State [Queue Date       ] [#N #T #C] #tasks Wall(h) Mem(GB) Rst Shrd Unicore|
-------------------------------------------------------------------------------+

normal  IDLE [09/02/14 10:55:32] [ 1  1  1] ->   1    70.0   0.00  yes -7131522  yes
normal  IDLE [09/02/14 10:41:42] [ 1  1  1] ->   1    70.0   0.00  yes -7130692  yes
normal  IDLE [09/02/14 10:40:11] [ 1 16  1] ->  16    48.0   0.00  yes -7130601  yes
normal  IDLE [09/02/14 10:37:07] [ 1  1  1] ->   1    72.0   0.00  yes -7130417  yes
normal  IDLE [09/02/14 08:45:36] [ 1  1  1] ->   1   120.0   0.00  yes -7123726  yes
normal  IDLE [09/02/14 08:44:53] [ 1  1  1] ->   1   120.0   0.00  yes -7123683  yes
normal  IDLE [09/02/14 10:26:44] [ 1 32  1] ->  32   120.0   0.00  yes -7093794  yes
normal  IDLE [09/02/14 08:57:07] [ 1 40  1] ->  40   120.0   0.00  yes -7088417  yes
normal  IDLE [09/01/14 21:33:48] [ 1  1  1] ->   1    80.0   0.00  yes -7083418  yes
normal  IDLE [09/01/14 19:07:37] [ 1  1  1] ->   1    50.0   0.00  yes -7074647  yes
normal  IDLE [09/01/14 16:50:04] [ 1 32  1] ->  32    72.0   0.00  yes -7051122  yes
normal  IDLE [09/01/14 16:49:49] [ 1 32  1] ->  32    72.0   0.00  yes -7051026  yes
normal  IDLE [09/01/14 14:58:51] [ 1  1  1] ->   1   120.0   0.00  yes -7023721  yes
normal  IDLE [09/01/14 14:50:51] [ 1  1  1] ->   1   120.0   0.00  yes -7023241  yes
normal  IDLE [09/01/14 12:40:15] [ 1  1  1] ->   1   120.0   0.00  yes -7015405  yes
normal  IDLE [09/01/14 10:08:41] [ 1  1  1] ->   1    70.0   0.00  yes -7006311  yes
normal  IDLE [09/01/14 00:27:35] [ 1  1  1] ->   1   120.0   0.00  yes -6971445  yes
normal  IDLE [09/02/14 08:06:06] [ 1 64  1] ->  64    15.0   0.00  yes -6905356  yes
normal  IDLE [09/02/14 02:04:42] [ 1  1  1] ->   1   120.0   0.00  yes -6883672  yes
normal  IDLE [09/02/14 00:23:07] [ 1  1  1] ->   1   120.0   0.00  yes -6877577  yes
normal  IDLE [09/02/14 00:06:03] [ 1  1  1] ->   1   120.0   0.00  yes -6876553  yes
normal NOTQU [09/01/14 23:15:53] [ 1 128  1] -> 128    72.0   0.00  yes -6873543  yes
normal NOTQU [09/01/14 23:15:51] [ 1 128  1] -> 128    72.0   0.00  yes -6873541  yes
normal  IDLE [09/01/14 23:15:49] [ 1 128  1] -> 128    72.0   0.00  yes -6873539  yes
normal  IDLE [09/01/14 23:15:46] [ 1 128  1] -> 128    72.0   0.00  yes -6873536  yes
normal  IDLE [09/01/14 23:15:44] [ 1 128  1] -> 128    72.0   0.00  yes -6873534  yes
normal  IDLE [09/01/14 23:15:42] [ 1 128  1] -> 128    72.0   0.00  yes -6873532  yes
normal  IDLE [09/01/14 21:50:50] [ 1  1  1] ->   1   120.0   0.00  yes -6868440  yes
normal NOTQU [09/01/14 21:02:46] [ 1 64  1] ->  64    72.0   0.00  yes -6865556  yes
```

(그림 4) 작업 모니터링

5. 결론

IBM 시스템에서 LoadLeveler를 이용하여 작업을 처리할 때 CPU 수에 따라 작업 우선순위를 적용하려고 할 때 시스템 자체에서 그러한 기능에 대한 환경 변수를 제공하지 않으나 시스템 프로그램을 작성하여 환경 변수 값을 구하여 구현하고자 하는 기능을 할 수 있게 CPU 수에 따른 작업 우선순위를 적용할 수 있었다.

향후에는 사용자 별 구분에 따라 작업 우선을 차별적으로 적용할 수 있는 기능을 연구하고자 한다.

참고문헌

[1] IBM, AIX Wprkload Manager.
[2] IBM, Using and Administering
[3] KISTI, IBM System User Guide, 2006
[4] KIPS, IBM 시스템에서 WLM을 이용한 LoadLeveler 최적 환경 구현, 2007
[5] KIPS, IBM 멀티 노드에서의 LoadLeveler 최적 작업 환경 구현, 2011