제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

클러스터 시스템을 위한 메모리 테스트 도구 설계

차광호*, 홍정우*, 이지수*
*한국과학기술정보연구원 슈퍼컴퓨팅센터
e-mail : khocha@kisti.re.kr

The Design of Memory Test Tool for Cluster System

Kwangho Cha*, Jeongwoo Hong*, Jysoo Lee*
*Supercomputing Center, Korea Institute of Science and Technology Information

요 약

가격대 성능 비를 고려할 때 저가격으로 병렬시스템을 제작할수 있다는 특징으로, 시작된 클러스터 시스템이 구성 장비의 특수화 및 전체 시스템의 대규모화로 인하여 더 이상 보조적인 소규모 시스템이 아닌, 슈퍼 컴퓨터의 한 종류를 이루는 비중있는 시스템으로 자리매김하고 있다. 이처럼 클러스터 시스템 개발의 전반적인 방향이 대규모화를 지향하는 점을 고려할 때, 각 구성 요소의 무결성, 즉 안정성 점검은 시스템의 정상적인 운영을 위해서 중요한 부분인다. 본 논문에서는 클러스터 시스템을 구성하는 각 계산노드의 메모리의 이상 유무을 관리 서버 측면에서 종합적으로 진단하기 위한 관리 도구의 개발을 다루고 있다.

1. 서론

초기 클러스터 시스템은 가격대 성능 비에서 우수하다는 특징으로 관심을 끌기는 했지만 실험적인 성격의 소규모 시스템이었다. 그러나 클러스터 시스템 전용의 특수 장비의 개발과 더불어 단위 계산 노드의 역할을 하는 서버의 성능이 향상되면서 시스템의 규모도 대형화되고 있으며, 슈퍼컴퓨터의 한 부류로 간주되어 지고 있다. 이는 Top500리스트를 살펴 볼 때 클러스터 시스템의 비중이 증가하고 있다는 점에서도 확인할 수 있다[1].

클러스터 시스템의 보급이 증가하면서 시스템의 규모도 급격히 증가하고 있어, 1024이상의 CPU를 사용하는 클러스터 시스템도 쉽게 찾아 볼 수 있다. 이처럼 클러스터 시스템에서 사용되는 하드웨어의 수가 증가함에 따라 각 구성요소에 문제가 발생할 확률은 그만큼 증가하게 되며, 특히 시스템의 발열량과 같은 물리적인 요소가 전체 시스템에 영향을 줄 수 있는 상황이다. 이러한 상황을 고려할 때, 안정된 시스템 운영을 위해서는, 시스템의 초기 구축 시는 물론, 정기 점검 시에 시스템 하드웨어의 이상 유무를 발견할 수 있는 방법이 필요하다.

본 논문에서는 이와 같은 대규모 단위 노드로 구성된 클러스터 시스템의 하드웨어 무결성 점검을 다루고 있다. 특히 각 계산 노드의 메모리의 상태 점검을 대상으로 하고 있다.

2. 관련연구

본 장에서는 현재 개발되어 오픈 소스 형태로 제공되는 단일 시스템용 메모리 테스트 도구에 대해 설명하도록 한다. 클러스터 시스템이 특정 운영체제를 요구하는 것은 아니지만, 일반적으로 많은 시스템들이 유닉스나 리눅스를 운영체제로 사용하고 있으므로, 본 논문에서는 리눅스용 도구를 대상으로 한다.

많이 알려진 메모리 테스트 도구로 Memtest86을 들 수 있다[2]. X86기반의 아키텍처를 주요 대상으로 하는데 가장 큰 특징은 커널을 배제시키고 테스트를 수

행한다는 점이다. 리눅스 상에서 테스트 도구를 생성한 후, 재 부팅하면 커널 대신 테스트 프로그램이 로딩된 후 메모리 테스트를 수행하는 구조이다. 커널과 독립적으로 테스트를 직접 수행함으로서 엄밀한 검사를 할 수 있다는 장점이 있으나, 커널이 수행하여야 하는 하드웨어 제어 루틴도 직접 포함하고 있으므로 하드웨어에 의존적인 도구이다. 현재는 X86 아키텍쳐용 프로그램만 제공되고 있다.

Memtest86가 수행하는 메모리 테스트 알고리즘과 상당히 유사한 기능을 가진 테스트 도구로 Memtester가 있다. Memtest86가 커널을 배제하고 독립적으로 테스트를 수행하는 반면, Memtester는 커널 상에서 동작하는 특징을 갖는다[3]. 커널 상에서 동작하는 유저 어플리케이션의 일종이므로 전체 메모리를 테스트하는 데는 제약이 따르지만, X86기반 아키텍쳐에서만 동작한다는 Memtest86의 단점을 보완할 수 있다. 또한 커널이 제공하는 서비스를 그대로 받을 수 있다는 장점도 있다. [표 1]은 Memtest86와 Memtester에서 각각 사용된 메모리 테스트 알고리즘들의 종류를 보여주고 있다.

[표 1] 메모리 테스트 알고리즘

Memtest86	Memtester
△ Address test ,walking ones(nc)	△ Stuck Address
△ Moving Inversions,ones&zeros(c)	△ Random value
△ Address test ,own address(nc)	△ XOR comparison
△ Moving Inversions,8 bit pattern(c)	△ SUB comparison
△ Moving Inversions,32 bit pattern(c)	△ MUL comparison
△ Block move ,64 moves(c)	△ DIV comparison
△ Modulo 20 ,ones&zeros(c)	△ OR comparison
△ Moving Inversions,ones&zeros(nc)	△ AND comparison
△ Block move ,512 moves(c)	△ Sequential Increment
△ Moving Inversions,8 bit pattern(nc)	△ Solid Bits
△ Modulo 20 ,8 bit(c)	△ Block Sequential
△ Moving Inversions,32 bit pattern(nc)	△ Checkerboard
	△ Bit Spread
※ c = cached, nc = no cache	△ Bit Flip
	△ Walking Ones
	△ Walking Zeroes

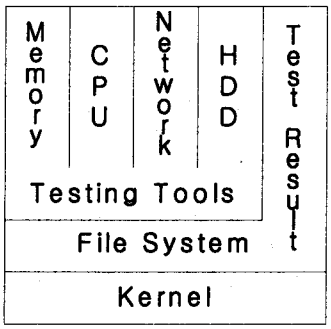
이상의 테스트 프로그램들 이외에도, 스크립트 수준의 간단한 프로그램들도 쉽게 찾아 볼수 있으나, 구체적인 설명은 제외한다[4].

3. 설계 및 구현

3.1. 클러스터 시스템용 메모리 테스트 도구의 설계

클러스터 시스템이 갖는 특징 중 하나는 각각의 계산 노드들이 각자 자신의 운영체제를 가지고 있다는 점이다. 이는 앞서 설명된 단일 시스템용 메모리 테스트 도구들을 별도의 수정 없이 그대로 사용할 수 있다는 의미가 된다. 그러나 계산 노드 수가 증가할 경우에는 제약이 따르게 된다.

테스트 도구를 설계하면서 고려한 주요 기능은, 관리 서버와 같은 특정 서버 시스템에서 다수의 계산 노드에 대한 자동화된 테스트 수행 및 결과 취합 기능이다. 또한 특정 시스템에 국한되는 구성을 배제하고 호환성을 유지하기 위하여, 기존의 공개 소프트웨어들을 수정하여 테스트 도구를 구성하는 방향으로 설계를 진행하였다.



[그림 1] 클러스터 시스템 테스트 도구의 구성

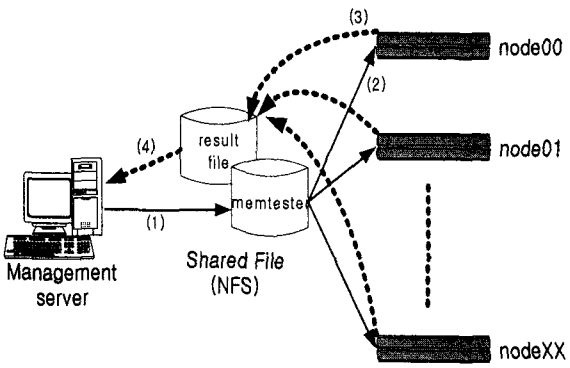
테스트 도구의 개략적인 구조는 [그림1]과 같다. 본 논문에서 다루고 있는 메모리 테스트 도구는 클러스터 시스템 테스트 도구의 일부분으로 사용될 계획이며, 전체 테스트 도구는 메모리 이외에도 CPU와 HDD 같은 하드웨어 요소에 대한 테스트 기능의 확장도 계획하고 있다.

앞서 설명한 메모리 테스트 도구들을 비교 분석한 결과, Memtester를 기반으로 메모리 테스트 모듈을 구현을 진행하였다. Memtest86가 보다 엄밀한 테스트를 진행함을 사실이나, 테스트 결과를 취합하는 과정에서 용이하지 못함과, X86기반의 아키텍쳐에서만 테스트가 가능하다는 점도 Memtester를 선택하는 한가지 요인으로 작용하였다. 이와 더불어 Memtester를 진단도구로 사용하는 관련 연구를 확인할 수 있었는데, Sandia National Labs에서 개발된 클러스터 통합 도구(CIT : Cluster Integration Toolkit)에서 진단 기능을 위하여

Memtester를 사용하고 있다[4,5]. 이 클러스터용 메모리 테스트 도구가 CPlant[6]형 클러스터를 대상으로 CIT의 일부 기능으로 제공되는 반면, 본 논문에서 계획하는 메모리 테스트 도구는 테스트 도구 집합의 일부분으로서 추후 개발될 테스트 도구들과 상호 연계 기능에 중점을 둘 계획이다.

3.2. 클러스터 시스템용 메모리 테스트 도구의 구현

각 노드에서 테스트 후 수행 결과를 공유 파일 시스템을 이용하여 관리 노드에 전달하도록 Memtester 프로그램을 수정하였다. 서버에서 테스트 프로그램을 계산 노드에서 수행시킬 때는 클러스터 시스템을 관리하는 툴로 많이 사용되는 Ptools[7]을 이용하도록 하였다.



[그림 2] 메모리 테스트 도구를 이용한 테스트 수행 절차

[그림 2]는 메모리 테스트 도구가 동작하는 절차를 단계적으로 보여주고 있다. 각 단계에서 수행하는 작업은 다음과 같다.

(1) 테스트를 수행하기에 앞서 수정된 Memtester를 공유 파일 시스템 상에 위치시킨다.

(2) 서버는 Ptools를 이용하여 각 계산 노드에서 수정된 Memtester를 수행한다.

(3) 수정된 Memtester프로그램은 각계산 노드에서의 수행 결과를 파일의 형태로 공유 파일 시스템 상에 기록한다.

(4) 서버는 생성된 결과 파일을 분석하여 각 계산 노드의 메모리 이상 유무를 확인한다.

이와 같은 과정을 수행하기 위하여 기존의 Memtester가 가지고 있는 로그 파일 생성 기능과는 별도로 테스트 결과의 요약을 임의의 파일에 기록하도록 하였으며, 서버 측에서는 이 파일을 분석하기 위한 도구를 개발 중에 있다.

4. 메모리 테스트 결과

테스트 도구의 동작을 확인하기 위하여, 정상적인 클러스터 시스템에서 테스트를 실행하였다. 보다 정확한 결과를 위해서는 비정상적인 요소를 가지고 있는 클러스터 시스템에서의 테스트가 불가피하나, 이는 향후 테스트 항목으로 계획 중이다. [그림 3]은 한국과학기술정보연구원에서 보유하고 있는 128노드 클러스터 시스템인 PLUTO에서 일부 계산 노드를 대상으로한 테스트 수행 결과를 서버측에서 확인한 결과이다.

```
[khocha@loginserver3 mem_test]$ ./fileread ./pluto.txt
Hostname = node033 | Run = 1 | Error = 0 | Time = 3
Hostname = node034 | Run = 1 | Error = 0 | Time = 2
Hostname = node035 | Run = 1 | Error = 0 | Time = 3
Hostname = node036 | Run = 1 | Error = 0 | Time = 3
Hostname = node037 | Run = 1 | Error = 0 | Time = 3
Hostname = node038 | Run = 1 | Error = 0 | Time = 2
Hostname = node039 | Run = 1 | Error = 0 | Time = 3
Hostname = node040 | Run = 1 | Error = 0 | Time = 3
Hostname = node041 | Run = 1 | Error = 0 | Time = 3
Hostname = node042 | Run = 1 | Error = 0 | Time = 3
Hostname = node043 | Run = 1 | Error = 0 | Time = 3
Hostname = node044 | Run = 1 | Error = 0 | Time = 3
Hostname = node045 | Run = 1 | Error = 0 | Time = 3
Hostname = node046 | Run = 1 | Error = 0 | Time = 3
Hostname = node047 | Run = 1 | Error = 0 | Time = 3
Hostname = node048 | Run = 1 | Error = 0 | Time = 2
Hostname = node049 | Run = 1 | Error = 0 | Time = 2
Hostname = node050 | Run = 1 | Error = 0 | Time = 2
Hostname = node051 | Run = 1 | Error = 0 | Time = 3
Hostname = node052 | Run = 1 | Error = 0 | Time = 3
Hostname = node053 | Run = 1 | Error = 0 | Time = 3
Hostname = node054 | Run = 1 | Error = 0 | Time = 2
Hostname = node055 | Run = 1 | Error = 0 | Time = 3
Hostname = node056 | Run = 1 | Error = 0 | Time = 2
Hostname = node057 | Run = 1 | Error = 0 | Time = 2
Hostname = node058 | Run = 1 | Error = 0 | Time = 2
Hostname = node059 | Run = 1 | Error = 0 | Time = 3
Hostname = node060 | Run = 1 | Error = 0 | Time = 2
Hostname = node061 | Run = 1 | Error = 0 | Time = 3
Hostname = node062 | Run = 1 | Error = 0 | Time = 2
Hostname = node063 | Run = 1 | Error = 0 | Time = 3
Hostname = node064 | Run = 1 | Error = 0 | Time = 2
```

[그림 3] 메모리 테스트 수행 결과의 예

5. 결론 및 향후 계획

본 논문에서는 클러스터 시스템을 위한 메모리 테스트 도구의 설계 및 구현에 관하여 설명하였다. 기존 리눅스 기반의 Memtester를 수정하여 분산된 각 노드들의 메모리 테스트 기능을 구현하여 테스트하였다.

향후 개선되어야 할, 부분으로는 메모리 테스트 모듈의 병렬 실행에 대한 부분, 수행 시간의 최적화 및 테스트 알고리즘별 효과 분석등이 있다.

참고문헌

[1] "TOP500 Supercomputer sites," http://www.top500.org/

[2] "Memtest86 - A Stand-alone Memory Diagnostic," http://www.memtest86.com/

[3] Charles Cazabon, "Memtester," http://www.qcc.ca/~charlesc/software/memtester

[4] Michael D. Crawford, "Using Test Suites to Validate the Linux Kernel," http://linuxquality.sunsite.dk/articles/testsuites

[4] James H. Laros III, Lee Ward, Nathan W. Dauchy, James Vasak, Ruth Klundt, Glen Laguna, Marcus Epperson, and Jon R. Stearley, "The Cluster Integration Toolkit - An Extensible, Portable, Scalable Cluster Management Software Implementation," Proc. 1st Cluster World Conference and Expo, June 2003, pp 23~26.

[5] James H. Laros III, Lee Ward, Nathan W. Dauchy, Ron Brightwell, Trammell Hudson, and Ruth Klundt, "An Extensible, Portable, Scalable, Cluster Management Software Architecture," Proc. IEEE International Conference on Cluster Computing, Sept. 2002, pp 287~295.

[6] "The Computational Plant project," http://www.cs.sandia.gov/cplant/

[7] "The Parallel Tools Consortium," http://www.ptools.org/