부하 분산 시스템의 확장

(이질적 환경으로의 확장과 고장 허용 기능)

박은지(EUN-JI PAK)

한국과학기술원 전산학과 컴퓨터 구조 연구실

2004년 8월 9일

요약

클러스터는 두 대 이상의 컴퓨터를 마치 하나의 시스템처럼 행동하도록 연결하여, 막대한 양의 계산이 필요하거나, 중단 없는 서비스를 수행하고자 할 때 사용하는 시스템이다. 연결하는 노드나, 네트워크의 종류에 따라 가격과 성능은 많이 다르지만, 슈퍼 컴퓨터에 비해 상대적으로 저렴한 비용으로 필요한 컴퓨팅 파워나 기능을 구현할 수 있다.

이런 클러스터 시스템에서는 자원을 적절히 분배하여 부하를 균등하게 유지하여 작업의 효율을 높이고자 부하 분산 시스템이 필요해지게 되고, 이런 요구에 맞춰 클러스터 환경에서 부하 분산 시스템을 구현하고 그 성능을 측정한 바 있다.

클러스터 시스템은 일반적인 PC를 사용하여 시스템을 구축하기 때문에 오래 사용하게 되면서 환경의 이질적 특성이 드러나게 된다. 이런 이질적 환경을 지원할 수 있도록 부하 분산 시스템의 기능을 확장하였다. 또한 구현된 부하 분산 시스템의 사용을 용이하기 위해 고장 허용 기능의 추가를 통해 클러스터 시스템의 고장에 능동적으로 대체할 수 있도록 부하 분산 시스템을 확장하였다.

목차

제 1 장

서론

클러스터 시스템은 빠른 속도의 네트워크로 저가의 PC를 묶어 고성능, 고가용성을 얻기 위한 시스템이다. 분산되어 있는 자원을 효율적으로 사용하고 높은 성능을 얻기 위해서는 자원의 상태에 따라, 그리고 각 노드의 부하 상태에 따라 자원을 적절히 분배하여 부하를 균등하게 유지하는 것이 중요하다. 이를 위해 각 노드의 부하 상태를 확인하여 작업을 적절히 분배하여 클러스터 시스템의 부하를 조절하는 부하 분산 시스템을 구현하고 벤치마크를 수행하여 그 성능을 평가하였다.

이렇게 구현된 부하 분산 시스템을 확장하여 사용자에게 편의를 제공하기 위해 부하 분산 시스템에 고장 허용 기능과 이질적 환경에서의 부하 분산 기능을 추가하였다. 클러스터 시스템은 일반적인 PC를 묶어 사용하기 때문에 네트워크와 프로세서가 발달함에 따라 그 이질적 특성이 드러나게 된다. 따라서 최근 이에 대한 많은 요구가 있어왔고, 사용자에게 편의를 제공함과 동시에 고성능을 얻기 위해 이질적 환경에서의 부하 분산 시스템으로 확장을 수행하였다. 이에 더해 시스템의 고장에 능동적으로 대처할 수 있는 고장 허용 기능을 추가하였다.

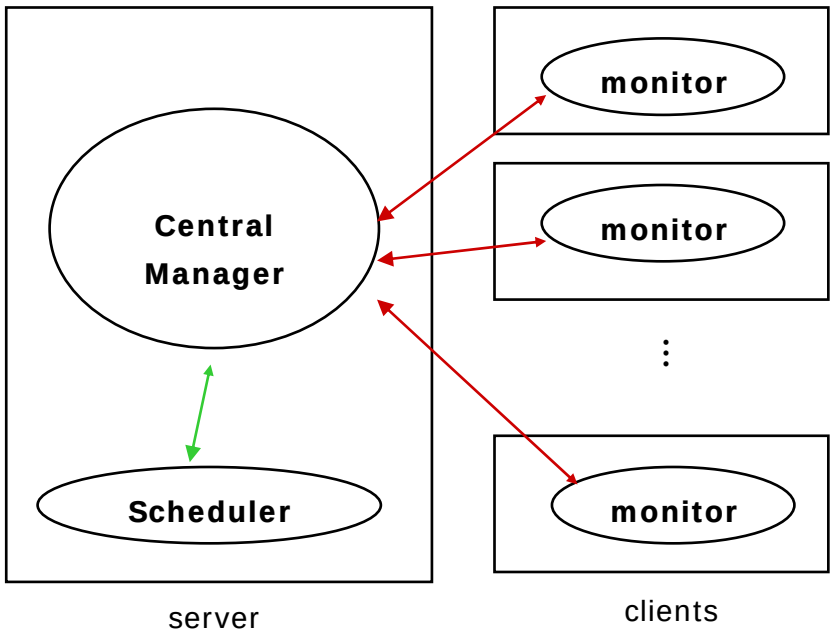
본 보고서의 구성은 다음과 같다. 2장에서는 확장된 기능에 대해 간단하게 설명하고, 3장에서는 기능의 확장을 통한 성능 향상을 분석한다. 그리고 4장에서 결론을 맺는다. 설치나 자세한 구현의 경우 기존의 부하 분산 시스템의 보고서에 포함되어 있으므로 간략하게 설명하고 넘어가기로 한다.

제 2 장

부하 분산 시스템의 설계

2.1 전체 시스템의 구성

부하 분산 시스템은 크게 서버와 클라이언트로 구성된다. 부하 분산 시스템을 사용하고자 하는 클러스터 중에서 한 노드가 서버가 되고 나머지가 클라이언트가 된다. 서버에는 central manager와 scheduler 데몬이 실행되고, 클라이언트에는 각각 monitor 데몬이 실행되며, central manager는 scheduler와 통신함으로서 작업을 어떻게 수행할 것인지에 대한 정책을 결정하고, monitor와의 통신을 통해 각 자원 정보를 관리하거나, 작업을 수행하고 작업 정보를 관리한다. 전체 구조는 그림 2.1과 같다.



[그림 2.1] 전체 시스템의 구조

2.2 고장 허용 기능

부하 분산 시스템에서 고장 허용 기능을 제공하기 위해 그림 2.1의 서버와 클라이언트를 기준으로 각각의 시스템에 문제가 생겼을 경우 확인하고 복구하는 기능을 추가하였다.

클라이언트 노드가 더 이상 작업을 할 수 없는 경우 서버는 이를 확인하고 클라이언트 쪽에서 수행하고 있던 작업은 자동으로 종료된다. 이 작업은 사용자가 필요로 하는 경우 다시 submit하여 다른 클라이언트를 통해 수행시킬 수 있게 된다. 또한 옵션에 따라 현재 수행되고 있는 작업들이 있는 경우 checkpoint/restart기능을 통해 수행중인 작업을 사용자의 개입 없이 자동으로 정상 동작하고 있는 다른 클라이언트 노드에서 수행할 수 있게 하였다.

서버 노드가 더 이상 작업을 할 수 없는 경우 클라이언트 노드는 이를 인식하여 부하 분산 시스템을 종료시키고 다시 서버 노드가 수행되면 사용자에 의해 클라이언트도 다시 수행되도록 해 두었다. 이에 대한 동적인 고장 허용 기능은 추가할 예정이다.

2.3 이질적 환경에서의 부하 분산 기능

클러스터 시스템은 일반적인 PC를 묶어 사용하기 때문에 네트워크와 프로세서가 발달함에 따라 그 이질적 특성이 드러나게 된다. 따라서 최근 이에 대한 많은 요구가 있어왔고, 사용자에게 편의를 제공함과 동시에 고성능을 얻기 위해 이질적 환경에서의 부하 분산 시스템으로 확장을 수행하였다.

SMP와 1 CPU를 그 기준으로 하여 기능의 추가를 수행하였으며, SMP와 1 CPU를 동시에 지원하기 위해 모든 부하 분산 기능을 user-level에서 수행하도록 하였다. 또한 SMP의 경우 동시에 두 개 이상의 작업이 수행되거나 MPI작업의 경우 더 효율적으로 수행하는 점을 감안하여 작업을 분배하였고, 서로 MHz가 다른 CPU는 성능이 다른 것을 반영하기 위해 metric에 이를 반영하였다. Metric을 CPU MHz에 따라 normalize한 값으로 사용함으로써 작업 분배에 있어 이를 효율적으로 반영할 수 있었다. Myrinet과 Ethernet의 다른 환경 역시 자연스럽게 지원된다.

제 3 장

부하 분산 시스템의 성능 측정

3.1 성능 측정 환경

성능 측정을 위해서 6개의 노드로 구성된 클러스터에서 부하 분산 시스템을 수행하였다. 노드 각각의 사양은 다음과 같다. 서버 노드를 can51.kaist.ac.kr로 설정하고 클라이언트 노드는 이질적 환경을 반영하기 위해 SMP 850MHz 두 노드, 1CPU 850MHz 두 노드, 1 CPU 1.5GHz 두 노드를 사용하였다.

Server node: can51.kaist.ac.kr	
CPU	Pentium Ⅳ (1.5GHz)
Memory	512MB
OS	Linux kernel 2.4.18

Client nodes (set 1): can11/can12.kaist.ac.kr	
CPU	SMP dual CPU (850MHz)
Memory	512MB
OS	Linux kernel 2.4.4
Client nodes (set 2): can18/can19.kaist.ac.kr	
CPU	Pentium III (850MHz)
Memory	512MB
OS	Linux kernel 2.4.2

Client nodes (set 3): can48/can49.kaist.ac.kr	
CPU	Pentium IV (1.5GHz)
Memory	512MB
OS	Linux kernel 2.4.20

성능 측정에는 SCALARPACK 벤치마크를 사용하였으며, 이 중에서 사용된 LINPACK은 Argonne Nat'l Lab에서 개발된 것으로 부동점 연산 능력을 행렬 계산을 통해 측정하는 벤치마크 프로그램으로 현재 세계 슈퍼 컴퓨팅 랭킹을 정하는 기본이 되고 있다.

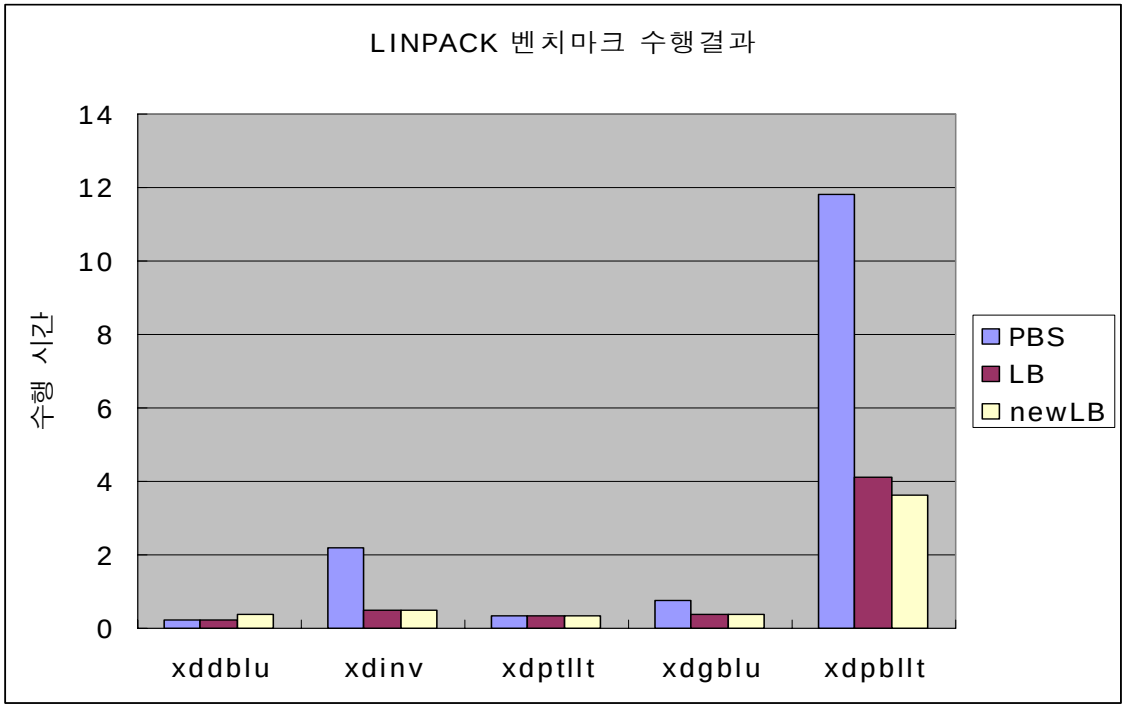
3.2 LINPACK 벤치마크

LINPACK은 주로 선형 방정식과 선형 최소 제곱 법 문제를 푸는 포트란 서브루틴들로 구성된 수치 해석 패키지의 하나로서 컴퓨터의 연산속도를 측정하는 벤치마크 프로그램으로 이용되기도 한다. SCALAPACK 은 선형대수 의 해를 구하는 패키지로 많은 부분이 부동소수점 연산으로 구성되어 있다. LINPACK 벤치마크 에서 중점적으로 사용되는 루틴들은 Gauss 소거법을 이용한 N 개의 선형방정식 의 해를 구하는 것으로 BLAS (Basic Linear Algebra Subprograms) 에 포함되어 있다. BLAS 는 LINPACK 벤치마크 에서 가장 기본이 되는 라이브러리로써 기본적인 선형대수 연산함수 들을 구현해놓은 집합이다. 이것은 Fortran으로 짜여 있으며 BLAS 라이브러리 내의 각 함수들은 연산자와 연산결과가 Vector 냐 Matrix냐 에 따라 계산 레벨이 나뉘어 진다.

3.3 성능 측정 결과

성능 측정을 위하여 PBS라는 프로그램을 설치하여 수행하였다. 6노드에 OpenPBS을 설치하여 벤치마크 프로그램을 수행시키고, 또 구현한 부하 분산 시스템에서 벤치마크 프로그램을 수행시켜 수행시간을 계산하였다. 부하 분산의 성능측정을 위해서 클라이언트 노드 6

개중 두 곳에 프로그램을 수행시켜 loadavg값을 임의로 2로 조정해 둔 상태에서 성능 측정을 하였다. 그림 3.1은 그 결과를 그래프로 나타낸 것이다.



[그림 3.1] 부하분산 시스템의 성능 측정

x축은 Benchmark에 포함되어 있는 5개의 서로 다른 프로그램을 나타내며, y축은 그 수행시간을 나타낸다. LB는 기존의 부하 분산 시스템을 나타내고 new LB는 새로 기능이 추가된 부하 분산 시스템을 나타낸다.

OpenPBS의 경우 MPI작업에 대한 고려가 전혀 이루어지고 있지 않기 때문에 random하게 선택된 노드에서 작업을 수행하게 되어 좋은 성능을 낼 수 없다. 부하 분산 시스템의 경우 OpenPBS의 경우보다 좋은 성능을 보이지만, 새로 기능을 확장한 부하 분산 시스템보다는 좋지 않은 성능을 보이는 것을 볼 수 있다. New LB의 경우 SMP노드에 MPI 작업을 우선 분배하고 서로 다른 노드의 특성(CPU 성능)을 고려하기 때문에 작업의 분배가 더 효율적으로 이루어진 것을 알 수 있었다.

그러나 일부 작업에서는 세 경우 다 비슷한 성능을 보이기도 했는데 이는 MPI작업이 수행중인 가장 성능이 낮은 노드의 성능에 많이 좌우되기 때문에 나타난 현상으로 보인다.

제 4 장

결론 및 향후 과제

본 보고서에서는 클러스터 환경 하에서 작업을 수행할 때 전체 클러스터 시스템의 성능을 높이기 위해 설계, 구현했던 부하 분산 시스템에 고장 허용 기능과 이질적 환경의 제공 기능을 추가하고 그것에 대해 설명하였다. 이질적 환경에서 부하를 균등하게 분배하여 시스템의 성능을 높이고 서로 다른 환경에서의 부하 분산을 용이하게 하였다. 또한 사용자 편의성을 위해 부하 분산 시스템의 수행 중의 고장에 대해 발견하고 해결하는 고장 허용 기능을 추가하였다.

이렇게 구현된 확장된 부하 분산 시스템을 이질적 환경에서 기존의 부하 분산 시스템과 OpenPBS 두 가지와 비교하기 위해 노드 6개에 부하 분산 시스템과 OpenPBS를 각각 설치하여 SCALARPACK 벤치마크를 사용하여 성능을 측정한 결과 이질적 환경을 효율적으로 지원하는 것을 확인하였다.

현재 구현되어 있는 이질적 환경에서의 부하 분산 시스템은 MPI작업에 대한 고려가 아직 완전히 이루어진 상태는 아니다. MPI작업의 경우 SMP노드를 우선으로 하는 수준에서 고려되었으나 이를 확장하고 더 구체적인 알고리즘을 구현하여 성능을 향상시킬 수 있을 것으로 기대된다. 또한 작업의 특성을 고려하여 CPU를 많이 사용하는 작업과 그렇지 않은 작업으로 분류하고 이를 통해 작업을 효율적으로 분배할 수 있는 알고리즘도 추가할 예정이다.