IBM 시스템에서 WLM을 이용한 LoadLeveler 최적 환경 구현

이영주*, 우준*, 성진우*, 이진훈**, 임경빈***, 박찬열*
*한국과학기술정보연구원
**한국IBM
***CIES
e-mail:yjlee@kisti.re.kr

LoadLeveler Optimization Using WLM in IBM System

Young-Joo Lee*, Joon Woo*, Jin-Woo Sung*, Jin-Hoon Lee**,
Kyung-Bin Im***, Chan-Yeol Park*
*Korea Institute of Science Technology Information
**IBM Korea
***CIES

요 약

시스템의 한정된 자원을 다수의 사용자들이 프로그램 실행 시 자원을 효율적으로 배분하기 위하여 작업관리 시스템을 이용한다. IBM 시스템은 작업관리 시스템으로서 주로 LoadLeveler를 사용한다. LoadLeveler에서의 메모리 관리는 WLM(Workload Manager)에서 설정하며, WLM의 환경 설정에 따라 작업의 실행에 많은 영향을 받는다.

본 논문에서는 WLM의 환경 설정에 의한 LoadLeveler에서의 작업을 실행하면서 메모리 변화와 실행 시간을 측정하고 분석하였다. 따라서 시스템의 특성과 사용자 작업에 알맞은 최적의 WLM 환경을 적용함으로써 시스템의 안정성을 유지하고 전체 작업처리 효율을 증가시켰다.

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 p690 시스템에서는 LoadLeveler, NEC SX-5/6 시스템은 NQS, 하멜 클러스터 시스템은 PBS 등이 사용되고 있다. 이러한 작업관리 시스템은 전체 작업의 흐름을 원활하게 유도하고 시스템의 작업처리에 많은 영향을 준다. 작업관리 시스템의 구성을 설계하고 관리하는데 많은 환경변수를 가지고 있지만 그 중에서도 CPU와 메모리가 가장 큰 환경 변수 요인이다.

본 논문에서는 작업관리 시스템을 통하여 작업을 실행할 때 WLM 기능을 이용하여 메모리를 제어함으로서 시스템의 안정성을 유지하고 따라서 전체 시스템의 작업 처리 효율성을 높이고자 한다.

2. 관련 연구

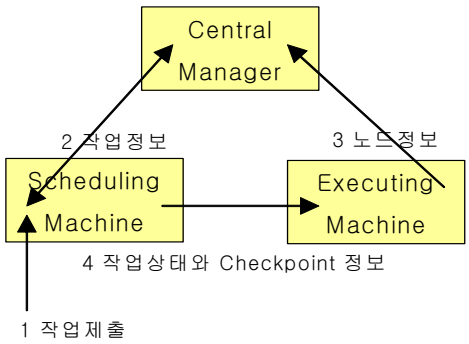
2.1 LoadLeveler

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 LoadLeveler에 대기하다가 해당 작업의 처리 조건이 가능한 클래스를 찾으면 해당 큐에서 작업을 실행한다.



(그림 1) LoadLeveler job cycle

2.3 WLM(Workload Manager)

WLM은 단일 O/S로 이루어진 단일 시스템에서 다수의 효율적인 자원의 분배를 허락하고 관리를 쉽게 할 수 있게 한다.

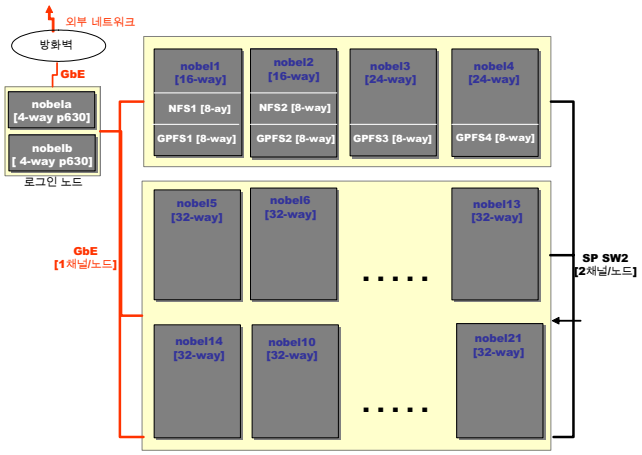
시스템 자원의 통제를 위해, LoadLeveler는 AIX WLM을 통해 CPU 및 메모리의 자원 할당을 통제할 수 있다. WLM은 시스템 자원을 감시하며 AIX 상에서 수행되는 프로세스들에게 할당되는 자원을 통제한다. 이는 수행되는 각각의 작업들이 필요로 하는 자원의 충돌이 발생했을 때 서로 충돌 없이 원활하게 수행하기 위함이다.

LoadLeveler는 하나의 작업이 수행하고자 하는 노드에 할당될 때, 특별한 자원의 사용권한을 가지는 WLM class를 동적으로 생성하며, 그 WLM class는 해당 작업의 process id를 가진다. 그 후, LoadLeveler는 정해진 LoadLeveler 자원 규제 정책에 따라 해당 class의 자원 공유 및 임계치를 정의한다

3. 시스템 환경 구성

3.1 IBM 시스템 구성

본 논문에서 실험 대상으로 하는 시스템은 (그림 2)에서와 같이 모두 21노드로 이루어진 IBM p630, p690 시스템이다. O/S는 AIX 5.2이고 작업관리 시스템은 LoadLeveler 3.3.2 이다. p630 시스템은 주로 로그인용 서버로 사용하고, p690 시스템은 계산노드로 사용한다. 하나의 노드는 32CPU로 구성되어 있다. 홈디렉토리는 NFS로 연결되고 사용자의 작업을 위한 스크래치 디렉토리는 GPFS를 통하여 연결되어 있다.



(그림 2) IBM 시스템 구성도

3.2 LoadLeveler 큐 구성

클래스는 크게 시스템의 구분에 따른 p630, p690과 작업의 특성에 따른 일반, 병렬의 4가지 유형과 별도의 특정 작업을 위한 클래스로 구분하였다. 각각의 클래스는 가용한 CPU 수와 실행시간을 설정하였다.

<표 1> LoadLeveler의 class 구분

Class Name	Priority	최대 CPU	허용시간	비고
normal	1	1	1080	일반1CPU
p_normal	2	448	720	일반병렬
p_normal_1.3	2	64	720	p630
p_normal_1.7	2	384	720	p690
grand	1	32	960	지정노드
grand1	1	32	960	지정노드

4. WLM의 기능을 적용한 Loadleveler 테스트

4.1 WLM 환경 구성

WLM의 기능을 담당하는 옵션에는 <표 2>처럼 크게 두가지 종류가 있으며, 그 각각의 기능을 <표 3>과 같은 조건표를 만들어 성능 테스트를 하였다. 실행 시간 측정에 사용한 모니터링 툴은 svmon,

nmon 그리고 시스템 모니터링 명령어 등을 사용하였다.

<표 2>의 첫 번째 옵션에서 shares는 프로그램 실행 중 메모리 사용을 서로 공유한다. 이 때 메모리 점유는 page 알고리즘에 따른다. soft는 프로그램에서 정의한 메모리보다 실행메모리가 더 필요할 경우 그 이상 사용할 수 있으며 다른 프로그램에서 메모리를 요구할 경우 메모리를 반환한다. hard는 프로그램에서 정의한 메모리보다 실행 메모리가 더 필요한 경우 swap 메모리를 사용한다.

<표 2>의 두 번 째 옵션에서 false는 실행메모리가 프로그램에서 정의한 메모리보다 클 경우 그 이상의 메모리를 점유하는 것을 허용하고, true는 실행메모리가 정의한 메모리의 한계를 넘는 것을 허용하지 않는다.

<표 2> WLM 주요 옵션

ENFORCE_RESOURCE _POLICY=shares|soft|hard ENFORCE_RESOURCE_MEMORY=false|true

<표 3> WLM 기능 설정에 따른 테스트 옵션

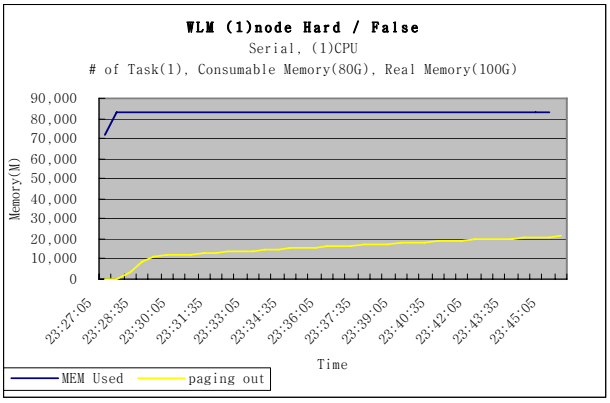
구분	ENFORCE_RESOURCE _POLICY	ENFORCE_RESOURCE _MEMORY
테스트1	soft	false
테스트2	hard	false
테스트3	soft | hard	true

<표 2>의 기능 옵션에서 shares를 제외한 것은 기존에 해당 옵션으로 구성하여 운영한 경험으로 문제점을 알고 있으며, 옵션 기능 특성 상 같은 작업 조건으로 paging in/out의 지연 시간을 측정하기가 어렵기 때문이다.

4.2 메모리 기능에 따른 작업 시간 테스트

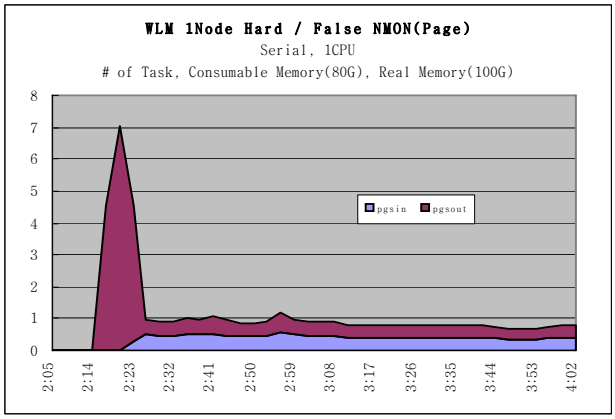
프로그램 테스트는 WLM에서 메모리 기능의 환경을 설정하고 LoadLeveler를 통하여 작업을 실행하면서 메모리의 사용변화와 실행시간을 측정하였다.

(그림 3)은 1CPU, Real Memory=100GB의 작업을 Consumable Memory=80GB에서 실행한 예이다. 프로그램에서 80GB 메모리만 확보하였기 때문에 필요한 Real Memory 100GB, 즉 추가로 20GB의 메모리를 확보하기 위하여 page out이 발생하게 된다. 이러한 빈번한 page out으로 인하여 프로그램의 수행 속도가 지연되는 원인이 된다.



(그림 3) svmon을 이용한 메모리 변화 측정

(그림 4)는 프로그램 실행 시 메모리의 page in/out의 변화를 나타낸 것이다. 작업을 실행하고 나서 프로그램에서 필요한 실행 메모리를 점유하는 과정에서 프로그램에서 정의한 메모리를 확보한 후 그 이상의 필요한 메모리를 점유하기 위하여 page out이 계속하여 발생하는 것을 알 수 있다.



(그림 4) WLM으로 메모리 변화 측정

4.3 프로그램 테스트 결과

4.3.1 테스트1(soft & false)

이 조건의 실제 테스트는 시행할 시스템의 paging space가 작은 관계로 생략하였다. 프로그램 작업 실행 중 또 다른 프로그램이 들어 올 경우, 처음 작업은 Consumable Memory보다 추가로 점유한 Real Memory를 반환하는데, 이 때 page out을 위한 paging space 크기가 작기 때문이다.

이 조건은 프로그램 실행에서 Consumable Memory 상에서 요구한 자원보다 사용되는 Real

Memory가 클 경우 기존 작업이 점유했던 추가 자원을 시스템에 반환하게 되는 과정에서 작업의 실행 시간이 크게 지연된다.

4.3.2 테스트2(hard & false)

이 조건은 작업이 Consumable Memory 상에서 요구한 자원보다 실제로 사용할 자원이 많더라도, 해당 작업에는 Consumable Memory 상에서 요구한 자원만 할당된다. 프로그램 실행에서 예측된 메모리를 사용하는 경우는 작업이 정상적으로 종료되지만, 실제로 사용할 Real Memory 자원이 Consumable Memory보다 클 경우 page out이 발생하며 이러한 과정에서 작업의 실행 성능은 크게 지연된다.

<표 4> 환경 조건에 따른 작업 실행 시간

구분	메모리 (GB)	작업 형태	노드수	소요 시간
테스트2	CM = 100 RM = 100	1CPU	1node	11분
	CM = 100 RM = 200	2CPU		11분
	CM = 100 RM = 100		2node	11분
	CM = 80 RM = 100	1CPU	1node	3시간50분
	CM = 80 RM = 200	2CPU		5시간37분
	CM = 80 RM = 100		2node	3시간58분
테스트3	CM = 100 RM = 100	1CPU	1node	11분
	CM = 100 RM = 200	2CPU		11분
	CM = 100 RM = 100		2node	11분
	CM = 80 RM = 100	1CPU	1node	작업중단
	CM = 80 RM = 200	2CPU		
	CM = 80 RM = 100		2node	

* CM=Consumable Memory
RM=Real Memory

4.3.3 테스트3(soft & true, hard & true)

해당 작업이 Consumable Memory상에서 요구한 메모리를 사용하는 경우는 작업 수행 시 예상된 시간에 정상적으로 종료되지만, Real Memory 사용량이 Consumable Memory보다 클 경우는 프로세스가 강제 종료되므로 작업 시 메모리 사용량을 충분히 확보하여야 한다. 다른 조건의 옵션과 비교했을 때 paging space를 사용하지 않아 시스템 성능에 영향을 주지 않는 장점이 있다.

5. 결론

WLM 기능을 조건별로 분류하여 LoadLelveler에서 작업 수행 시간을 측정 분석하였다. 사용자가 프로그램 실행 시 프로그램 환경에서 정의한 메모리가 실제로 필요한 메모리보다 작게 할당했을 경우 최소 20배 이상의 프로그램 실행시간 지연이 발생하였다.

<표 3>의 테스트3과 같은 설정을 일부 노드에 6개월 정도 적용한 결과 한번도 시스템에 hang 현상이 발생하지 않았다. 이와 같은 기능 조건을 적용하지 않았을 경우는 메모리 부하로 인하여 월 1회 정도의 메모리 hang이 발생하였다.

향후의 연구에는 충분한 환경을 가진 시스템에서 <표 3>의 테스트1을 시험하여 메모리의 입출력 부하와 시스템 hang과의 연계성을 분석하고자 한다.

참고문헌

[1] IBM, LoadLeveler Using and Administering
[2] IBM, AIX Workload Manager.
[3] IBM, LoadLeveler Guide
[4] IBM, AIX Performance Manager
[5] KISTI, IBM System User Guide, 2006