

최종 연구보고서

클러스터 자원 제공을 통한 컴퓨팅 그리드 구축 (Implementation of a Computational Grid based on Support from Cluster Resource)

2003. 12. 11

연구기관 : 서울대학교

주 의

1. 이 보고서는 한국과학기술정보연구원에서 위탁연구과제로 시행한 연구보고서입니다.
2. 이 보고서 내용을 발표할 때에는 반드시 한국과학기술정보연구원에서 시행한 위탁연구과제의 연구결과임을 밝혀야 합니다.

클러스터

자원
제공을

통한

컴퓨팅
그리드

구축

2003

한국과학기술정보연구원

최종 연구보고서

클러스터 자원 제공을 통한 컴퓨팅 그리드 구축
(Implementation of a Computational Grid based on
Support from Cluster Resource)

2003. 12. 11

연구기관 : 서울대학교

한국과학기술정보연구원

제 출 문

한국과학기술정보연구원장 귀하

본 보고서를 “클러스터 자원 제공을 통한 컴퓨팅 그리드 구축”에 대한 연구의 중간보고서로 제출합니다.

2003년 12월 11일

수탁기관 : 서울대학교

수탁기관장 : 정운찬(인)

연구책임자 : 김승조(기계항공공학부)

참여연구원 : 박시형(기계항공공학부)

윤영하(기계항공공학부)

문종근(기계항공공학부)

구기범(KISTI)

김주석(KISTI)

보고서 초록

연구과제코드	G-03-SD-01-29R-2			연구기간	2003.6.15 ~2003.12.15	
연구과제명	클러스터 자원 제공을 통한 컴퓨팅 그리드 구축					
연구책임자	김승조	참여연구원수	총 : 6 명 내부 : 1 명 외부 : 5 명	연구비	정부: 20000 천원 기업: 천원 계: 20000 천원	
연구기관명 및 소속부서명	한국과학기술 정보연구원		참여기관명	서울대학교		
요약(연구결과를 중심으로 개조식 600자이내)					보고서 면수	30
<p>◦ 기존 연구용 Pentium4급 16노드 클러스터 시스템을 그리드 전용 자원으로 활용하기 위해 네트워크 장비 업그레이드 및 로그인 서버 추가 설치.</p> <p>◦ RedHat 7.3 탑재 및 2.4.20으로의 커널 업그레이드, 네트워크 튜닝을 통한 성능 개선</p> <p>◦ NFS에 의한 로그인서버의 /home 공유와 병렬 관리에 필요한 Python 스크립트 툴의 개발로 클러스터를 효율적으로 운영</p> <p>◦ tcp wrapper와 iptables에 의한 방화벽 설치를 통해 그리드 자원의 보안상 안정성 유지</p> <p>◦ Globus toolkit-2.4, MPICH_G2, OpenPBS-2.3.16의 연동을 통한 그리드 환경 구축</p> <p>◦ 장시간의 부하 테스트를 거쳐 시스템 안정성 확인</p> <p>◦ latency와 Linpack 벤치 마크를 통해 클러스터 자체의 대략적인 네트워크 성능 확인</p> <p>◦ 시스템간 네트워크 latency 및 bandwidth 측정</p> <p>◦ 유저의 주문이 있을 경우 혹은 시스템 장애 발생시 신속한 대응을 할 수 있도록 전용 웹 페이지 운영(http://gridtest.gridcenter.or.kr)</p> <p>◦ 고성능 저비용 컴퓨팅 환경의 구축, 활용 및 운영 사례의 모범 사례가 될 것</p> <p>◦ 이번 자원 제공을 통한 그리드 환경 인프라 구축으로 인해서, 세계적 핫이슈인 그리드 프로젝트를 선도하기 위한 국가 기술발전에 이바지 할 것으로 사료됨</p>						
색인어 (각 5개 이상)	한 글	그리드, 클러스터, 분산 컴퓨팅, 글로버스, 고성능				
	영 어	grid, cluster, distributed computing, globus, high performance				

요 약 문

I. 제 목

클러스터 자원제공을 통한 컴퓨팅 그리드 구축

II. 연구개발의 중요성 및 목적

과거 슈퍼 컴퓨터를 이용한 계산들이 근래에는 클러스터를 이용하여 해결하는 경우가 많아졌다. 슈퍼 컴퓨터와 성능의 차이는 크지 않지만 시스템 자원의 사용 비용 면에서는 매우 저렴하게 이용할 수 있기 때문이다. 이는 최근의 개인용 컴퓨터의 대중화와 부품 가격의 하락 그리고 네트워크 장비의 고성능화가 급속히 이루어졌기 때문인데, 규모가 크지 않은 대학의 실험실이나 연구소 등에서 큰 컴퓨팅 파워를 필요로 하는 BT, NT, ST 등의 기술 연구를 위해 클러스터 컴퓨팅 시스템을 요긴하게 쓸 수 있는 환경이 만들어지고 있다.

한편, 최근에는 단순 분산 시스템이나 www을 아우르는 infrastructure인 그리드 연구가 활발히 진행되고 있다. 본 연구는 클러스터 시스템들을 다시 광역적으로 엮어 이기종 시스템간의 연동 컴퓨팅을 구현하는 KISTI의 국가 그리드 기반 구축 프로젝트의 일부로서 자원의 안정적인 제공의 임무를 띄고 있다.

이를 위해 CPU대 네트워크의 병목현상을 해결하기 위하여 기존 보유 클러스터의 네트워크 장비를 업그레이드 하고 globus toolkit을 주축으로 하는 그리드 연동 필수 소프트웨어의 세팅을 완료하는 것을 일차적인 목표로 하고, 프로젝트에 참여하는 타 기관과의 유연한 연동을 추구함으로써 초대형 고성능 컴퓨팅 자원의 구축에 이바지 하는 것을 목표로 한다.

III. 연구개발의 내용 및 과정

이번 과제에 사용하게 된 Pentium4 클러스터는 Intel 1.5GHz CPU 탑재한 노드 16가 100Mbps fast ethernet 네트워크로 연결되어 있던 시스템으로서 기존에 실험실의 연구에 활용되었던 것이다. 이를 Nortel사의 BayStack 380-24T 기가비트 스위치와 Intel 기가비트 랜카드 등을 이용하여 네트워크를 업그레이드 시키는 한편, 3.06GHz(dual)급 CPU를 갖고 있는 로긴 서버를 새로이 설치하고 로긴 서버의 /home을 각 계산 노드가 NFS로 공유하도록 하여 관리의 편의성을 도모하였다.

시스템의 O/S로는 경험적인 검증을 통해 클러스터 시스템 운영에 적합하다고 판단된 Redhat 7.3을 O/S로 탑재하였다. 클러스터간 연동의 핵심 요소인 그리드 미들웨어로는 globus toolkit2.4을 설치하였고, local job scheduler로서는 OpenPBS-2.3.16을 사용하였으며, MPI 프로그램으로는 grid-enabled 기능을 갖고 있는 MPICH_G2를 설치하였다. 또한 단일 클러스터로서의 사용 편의성을 위해서 구성 노드간 rsh를 설정하였고, Python 스크립트 툴인 clustexec를 자체적으로 개발하여 일괄 작업에 유용하게 사용할 수 있도록 하였다.

그리드에 의한 자원 연동은, 미리 교환된 인증서를 보유하고 있다는 가정 하에 그리드 풀 안에 있는 자원을 자유롭게 이용할 수 있기 때문에, 네트워크 침입자에 대한 방화벽을 튼튼히 쌓아야 하는데, tcp wrapper와 iptables을 이용하여 이를 해결하였다.

시스템의 설정을 완료한 후에는 그리드 유저들의 본격적인 이용을 하기 전에 그리드 툴이 정상적으로 세팅 되어졌는가와 시스템이 안정적으로 동작을 하는지의 여부를 확인 하였고, 네트워크의 latency를 측정하는 Netpipe와 시스템의 전체적인 성능을 측정하는 HPL등으로 간단한 벤치 마크를 실시하였다.

IV. 연구개발 결과

위의 과정을 통하여 이번 그리드 프로젝트의 핵심적인 부분인 클러스터 기반 컴퓨팅 자원의 원활한 제공 준비가 완료되었음을 확인할 수 있었다.

VI. 기 대 효 과

본 연구는, 개발 사업적인 측면에서, 고성능의 저렴한 컴퓨팅 자원을 필요로 하는 대학원 실험실 레벨의 연구 기관에 시스템 구축 및 운용의 모범 사례가 될 것이며, 협업적인 측면에서는, 국가 그리드 프로젝트의 기본 인프라인 안정적인 자원 제공이라는 역할을 수행함으로써, 조만간 불어 닥칠 IT기술의 핫이슈인 그리드 연구에 앞서 국가 기술의 초석을 다지는데 일정 역할을 할 것이라 사료된다.

Summary

I . Title

Implementation of a Computational Grid based on Support from Cluster Resource

II. Objective of the study and its importance

In these days, one tends to use cluster computing power in instead of super computer. Because the cost of cluster is much lower than that of super computer though the computing performance of cluster is not as different as that of super computer. This is possible by the popularization of PCs, decline of the components and increase of performance of network device. The laboratory of university or small scale institute which is needs large scale computing power for BT, NT, ST technologies, can use cluster computing system essentially.

The research of Grid which is the infrastructure that unites simple distributed system covering with www, become very active recently. In this study, we provide stable cluster system for the part of the project of national grid basis construction by KISTI. This project realizes the collaboration of widely distributed clusters and supercomputers. For these purpose, the bottleneck between CPU and network must be reduced by upgrade of network devices. And essential software for grid connection must be set for the first goal. By the connection with other institution participates in this project, we can help to construct the large scale and high performance computing.

III. Content of the study and procedure

The cluster which be used in this work is composed of 16 nodes of 1.5GHz-Intel Pentium4 CPU and connected each other on 100Mbps fast ethernet network environment. It has worked for laboratory parallel algorithm application program. Now this system was upgraded to Gigabit connectivity in Network environment and 3.06GHz(dual) Pentium4 login server was newly installed. This login server takes rolls of NFS mount server for /home directory for local nodes as well as gate-point to the cluster.

RedHat linux 7.3 was adopted for O/S because it was estimated stable one for cluster computing by long experiences. The main toolkit for grid environment, globus-2.4 was installed and OpenPBS-2.3.16 was set up as local job scheduler, and MPICH_G2 which is grid-enabled MPI library was installed with device option of globus2. To help system managements easily, a simple script 'clustexec' was made with python. This scripts allow one command to do automatic consecutive work to local nodes with threading.

The collective resource allocation system permit resource usage freely only to pre-authorized one based on GSI. If local area of grid pool was invaded, the whole resource would be stolen.

IV. Result of the study and next object

In this work, and it can be shown that supporting from cluster resources is ready to work for implementation of a computational grid project.

V. Expected Effects

The system management and operating know-how would be good examples for laboratory level, which requires cluster system of high performance computing in low cost. In the view of resource collaboration, the basic infrastructure of grid environment was established through stable supply of computing power. It would make the position of national grid technology higher against rapid varying IT environment.

Contents

Chapter I Introduction	1
section 1 Importance of the study	1
section 2 Objectives of the study	2
Chapter II Implementation of cluster for testbed only	4
section 1 Specification of cluster	4
section 2 Details of system setting and firewall policy	6
a. System setting	6
① kernel compile	6
② RSH connection	8
③/etc/hosts file setting and adding user group	9
④ usage of NFS(Network File System)	9
⑤ setting of NIC driver option	9
⑥ the other settings	11
b. Firewall setting	11
c. Python script tool for management	11
Chapter 3 Build up grid environment in globus-2.4	11
a. Installation of OpenPBS	11
b. Installation of globus-2.4	11
① GPT(Globus Package Toolkit)	11
② installation of Globus-2.4	11
c. Installation of MPICH_G2	11
Chapter III System operating and stability test	12
section 1 Operating test of grid environments	12
a. Globus	12

b. MPICH_G2	22
c. OpenPBS	3
section 2 Stability test	42
section 3 Benchmark test	52
a. Netpipe test	5
b. HPL test	8
c. coallaboration with other testbeds	6 2
Chapter IV Conclusion and expected effects	9 2

목 차

제 I 장 서론	1
제 1 절 연구의 중요성	1
제 2 절 연구의 목적	2
제 II 장 테스트 베드 전용 클러스터의 구성	4
제 1 절 시스템의 구성과 사양	4
제 2 절 시스템 세부적인 설정 및 보안 정책	6
가. 시스템 세팅	6
① kernel compile	6
② RSH 연결	8
③/etc/hosts 파일 설정 및 사용자 계정 운영	9
④ NFS(Network File System)의 이용	9
⑤ NIC 드라이버 옵션 설정	9
⑥기타 설정	10
나. 방화벽 설정	10
다. 시스템 관리를 위한 스크립트 tool	11
제 3 절 Globus 2.4를 이용한 그리드 환경 구성	4
가. OpenPBS의 설치	11
나. globus 2.4의 설치	16
① GPT(Globus Package Toolkit)의 설치	16
② Globus-2.4의 설치 및 세팅	16
나. MPICH_G2의 설치	20
제 III 장 시스템의 동작 및 안정성 테스트	21
제 1 절 그리드 환경 동작 테스트	21
가. 글로버스	21

나. MPICH_G2	22
다. OpenPBS	23
제 2 절 안정성 테스트	24
제 3 절 클러스터 시스템의 벤치마크 테스트	25
가. Netpipe 테스트	25
나. HPL 테스트	26
다. 타 시스템과의 연동 테스트	26
제 장 결론 및 과급 효과	29

표차례

<표 1> 국가 그리드 프로젝트에 연동되는 기관 및 시스템 사양	4
<표 2> 서울대학교 Cluster 로그인 노드 및 네트워크 장비 사양	5
<표 3> 서울대학교 Cluster 계산 노드 자원 사양	5
<표 4> iptables를 이용한 방화벽 스크립트	11
<표 5> 시스템 관리를 위한 스크립트(clusterexec)	31
<표 6> clusterexec의 이용법	4
<표 7> grid-mapfile의 내용	20
<표 8> SSC4, Jupiter, Venus를 연동했을 때의 RSL file	32
<표 9> HPL test	27
<표 10> 타 시스템과의 연동시 HPL 성능 테스트 결과	27
<표 11> 타 시스템과의 연동시 테스트베드 사이의 TCP/IP latency	28

그림 차례

<그림 1> 광역 그리드 컴퓨팅	2
<그림 2> 그리드 프로젝트 구성도	3
<그림 3> Netpipe test	8

제 I 장. 서론

제 1 절 연구의 중요성

과거에는 슈퍼컴퓨팅을 위한 고속의 연산을 위해 고가의 장비를 이용해서 구축하였으나 막대한 비용의 제약이 따르므로 근래에는 주변에서 쉽게 구할수 있는 컴퓨터 장비들을 연결하여 클러스터링 시스템을 만들어 슈퍼컴퓨팅의 성능을 내고자 하는 노력들이 있어왔다. 특히 최근에는 일반 개인용 컴퓨터의 부품이 성능 면에서 비약적인 발전을 거듭한 한편, 비용 면에서는 대량 생산 따른 가격하락으로 주변에서 손쉽게 저렴한 가격으로 구입할 수 있게 되었다. 이에 따라 큰 규모의 연구소나 기관이 아니라 하더라도 약간의 노력을 기울이면 기존의 대형 high performance supercomputer 수준의 계산능력을 갖는 고성능 클러스터링 시스템을 구축할 수 있게 되었다. 더욱이 예전의 병렬 클러스터링 시스템에서는 CPU의 계산 성능을 네트워크 시스템이 따라가지 못해 병목 현상이 심하게 발생하였는데, 최근에는 네트워크 장비 역시 많은 성능 향상을 이루었기 때문에, 각 노드 간의 무거운 통신 패턴이 오가는 자유도가 큰 응용 문제를 푸는데 있어서도 큰 효용성을 갖게 되었다. 그리고 이러한 클러스터 자원은 고성능의 수치 해석을 필요로 하는 요즘의 연구 추세와 맞물려 대학원의 실험실 규모에서 자작 시스템을 구축하여 BT, NT, ST 등의 차세대 핵심 국가 기술을 위해 요긴하게 쓸 수 있는 환경이 만들어지고 있다.

한편, 이러한 클러스터 시스템이 일반적으로 운영 및 관리를 위한 중심점이 있고, 시스템 자원의 활용이 순수하게 이 중심점을 위해 사용되어지는데 반해, 그리드 시스템은 이러한 클러스터 시스템을 광역적으로 연동시켜 보다 많은 정보 처리량, 정보 보급의 다양성, 자원 활용의 분권화, 어플리케이션의 유연성 등이 종합적으로 고려되어진다. 그리드는 기존의 단순 분산 시스템이나, WWW을 아우르는 기반 infrastructure이며, 프로세서, 스토리지 등 다양한 자원을 광역적으로 공유하기 때문에, 그리드 풀에 연동되는 수천 수만개의 노드들을 이용하여 훨씬 규모가 큰 고성능 컴퓨팅 자원을 제공 할 수 있고, 풀에 연동되는 자원의 대상이 본질적으로 크로스 플랫폼의 속성을 가지고 있기 때문에, 다양한 형태의 서비스를 제공 받을 수 있다. 이미 세계적인 IT업체들인 IBM, 컴팩, 선마이크로시스템즈 등은 향후 IT 산업의 핫이슈로 부각할 것에 대비, 새로운 사업본부 발족하고(IBM) 연구소를 설립하고(컴팩) 새 소프트웨어 발표하는(썬)등, 세력 확대를 꾀 하고 있다.

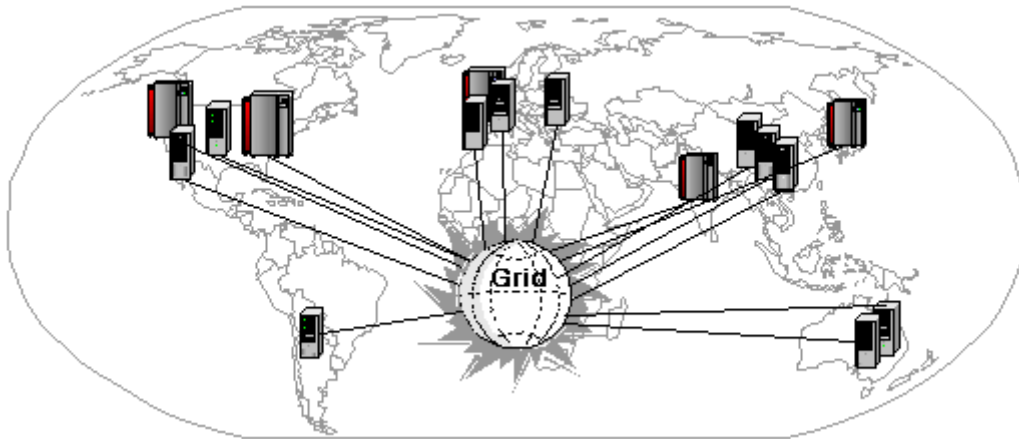


그림 1 광역 그리드 컴퓨팅

본 연구는 강력한 컴퓨팅 자원을 제공하는 단일 클러스터의 운용에 관한 노하우를 축적할 수 있다는 점과 그리드 프로젝트라는 세계적 흐름 속에 국내 그리드 연구 발전의 초석이 될 수 있다는 점에서 매우 중요한 의미를 가진다고 할 수 있다.

제 2 절 연구의 목적

본 연구의 핵심 목표는 클러스터 자원을 제공함으로써 KISTI에서 주관하고 있는 국가 그리드 프로젝트의 기본 인프라를 구축하는 것이다. 이러한 기본 인프라 구축은 시스템의 안정적인 자원 제공이 필수적인데, 이를 위해 그리드 운용에 필요한 필수 소프트웨어의 설치 및 세팅, 정기적인 관리가 요구 된다. 또한 그리드 연구 자체가 고성능 자원 제공의 의미를 내포하므로 각 그리드 풀을 구성하고 있는 분산 자원 역시 높은 컴퓨팅 퀄리티의 자원을 제공해야 한다. 이를 위해 시스템의 성능 향상 튜닝이 필요하며, CPU 성능대 네트워크 병목 현상을 해결하기 위하여 네트워크 장비의 업그레이드가 필요하다.

분산 자원 간의 유연한 연동을 위해서는 globus toolkit을 주축으로 하는 middle ware와 네트워크 시스템의 설정 등을 통해 광역적으로 흩어져 있는 자원들을 하나의 단일 계산 망으로 묶어야 한다. end-user가 의뢰한 작업을 웹 등을 통한 편리한 인터페이스를 통해 globus에 전달하고 globus는 다시 로컬 배치 스케줄러에 명령을 내림으로써, 응용 어플리케이션의 실행이 시스템의 하드웨어적인 자원을 유연하게 활용할수 있도록 한다. 이 부분은 그리드 협업 과제의 다른 연구팀들 연구 성과가 요구된다.

사용자 지원 차원에서 로컬 클러스터의 이상이 발견되었을 시 신속한 대응을 할 수 있는 장치가 필요한데, 로컬 클러스터 자원 전용 웹을 운영하여 의견 교환을 이루고자 한다.

마지막으로 다른 기관의 테스트베드 제공 팀들과의 교류를 통해 안정적인 자원 제공에 대한 문제점을 공유하고 해결책을 모색하는 것이 필요하다.

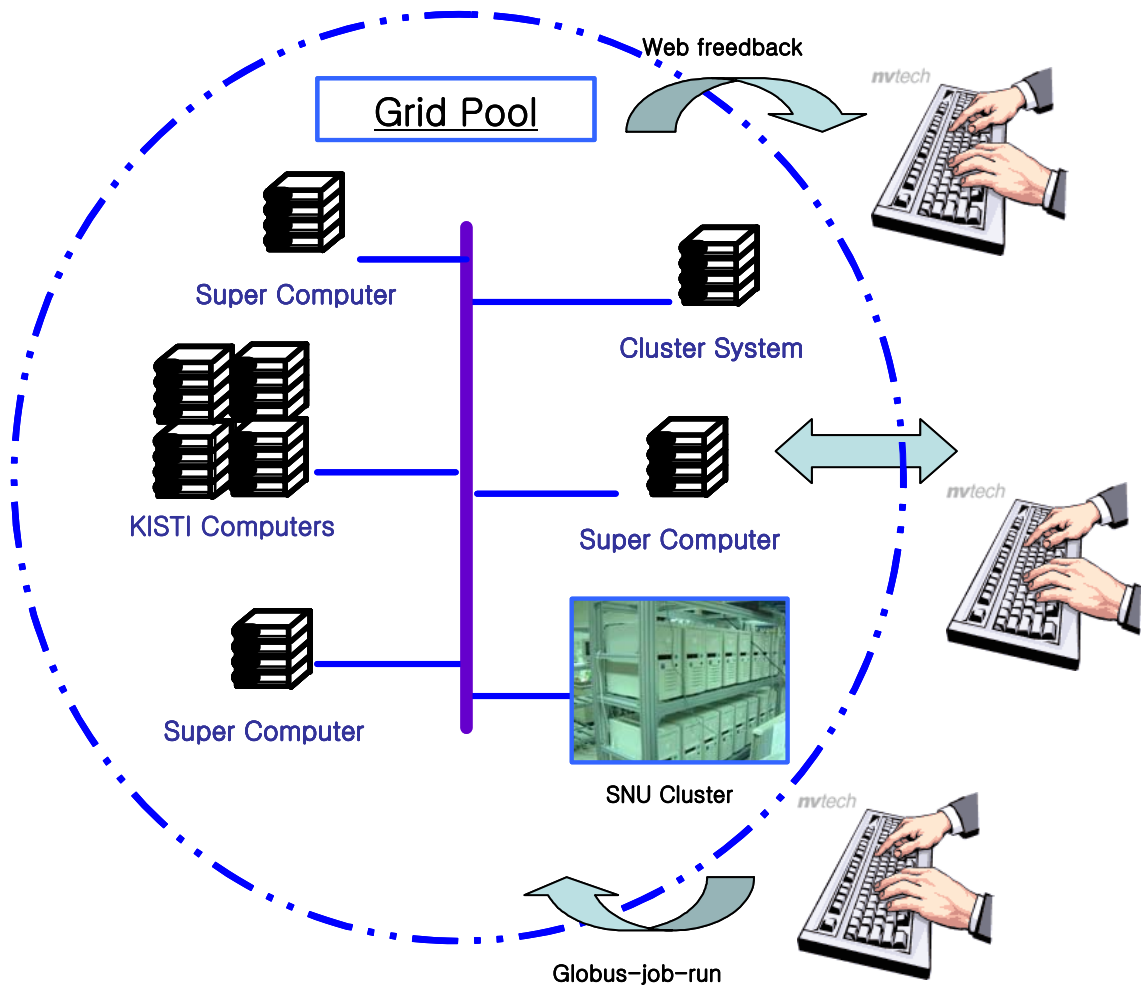


그림 2 클러스터 연동을 통한 그리드 pool 구성도

제 II 장. 테스트 베드 전용 클러스터의 구성

제 1 절 시스템의 구성과 사양

이번 과제에서 컴퓨팅 그리드 풀에 제공될 클러스터로, 기존에 실험실의 응용문제 계산용으로 사용했던 Pentium-4 시스템을 네트워크 성능을 개선하여 내놓게 되었다. 이 시스템은 본 연구실에서 직접 제작했던 것으로서, 1.5 GHz Intel Pentium-4 CPU를 가지고 있는 16개의 노드가 100Mb fast ethernet 네트워크로 구성되어 있었다. 여기에 Nortel사의 BayStack 380-24T 기가비트 지원가능 스위치를 물려서 클러스터 내부의 네트워크를 획기적으로 향상시키고, 클러스터의 실제 계산 수행 노드 외에 별도의 로그인 서버를 구성하여 로그인 관리 및 job management의 편의성을 도모하였다. 로그인 서버에는 Network File System(NFS)를 이용하여 /home을 공유 가능한 상태로 열어 두었고, 각 노드에서는 이를 공유하여 관리의 편의성을 도모하였다. 또한 기존의 구성은 실험실 내부에서 지역적인 계산 수행만을 목적으로 이용되었기에 각 계산 노드들은 사설 IP가 할당되어 있었으나, 이번 과제에서는 타 기관과의 연동이 필요하므로 각 노드에도 real IP를 할당하였다. 테스트베드의 구체적인 사양은 표2,3과 같다. 표1은 이번 그리드 자원 제공에 참여한 타 기관들을 나타낸것이다.

자원	사양
KSITI, p630, IBM	power4 1GHz, 4CPU, 2node
포항공과대학교, v2500 HP	pa8600 875MHz, 16CPU, 16node
동명정보대학교, SP2, IBM	power2 120MHz, 26CPU, 26node
명지대학교, Origin3400, GSI	RI 2000 400MHz, 16CPU, 16node
전북대학교, SP, IBM	power3 200MHz, 64CPU, 32node
금호그룹, SGI	RI 4000 500MHz, 6CPU, 6node
KISTI, linux cluster	Intel Pentium4 1.7GHz, 64CPU, 64node
KISTI, linux cluster	Intel Pentium4 2.0GHz, 16CPU, 16node
포항공과대학교, linux cluster	Intel Pentium3 733MHz, 24CPU, 24node
포항공과대학교, linux cluster	Intel Pentium3 900MHz, 32CPU, 16node
부산대학교, linux cluster	AMD Athlon 1.5GHz, 26CPU, 16node
서울대학교, linux cluster	Intel Pentium4 1.5GHz, 16CPU, 16node

표 1 국가 그리드 프로젝트의 그리드 풀에 연동되는 기관 및 시스템 사양

테스트베드 로그인노드 사양		
Architecture		Linux (kernel 2.4.20-smp)
CPU	CPU	Pentium-4 (dual)
	Clock	3.06 GHz
RAM	Total	1 GB
Hard Disk		60GB (7200rpm)
Network 세팅	Hostname	ssc4
	Domainname	snu.ac.kr
	IP address	147.46.128.210
	방화벽	iptables, tcp wrapper
구성 네트워크 하드웨어 사양		
스위치		Nortel, BayStack-24T Gigabit auto sensing, scalability to 512Gbps(theoretical)
연결 lan 케이블		LG 전선, Category 6 (Gigabit용)
랜 카드		Intel, 10/100/1000 PIC (e1000 module)

표 2 서울대학교 Cluster 로그인 노드 및 네트워크 장비 사양

테스트베드 계산노드 사양		
Architecture		Linux (kernel 2.4.20)
CPU	CPU	Pentium-4
	Clock	1.5GHz
	CPU/Node	1CPU/node
	Node 수	16
	총 CPU 수	16
	Peak Perf	
RAM	RAM/Node	1024MB/node
	Total	16GB
Hard Disk		40GB/node
Network	Hostname	ssc401~ssc416
	Domainname	snu.ac.kr
	IP address	147.46.128.194 ~ 147.46.128.209
	방화벽	iptables, tcp wrapper
S/W	Queue	OpenPBS
	MPI	MPICH_G2
	OS	Linux (Redhat7.3)

표 3 서울대학교 Cluster 계산 노드 자원 사양

제 2 절 시스템 세부적인 설정 및 보안 정책

가. 시스템 세팅

적절한 Hardware를 선택하고 각각의 Cluster Node의 물리적 구성이 완성된 후에 우선적으로 해야할 일은 Linux 배포판을 설치하는 것이다. 현재 여러 가지 버전이 나와 있지만 그 중에서 가장 많이 사용되고 있는 것이 Redhat 계열이다. 특히 7.3버전은 2002년 상반기에 출시된 이래 클러스터용 OS로 많이 쓰여지고 있고 경험적으로 안정성이 확인된 버전이다.

① kernel compile

리눅스 커널은 내달 발표될 예정인 2.6버전이 대기중에 있고, 현재 2.4.x대의 안정적인 버전들이 나와 있으나 역시 경험적으로 많이 사용되어져 온 안정버전인 2.4.20을 선택하였다. 커널을 새로 컴파일 하기위해서 다운 받은 커널 압축 소스를 /usr/src 밑에 풀고, 다음의 순서를 따라간다.

```
$make menuconfig  
$make bzImage  
$make modules  
$make modules_install  
$make install
```

우선 새 커널에 어떠한 기능들을 포함시킬것인지를 설정해야 하는데, 'make menuconfig'를 치면 다음과 같은 화면이 나타난다. 각 메뉴는 하위 설정 메뉴를 가지고 있다. 특히 눈여겨 보아야 할 항목은 Networking options 항목과 Network device support인데, Networking options의 하위메뉴에서 IP:netfilter configuration을 모듈로서 탑재해야 나중에 언급할 iptable을 사용할 수 있다(시스템을 스쳐가는 패킷을 필터링하려면 네트워크에 어느정도의 부하가 걸리기 때문에, 커널 수준에서 처리한다). 또한 Network device support의 하위메뉴에서 현재 자신이 사용하고 있는 NIC를 선택해야 네트워크를 사용하기 위한 드라이버를 탑재하게 된다. 만일 추후에 NIC가 교체될 예정이라면 해당 드라이버를 모듈 형태로 선택한다.

```

Code maturity level options --->
Loadable module support --->
Processor type and features --->
General setup --->
Memory Technology Devices (MTD) --->
Multi-device support (RAID and LVM) --->
Networking options --->
Telephony Support --->
ATA/IDE/MFM/RLL support --->
SCSI support --->
Fusion MPT device support --->
I2O device support --->
Network device support ---->
Amateur Radio support --->
IrDA (infrared) support --->
Input core support --->
File systems ---->

```

'make bzImage'는 위에서 설정된 사항을 따라서 실제로 커널을 컴파일하는 과정이다. 성공적으로 컴파일된 압축 커널은 /boot 밑에 vmlinuz-2.4.20로 생성된다. 'make modules'과 'make modules_install'은 기본 압축 커널 외에 'make menuconfig'시 모듈 형태로 탑재되었던 것들에 대한 설치를 한다. 마지막으로 'make install'은 커널 컴파일을 마무리한다. 커널 컴파일 후에는 /etc/lilo.conf 파일을 수정하여 새로 생성된 커널 이미지를 불러들일 수 있도록 해야 한다. 다음과 같이 vmlinuz-2.4.20에 대한 이미지를 추가하고 default label을 2.4.20으로 교체한다.

```

default=2.4.20
#default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
linear

image=/boot/vmlinuz-2.4.18-3smp
    label=linux
    read-only
    root=/dev/hda1
image=/boot/vmlinuz-2.4.20
    label=2.4.20
    read-only
    root=/dev/hda1

```

마지막으로 다음을 실행 시켜 재부팅시 새로 설정된 커널이 올라오도록 한다.

```
#/sbin/lilo
```

② RSH 연결

기본적으로 클러스터 형태의 시스템은 계산 자원으로의 접근 편의성을 위하여 ssh 혹은 rsh을 설정하는데 SSC4 시스템은 rsh을 이용하였다. 이를 위하여 각 계정의 홈 밑에는 로컬 클러스터 전체의 노드 목록과 username이 기록된 .rhosts 파일을 넣어두었으며 root의 rsh를 위해서 /etc/pam.d/rlogin, /etc/securetty 파일등을 수정해주었다.

다음은 sjkim이란 user의 홈디렉토리에 설정된 .rhosts 파일이다. ssc400 ~ ssc416의 호스트의 sjkim 유저에 대하여 rsh 접속을 허용함을 나타내고 있다.

```
ssc400 sjkim
ssc401 sjkim
..
..
ssc416 sjkim
```

root의 rsh 연결을 위해서는 /etc/securetty 파일의 마지막에 rlogin, rsh를 적어주고, /etc/pam.d/rlogin의 2번째 4번째 줄을 첫번째, 두 번째로 각각 올려주어야 한다. 이들 파일의 권한은 644로 설정되어져야 한다.

```
vc/1      // securitty file
..
..
tty11
rlogin
rsh
```

```
auth      sufficient /lib/security/pam_rhosts_auth.so      //rlogin file
auth      required /lib/security/pam_securetty.so
auth      required /lib/security/pam_nologin.so
auth      required /lib/security/pam_env.so
auth      required /lib/security/pam_stack.so service=system-auth
account    required /lib/security/pam_stack.so service=system-auth
password   required /lib/security/pam_stack.so service=system-auth
session    required /lib/security/pam_stack.so service=system-auth
```


③/etc/hosts 파일 설정 및 사용자 계정 운영

한편 /etc/hosts에는 불필요한 DNS 쿼리를 줄이고, 효율성의 증대를 위하여 로컬 클러스터의 노드 목록 뿐 아니라, N*Grid의 모든 호스트 이름을 등록하였다. 또한 본 과제의 목적이 그리드 응용문제 유저들에게 자원을 제공하는 것이 목적인만큼, 시스템의 주 사용층은 이들 유저와 KISTI의 운영진들이 된다. 따라서 전자에게는 ngrid 그룹으로서 계정을 생성하고, 후자에게는 others 그룹으로서 계정을 생성시킴으로서 계정 관리를 체계적으로 정리하였다.

④ NFS(Network File System)의 이용

수행 프로그램의 입력 파일과 같이 모든 계산 노드가 동일한 읽기 작업을 수행할 경우 NFS는 매우 편리하게 사용되어질 수 있다. NFS설정을 위해선 서버 노드의 portmap, rpc.mountd, rpc.nfsd, rpc.nfsd, rpc.rquotad, rpc.lockd, rpc.statd 등의 서비스가 필요하다. portmap은 RPC(remote procedure call)프로그램을 TCP/UDP포트에 연결시켜주는 역할을 하는데, RPC를 이용하는 프로그램이 시작되면 그 프로그램은 자신이 제공하는 서비스와 자신이 사용하는 포트를 portmap에 등록을 하고, 클라이언트는 portmap에 문의해 원하는 서버에 접근할 수 있는 방법을 알아낼 수가 있다. portmap은 RPC 호출을 만들기 위해서 가동된다. RPC 서버가 가동되면 포트맵은 listen되고 있는 포트가 무엇이고 서비스를 할 RPC 프로그램 번호가 무엇인지를 말해준다. 클라이언트가 포트번호를 주어 RPC를 호출하면 어디에서 RPC 패킷이 보내졌는지를 포트번호를 결정해서 서버 머신은 첫 번째 포트맵에 접속할 것이다. 포트맵은 항상 RPC 서버를 가동시키기 전에 실행되고 있어야 한다. rpc.mountd는 NFS 마운트 프로토콜로서 외부에서 요청이 들어오게 되면 자신의 한 부분을 클라이언트들이 사용할 수 있도록 공유를 시켜주는 역할을 하는데, 리눅스의 경우 NFS 서버가 될 노드의 /etc/exports에 보통 다음과 같이 설정을 해줌으로서 147.46.128.* 대역에 있는 클라이언트가 /home을 읽고 쓸 수 있도록 할 수 있다. 클라이언트의 접근 권한이라던가 동기화 등에 대한 옵션을 지정할 수 있다.

/home	147.46.128.*(rw,no_root_squash)
-------	---------------------------------

일단 rpc.mountd에 의해 마운트가 되었다면 클라이언트는 /etc/exports의 내용에 따라 다양한 작업을 할 수 있다. 이때 클라이언트가 작업을 수행하면서 서버쪽 파일시스템 상에 무엇인가를 요구한다면 rpc.nfsd이 이를 맡아서 처리를 해준다. rpc.rquotad는 원격 쿼터 서버로서 로컬 유저의 쿼터를 리턴한다. rpc.lockd는 파일 잠금을 통해 여러 명이 동시에 한 파일을 수정하는 것을 막아준다. rpc.statd는 rpc.lockd와 함께 사용되는데, NFS서버가 크래쉬되거나 리부팅되었을 때 잠겼던 것을 다시 복구시키는 역할을 한다.

⑤ NIC 드라이버 옵션 설정

구성된 기가비트 네트워크의 성능을 최대한 활용하기 위해서는 장치가 지원하는 가장 최신의 드라이버 모듈을 사용하여야 한다. SSC4 시스템은 Intel Gigabit NIC를 갖고 있기 때문에, e1000 모듈을 사용한다. 그런데 최근에 커널 2.4.20에 탑재되어 있는 e1000 드라이버보다 더 최신인 e1000-4.4.19 버전이 나왔기 때문에, 제조사측 홈페이지에서 이를 다운받아 수동으로 설치하였다. 한편 /etc/modules.conf에는 다음과 같은 optoinn을 추가한다. RxIntDelay, TxIntDelay는 각각 패킷을 보내고 받을때 사용할 버퍼의 양을 나타내고, InterruptThrottRate는 오류가 있는 패킷을 검사하는 빈도를 나타내는데, 보통 클러스터의 네트워크 환경에서는 모두 0으로 세팅하는 것이 latency와 bandwidth의 성능 향상을 가져온다.

```
options e1000 RxIntDelay=0 TxIntDelay=0 InterruptThrottRate=0
```

⑥기타

globus-2.4, MPICH_G2 등을 포함하여 소프트웨어 설치하는 기본적으로 GNU gcc-2.95.3 버전을 이용하였다.

나. 방화벽 설정

그리드 시스템은 흩어져 있는 광역 자원을 미리 교환된 인증서를 보유하고 있다는 가정하에 자유롭게 이용할 수 있도록 하는 것이 핵심이다. 이러한 특성은 그리드 풀을 구성하고 있는 로컬 자원의 일부분이 외부의 침입을 당하게 되면, 최악의 경우 허가되지 않은 침입자에 의해 전체 자원의 사용권을 내주게 되는 결과를 초래할 수도 있는 맹점이 있다. 따라서 안전한 그리드 시스템을 운용하기 위해선 각 자원의 보안 설정을 공고히 할 필요가 있다. 본 테스트베드의 방화벽은 tcp wrapper 혹은 iptables를 이용하여 설정되었으며 iptables을 이용할 시 방화벽에 대한 기본 정책은 다음과 같이 정하였다.

- FS의 세팅과, rsh 등을 위해서 클러스터 내부의 노드 서로간에는 모든 패킷을 허용한다
- 그리드 풀에 속하게 되는 타 기관의 자원에 대해서는 gsissh, gsincftp, globusrun 등 그리드 운용을 위해서 꼭 필요한 포트인 2119, 2135, 2222, 2811, 20000~25000번에 대하여만 패킷을 허용한다. (풀에 속하게 되는 자원이라도 telnet, ssh다른 포트는 차단한다.)
- 시스템 administrator가 원격으로 접속하여 시스템을 관리할 수 있도록 관리자 컴퓨터의 IP에 대하여 22번(sshd) 포트를 개방한다.

- 관리자의 컴퓨터에도 방화벽을 공고히 설치하여 관리자의 컴퓨터를 통해 그리드 자원에 접근할 수 있는 가능성을 제거한다.

한편, 그리드 풀에 참여하게 되는 타 기관의 자원의 노드가 수백여 개에 달하기 때문에 일일이 iptable에 적어주기가 번거로우므로 간단한 스크립트를 이용하여 grid.txt라는 파일로부터 IP 리스트를 불러들여 설정을 하도록 하였다. iptables를 이용하기 위하여 kernerl 2.4.20 버전의 network options의 옵션을 다시 설정하고 커널을 rebuild 했으며, 위의 정책이 반영된 다음과 같은 스크립트를 rc.firewall로 저장하고 /etc/rc.d/rc.local에서 실행되도록 함으로써 시스템이 시작할 때마다 자동 설정되도록 하였다.

```
#!/bin/bash
/sbin/depmod -a
/sbin/insmod ip_tables
/sbin/iptables -F

# Accept : NFS(2049), portmap(111), rsh for ssc401~ssc416
host=`cat /etc/rc.d/ssc4_list.txt`
for i in $host;
do
/sbin/iptables -A INPUT -p tcp -s $i -j ACCEPT
done

# Accept : N*Grid TestBed
grid=`cat /etc/rc.d/grid.txt`
for a in $grid;
do
/sbin/iptables -A INPUT -p tcp -s $a --dport 2119 -j ACCEPT
/sbin/iptables -A INPUT -p tcp -s $a --dport 2235 -j ACCEPT
/sbin/iptables -A INPUT -p tcp -s $a --dport 2811 -j ACCEPT
/sbin/iptables -A INPUT -p tcp -s $a --dport 20000:25000 -j ACCEPT
done

# Authentication for Administrator IP (ssh)
/sbin/iptables -A INPUT -p tcp -s 147.46.120.243 --dport 22 -j ACCEPT

# Drop the other packets
/sbin/iptables -A INPUT -p tcp --syn -j DROP
```

표 4 방화벽 설정을 위한 스크립트

다. 시스템 관리를 위한 스크립트 tool

클러스터의 관리 툴에는 몇 가지가 있으나 좀 더 효과적이고 범용적인 활용을 위하여 Python을 이용하여 간단한 스크립트를 개발하였다. 이는 기존의 스크립트에 비해 편의성을 높였을 뿐 아니라, 구성 노드의 수가 많을 때에도 신속하게 작업을 수행 할 수 있도록 thread 기능을 옵션으로 채택할 수 있도록 제작되었다.

```

#!/usr/bin/python

import sys, thread, os, time, threading
arg = sys.argv[1:]
argdic = {}
def printhelp():          # print help
print "syntax: " + sys.argv[0]
print """ [Options]
-c COMMAND                (in quotes for multiple word commands)
-n NODE_LIST_FILENAME    (default is /etc/NODELIST)
-t TEST_NODE              (run COMMAND on TEST_NODE for test)
-s SHELL                  (default is /usr/bin/rsh)
-T                        (run threaded)
version 1.1 <jwpark@aeroguy.snu.ac.kr>
"""

return
def parseNodelistFile( filename ):    # parse nodelist file
try:
f = open (filename)
except IOError:
return None
nodes = []
while 1:
node = f.readline()
if node == '' : break
if node[-1] == '\n':
nodes.append(node[:-1])
else: nodes.append(node)
return nodes
# Parse arguments
for a in arg:
if a[0] is '-':
try:
if arg.index(a)+1 is len(arg): r = ''
else : r = arg[ arg.index(a)+1 ]
try:
if r[0] is '-': r = ''
except: pass
except:
break
argdic[a[1:]] = r

if argdic.has_key('c'):
command_string = argdic['c']
if command_string == '':
sys.exit(1)
else:

```

```

printhelp()
sys.exit(1)
if argdic.has_key('s'):
    shell = argdic['s']
else: shell = '/usr/bin/rsh'          # default shell is rsh
if argdic.has_key('n'):
    node_list_file = argdic['n']
else: node_list_file = "/usr/local/bin/NODELIST" # default file is /etc/NODELIST
if argdic.has_key('T'):
    threaded = 1
else: threaded = 0
if argdic.has_key('t'):
    test_node = argdic['t']
else: test_node = None

# main run
if test_node is None:
    nodes = parseNodelistFile( node_list_file )
if nodes is None:
    print "Invalid nodelist file: probably file doesn't exist"
    sys.exit(1)
else:
    nodes = [test_node]
    class Thr(threading.Thread):
    def __init__(self, node):
        self.node = node
        threading.Thread.__init__(self)
    def run(self):
        comline = shell+' '+self.node+' 'W
        + ""s""%(command_string)
        print "Running on %s : %s"%(self.node, command_string)
        os.system(comline)

    for node in nodes:
        tlist=[]
        if threaded:
            tt=Thr(node)
            tt.start()
            tlist.append(tt)
        else:
            comline = shell+' '+node+' '+'""s""%(command_string)
            print "Running on %s : %s"%(node, command_string)
            os.system(comline)

    if threaded:
        while 1:
            finished=1
            for tt in tlist:
                if tt.isAlive():
                    finished=0
            if finished: break
            time.sleep(.1)

```

표 5 시스템 관리를 위한 스크립트(clusterexec)

이용법은 다음과 같다.

Syntax	\$clusterexec -c "명령어" <optoin>
옵션	-t 테스트모드로 실행
	-s rsh 혹은 ssh 등의 shell을 선택하여 수행
	-n 명령어를 전달할 nodelist를 지정 기본적으로 /usr/local/bin/NODELIST로 세팅되어 있음
	-T Thread를 이용하여 각 노드에 신속하게 명령어를 전달 명령의 정확하게 적용했다고 가정할 수 있을 때 유용하게 이용가능

표 6 clusterexec의 이용법

예를 들어 nodelist파일에 ssc100 ~ ssc110까지의 노드가 적혀 있고, 이 노드들을 재부팅하고자 한다면 다음과 같은 명령을 내린다.

```
$ clusterexec -c "/sbin/shutdown -r now" -n nodelist
```

제 3 절 그리드 환경의 설정

그리드 컴퓨팅은 광역적 분포 자원을 유저가 손쉽게 이용할 수 있도록 middle ware의 역할이 중요한데, 기존의 분산처리 방식과는 다르게 개인간 혹은 연구 기간 사이의 virtual organization(VO)개념에 의하여 연계를 가지게 된다. 이번 과제 역시 글로벌스를 이용하여 다양한 형태의 컴퓨팅 자원을 연동하여 Data를 공유하고 응용 프로그램을 활용하는 것이 핵심 이슈이다.

가. OpenPBS의 설치

PBS(Portable Batch System)는 일반적으로 각 로컬 노드에 작업을 할당하고 간단한 스케줄링 및 모니터링 역할을 한다.

- 마스터 노드에는 Job scheduling master의 역할을 하게 되므로, 다음과 같은 옵션으로 컴파일 한다.

```
$ ./configure --prefix=/usr/pbs --enable-docs --disable-mom
$ make
# make install
```

/usr/spool/PBS/server_name에 서버의 이름을 등록하고,

/etc/spool/PBS/server_priv/nodes에는 각각의 계산 노드를 나열한다.

작업 큐를 생성시키고 생성된 큐의 기본설정을 위하여,

/usr/spool/PBS/pbs_server.conf에 다음과 같이 적어준다.

```
# Create queues and set their attributes.
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
#
# Set server attributes.
#
set server scheduling = True
set server default_queue = workq
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
```

- 위의 설정파일로 serverdb 파일, 큐를 생성시키고 초기화하며, pbs_server, pbs_sched를 시작한다. 이 명령어들 역시 /etc/rc.local에 등록하여 시스템 부팅시 자동으로 뜰 수 있도록 처리한다.

```
# /usr/pbs/sbin/pbs_server -t create
# /usr/pbs/bin/qmgr < /usr/spool/PBS/pbs_server.conf
Max open servers: 4
# /usr/pbs/bin/qterm
# /usr/pbs/sbin/pbs_server
# /usr/pbs/sbin/pbs_sched
# ps -ax | grep pbs

15038 ?      S      0:00 /usr/pbs/sbin/pbs_server
15040 ?      S      0:00 /usr/pbs/sbin/pbs_sched
```

- 로컬 노드는 pbs_server 대신 pbs_mom이 필요하므로 다음과 같이 컴파일을 하며 default-server로 로그인 서버를 지정한다.

```
$ ./configure --prefix=/usr/pbs --disable-server --enable-mom W
--set-default-server=sdd114.foobar.com
$ make
# make install
```

- 로컬 노드에 서버의 이름을 등록하고 pbs_mom으로 클라이언트 데몬을 시작한다.

나. globus 2.4의 설치

글로버스는 globus 유저를 통해 설치된다. 설치 전에 점검해야할 사항으로는 시스템에 perl이 설치되어 있는지, /etc/hosts에 호스트명이 등록되어 있는지, 그리고 \$GLOBUS_LOCATION이 환경 변수로 지정되어 있는지 등이다.

①GPT(Globus Package Toolkit)의 설치

글로버스의 설치는 GPT를 미리 설치해 놓고 이를 통해 손쉽게 설치할 수 있다. 먼저 GPT를 설치하기 위해선 GPT_LOCATION이 환경변수로 잡혀 있어야 한다. 여기서는 2.2.10버전을 이용하였는데, /usr/local/gpt-2.2.10로 경로를 잡았다. /etc/bashrc에 다음을 추가한다.

```
export GPT_LOCATION=/usr/local/gpt-2.2.10
```

소스를 받아서 압축을 푼 후 build_gpt라고 치면 간단하게 설치가 끝난다.

```
$ tar zxvf gpt-2.2.2-src.tar.gz
$ cd gpt-2.2.2
$ ./build_gpt
```

② Globus-2.4의 설치 및 세팅

- Globus 2.x대 버전은 Data Management, Information Services, Resource Management 에 대하여 각각의 Client, SDK, Server가 있어 총 9가지 종류의 번들을 설치할 수 있다. 다음과 같이 각각의 번들에 대하여 적절한 옵션과 flavors를 주어가며 설치할 수 있는데, 여기서 <flavors>는 패키지를 컴파일시킬 시 사용되는 컴파일러를 결정하고 architecture를 선택하며 디버깅 모드로 설치할 것인지 그렇지 않은지 thred 모드로 설치할 것인지 그렇지 않은지 등을 선택한다. 예를 들어 <flavors>로 gcc32dbg를 이용했다면 compiler로는 gcc를 사용하고 32비트 환경에서 컴파일하며 디버깅 모드로 설치하게 된다.

```
$ /usr/local/GPT/sbin/gpt-build bundle options <flavors>
```

/etc/bashrc에 GLOBUS_LOCATION을 추가한다.

```
export GLOBUS_LOCATION=/usr/local/Globus-2.4
```

여기에서는 다음의 script를 만들어 script를 실행시킴으로써 패키지 별로 하나하나

빌드하지 않고 수시간에 걸쳐 한꺼번에 설치되도록 한다.

```
#!/bin/sh
gpt-build globus-data-management-client-2.4.2-src_bundle.tar.gz gcc32dbg
gpt-build globus-data-management-sdk-2.4.2-src_bundle.tar.gz gcc32dbg
gpt-build globus-data-management-server-2.4.2-src_bundle.tar.gz gcc32dbg
gpt-build globus-information-services-client-2.4.2-src_bundle.tar.gz gcc32dbgpthr
gpt-build globus-information-services-sdk-2.4.2-src_bundle.tar.gz gcc32dbgpthr
gpt-build globus-information-services-server-2.4.2-src_bundle.tar.gz gcc32dbgpthr
gpt-build globus-resource-management-2.4.2-src_bundle.tar.gz gcc32dbg
gpt-build globus-resource-management-sdk-2.4.2-src_bundle.tar.gz gcc32dbg
gpt-build globus-resource-management-server-2.4.2-src_bundle.tar.gz gcc32dbg
```

- globus-build뒤의 설정을 한다.

```
$ /usr/local/GPT/sbin/gpt-postinstall
```

슈퍼 유저 권한으로 GSI 소프트웨어의 setup을 확인하고, 다시 글로버스 계정으로 패키지 설치가 제대로 되었는지 체크한다. setup-gsi는 /etc/grid-security에 globus의 인증 및 기타 설정파일들을 생성시키는 역할을 한다.

```
$ /usr/local/globus/setup/globus/setup-gsi
$ su
# /usr/local/GPT/sbin/gpt_verify
```

- 풀에 연동된 자원을 사용하기 위해선 인증서가 필요한데, 우선 사용자 인증서는, 다음의 명령을 통하여 홈 계정의 .globus밑에 usercert_request.pem을 생성 시키고 이 CRS를 KISTI의 CA에 등록함으로 요청하게 된다. CA로부터 usercert.pem를 받으면 이를 ~/.globus/usercert.pem파일에 복사하여 넣음으로서 사용자 인증을 완료한다.

```
$/grid-cert-request -cn "사용자명"
```

마찬가지 방법으로 host에 대한 gatekeeper 인증서와 CA인증서를 신청받아 해당 디렉토리에 넣어둔다.

```
$/grid-cert-request -service host -host ssc4.snu.ac.kr
```

- 인증서를 인증하는 쪽에서는 다음과 같이 -in 옵션으로 인증요청서를 지정하여

grid-ca-sign을 실행시키면 -out에 지정된 파일로 인증서가 생성된다. 동시에 이 인증서는 /root/.globus/simpleCA/newcerts밑에도 저장된다.

```
$grid-ca-sign -in usercert_request.pem -out usercert.pem
```

- gatekeeper를 xined 데몬 서비스로 등록하기 위해 /etc/services파일에 다음을 추가한다.

```
globus-gatekeeper    2119/tcp    #globus gatekeeper
```

/etc/xinetd.d/globus-gatekeeper파일을 다음과 같이 작성한다.

```
service globus-gatekeeper
{
    socket_type = stream
    protocol = tcp
    wait      = no
    user      = root
    server    = /usr/local/Globus-2.4/sbin/globus-gatekeeper
    disable   = no
}
```

- 마찬가지로 방법으로 grid-ftp를 xinetd데몬에 등록시키기 위해 다음과 같은 파일을 작성하여 /etc/xinetd.d/grid-ftp로 저장한다.

```
service gsiftp
{
    disable      = no
    instances    = 1000
    socket_type  = stream
    wait        = no
    user        = root
    env         = LD_LIBRARY_PATH=/usr/local/GLOBUS/lib
    server      = /usr/local/GLOBUS/sbin/in.ftpd
    server_args = -l -a -d -G /usr/local/GLOBUS
    log_on_success += DURATION USERID
    log_on_failure += USERID
    nice          = 10
}
```

/etc/services에는 2911포트를 예약한다.

```
gsiftp    2911/tcp    #globus grid ftp
```

등록된 gatekeeper와 grid-ftp가 서비스를 시작할 수 있도록 xinetd를 restart한다.

- 글로버스를 설치한 것과 마찬가지로 GPT를 이용하여 GRAM job manager scheduler 서포트 모듈과, MDS에 정보를 제공하는 역할을 하는 GRAM 리포터를 설치한다.

```
$gpt-build globus_gram_job_manager_setup_pbs-1.5.tar.gz gcc32dbg
$gpt-build globus_gram_reporter-2.0.tar.gz gcc32dbg
$gpt-build globus_gram_reporter_setup_fork-1.0.tar.gz gcc32dbg
```

새로운 패키지가 깔리면 항상 gpt-postinstall을 해준다.

- MDS 설정을 설정한다.

```
$ cd /usr/local/globus/etc/
$ vi grid-info-slapd.conf
...
#(GRIS설정이므로 GLIS관련 설정부분을 모두 주석처리 한다.)
# database      giis
# suffix        "Mds-Vo-name=site, o=Grid"
# conf          /usr/local/globus/etc/grid-info-site-giis.conf
# policyfile    /usr/local/globus/etc/grid-info-site-policy.conf
# anonymousbind yes
# access to * by * write
modulepath항목이 다음과 같이 설정되어 있는지 확인한다.
modulepath /usr/local/globus/libexec/openldap/gcc32dbgpthr
```

위의 유저인증 작업과 마찬가지로 MDS서비스를 위하여 ldap인증을 받는다.

```
$ grid-cert-request -service ldap -host ssc4.snu.ac.kr
```

GRIS 서비스를 가동하고, 이를 /etc/rc.local에 다음과 같이 등록하여 시스템이 시작할 때 자동으로 실행되도록 한다.

```
/usr/local/globus/sbin/SXXgris start
```

- /etc/grid-security/grid-mapfile은 그리드 풀에 속하게 되는 다른 기관의 유저들에게 자원 사용을 위한 접근을 허용하는 역할을 하는데, 'subject'를 갖고 있는 원격 사이트에서 자원 이용을 요청하면 현재 시스템의 'user'계정을 이용하도록 매핑시킨다. 해당기관의 subject와 그에 상응하는 username을 일일이 추가한다. 현재 SSC4에는 KISTI의 administrator 그룹과 N*Grid 응용문제 이용자들의 총 28개가 등록되어 있다. 그리고 이들 이용자들에 대한 계정은 앞서 언급한 바와 같이 kisti혹은

ngrid 그룹에 포함하여 생성되어진 것들이다.

```
# TestBed Administrators
"/O=Grid/O=Globus/OU=gridcenter.or.kr/CN=Jae Hyuck Kwak" globus
"/O=Grid/O=Globus/OU=gridcenter.or.kr/CN=Gee-Bum Koo" globus
"/O=Grid/O=Globus/OU=gridcenter.or.kr/CN=globus" globus

# KISTI users
"/O=Grid/O=Globus/OU=hpcnet.ne.kr/CN=Hong Suk Yi at KISTI" kis001
"/O=Grid/O=Globus/OU=cnu.ac.kr/CN=globuslinux" kis002
"/O=Grid/O=Globus/OU=hpcnet.ne.kr/CN=Kum Won Cho" kis003
"/O=Grid/O=Globus/OU=gridcenter.or.kr/CN=Sangwan Kim" kis004
"/O=Grid/O=Globus/OU=hpcnet.ne.kr/CN=Sang Min Lee" kis005

# N*Grid users
"/O=Grid/O=Globus/OU=postech.ac.kr/CN=KSC" ngrid001
"/O=Grid/O=Globus/OU=snu.ac.kr/CN=floydfan" ngrid002
"/O=Grid/O=Globus/OU=kaist.ac.kr/CN=kaistCFD" ngrid003
"/O=Grid/O=Globus/OU=gridcenter.or.kr/CN=cfdkwon" ngrid003
"/O=Grid/O=Globus/OU=me.pusan.ac.kr/CN=sdhong" ngrid004
"/O=Grid/O=Globus/OU=sookmyung.ac.kr/CN=grid1" ngrid005
"/O=Grid/O=Globus/OU=grid.pusan.ac.kr/CN=gridnode1" ngrid006
"/O=Grid/O=Globus/OU=SNU/CN=dcslab" ngrid007
"/O=Grid/O=Globus/OU=kaist.ac.kr/CN=globus_at_wwf" ngrid008
"/O=Grid/O=Globus/OU=mju.ac.kr/CN=globus" ngrid010
"/O=Grid/O=Globus/OU=konkuk.ac.kr/CN=konkukimc" ngrid011
"/O=Grid/O=Globus/OU=kumho.co.kr/CN=globus" ngrid013
"/O=Grid/O=Globus/OU=cs.kookmin.ac.kr/CN=lucifer at Kookmin Univ" ngrid015
"/O=Grid/O=Globus/OU=Nanomics/CN=Taesung Moon" ngrid016
"/O=Grid/O=Globus/OU=pknu.ac.kr/CN=cgtest" ngrid017
"/O=Grid/O=Globus/OU=ssc.ac.kr/CN=SSU Workflow" ngrid018
"/O=Grid/O=Globus/OU=kaist.ac.kr/CN=globus_at_dalton1" ngrid019
"/O=Grid/O=Globus/OU=kaist.ac.kr/CN=eugene" ngrid020
"/O=Grid/O=Globus/OU=uos.ac.kr/CN=ccsgrid" ngrid021
```

표 7 grid-mapfile

다. MPICH_G2 설치

mpich_G2는 globus toolkit이 제공하는 service를 사용할 수 있도록 MPI 표준을 구현한것이다. mpich_G2는 일반 mpich-1.2.5 일반 버전을 device로 globus toolkit을 택하여 컴파일 하면된다.

```
$ gunzip -c mpich.tar.gz | tar xvf -
$ patch -p0 < patch.all
$ ./configure --with-device=globus2:-flavor=gcc32dbg
$ make
# make install
```

제 III 장. 제공 시스템의 동작 및 안정성 테스트

제 1 절 그리드 환경의 동작 테스트

가. 글로버스

유저 프락시와 게이트 키퍼를 작동시키고 localhost에서 /bin/date를 실행 시킨 결과이다.

```
$ /usr/local/globus/bin/grid-proxy-init
$ /usr/local/globus/bin/globus-personal-gatekeeper -start
GRAMcontact: sdd125.hpcnet.ne.kr:32915:/O=Grid/O=Globus
/OU=hpcnet.ne.kr/CN=
globus at sdd125
$ /usr/local/globus/bin/globusrun -o -r W
"localhost:32915:/O=Grid/O=Globus/OU=hpcnet.ne.kr/CN=globus at
sdd125" '&(executable=/bin/date)'
```

여기에서 grid-mapfile에 등록된 다른 기관의 자원에 job을 실행시키려면 localhost 대신에 해당 hostname을 기입하면 되는데, KISTI administrator의 테스트 결과 정상 작동됨을 확인할 수 있었다.

gsincftp는 기존의 ncftp에 GSI부분을 추가하여 그리드 환경에 맞도록 개선한것으로서 유저인증이 확인된 사용자에게 대하여 data 전송을 손쉽게 허용한다. 다음은 gsincftp의 테스트 결과이고 마찬가지로 정상 작동됨을 확인할 수 있다.

```
$ gsincftp localhost
NcFTP 3.0.3 (April 15, 2001) by Mike Gleason (ncftp@ncftp.com).
Connecting to 127.0.0.1....localhost FTP server
(GridFTP Server 1.0 [GSI patch v0.5] wu-2.6.1(2) Wed May 15 19:23:46
KST 2002) ready...Logging in...
User globus logged in.
Logged in to localhost.
ncftp /home/globus >
```

gssssh는 ssh에 GSI 기능을 추가 시킨것으로서 마찬가지로 정상작동됨을 확인할 수 있었다.

```
$ gssssh jupiter.gridcenter.or.kr
```

이 외에도 GRIS 서비스를 가동한 상태에서 MDS를 통하여 다양한 정보들이 교

환될 수 있음을 확인하였다.

```
$ grid-info-search -h ngrid-mds.gridcenter.or.kr -p 2135 -x -b W  
"Mds-Vo-name=local, o=Grid"  
$ grid-info-search -h ngrid-mds.gridcenter.or.kr -p 2135 -x -b W  
"Mds-Vo-name=ngrid-mds, o=Grid"  
$ grid-info-search -x -b "Mds-Vo-name=ngrid-mds, o=Grid" -s base W  
giisregistrationstatus
```

나. MPICH_G2

\$mpich_location/bin/machines 파일에 사용할 계산 노드와 가중치를 부여하고 mpich_G2 공식 사이트의 매뉴얼에 나와있는 ring.c 코드를 돌려보았다.

```
$ make ring  
$ /usr/local/mpich_G2/bin/mpirun -np 4 ring  
Master: end of trip 1 of 1: after receiving passed_num=4  
(should be =trip*numprocs=4) from source=3
```

또한 MPICH_G2의 mpirun은 RSL(Resource Specification Language)를 생성해서 globus에 보낼 수 있는데, RSL script는 하나 이상의 sub-job으로 구성되어 있으며 일반적으로 하나의 subjob은 하나의 원격 자원에 할당되게 되어 있다. SSC4, Jupiter, Venus를 연동하여 HPL을 돌리기 위해 RSL 파일을 만들었다. 다음과 같이 -dumprsl 옵션을 이용해서 생성시킬 수 있다.

```
$mpirun -dumprsl -np 8 xhpl > RSLfile.rsl
```

생성된 RSL 파일을 세 군데의 원격 자원에 나누어 돌리기 위해 다음과 같이 수정하였다. count는 해당 원격 사이트가 사용할 계산노드를 나타내는데, SSC4의 경우 4개, Jupiter 2개, Venus 2개를 사용한다. 사용할 포트는 방화벽에서 허용한 20000에서 25000 사이의 것을 이용하도록 한다.

```

+
( &(resourceManagerContact="ssc4.snu.ac.kr")
  (count=4)
  (label="subjob 0")
  (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
    (GLOBUS_TCP_PORT_RANGE "20000,25000")
    (LD_LIBRARY_PATH /usr/local/GLOBUS/lib/))
  (directory="/home/sjkim/hpl/bin/jupiter")
  (executable="/home/sjkim/hpl/bin/jupiter/xhpl")
)
( &(resourceManagerContact="jupiter.gridcenter.or.kr")
  (count=2)
  (label="subjob 1")
  (environment=(GLOBUS_DUROC_SUBJOB_INDEX 1)
    (GLOBUS_TCP_PORT_RANGE "20000,25000")
    (LD_LIBRARY_PATH /usr/local/GLOBUS/lib/))
  (directory="/home/ngrid/ngrid014/hpl/bin/jupiter")
  (executable="/home/ngrid/ngrid014/hpl/bin/jupiter/xhpl")
)
( &(resourceManagerContact="venus.gridcenter.or.kr")
  (count=2)
  (label="subjob 2")
  (environment=(GLOBUS_DUROC_SUBJOB_INDEX 2)
    (GLOBUS_TCP_PORT_RANGE "20000,25000")
    (LD_LIBRARY_PATH /usr/local/GLOBUS/lib/))
  (directory="/home/ngrid/ngrid014/hpl/bin/venus")
  (executable="/home/ngrid/ngrid014/hpl/bin/venus/xhpl")
)

```

표 8 SSC4, Jupiter, Venus를 연동했을 때의 RSL file

다. OpenPBS

pbs_server, pbs_sched와 pbs_mom이 실행되고 있는 상황에서 PBS master 노드 (로그인 서버)에서 다음과 같은 명령을 하면, 배치작업을 하달할 수 있는 가용 노드와 각 노드의 상태 정보가 나타난다.

```

# /usr/pbs/bin/pbsnodes -a
ssc101
  state = free
  np = 1
  ntype = cluster
ssc102
  state = free
  np = 1
  ntype = cluster
ssc103
  state = free
  np = 1
  ntype = cluster
...
...(이하생략)

```

사용자 튜토리얼에 나와 있는 원주율을 구하는 코드를 mpich로 컴파일하고, PBS

master 노드에서 다음과 같은 작업 스크립트를 작성하여 16개의 노드에 대하여 일을 시킨다.

```
$ vi test.sh
#!/bin/sh
#PBS -l nodes=3 # 실행중인 노드를 exclusive 하게 이용하기 위해 넣어준다.
hostname
cat $PBS_NODEFILE
NPROCS=`wc -l < $PBS_NODEFILE` #프로세서의 수를 정의한다.
date # 실행 시작 시간
/usr/local/mpich/bin/mpirun -v -machinefile $PBS_NODEFILE -np $NPROCS
$PBS_O_WORKDIR/cpi
sleep 10
date # 실행 종료 시간
$:wq
$ qsub test.sh
113.ssc401
```

다음과 같은 결과를 얻을 수 있었다.

```
$ cat test.sh.o113
ssc401
ssc402
ssc403
...
...(생략)
...
Thu July 29 09:54:50 KST 2003
running /home/sjkim/pbs_test/cpi on 16 LINUX ch_p4 processors
Created /home/sjkim/PI23221
pi is approximately 3.1416009869231249, Error is 0.0000083333333318
wall clock time = 0.000371
Thu July 29 09:55:01 KST 2003
```

현재 시스템에서 작업이 돌고 있거나 대기 중인 작업이 있을 경우 qstat 명령어를 통해 알 수 있다.

```
$ qstat -n

ssc4.snu.ac.kr:
Job ID   Username Queue Jobname SessID NDS TSK Memory Time   S Time
-----
2216.ssc4.snu.a ngrid004 workq STDIN 17912 4 -- -- -- R 490:2
ssc408/0+ssc407/0+ssc405/0+ssc401/0
3138.ssc4.snu.a kis001 workq STDIN 6405 1 -- -- -- R --
ssc406/0
```

제 2 절 안정성 테스트

컴퓨팅 자원 제공 시스템의 경우 안정적인 운영의 보장이 중요한데, 특히 이번 연구과제에서와 같이 타 기관과의 협업을 통해 각 기관의 자원이 대규모로 연동이 되어 응용문제 유저들에게 제공되는 경우에는, 로컬 노드의 failure가 end-user 입장에서 치명적일 수 밖에 없다. Linpack Benchmark 프로그램은 기본적으로 $Ax=b$ 의 형태로 주어지는 n차의 선형 System을 행 방향의 LU factorization 방법으로 해를 구함으로서 Super Computer의 성능을 측정하는데 이용되는데, CPU의 작업능력을 최대한으로 이용하는 이 프로그램을 이용하여 수일에 걸쳐 돌리면서 시스템의 안정성을 살펴보았다. 과도한 하중에도 시스템의 특별한 이상을 발견하지는 못하였다.

제 3 절 클러스터 시스템의 벤치 마크 테스트

가. Netpipe 테스트

netpipe는 tcp 패킷의 크기를 변화시켜가며 네트워크의 성능을 측정하는 도구이다. ssc401에 netpipe reciever를 띄워놓고 ssc402에서 transmitter로 패킷을 보내가며 구성된 기가비트 네트워크의 성능을 살펴보았다. 다음은 테스트 결과로 나온 패킷의 크기 대 대역폭을 나타낸 그림이다.

본 기가비트 네트워크 시스템의 경우 약 $32\mu s$ 의 latency를 나타내며 100Mbps fast ethernet 환경보다 훨씬 좋은 성능을 보여주고 있다. 다만 사설 IP만으로 연결되었을 경우의 Gigabit 네트워크 환경보다는 성능이 떨어지는데, 이는 엄격한 방화벽 적용에 따른 네트워크 성능 저하로 추정된다.

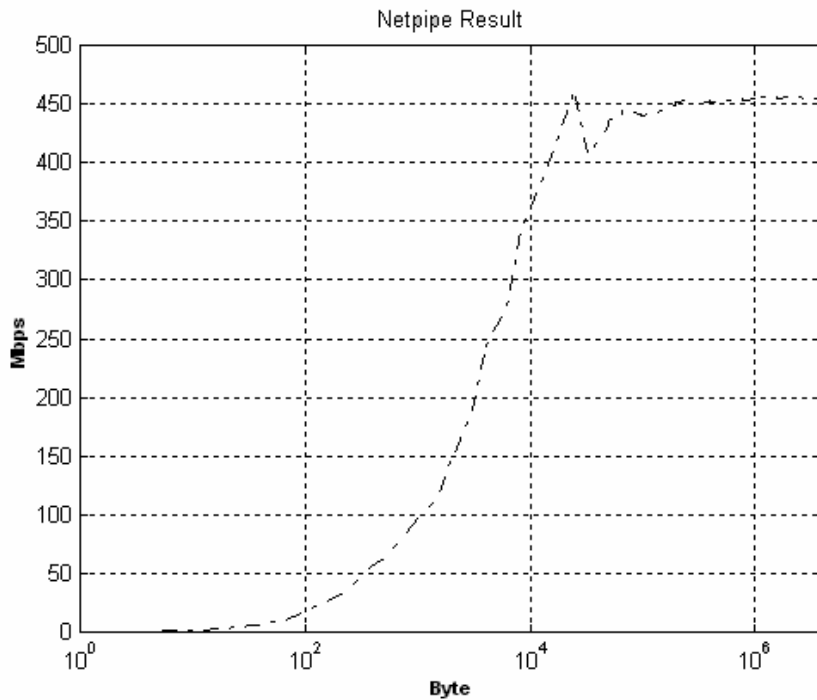


그림 3 netpipe test

나. Linpack 테스트

안정성 테스트에도 사용되었던 린팩 벤치마크 테스트를 이용하여 block 크기, 문제 크기, 격자의 행렬 크기 등을 조절해가며 튜닝을 시도하였다. block 크기 104, 문제 크기 20500, 4x4의 P,Q 조합으로부터 다음과 같이 26.42Gflops를 나타냈다. 이는 초당 약 260억 번의 부동소수점 연산을 수행할 수 있음을 나타내고 있다.

```

N      : 20500
NB     : 104
P      : 4
Q      : 4
PFACT  : Left
NBMIN  : 2
NDIV   : 2
RFACT  : Left
BCAST  : 1ring
DEPTH  : 0
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes

```

ALIGN : 8 double precision words						

- The matrix A is randomly generated for each test.						
- The following scaled residual checks will be computed:						
1) Ax-b _oo / (eps * A _1 * N)						
2) Ax-b _oo / (eps * A _1 * x _1)						
3) Ax-b _oo / (eps * A _oo * x _oo)						
- The relative machine precision (eps) is taken to be 1.110223e-16						
- Computational tests pass if scaled residuals are less than 16.0						
=====						
T/V	N	NB	P	Q	Time	Gflops

W00L2L2	20500	104	4	4	217.44	2.642e+01

Ax-b _oo / (eps * A _1 * N) = 0.0038019 PASSED						
Ax-b _oo / (eps * A _1 * x _1) = 0.0045296 PASSED						
Ax-b _oo / (eps * A _oo * x _oo) = 0.0008809 PASSED						
=====						

표 9 HPL test

다. 타 시스템과의 연동 테스트

- 미들웨어의 설치 및 설정을 확인하기 위해 SSC4, Jupiter, Venus를 연동하여 HPL 테스트를 하였다. 아래의 표와 같이 단일 시스템에서 돌리는 것보다 성능이 떨어짐을 알 수 있다. 이것은 시스템 사이의 네트워크 대역폭에 의해 병목 현상이 일어나기 때문이다. 같은 백본망을 공유하고 있는 KISTI의 Jupiter와 Venus를 연동한 경우 성능 저하가 그리 심하지 않음을 알 수 있다.

계산참여그룹	이론성능	측정성능	효율
SSC4 8CPU	24	13.1	54.6%
Jupiter 8CPU	27.2	13.0	47.8%
Venus 8CPU	32	12.7	39.7%
SSC4 4 + Jupiter 4	25.6	8.96	35.0%
SSC4 4 + Venus 4	28	9.67	34.5%
Jupiter 4 + Venus 4	29.6	11.30	38.2%
SSC4 4 + Jupiter 2 + Venus 2	26.8	8.8	32.8%

표 10 HPL 성능 테스트 결과(단위:Gflops)

- SSC4, Jupiter, Venus, PNU, Postech의 시스템들에 대하여 Netpipe를 이용하여 서로간에 통신 latency를 살펴보았다. 수백 μ s에서 수천 μ s의 응답반응 시간이 나타난다. 마찬가지로 Jupiter와 Venus사이의 환경이 제일 좋다.

	SSC4	Jupiter	Venus	PNU	Postech
SSC4	.	2552	2360	4251	6108
Jupiter	2552	.	200	4372	2232
Venus	2360	200	.	6562	2227
PNU	4251	4372	6562	.	7832
Postech	6108	2232	2227	7832	.

표 11 테스트베드 사이의 TCP/IP latency(μ s)

제 IV 장. 결론 및 개선 사항

이번 그리드 프로젝트의 핵심적인 부분인 기반 컴퓨팅 자원 시스템의 구축이 계획대로 진척되었음을 확인 할 수 있었다. 그리고 전용 클러스터의 제공을 위한 네트워크 업그레이드와 Globus, MPICH_G2, OpenPBS의 설치 및 세팅 등과 같은 세부 설정 등을 무사히 마칠 수 있었다. 아울러 사전에 자원의 안정성 테스트를 미리 거쳐, 응용 문제 유저들이 불편을 겪지 않도록 노력하였다.

그리드 연동을 통한 자원 공유는 그 편의성과 막강한 성능의 이면에 완전한 보안을 유지할 수 있다는 전제가 밑바탕이 된다. 그리드 구성의 각 제공 자원들은 개별적인 보안의 유지에 심혈을 기울여야 하는데, 이러한 측면에 있어서 본 시스템은 방화벽을 공고히 구축하여 안전장치를 마련하였다.

또한 KISTI 그리드 실무 관리자들의 도움을 얻어 타 기관과의 연동을 시도하였으며, 이기종 시스템을 비롯하여 그리드 풀 가담 자원 중 상당수와 작업 연동이 가능함을 확인 할 수 있었다.

유저들과의 원활한 의사 소통을 위하여 자원제공 테스트베드 팀별로 웹 사이트를 개설해 이용자와의 유기적인 대화 채널을 만들려고 했었는데, 현재는 KISTI의 testbed웹에 통합적으로 구축된 상태이다.

이번 그리드 자원 제공 과제를 통하여 그리드 응용문제 혹은 미들웨어 이용자들에게 실험을 위한 공간을 제공하게 되었고, 작년의 1차년도에 비해서는 많은 성과가 있었음을 확인하였다. 하지만 과제의 특성 상 다른 과제 이용자들에게 서비스를 제공하는 성격이 강하고 자원 제공기관들 사이 그리고 자원 제공기관과 KISTI의 실무진 사이 원활한 협업을 해야 하기 때문에, 업무 지원과 협력에 대한 문제점 등이 드러났다. 또한 그리드의 이용이 아직까지는 그리드 컴퓨팅에 초점이 맞추어져 있는 상태에서, 시스템 사이의 병목에 의한 성능 저하나, 단일 시스템의 성능 자체가 현저하게 떨어지는 제공기관이 연결되었을 경우 그리드 연동의 실제적인 의미를 찾을 수 없게 되어 기존의 슈퍼 컴퓨터에서 추구하던 High Performance Computing을 위해서는 앞으로도 많은 연구가 필요할 것이다.

참고문헌

1. 김승조, 이창성, 임상영, 이상산, “PC farm을 이용한 그리드 컴퓨팅 구축”, KSAIM, 2002
2. 김병부, 김세연, 송화동, 조윤석, 홍성순, “Linux Server Bible”, *youngjin.com*, 2002
3. <http://www.gridtest.gridcenter.or.kr>
4. <http://www.globus.org>
5. Cameron Newham, Bill Rosenblatt, "Learning The Bash", O'REILLY, June 2001
6. 윤영하, 김승조, 박시형, 구기범 "국가 그리드 테스트베드를 이용한 구조 해석 수행," 한국항공우주학회 추계학술대회, 2003년 11월 15일, 경주