개인용 슈퍼컴퓨팅 개발가이드(초안 v1.0 beta 24)

Document Status

유형	Draft (초안)	보안	Confidential

NVIDIA, the NVIDIA logo, CUDA, and GeForce are trademarks or registered trademarks of NVIDIA Corporation.

Information furnished is believed to be accurate and reliable. However, PSC Group assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of PSC Group. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. PSC Group products are not authorized for use as critical components in life support devices or systems without express written approval of PSC Group.

The PSC Group logo is a registered trademark of PSC Group.
All other names are the property of their respective owners
© 2011 PSC Group - All rights reserved

개인용 슈퍼컴퓨팅 개발가이드(초안 v1.0 beta 24)

내용

Introducing....4

GPGPU 개발환경 구축 : Source Directory 구조....4

GPGPU 개발환경 구축 : Windows 7 64-bit, CUDA 기준....5

GPGPU 개발환경 구축 : Linux 64-bit Ubuntu 11.04, CUDA 기준....19

GPGPU 개발환경 구축 : OpenMP 환경 설치....24

GPGPU 개발환경 구축 : MPI 환경 설치....33

PSC Platform 구축 : Hadoop-mapreduce 환경 설치(Linux)....41

PSC Platform 구축 : Hadoop에서 GPGPU를 사용하기 위한 작업....57

PSC Platform 구축 : Hadoop source 빌드 방법 (참조)....58

PSC Platform 구축 : Mars(MapReduce + CUDA) 빌드....59

PSC Platform 구축 : JCUDA(java를 이용한 CUDA) 설치 및 예제....60

PSC Platform 구축 : Hadoop C++ 예제 컴파일 및 실행....76

PSC Platform 구축 : 멀티 유저 계정에서 Hadoop 시스템 사용....79

PSC Platform 구축 : 사용상 주의점....80

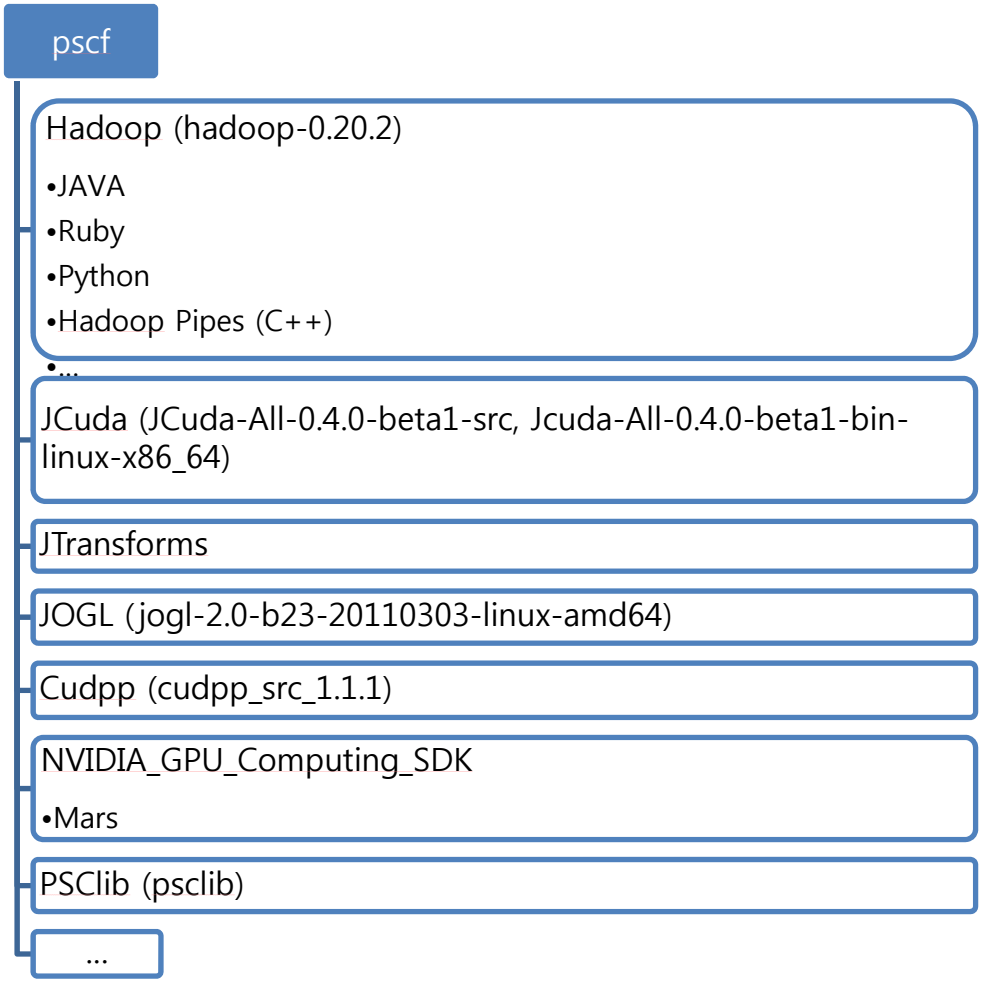
Introducing

본 문서는 '공개소프트웨어 기반의 개인용 슈퍼 컴퓨팅 플랫폼 구축 및 커뮤니티 운영' 과제의 개발환경, Personal Super Computing(PSC) Platform 구축과 가이드 내용을 서술한 것임.

GPGPU 개발환경 구축 : Source Directory 구조

1 구조

A. Directory(Linux Directory)



2 문서 내용

A. 기술된 개발 환경 중에서 Windows 관련 내용은 디버깅을 위한 환경 구축이고 Linux 내용이 실제로 본 과제에서 구축하기 위한 목표가 되므로 Linux 부분만 참조해야 함.

GPGPU 개발환경 구축 : Windows 7 64-bit, CUDA 기준

1. 우선 Microsoft Visual Studio 2008 또는/그리고 Microsoft Visual Studio 2010을 install 함.
   A. 실제 설치 순서에 있어서, 나중에 Install 하게 될 Parallel Nsight 보다는 우선적으로 Visual Studio를 Install하면 문제는 없음.
2. http://www.nvidia.com 에서 아래 두 개의 그림과 같이 (1), (2), (3), (4), (5)의 순서대로 선택해 나감.

3. 영문으로 설명하는 페이지가 표시되는데 그 중에서 다음과 같은 표를 확인.

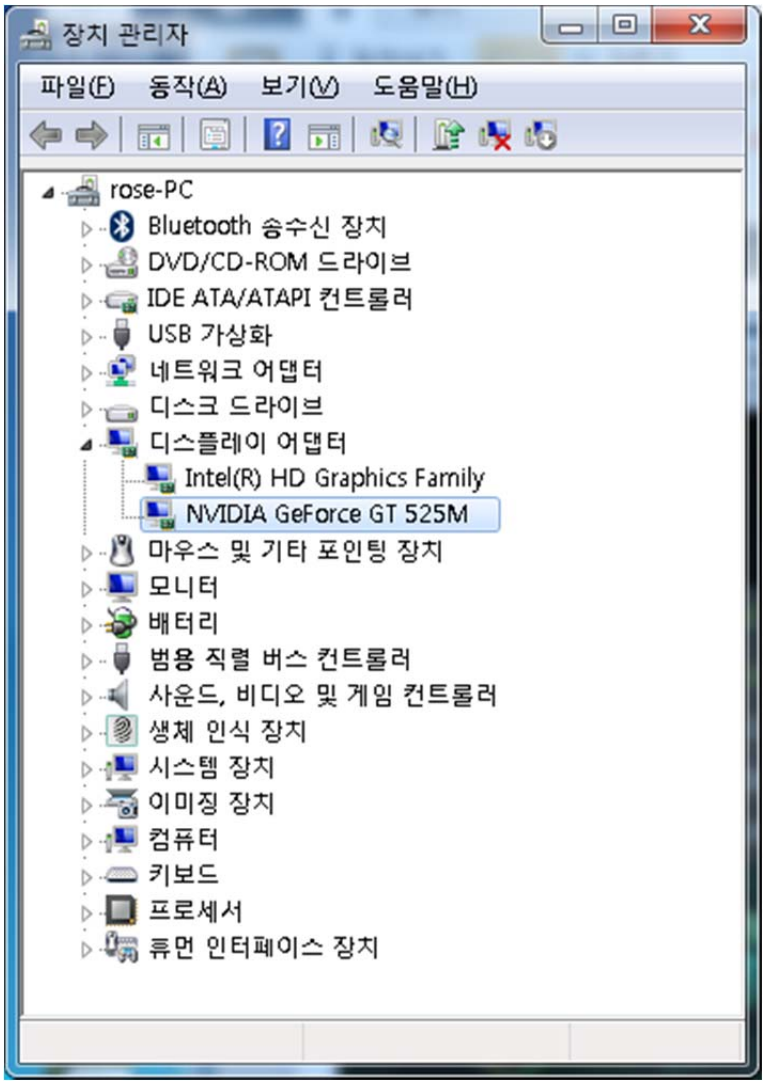
Windows 7, VISTA, Windows XP	Downloads
Developer Drivers for WinXP (270.81)	32-bit 64-bit
Developer Drivers for WinVista and Win7 (270.81)	32-bit 64-bit
Notebook Developer Drivers for WinVista and Win7 (270.61)	please check again later
CUDA Toolkit • C/C++ compiler • Visual Profiler • GPU-accelerated BLAS library • GPU-accelerated FFT library • GPU-accelerated Sparse Matrix library • GPU-accelerated RNG library • Additional tools and documentation	32-bit 64-bit documentation
CUDA Tools SDK	32-bit 64-bit
GPU Computing SDK code samples	32-bit 64-bit
Parallel Nsight 2.0	download
Other Tools and Libraries	link to page

< Table 1 : WINDOWS, VISTA, WINDOWS XP Download>

4. Developer Driver를 설치하기 위해서는 위의 <Table 1>보다는 아래와 같이 다운로드

   A. 그래픽 칩을 수동으로 확인하는 방법

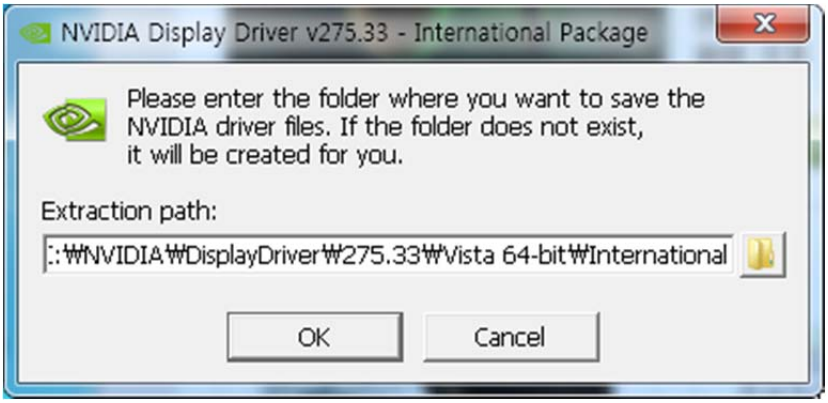
      i. [컴퓨터]->[시스템 속성]->[장치 관리자]->[디스플레이 어댑터]에서 NVIDIA 그래픽 카드를 확인(여기에서는 NVIDIA GeForce GT 525M)



ii. 아래 그림과 같이 (1), (2)를 선택 후 옵션 1과 같이 (3), (4), (5), (6), (7), (8)의 순서대로 또는 옵션 2와 같이 (a)의 자동 검색을 사용하여 드라이버 다운로드.

B. 또는 (1), (2)를 선택 후 옵션 2와 같이 (a)의 자동 검색을 사용하여 드라이버 다운로드

5. 다운로드된 Developer Driver 설치



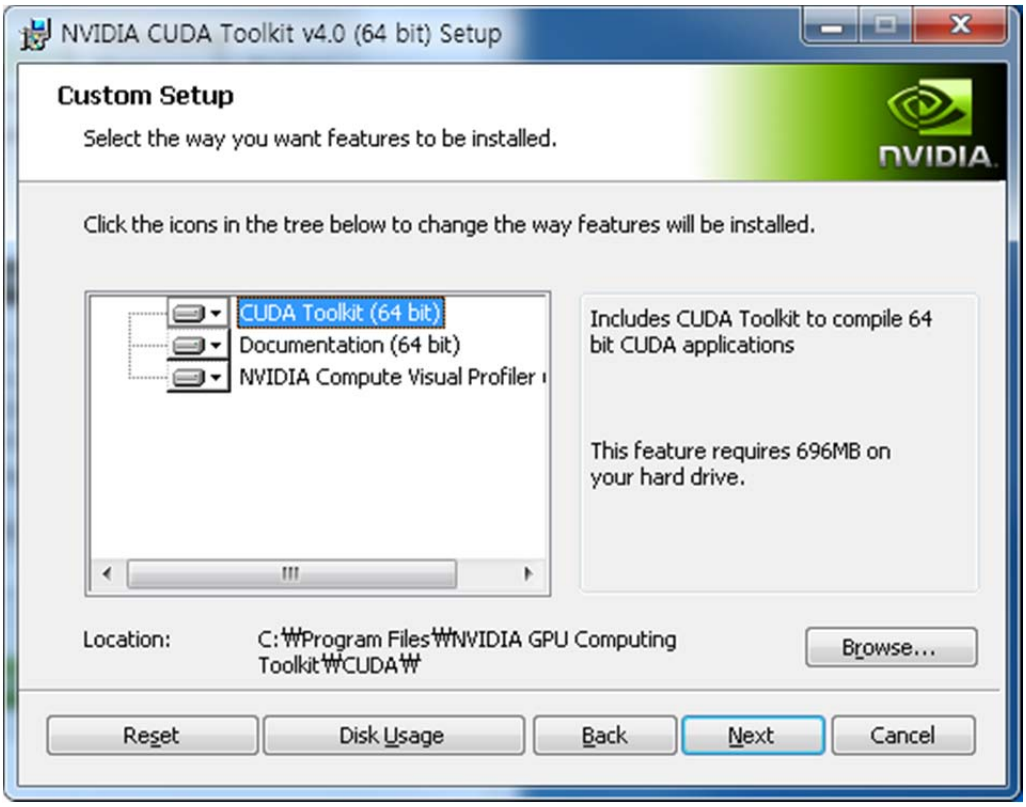
A. 'C:₩NVIDIA₩...'에 압축을 풀어서 인스톨을 시작하려고 하는데 웬만하면 이 Default 디렉토리에서 install을 완료함.

B. Install 완료와 동시에 컴퓨터 재시작을 요구함.

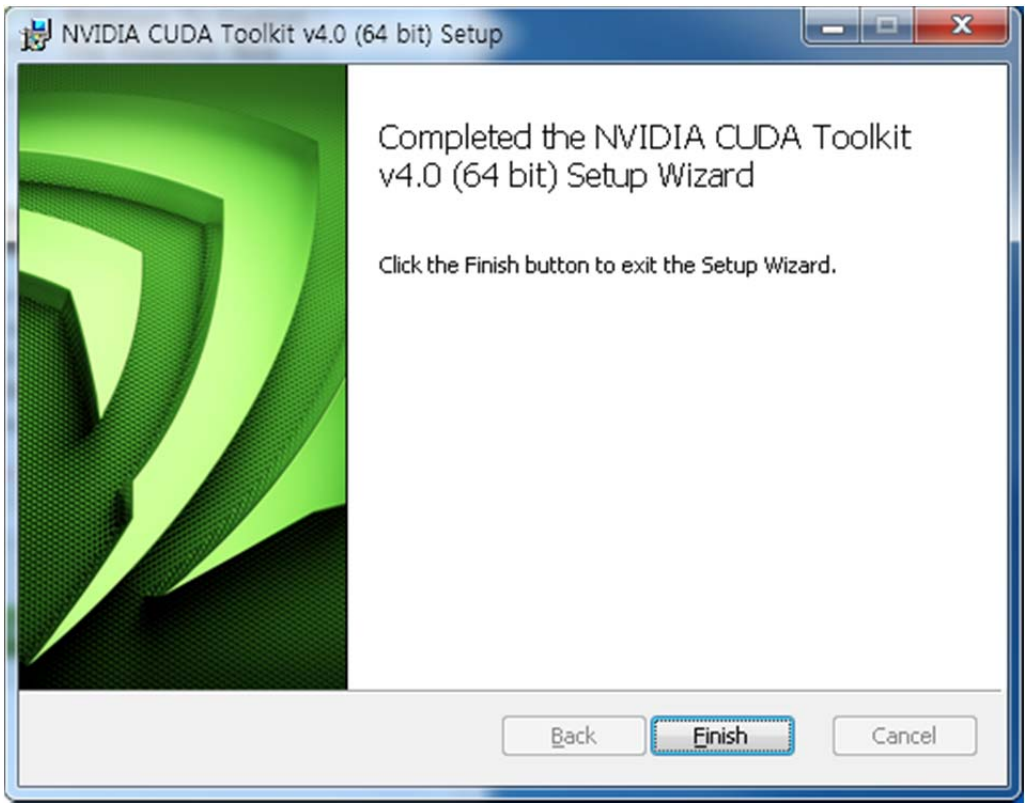
6. <Table 1>에서 CUDA Toolkit 중 OS에 맞게 32-bit 또는 64-bit를 다운로드 후 설치

A. Default로 'C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA'에 설치됨.

B. 설치 초기 화면 (모두 설치)



C. 설치 완료 화면



D. 설치 완료 후 바탕화면에 생성된 아이콘

7. <Table 1>에서 CUDA Toolkit 중 필요한 Document 다운로드

   A. 아래 문서들은 모든 install이 완료된 후 필요에 따라 숙독.

   B. 제공되는 Document (NVIDIA GPU Computing Documentation - http://developer.nvidia.com/nvidia-gpu-computing-documentation 참조)

CUDA Getting Started Guide (Windows) This guide will show you how to install and check the correct operation of the CUDA development tools in Windows.	Download
CUDA Getting Started Guide (Linux) This guide will show you how to install and check the correct operation of the CUDA development tools in Linux.	Download
CUDA Getting Started Guide (Mac OS X) This guide will show you how to install and check the correct operation of the CUDA development tools in Mac OS X.	Download
Getting Started with CUDA SDK samples This guide covers the introductary CUDA SDK samples beginning CUDA developers should review before developing your own projects.	Download
SDK Code Sample Guide New Features in CUDA Toolkit 4.0 This guide covers what is new in CUDA Toolkit 4.0 and the new code samples that are part of the CUDA SDK 4.0.	Download
CUDA Toolkit 4.0 Release Notes NVIDIA CUDA Toolkit version 4.0 Release Notes for all OS Platforms	Download
CUDA Toolkit 4.0 Release Notes Errata	Download

NVIDIA CUDA Toolkit version 4.0 Release Notes Errata for all OS Platforms

---

CUDA Toolkit 4.0 Readiness for CUDA Applications

In NVIDIA CUDA Toolkit version 4.0, a major emphasis has been placed on improving the programmability of multi-threaded and multi-GPU applications and on improving the ease of porting existing code to CUDA C/C++. This document describes the key API changes and improvements that have been made toward that end, particularly where they have the potential to impact existing applications. This document also highlights a few of the improvements that have been made to the libraries bundled with the CUDA Toolkit.

Download

CUDA C Programming Guide

Download

This is a detailed programming guide for CUDA C developers.

CUDA C Best Practices Guide

Download

This is a manual to help developers obtain the best performance from the NVIDIA CUDA Architecture. It presents established optimization techniques and explains coding metaphors and idioms that can greatly simplify programming for the CUDA architecture.

CUDA Occupancy Calculator

The CUDA Occupancy Calculator allows you to compute the multiprocessor occupancy of a GPU by a given CUDA kernel. This tool provides guidance for optimizing the best kernel launch configuration for the best possible occupancy for the GPU.

Download

CUDA Developer Guide for Optimus Platforms

This document provides guidance to CUDA developers and explains how NVIDIA CUDA APIs can be used to query for GPU capabilities in Optimus systems. It is strongly recommended to follow these guidelines to ensure CUDA applications are compatible with all notebooks featuring Optimus.

Download

OpenCL Programming Guide

Download

This is a detailed programming guide for OpenCL developers.

OpenCL Best Practices Guide Download

This is a manual to help developers obtain the best performance from OpenCL.

OpenCL Overview for the CUDA Architecture Download

This whitepaper summarizes the guidelines for how to choose the best implementations for NVIDIA GPUs.

OpenCL Implementation Notes

This document describes the "Implementation Defined" behavior for the NVIDIA OpenCL implementation as required by the OpenCL specification Version: 1.0. The implementation defined behavior is referenced below in the order of it's reference in the OpenCL specification and is grouped by the section number for the specification. Download

---

DirectCompute Programming Guide Download

This is a detailed programming guide for DirectCompute developers.

CUDA API Reference Manual (HTML) Download

This is the CUDA Runtime and Driver API reference manual, an online HTML version

---

CUDA API Reference Manual (PDF) Download

This is the CUDA Runtime and Driver API reference manual in PDF format.

CUDA API Reference Manual (CHM) Download

This is the CUDA Runtime and Driver API reference manual in CHM format (Microsoft Compiled HTML help).

PTX: Parallel Thread Execution ISA Version 2.3

This document describes PTX, a low-level parallel thread execution virtual machine and instruction set architecture (ISA). PTX exposes the GPU as a data-parallel computing device. Download

---

CUDA-memcheck User Manual

The CUDA debugger tool, cuda-gdb, includes a memory-checking feature for

detecting and debugging memory errors in CUDA applications. This document describes that feature and tool, called cuda-memcheck. The cuda-memcheck tool is designed to detect such memory access errors in your CUDA application.	Download
CUDA-gdb Debugger User Manual CUDA-GDB is the NVIDIA tool for debugging CUDA applications running on Linux and Mac. The tool provides developers with a mechanism for debugging CUDA applications running on actual hardware. CUDA-GDB runs on Linux and Mac OS X, 32-bit and 64-bit. The Linux edition is based on GDB 6.6 whereas the Mac edition is based on GDB 6.3.5	Download
Compute Visual Profiler The Compute Visual Profiler is a graphical user interface based profiling tool that can be used to measure performance and find potential opportunities for CUDA and OpenCL optimizations, to achieve maximum performance from NVIDIA GPUs. Compute Visual Profiler provides metrics in the form of plots and counter values presented in tables and as graphs. It tracks events with hardware counters on signals in the chip; this is explained in detail in the chapter entitled, "Compute Visual Profiler Counters."	Download
CUDA Fermi Compatibility Guide The Fermi Compatibility Guide for CUDA Applications is intended to help developers ensure that their NVIDIA CUDA applications will run effectively on GPUs based on the NVIDIA Fermi Architecture. This document provides guidance to developers who are already familiar with programming in CUDA C/C++ and want to make sure that their software applications are compatible with Fermi.	Download
CUDA Fermi Tuning Guide An overview on how to tune applications for Fermi to further increase these speedups is provided. More details are available in the CUDA C Programming Guide (version 3.2 and later) as noted throughout the document..	Download
CUBLAS Library User Guide	Download

The CUBLAS library is an implementation of BLAS (Basic Linear Algebra Subprograms) on top of the NVIDIA CUDA runtime. It allows the user to access the computational resources of NVIDIA Graphical Processing Unit (GPU), but does not auto-parallelize across multiple GPUs.

CUFFT Library User Guide

This document describes CUFFT, the NVIDIA CUDA Fast Fourier Transform (FFT) library. The FFT is a divide-and-conquer algorithm for efficiently computing discrete Fourier transforms of complex or real-valued data sets, and it is one of the most important and widely used numerical algorithms, with applications that include computational physics and general signal processing. The CUFFT library provides a simple interface for computing parallel FFTs on an NVIDIA GPU, which allows users to leverage the floating-point power and parallelism of the GPU without having to develop a custom, GPUbased FFT implementation.

Download

CUSPARSE Library User Guide

The NVIDIA CUDA CUSPARSE library contains a set of basic linear algebra subroutines used for handling sparse matrices and is designed to be called from C or C++. These subroutines can be classified in four categories.

Download

CURAND Library User Guide

The NVIDIA CURAND library provides facilities that focus on the simple and efficient generation of high-quality pseudorandom and quasirandom numbers.

Download

NVIDIA Performance Primitives (NPP) Library User Guide

NVIDIA NPP is a library of functions for performing CUDA accelerated processing. The initial set of functionality in the library focuses on imaging and video processing and is widely applicable for developers in these areas. NPP will evolve over time to encompass more of the compute heavy tasks in a variety of problem domains. The NPP library is written to maximize flexibility, while maintaining high performance.

Download

Thrust Quick Start Guide

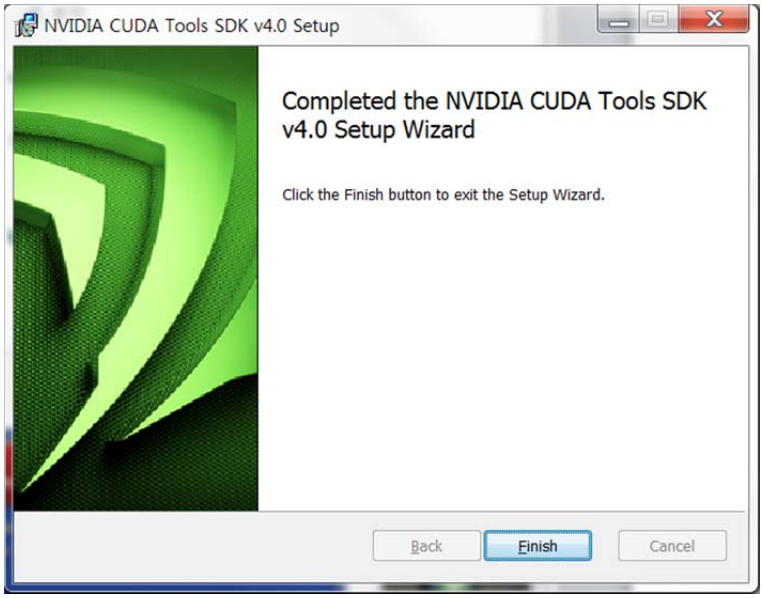
Thrust is a C++ template library for CUDA based on the Standard Template

Download

Library (STL). Thrust allows you to implement high performance parallel applications with minimal programming effort through a high-level interface that is fully interoperable with CUDA C.	
NVIDIA CUDA H.264 Video Encoder Library User Guide The NVIDIA CUDA H.264 Video Encoder is a library for performing CUDA accelerated video encoding. The functionality in the library takes raw YUV frames as input and generates NAL packets. This encoder supports up to various profiles up to High Profile @ Level 4.1.	Download
NVIDIA CUDA Video Decoder Library User Guide The CUDA Video Decoder API gives developers access to hardware video decoding capabilities on NVIDIA GPU. The actual hardware decode can run on either Video Processor (VP) or CUDA hardware, depending on the hardware capabilities and the codecs. This API supports the following video stream formats for Linux and Windows platforms: MPEG-2, VC-1, and H.264 (AVCHD).	Download
CUDA C SDK Release Notes	Download
DirectCompute SDK Release Notes	Download
OpenCL SDK Release Notes	Download
CUDA Toolkit Software License Agreement This is the Software License Agreement for developers that use the CUDA Toolkit. This License agreement also include the distribution license for CUDA Accelerated Libraries.	Download
GPU Computing SDK End User License Agreement This is the Software License Agreement for developers or licensees.	Download

8. <Table 1>에서 CUDA Tools SDK 중 OS에 맞게 32-bit 또는 64-bit를 다운로드 후 설치
   A. 'Choose Setup Type'에서 'Complete install' 실행
   B. Default로 'C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA Tools SDK'에 install 됨.

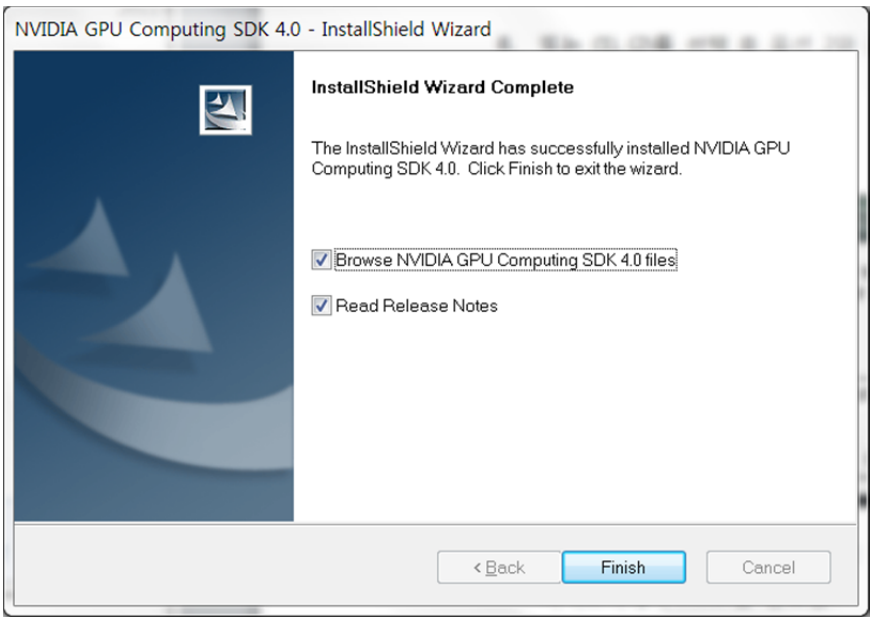
C. 다음과 같은 화면과 함께 install 완료됨



9. <Table 1>에서 'GPU Computing SDK code samples' 중 OS에 맞게 32-bit 또는 64-bit를 다운로드 후 설치

A. Default로 'C:₩ProgramData₩NVIDIA Corporation₩NVIDIA GPU Computing SDK 4.0'에 install 됨.

B. 다음과 같은 화면과 함께 완료 됨.



10. <Table 1>에서 'Parallel Nsight 2.0'를 Download

A. 등록 또는 Login 후 OS에 맞게 32-bit 또는 64-bit 버전 다운로드

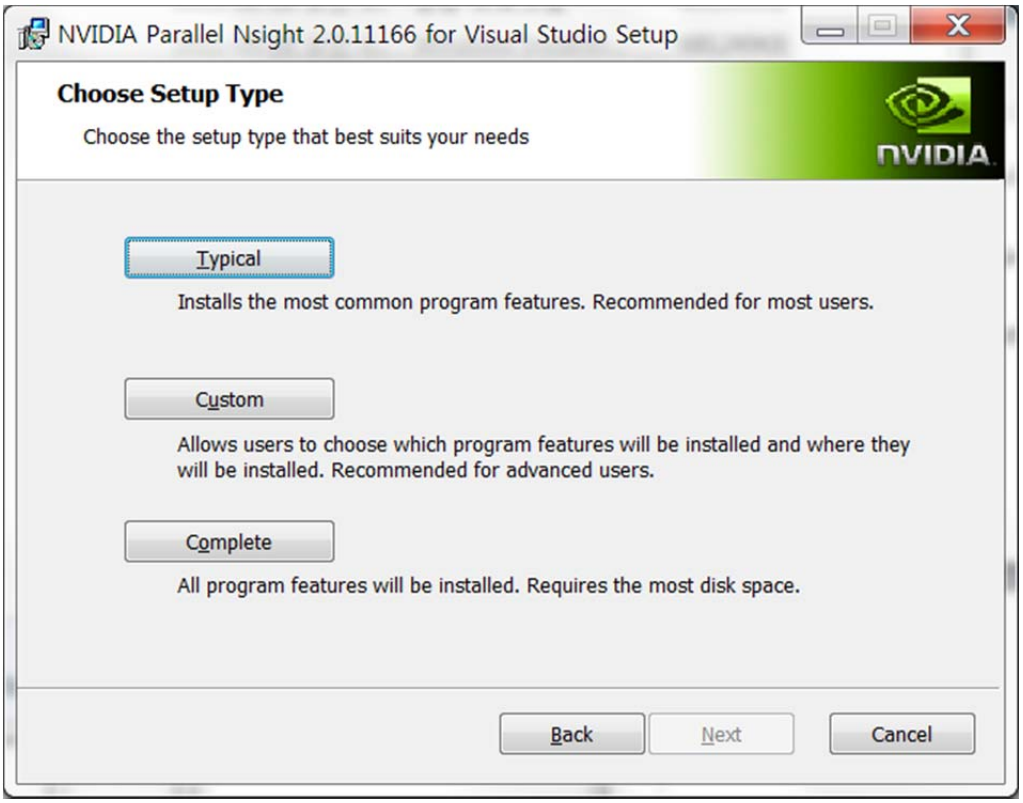
B. 관련 Document와 참고 자료 (http://developer.nvidia.com/nvidia-parallel-nsight)

NVIDIA Parallel Nsight

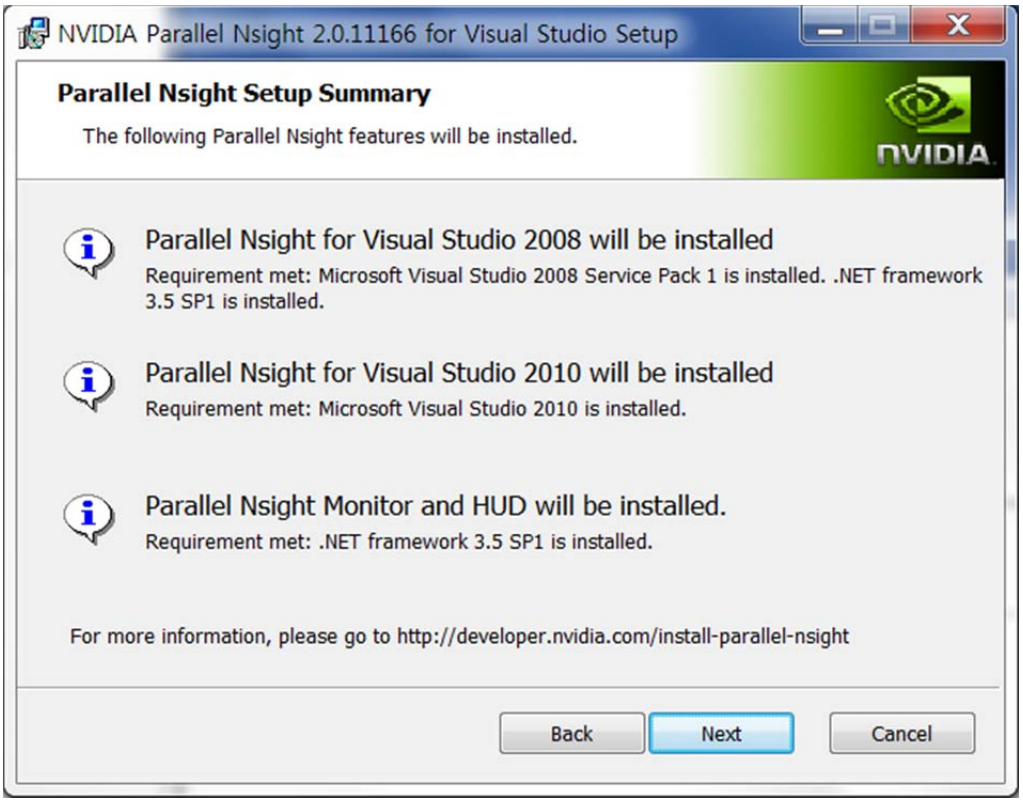
NVIDIA Parallel Nsight brings GPU Computing into Microsoft Visual Studio. Debug, profile and analyze GPGPU or graphics applications using CUDA C, OpenCL, DirectCompute, Direct3D, and OpenGL.

- Product Overview
  Introduction to Parallel Nsight on NVIDIA.com
- Support and Documentation
  User Guide, Forums, and more
- Videos
  Teaser and Instructional Videos showing Parallel Nsight in action
- Webinars
  Past Parallel Nsight Webinars
- Licensing
  License and Pricing information

C. 'Choose Setup Type'에서 'complete' 선택



D. 'Parallel Nsight Setup Summary' 다이얼로그 박스에서 아래와 같이 이미 install 되어 있는 MS Visual Studio 버전을 찾아서 진행 함.

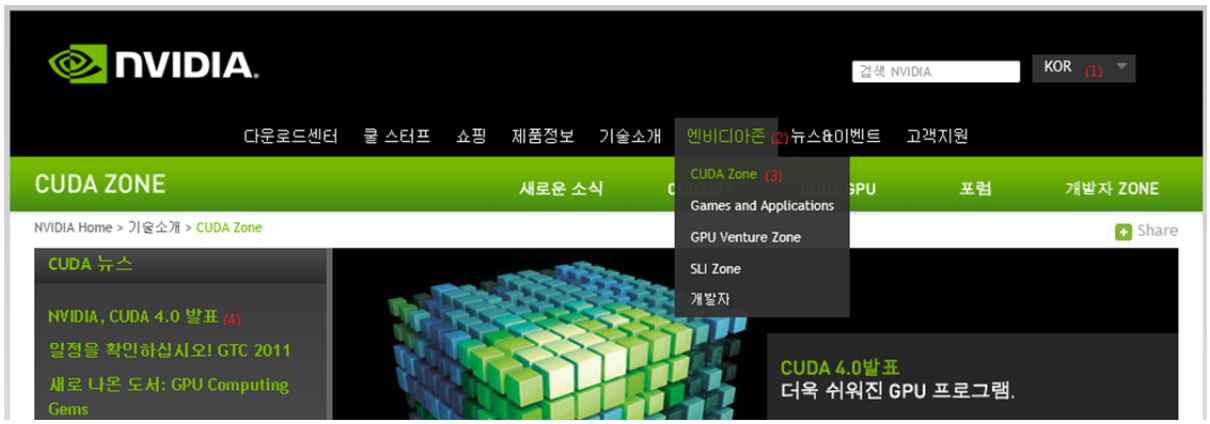


11. <Table 1>에서 'Other Tools and Libraries' link로 가서 필요한 툴 Download

GPGPU 개발환경 구축 : Linux 64-bit Ubuntu 11.04, CUDA 기준

1 우선 Ubuntu 를 인스톨 함(본 문서에서는 version 11.04, 64-bit)

2 http://www.nvidia.com 에서 아래 두 개의 그림과 같이 (1), (2), (3), (4), (5)의 순서대로 선택해 나감.



3 위 Windows 7 과 같은 방식으로 Nvidia 웹 메뉴를 선택

4 영문으로 설명하는 페이지가 표시되는데 그 중에서 다음과 같은 표를 확인.

Linux	Downloads
Developer Drivers for Linux (270.41.19)	32-bit 64-bit
CUDA Toolkit	documentation

Linux	Downloads
• C/C++ compiler • CUDA-GDB debugger • Visual Profiler • GPU-accelerated BLAS library • GPU-accelerated FFT library • GPU-accelerated Sparse Matrix library • GPU-accelerated RNG library • Additional tools and documentation	
CUDA Toolkit for Fedora 13	32-bit 64-bit
CUDA Toolkit for RedHat Enterprise Linux 6.0	64-bit
CUDA Toolkit for RedHat Enterprise Linux 5.5	32-bit 64-bit
CUDA Toolkit for RedHat Enterprise Linux 4.8	64-bit
CUDA Toolkit for Ubuntu Linux 10.10	32-bit 64-bit
CUDA Toolkit for OpenSUSE 11.2	32-bit 64-bit
CUDA Toolkit for SUSE Linux Enterprise Server 11 SP1	32-bit 64-bit
CUDA Tools SDK	32-bit 64-bit
GPU Computing SDK code samples	download
Other Tools and Libraries	link to page

< Table 3 : Linux Download>

1 위 <Table 3>의 'Developer Drivers for Linux (270.41.19)'을 download

2 설치 전에 혹시 설치되어 CUDA 가 설치 되어있다면 지워주어야 함

A. apt-get –purge remove nvidia-* (문제 발생 시 –purge 제외)

3 'Developer Drivers for Linux (270.41.19)' 를 설치

A. 참조문서

i. CUDA Getting Started Guide (Linux)

http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Getting_Started_Linux.pdf

ii. NVIDIA Accelerated Linux Graphics Driver README and Installation Guide

http://us.download.nvidia.com/XFree86/Linux-x86/256.35/README index.html

B. GUI 환경에서 빠져 나가야 함

i. 'Terminal' 프로그램을 시작한 후 'sudo /etc/init.d/gdm stop ' 또는 '/sbin/init 3 ' 실행해서 GUI 를 빠져나감.

ii. 또는 재부팅 후 recovery 에서 netroot 로 부팅

C. Download 한 Driver 프로그램의 폴더를 확인한 후, 'sudo sh <directory>/devdriver_4.0_linux_64_270.41.19.run' 를 실행(Driver file name 은 버전에 따라 변경될 수 있음)

i. NVIDIA Software Installer for Unix/Linux

① "You appear to be running in runlevel 1; this may cause problems. ..." 이 표시되면 "No" 선택.

② "Please read the following LICENSE and ..." 에서 "Accept" 선택

③ 이미 드라이버가 설치되어 있으면 "There appears to already be a driver installed ..." 표시가 되는데 "Yes" 선택

④ "Install NVIDIA's 32-bit compatibility OpenGL libraries?" 에 "Yes" 선택.

- "ERROR: File '/usr/lib/xorg/modules/extensions/libglx.so' is not a symbolic link." 발생하기도 함.

⑤ "WARNINGS: Your driver installation has been altered since..." 발생하기도 함.

⑥ "Would you like to run the nvidia-xconfig utility to automatically update your X configuration file so that the NVIDIA X driver will be used when you restart X? Any pre-existing X configuration file will be backed up." 표시되면 "Yes" 선택.

⑦ 인스톨 완료 시 "Your X configuration file has been successfully updated. Installation of the NVIDIA accelerated Graphics Driver for Linux-x86_64 (version:270.41.19) is now complete." 표시되고 Shell 로 나옴.

⑧ SLI 방식일 경우 SLI 설치 필요.

⑨ #startx 실행

⑩ 만약 화면에 아이콘들이 표시되지 않을 경우 "Ctrl"+"Alt"+"Del"로 Shut Down 후 다시 시작.

D. 맞는 버전의 Driver 가 설치되었는지 'cat /proc/driver/nvidia/version ' 명령어로 확인

예) 다음과 같이 표시됨.

$ cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module 270.41.19 Mon May 16 23:32:08 PDT 2011
GCC version: gcc version 4.5.2 (Ubuntu/Linaro 4.5.2-8ubuntu4)

E. CUDA 가 돌아갈 수 있도록 환경변수를 지정해 주어야 함

sudo nano ~/.bashrc 또는 sudo nano ~/bash_profile 에

맨 아랫줄에 다음 명령어를 적음

export CUDA_HOME="/usr/local/cuda"

export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${CUDA_HOME}/lib64"

//32 비트 이신 분들은 64 를 지워주시면 됩니다.

export PATH=${CUDA_HOME}/bin:${PATH}

그 후 적용

source ~/.bashrc 또는 source ~/.bash_profile

4 CUDA Toolkit 을 설치

A. sudo sh cudatoolkit_4.0.17_linux_64_ubuntu10.10.run

5 CUDA Tools SDK 를 설치

A. sudo sh cudatools_4.0.17_linux_64.run

6 GPU Computing SDK code samples 를 설치

A. sudo sh gpucomputingsdk_4.0.17_linux.run

7 현재(2011-07-11) 아직 CUDA 가 GCC 4.5 버전 이상을 지원하지 않으므로 GCC 4.4 버전을 받음

A. sudo apt-get install build-essential gcc-4.4 g++-4.4 libxi-dev libxmu-dev freeglut3-dev

B. 새로 폴더를 만들고 설치한 파일을 넣는다.

i. mkdir gcc44

ii. cd gcc44

iii. ln –s /usr/bin/cpp-4.4 cpp

iv. ln –s /usr/bin/gcc-4.4 gcc

v. ln –s /usr/bin/g++-4.4 g++

C. nvcc 에 등록

i. nano /usr/local/cuda/bin/nvcc.profile

ii. compiler-bindir = /home/xxx/gcc44 //gcc 폴더가 있는 곳을 작성

D. GUI 환경으로 돌아가기 위해 'startx' or 'init 5' or 'sudo /etc/init.d/gdm start ' 등 시스템 환경에 맞는 명령 실행

8 컴파일을 하고 실행 확인

A. Sample SDK 가 설치되 있는 폴더 -> C 폴더 -> sudo make 를 하시면 컴파일이 됨

B. 컴파일이 완료된 후 C -> bin -> linux -> release 에서 ./deviceQuery 하면 됨

9 기타 참고사항

A. SLI 를 인식하지 못하는 현상

i. sudo nvidia-xconfig --enable-all-gpus

ii. sudo nvidia-xconfig --sli=On

iii. sudo pico /etc/default/grub 여기서
'GRUB_CMDLINE_LINUX_DEFAULT="quiet splash vmalloc=256M" 라고 쳐주면 됨.
혹시 256M 으로 해서 안되면 192M 으로 바꾸면 됨.

iv. sudo update-grub

B. 드라이버를 잘못 설치하여 부팅이 되는 않는 현상

i. 전에 내용을 삭제 후 재설치

ii. 개발자 드라이버를 설치했는데 이렇게 되면 자신의 그래픽 카드에 맞는 드라이버를 찾아 설치해도 됨

C. 컴파일 중 에러

i. can not found –lcuda

① cuda 라이브러리를 못 찾는 경우라 드라이버를 다시 설치

D. 실행 시 에러

i. Libcudart.so.4

① PATH 설정이 잘못 되었을 경우가 크므로 자신의 컴퓨터의 환경에 맞게 해주어야 함.

10 참고 문헌

A. http://ubuntuforums.org/archive/index.php/t-1741962.html

B. http://hdfpga.blogspot.com/2011/05/install-cuda-40-on-ubuntu-1104.html

GPGPU 개발환경 구축 : OpenMP 환경 설치

<현재는 OpenMP 환경을 본 과제 플랫폼에 사용하지 않으므로 테스트 및 참조 용>

1. 관련 웹 사이트

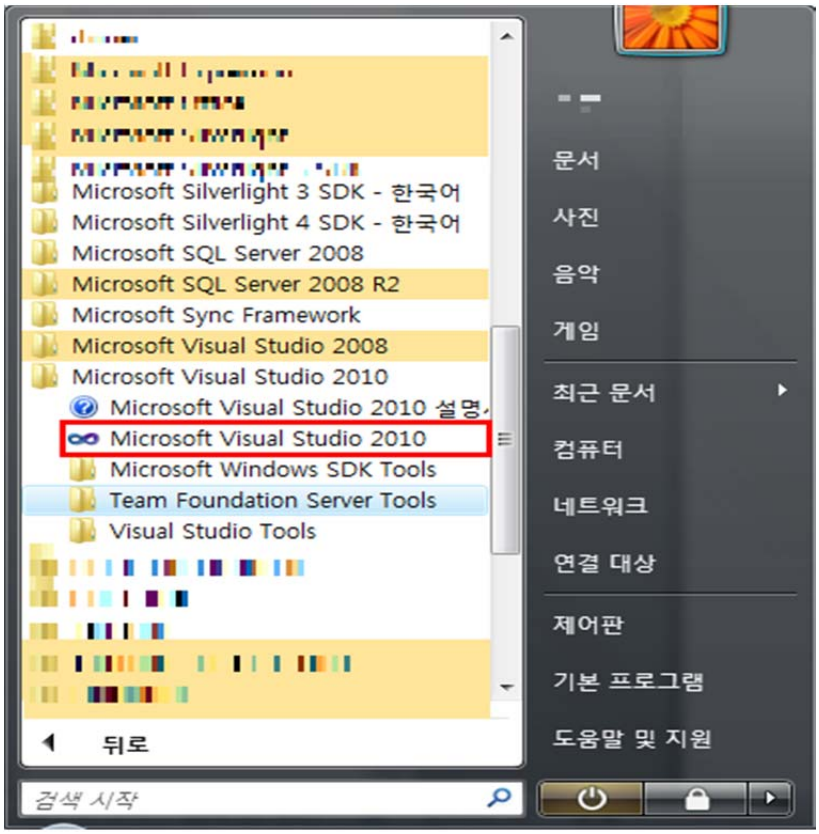
1.1 Home Page : http://www.openmp.org

1.2 Compilers Download Page : http://openmp.org/wp/openmp-compilers/

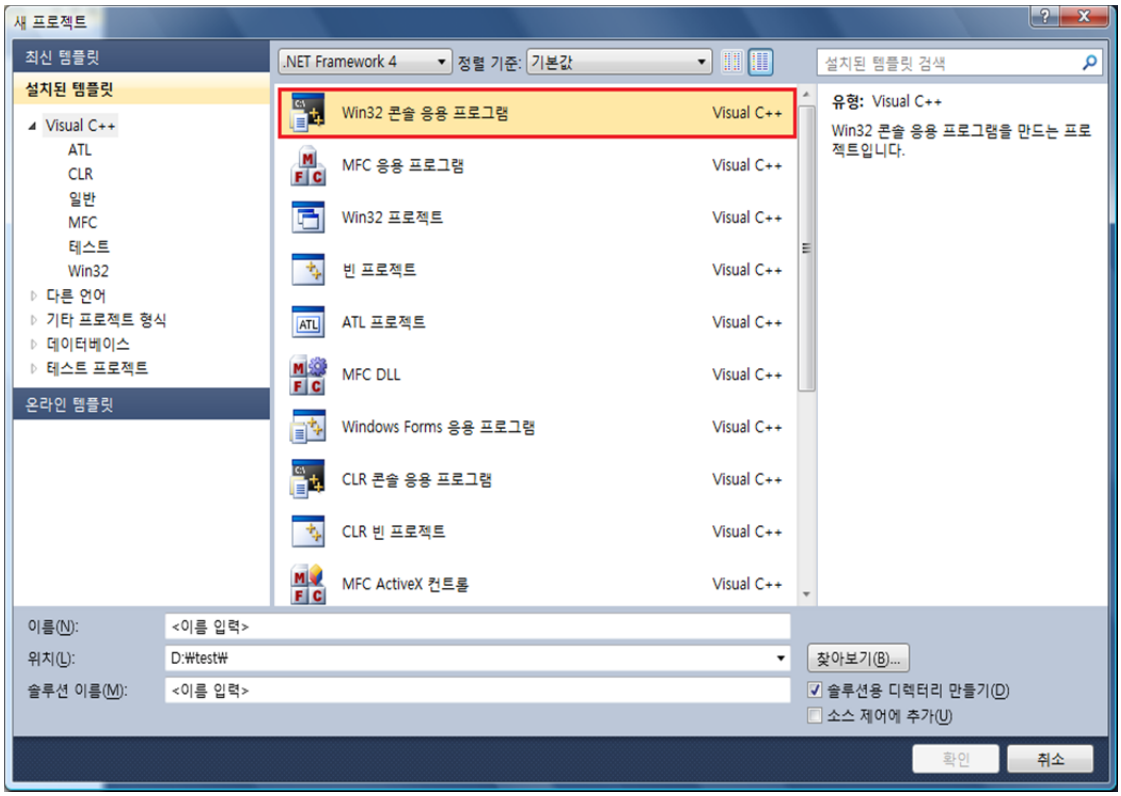
2. Visual Studio에서의 Compiling

2.1 OpenMP test

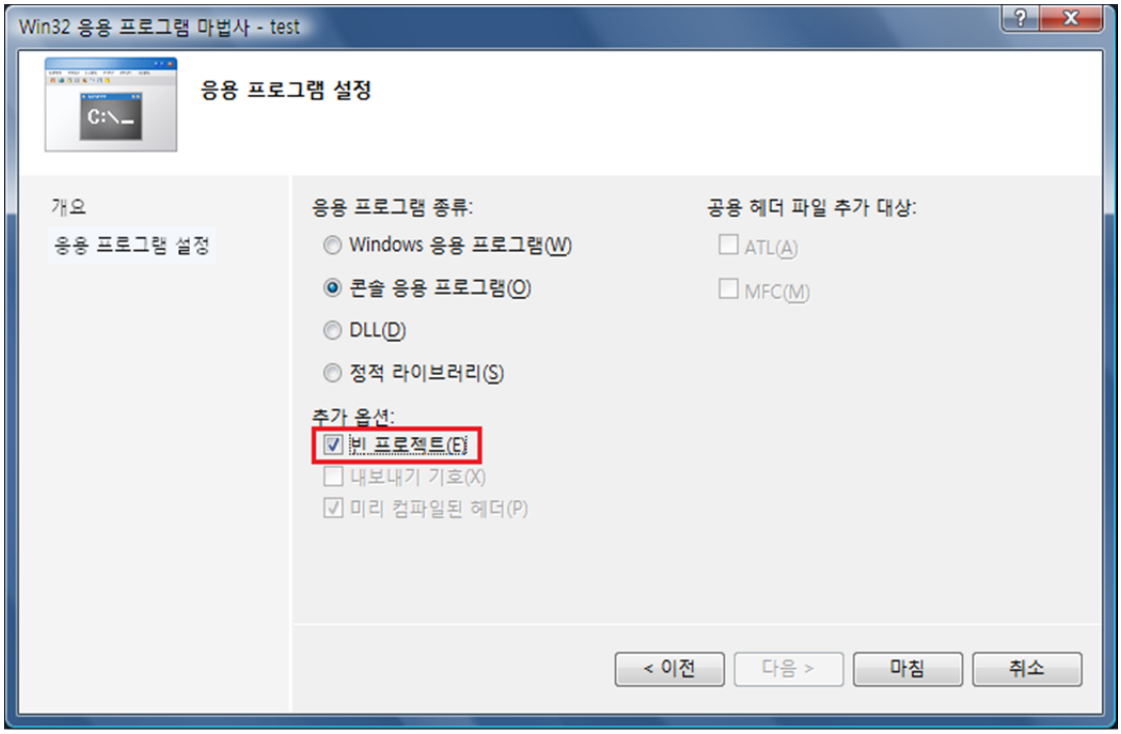
2.1.1 Visual Studio 2010(or 2008)을 실행한다



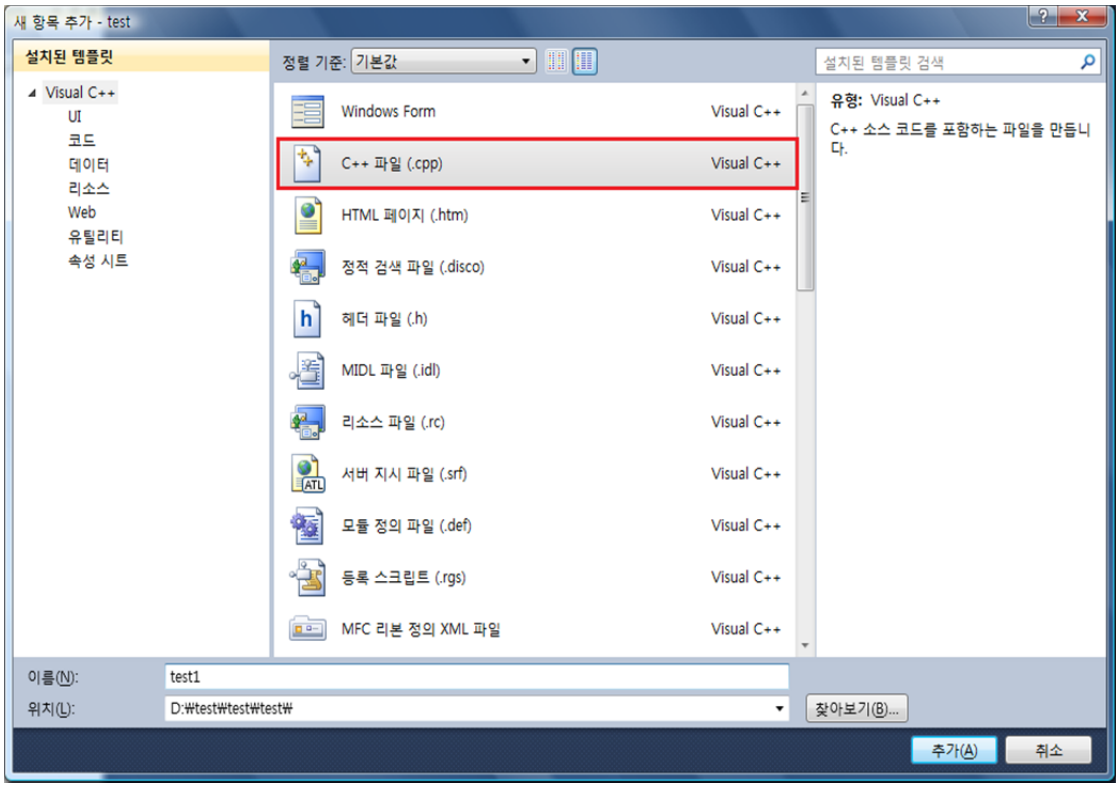
2.1.2 [파일] -> [새로 만들기] -> [프로젝트] -> [설치된 템플릿에서 Visual C++] -> [Win32 콘솔 응용 프로그램] 클릭 후 확인



2.1.3 추가 옵션에서 빈 프로젝트 체크 후 마침

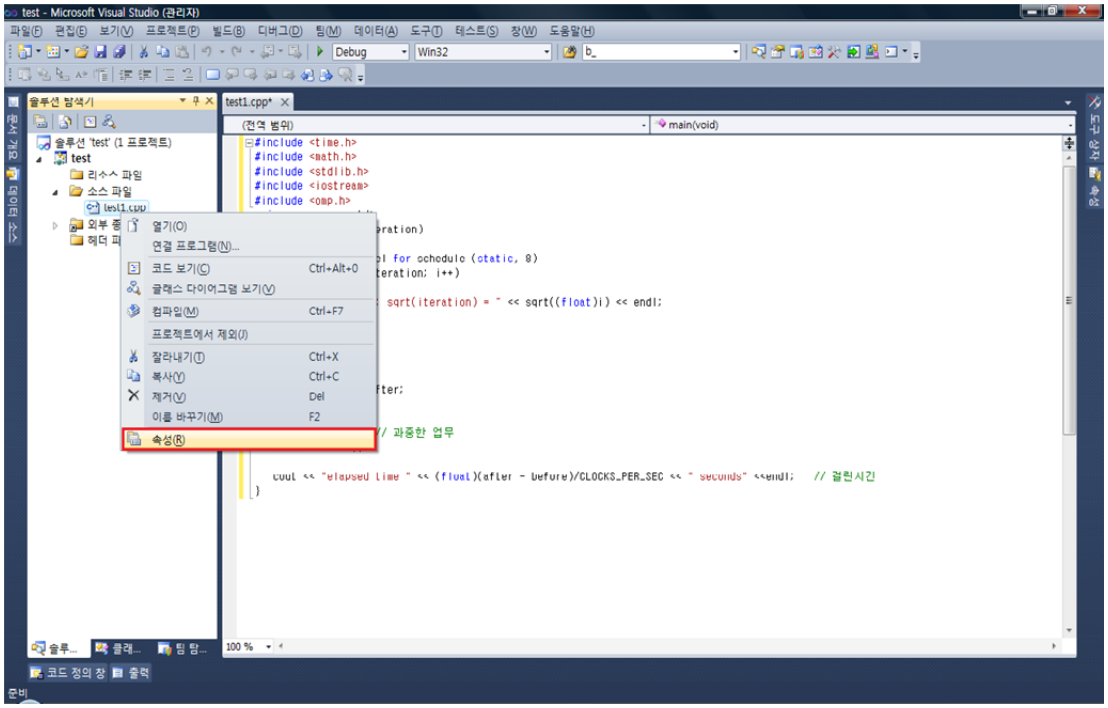


2.1.4 솔루션 탐색기 중 소스파일에서 마우스 오른쪽 클릭 -> [추가] -> [새 항목] 클릭 후 C++ 파일을 생성, 파일명은 test1

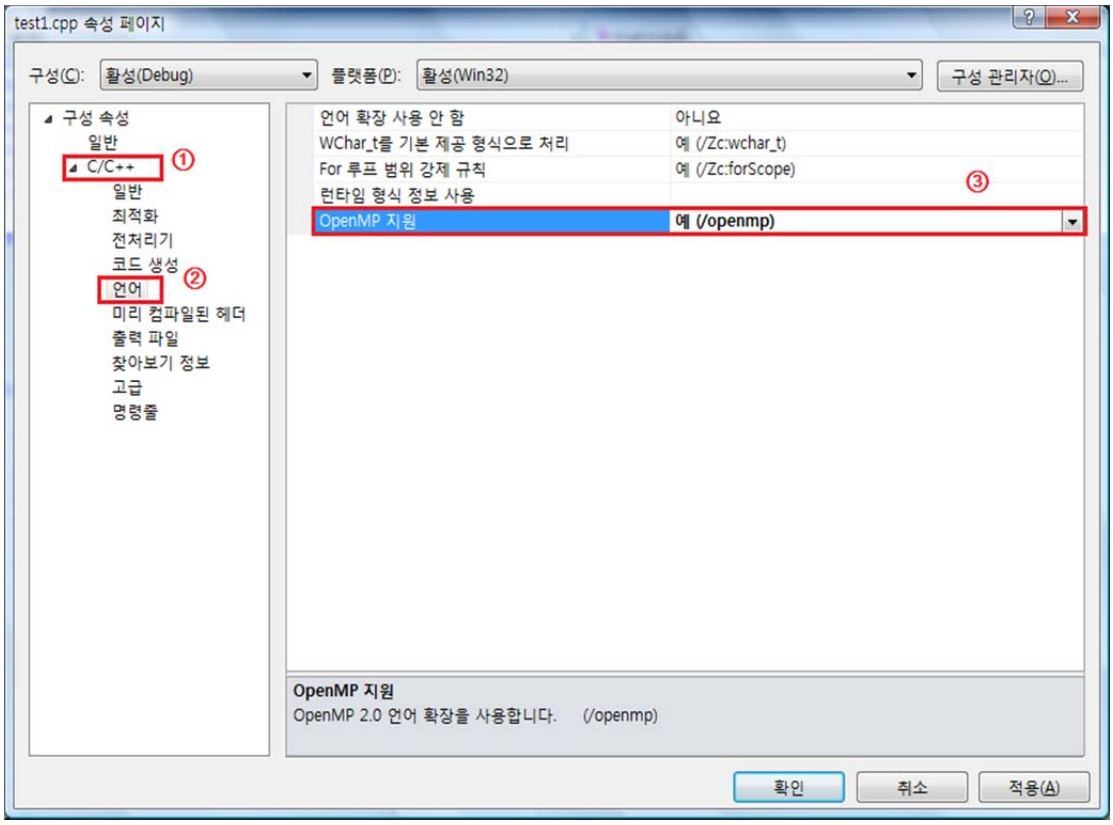


그리고 간단한 소스코드 *1)

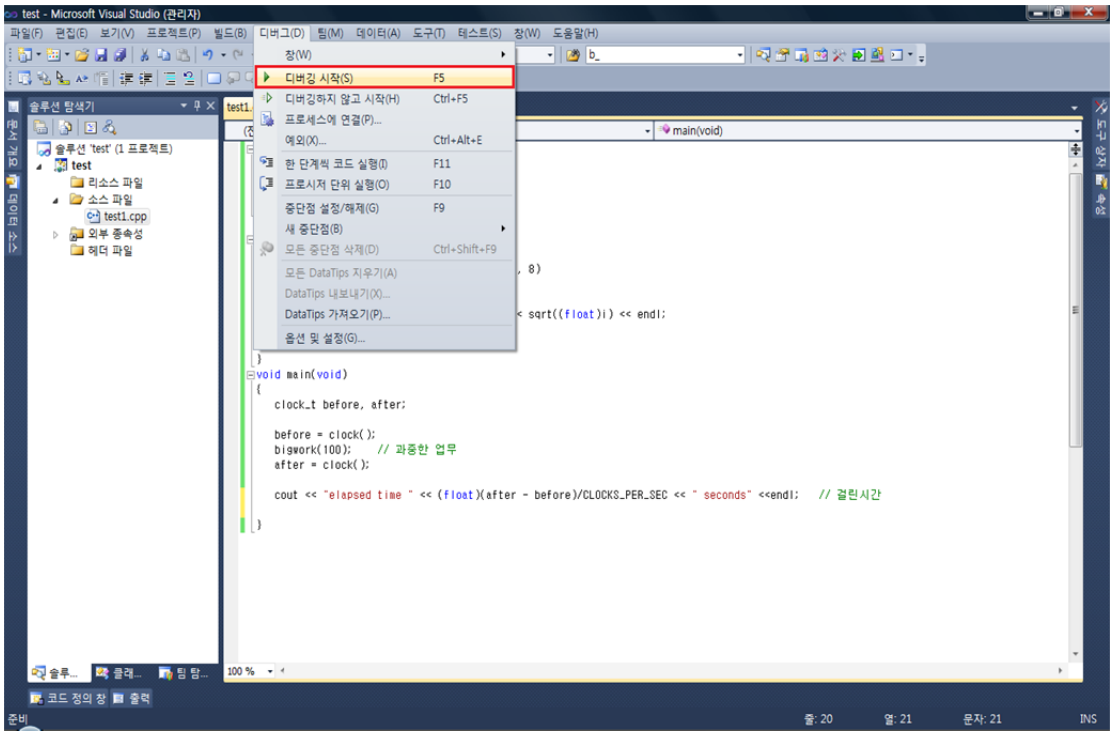
2.1.5 컴파일 전 test1.cpp 파일을 마우스 우클릭 후 속성을 클릭



2.1.6 [구성 속성] -> [C/C++] -> [언어] -> [OpenMP 지원] -> [예(/openmp)] 클릭

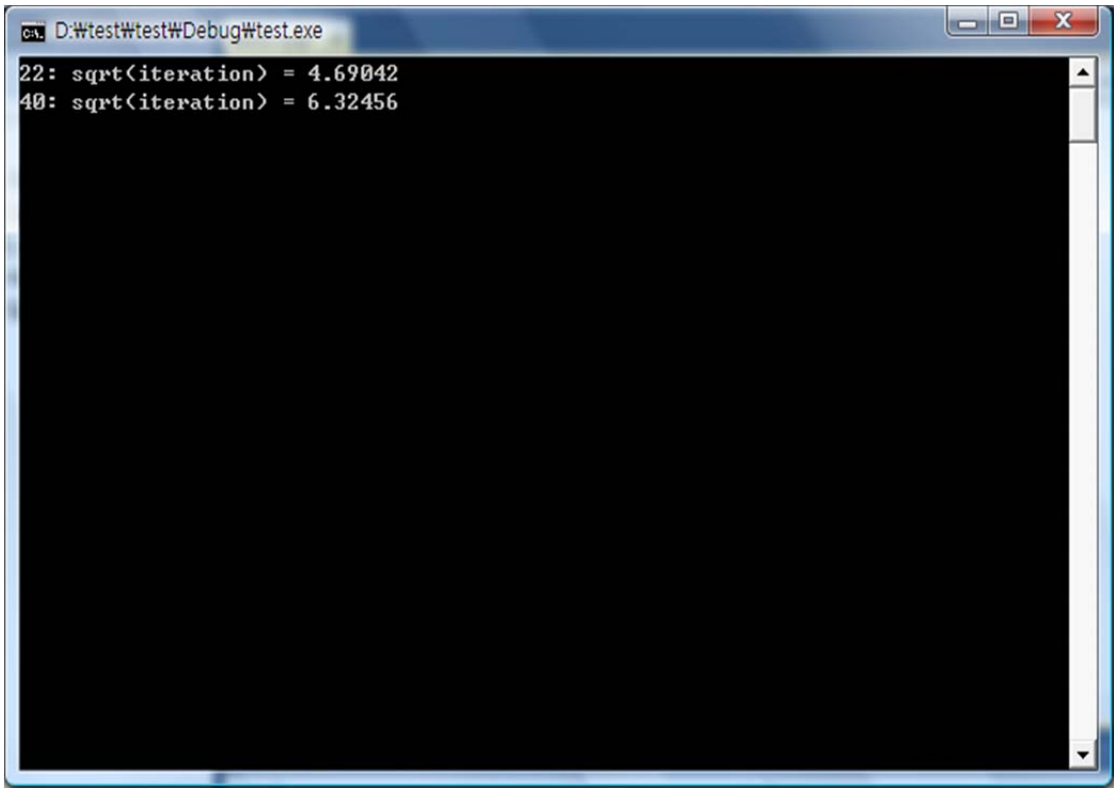


2.1.7 [디버그] -> [디버깅 시작] 클릭. 단축키로는 F5

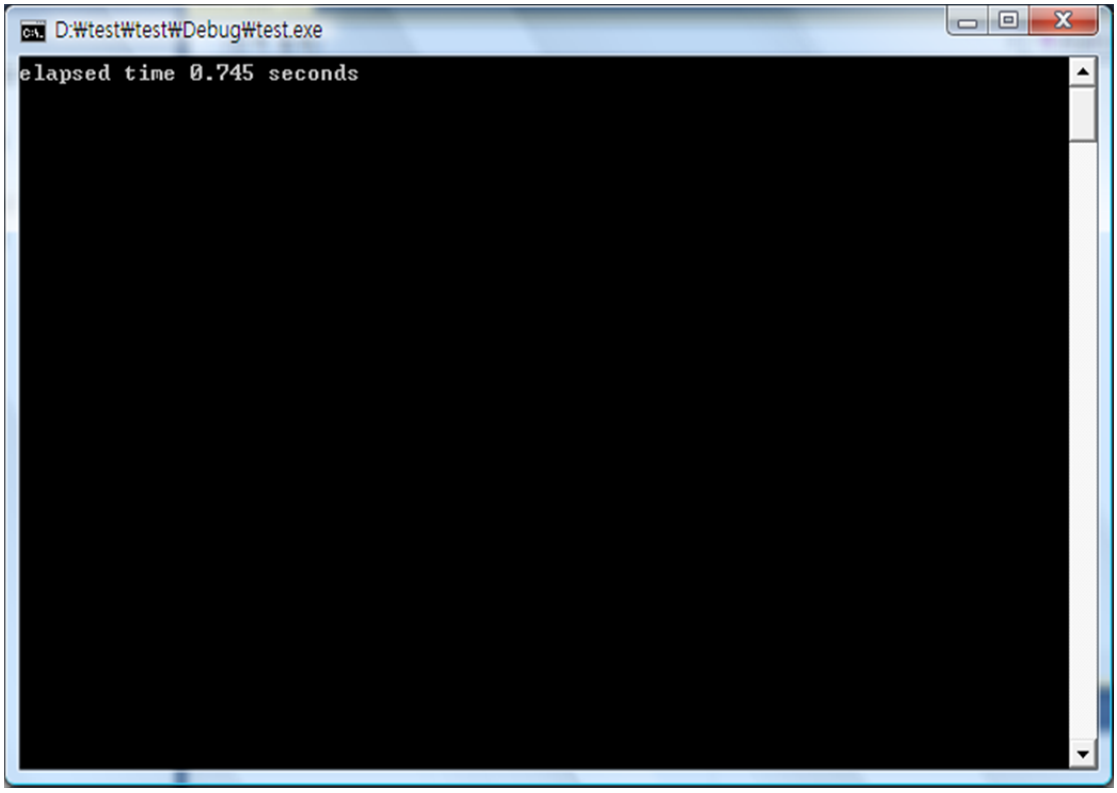


2.1.8 실행 중 캡쳐

실행중



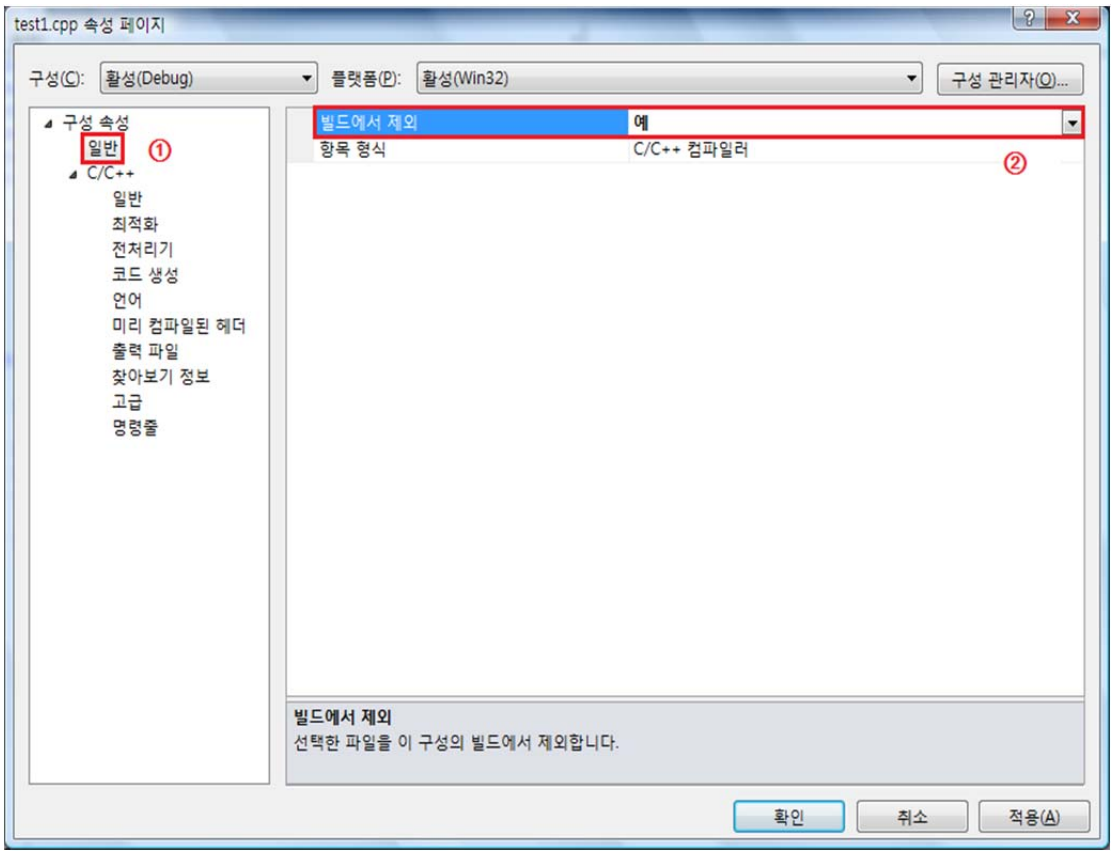
실행 후



2.2 OpenMP를 사용하지 않은 파일과의 비교

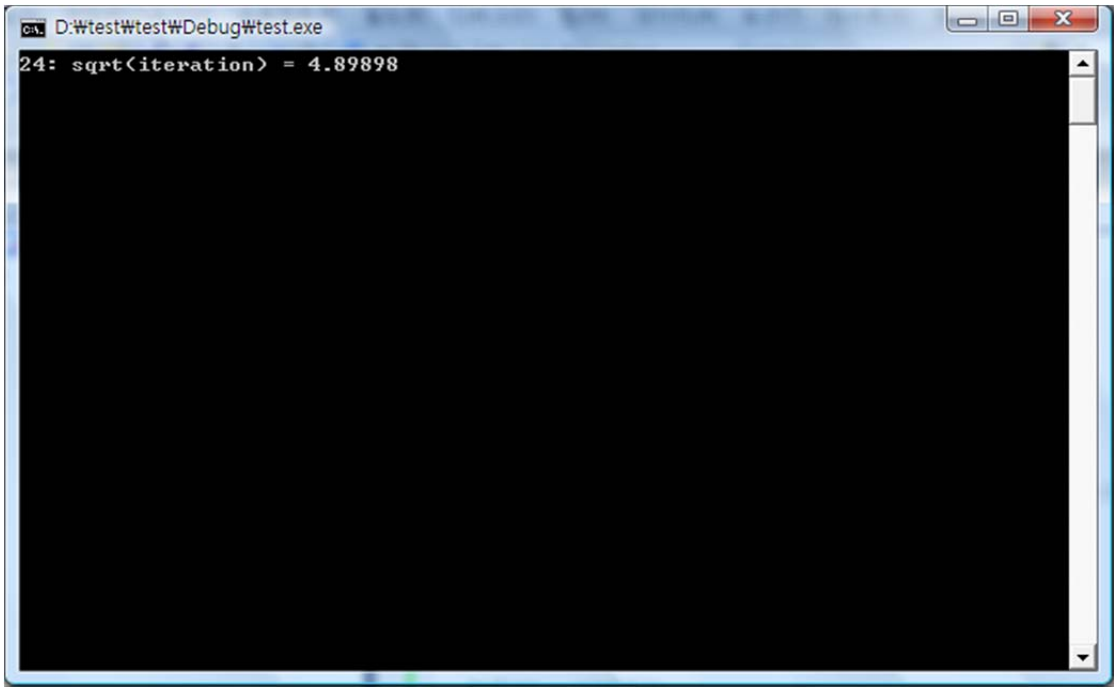
2.2.1 위의 2.1에서 4번만 하되, 파일명은 test2.cpp로 한다. *2)

2.2.2 컴파일 하기 전 test1.cpp파일을 빌드에서 제외 시킨다.

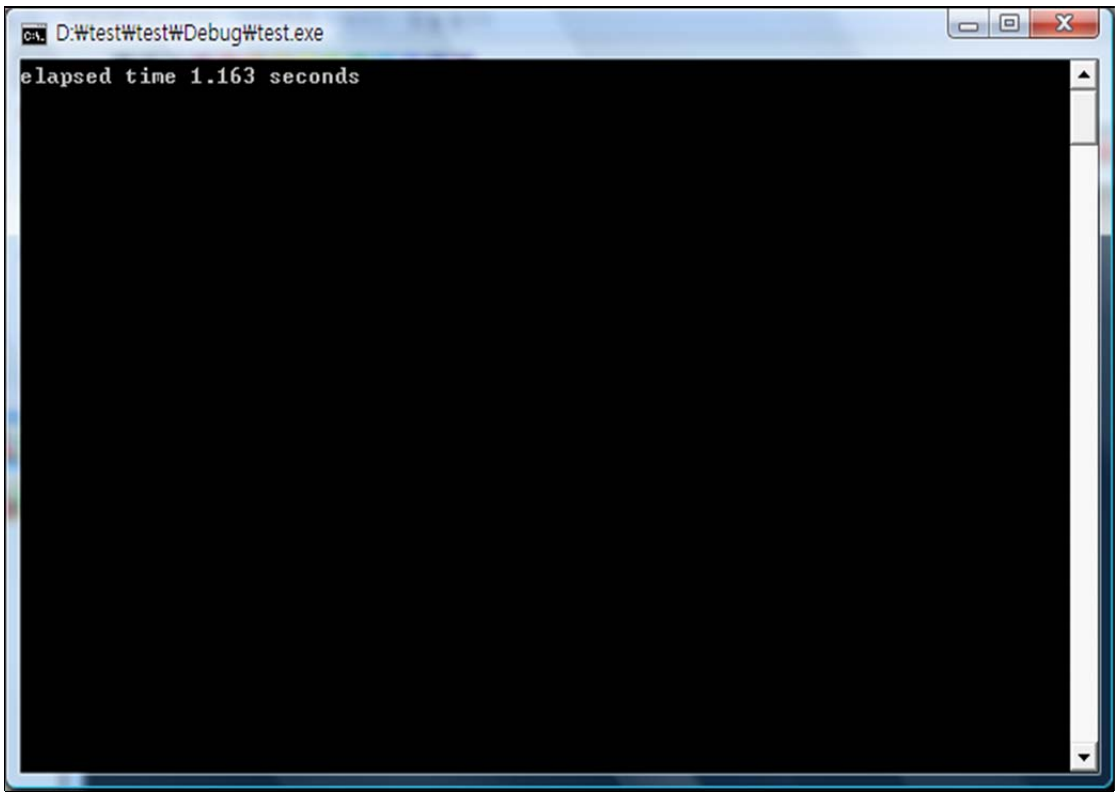


2.2.3 컴파일을 한다. [단축키 F5]

실행 중



실행 후



3. 두 파일의 실행 후 비교

3.1 실행 후의 비교 표

파일명 / 순서	test1.cpp	test2.cpp
OpenMP 사용유무	O	X
bigwork(n)	100	100
실행 결과 시간(s)	0.745[약 36%단축]	1.163
bigwork(n)	1000	1000
실행 결과 시간(s)	7.826[약 27%단축]	10.67

부록

*1)

```
#include <time.h>
#include <math.h>
#include <stdlib.h>
#include <iostream>
#include <omp.h>
using namespace std;
void bigwork(int nIteration)
{
  #pragma omp parallel for schedule (static, 8) //숫자 8은 스레드의 개수를 말함
    for(int i=0; i<nIteration; i++)
    {
        cout << i << ": sqrt(iteration) = " << sqrt((float)i) << endl;
        system("cls");
    }
}
void main(void)
{
    clock_t before, after;

    before = clock();
    bigwork(100);      // 과중한 업무
    after = clock();
    cout << "elapsed time " << (float)(after - before)/CLOCKS_PER_SEC << " seconds" <<endl;    //
걸린시간
}
```

[출처 : http://sapeyes.blog.me/70033771510]

부록

*2)

```
#include <time.h>
#include <math.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
void bigwork(int nIteration)
{
    for(int i=0; i<nIteration; i++)
    {
        cout << i << ": sqrt(iteration) = " << sqrt((float)i) << endl;
        system("cls");
    }
}
```

```
void main(void)
{
    clock_t before, after;

    before = clock();
    bigwork(100);      // 과중한 업무
    after = clock();
    cout << "elapsed time " << (float)(after - before)/CLOCKS_PER_SEC << " seconds" <<endl;    //
걸린시간
}
```

GPGPU 개발환경 구축 : MPI 환경 설치

<현재는 MPI 환경을 본 과제 플랫폼에 사용하지 않으므로 테스트 및 참조용>

1 Home Page

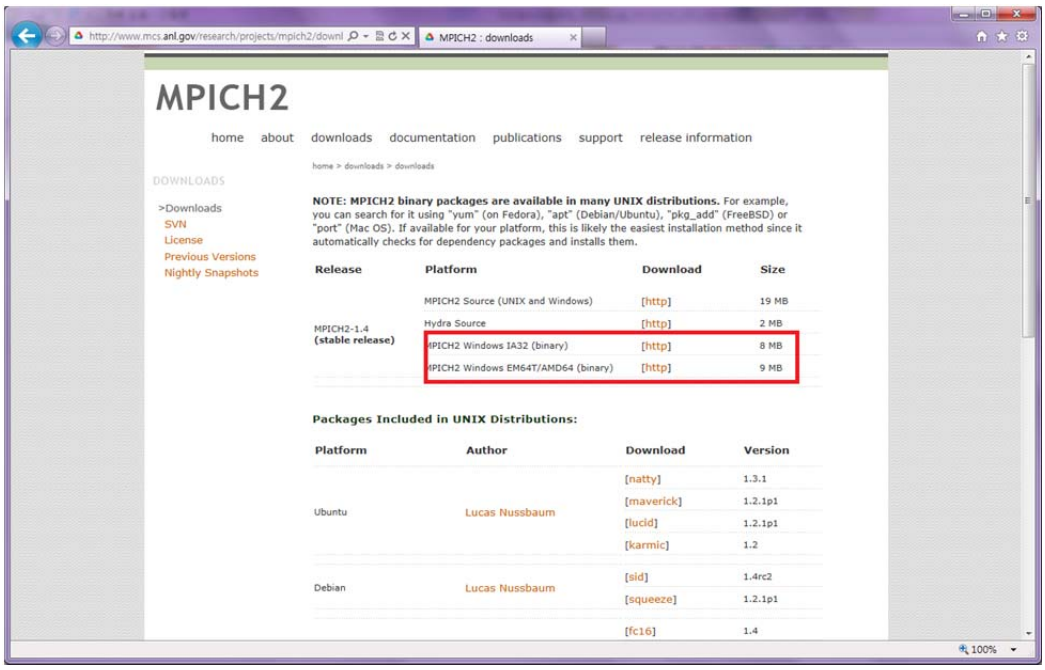
A. MPICH2 : http://www.mcs.anl.gov/research/projects/mpich2/downloads/index.php?s=downloads

B. Microsoft HPC SDK Pack : http://www.microsoft.com/download/en/details.aspx?id=10505

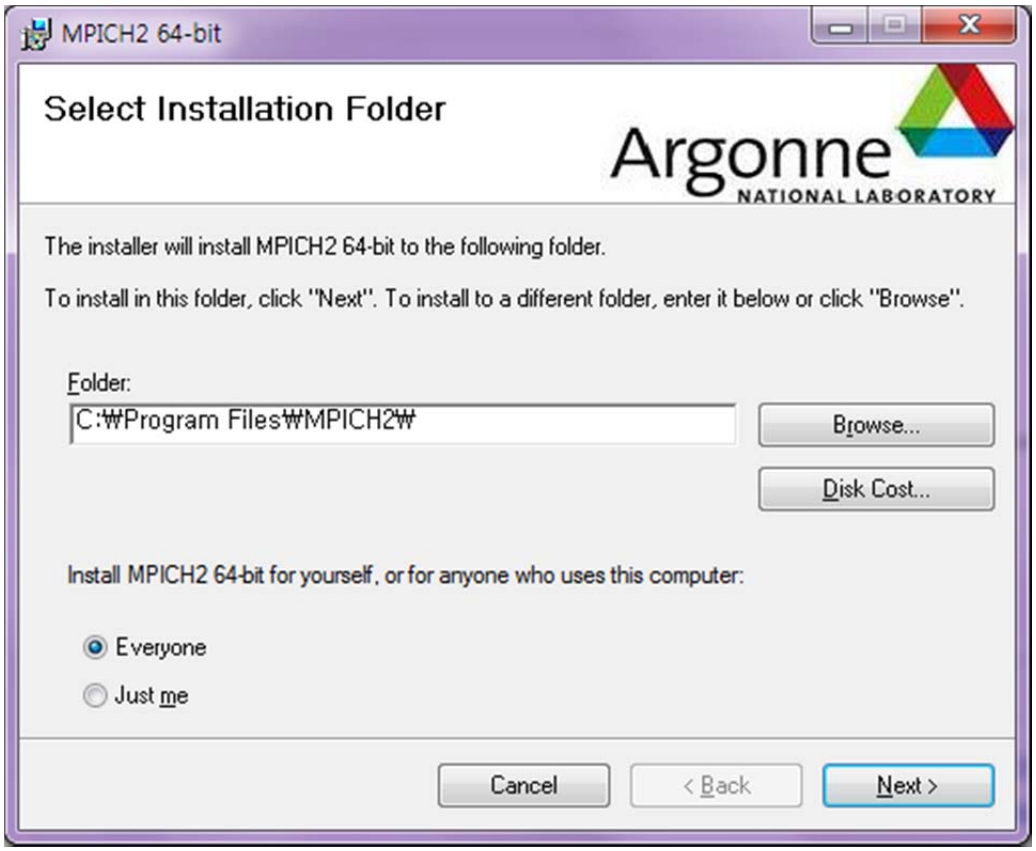
2 설치 및 환경 구성 방법

A. MPICH2

i. 해당 사이트에서 자신의 컴퓨터에 맞는 프로그램을 다운을 받는다.

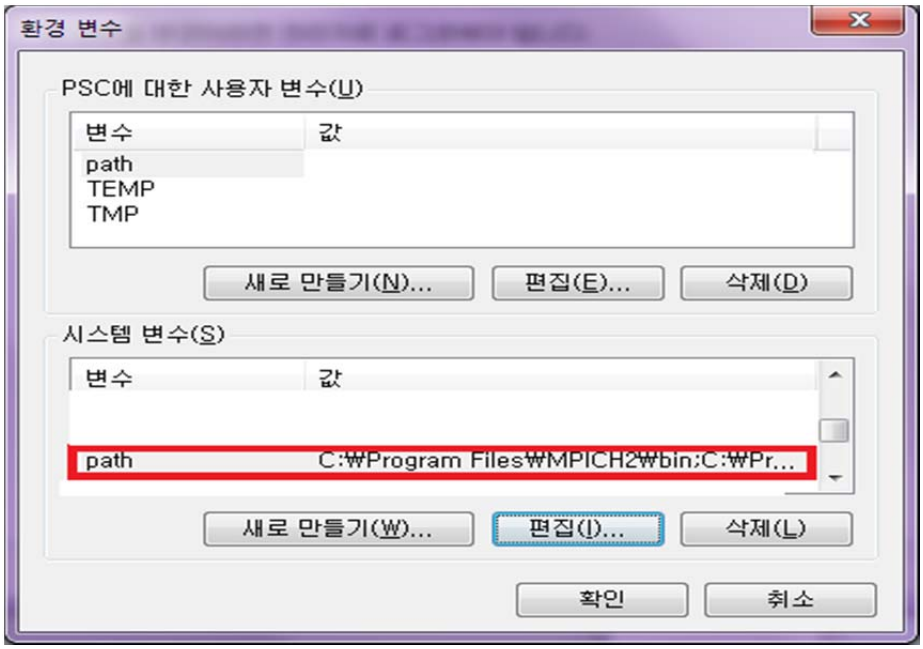


ii. 다운 받은 파일을 실행 및 설치



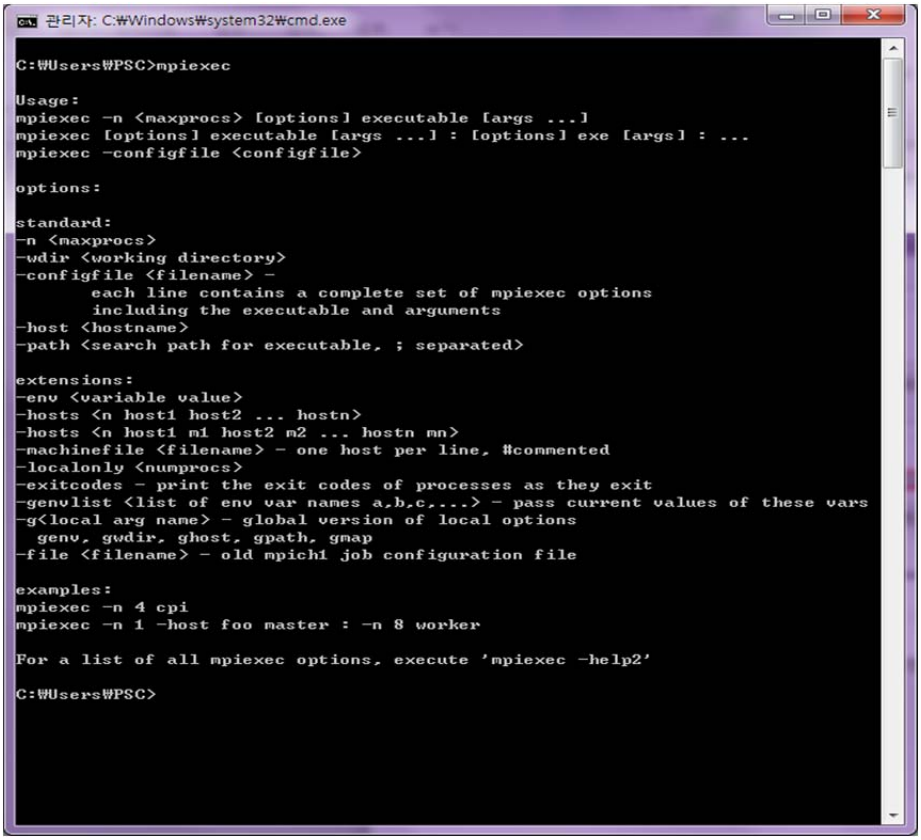
① 설치폴더는 기본으로 하되 그림과 같이 Everyone 으로 하는 것이 좋음

iii. 설치를 마친 뒤에는 환경변수 path 에 C:₩Program Files₩MPICH2₩bin 를 추가



iv. 환경변수 등록을 마쳤으면 프로그램이 돌아가는지 확인

① 실행창에서 cmd 를 실행시킨 뒤에 mpiexec 를 타이핑



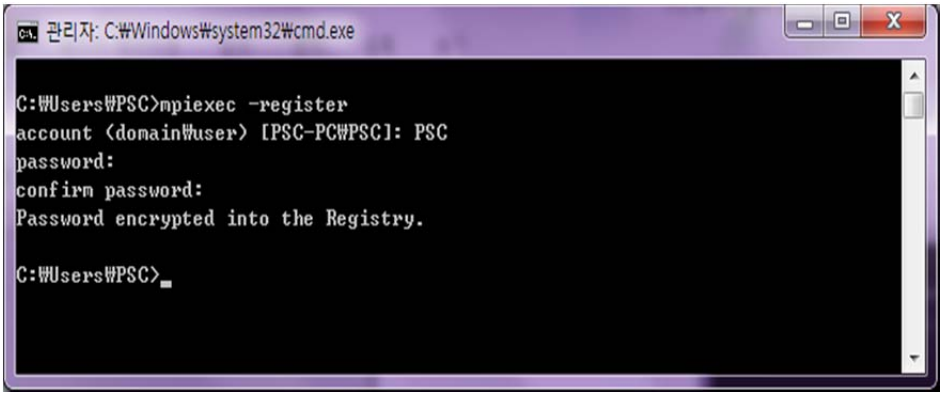
② 이러한 내용이 나왔다면 성공

v. 그 후 계정을 등록

① 이어서 mpiexec –register 를 타이핑

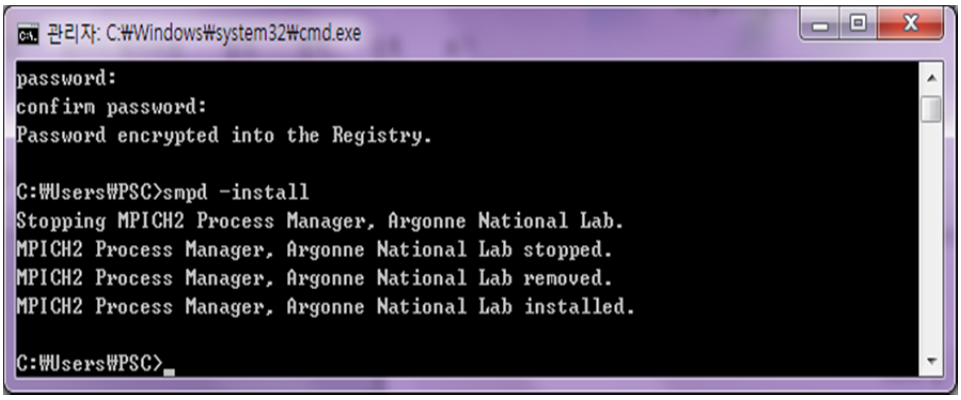
② 계정은 지금 현재 로그인 되어있는 윈도우 계정을 등록하는데 비밀번호 역시 같게 함.

③ 비밀번호가 없을 시에는 등록해야 함

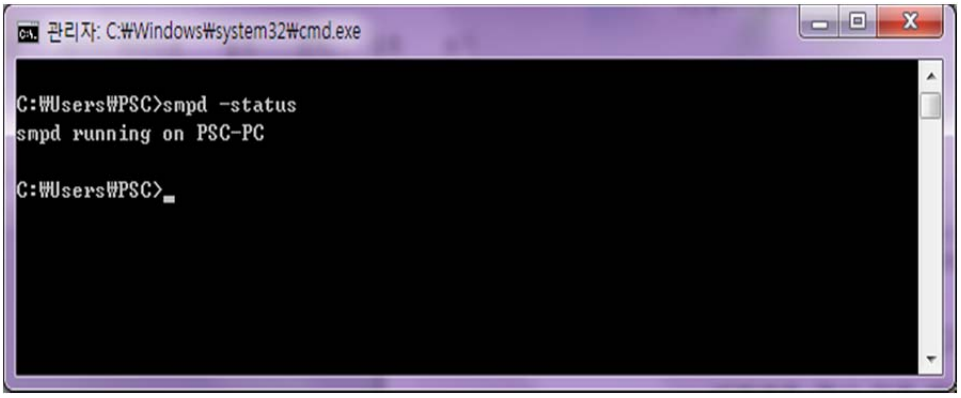


vi. 등록을 마치면 smpd 를 설치하고 작동상태를 확인

① 명령창에 smpd –install 이라고 치면 설치가 됨



② 그 후 smpd –status 를 쳐서 작동이 되는지 확인



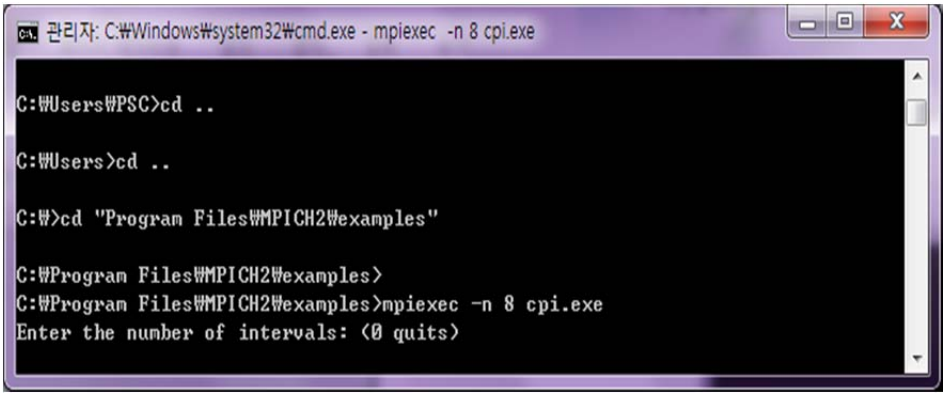
③ 그 다음 mpiexec –validate 를 쳐서 mpi 가 작동되는지 확인

vii. 작동이 정상적인지 확인

① Example 이 있는 폴더로 이동해서 작동

mpiexec –n x cpi.exe

(x : CPU 개수)



viii. 다른 컴퓨터와 연결 하여 MPI 를 작동시킬 때

① 같은 WORKGRUOP 에 있어야 할 것

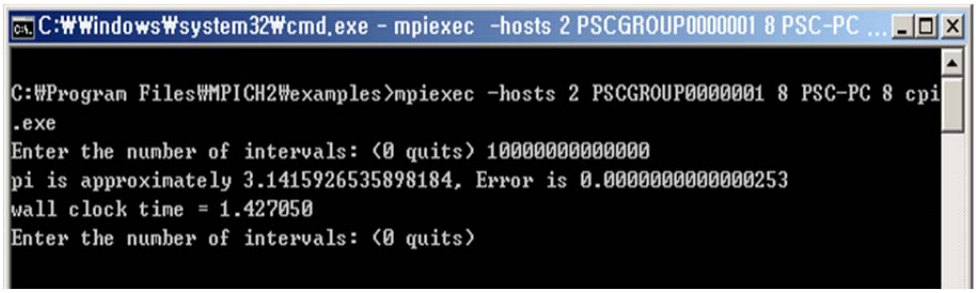
② 같은 계정과 비밀번호를 써야 할 것

③ 그 후 다음과 같은 코드를 씀.

mpiexec –hosts n xxx m yyy l cpi.exe

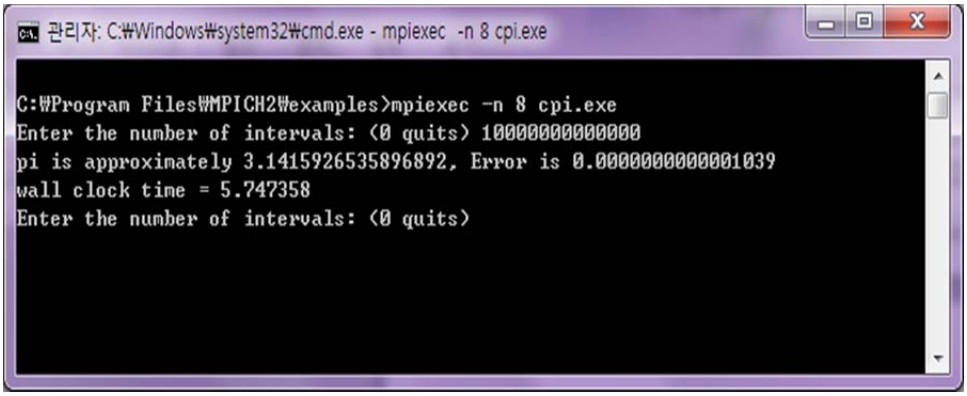
(n : 컴퓨터 개수 xxx, yyy :컴퓨터 명 m, l : CPU 개수)

④ 속도 비교 => 1.427050 : 5.747358 => 약 4 배 이상 성능 향상



< Intel I7 2630QN(Quad cores x 2 hyper-threading)

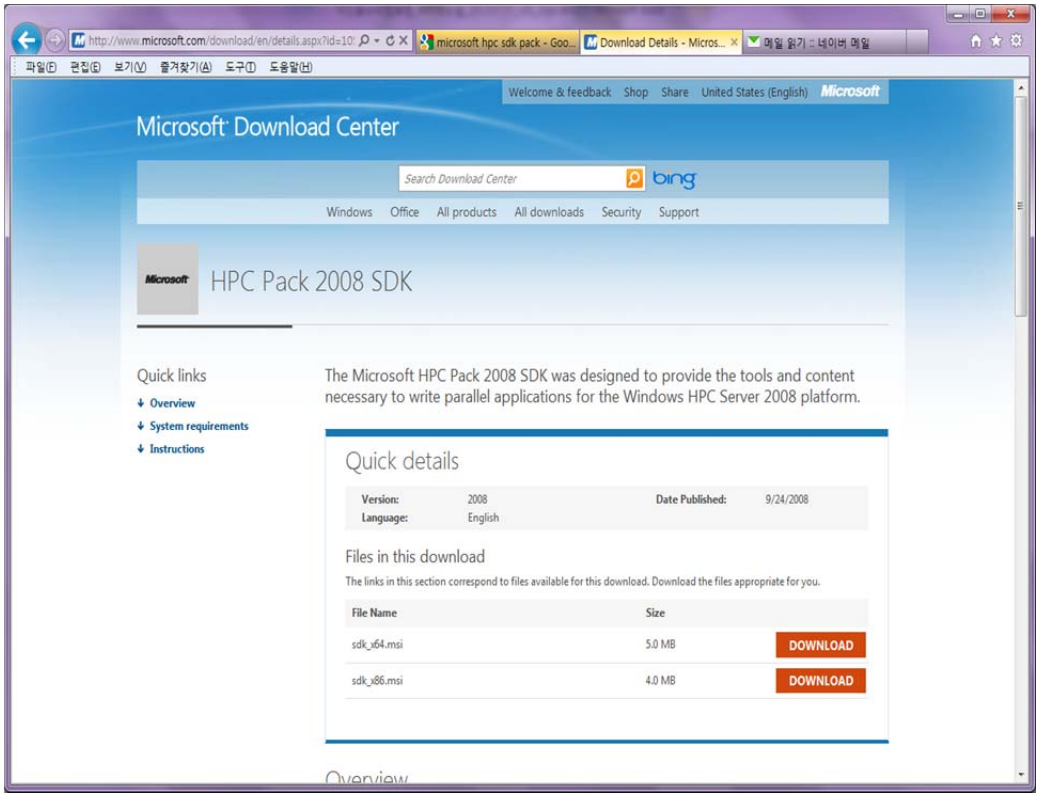
+ Intel Xeon E5620(Quad cores x 2 hyper-threading)>



<Intel I7 2630QN(Quad cores x 2 hyper-threading)>

B. Microsoft HPC SDK Pack

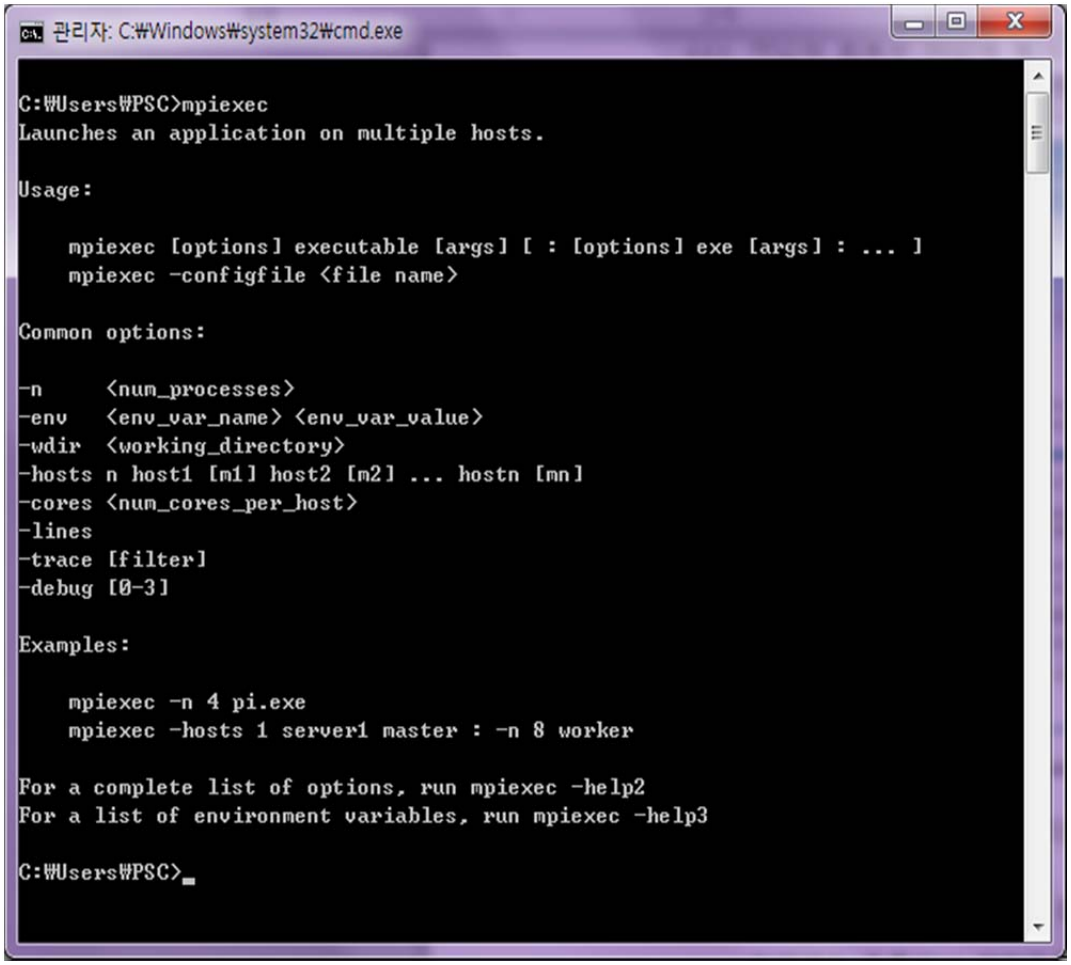
i. 자신의 윈도우 버전에 맞게 프로그램을 다운로드 받는다



ii. 다운로드 받은 파일을 설치를 한다

iii. HPC Pack 은 MPICH2 와 다르게 자동으로 셋팅을 다 해줘서 따로 환경변수 등록이나 계정등록을 할 필요가 없다

iv. 명령창에서 작동이 잘 되었는지 확인



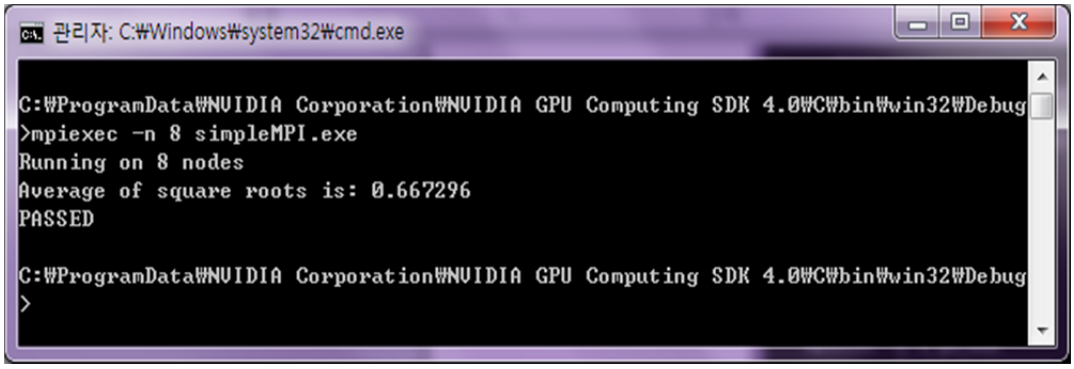
v. 이런 식으로 나오면 작동이 잘 되는 것인데, MPICH2 와 별로 차이가 보이지 않아서 혼동될 경우 아랫줄에 help 를 보면 됨. 앞에서 MPICH2 는 help3 이 존재하지 않음

vi. 여기서 테스트 할 샘플은 CUDA 에 있는 simpleMPI 임. 그래서 앞의 문서를 보고 CUDA 를 설치해야 함.

vii. 설치를 마치고 cuda 예제가 있는 폴더로 이동한 뒤 작동 시도

viii. 프로그램 실행은 MPICH2 와 같음

mpiexec –n x simpleMPI.exe



PSC Platform 구축 : Hadoop-mapreduce 환경 설치(Linux)

1 참조 Home Page

A. Hadoop : http://hadoop.apache.org/

B. Java : http://www.java.com/ko/

2 Process

A. 디렉토리 구성에서 주의할 점.

i. 본 문서에서는 구축되는 모든 시스템의 디렉토리를 /home/psc/pscf_01 라고 만들어서 통일시키고 그 아래에서 생성할 것임.(Hadoop 만을 사용한다면 유저 이름(예 : psc)과 디렉토리(pscf_01)는 시스템에 따라 각각 다른 것으로 사용해도 문제는 없지만 GPGPU 를 사용하기 위해서는 동일하게 지정 해야 함. 이 문제는 Hadoop 의 에러로 추정되는 현상으로 비롯된 것이므로 차기 버전에서는 해결될 수도 있음.)

ii. Hadoop+GPGPU 시스템을 구축하게 되는데 기본 플랫폼이 되는 Hadoop 시스템에 맞추기 위해서는 모든 Master, Slave 서버들은

동일한 디렉토리 구조와 동일한 login name(user name)을 가져야 함.

iii. ~/.bashrc 에 다음을 추가.(다음의 환경변수를 적용하려면 시스템을 reboot 하거나 아래의 export 문을 Shell 에서 실행.)

export PSCF_HOME=/home/psc/pscf_01

B. Hadoop 을 다운받음.

i. 첫 홈페이지에서 Common 페이지에 들어가면 Getting Start 부분에 Download 가 있음

ii. 다운받은 Hadoop 의 압축을 해제($PSCF_HOME 아래 디렉토리에서 압축 해제)

① tar xzf hadoop-0.20.2.tar.gz (더 최근 버전 사용 시 문제가 발생하므로 이 버전 사용 요망)

iii. Java runtime environment 가 없을 경우 먼저 인스톨 해야 함.

① http://www.oracle.com/technetwork/java/javase/downloads/index.html 에서 최신 JDK 를 다운받아서 설치해야 함.

- linux x64 버전의 jdk-7u1-linux-x64.tar.gz 를 /usr/local 에 download 했음.

② 여기에서는 다음과 같이 install.

- $ cd /usr/local
- $ sudo tar xzf jdk-7u1-linux-x64.tar.gz

iv. Hadoop 을 사용하려면 java 의 환경변수를 등록해주어야 함.

① cd hadoop-0.20.2

② nano ~/.bashrc 에서 다른 CUDA, JCUDA 환경변수보다 먼저(PSCF_HOME 보다는 아래) 추가(Hadoop install 후 처음 한 번만 추가)

- Hadoop, Java 환경변수를 적어 줌. 정확한 디렉토리 확인 필요.

  export JAVA_HOME=/usr/local/jdk1.7.0_01

  export HADOOP_INSTALL=/home/psc/pscf_01/hadoop-0.20.2

  export HADOOP_HOME=/home/psc/pscf_01/hadoop-0.20.2

  export PATH=${HADOOP_HOME}/bin:${PATH}

- Hadoop Library 추가 등록

  Export LD_LIBRARY_PATH=[기존 등록된 Path]:${HADOOP_INSTALL}/lib/native

- $source ~/.bashrc
- $sudo ldconfig

③ Hadoop environment setting

- vi conf/hadoop-env.sh

  1. Uncomment "export HADOOP_HEAPSIZE=2000"
  2. Comment "#export HADOOP_LSAVES=${HADOOP_HOME}/conf/slaves"

- $source conf/hadoop-env.sh

④ 진행 중 발생할 수 다음과 같은 에러, '여러분의 현재 네트워크는 .local 도메인을 가지고 있습니다. 이는 Avahi 네트워크 서비스 탐색 서비스에 문제를 일으킬..."에 대한 대처.

- '네트워크 서비스 탐색 비활성화' 오류 알림은 우분투 9.04 이후 발견된 공식 버그로, ISP(Internet Service Provider, 인터넷 서비스 공급업체)에 따라 발생할 수 있는 문제로 알려짐.

해결 방법 1

버그가 알려진 이후 우분투 한국 사용자 모임을 통해 알게 된 해결 방법이다.

이 방법은 /etc/default/avahi-daemon 파일의 설정 내용을 수정하는 것으로, 터미널에서 아래 명령어를 실행해 해당 파일을 불러온다.

$ sudo gedit /etc/default/avahi-daemon

그리고 AVAHI_DAEMON_DETECT_LOCAL=1 의 숫자를 '0'으로 수정한다.

AVAHI_DAEMON_DETECT_LOCAL=1

AVAHI_DAEMON_DETECT_LOCAL=0

하지만 우분투 9.10 이후로는 이 설정의 위치가 변경된 것으로 판단된다. 이렇게 되면 빈 /etc/default/avahi-daemon 파일을 불러오는데, 그냥 'AVAHI_DAEMON_DETECT_LOCAL=0'을 입력하고 저장하면 다음 로그인 때부터는 오류 알림이 나타나지 않는다.

해결 방법 2

최근 떠돌이(@bugbear5) 님의 블로그에서 ferret 님을 통해 알게 된 해결 방법이다.

이 방법은 /usr/lib/avahi/avahi-daemon-check-dns.sh 파일의 설정 내용을 수정하는 것으로, 나머지는 방법 1 과 같다. 터미널에서 아래 명령어를 실행해 해당 파일을 불러온다.

$ sudo gedit /usr/lib/avahi/avahi-daemon-check-dns.sh

그리고 12 번째 줄 AVAHI_DAEMON_DETECT_LOCAL=1 의 숫자를 '0'으로 수정한다.

AVAHI_DAEMON_DETECT_LOCAL=1

AVAHI_DAEMON_DETECT_LOCAL=0

파일을 저장하고 나면 다음 로그인 때부터는 오류 알림이 나타나지 않는다.

* 참조 : http://cafe.naver.com/udtssueod.cafe?iframe_url=/ArticleRead.nhn%3Farticleid=578&

⑤ Hadoop 은 3 가지의 실행모드가 있음. : **테스트 시에는 Pseudo-Distributed Operation 권장. 2 Nodes(Workstations or racks) 이상일 경우에는 Fully-Distributed Operation 사용**

⑥ Standalone Operation

- mkdir input
- cp conf/*.xml input
- bin/hadoop jar hadoop-examples-0.20.203.0.jar grep input output 'dfs[a-z.]+'
- cat output/*

⑦ Pseudo-Distributed Operation (1 대의 서버로 테스트 시 권장)

- /etc/hosts 의 로컬호스트 아이피가 주석처리(#) 되어 있다면 아래와 같이 주석처리를 빼고 다시 되돌림.
  127.0.0.1 localhost
  127.0.1.1 XXXXXX

- Setting

1. conf/core-site.xml

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

2. conf/hdfs-site.xml

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

3. conf/mapred-site.xml

```
<configuration>
    <property>
        <name>mapred.job.tracker</name>
        <value>localhost:9001</value>
    </property>
</configuration>
```

4. conf/master 에 다음과 같이 되어 있는지 확인.

   localhost

5. conf/slaves 에 다음과 같이 되어 있는지 확인

   localhost

- SSH 연결 설정(이미 설정이 되어 있으면 Skip. 경우에 따라 Hadoop 실행의 문제 발생 시 재 수행 필요)

  1. $rm –r /home/[login_id]/.ssh
  2. $ ssh-keygen –t rsa
     (1) "Enter file in which to save the key (/home/[login_id]/.ssh/id_rsa):" 에서 Enter 로 Default 입력.
     (2) "Enter passphrase (empty for no passphrase) :"에서 Enter 로 Default 입력
  3. $cp /home/[login_id]/.ssh/id_rsa.pub /home/[login_id]/.ssh/authorized_keys

- HDFS 포맷

  1. $ bin/hadoop namenode –format

- 데몬 프로그램 실행

  1. **Install 후에는 위의 과정을 생략하고,**

     "$source conf/hadoop-env.sh"만 실행

  2. $bin/start-all.sh
     (1) Start 후 http://localhost:50070 또는 http://localhost:50030 (Hadoop 이 동작하는 서버가 아닌 다른 컴퓨터에서는 "localhost" 대신 해당 IP Address 로 대체)으로 보면 "Live Nodes" 또는 "Nodes"가 "0" 이 아닌 "(node 개수)" 이어야 함. => 만약 "0"일 경우에는 잠시(최대 약 2~3 분) 기다리면

"(node 개수)"로 됨. "(node 개수)"가 되지 않을 경우에는 다시 start-all.sh 실행.

(2) Error message 로 "hadoop-0.20.203.0/bin/../logs 의 소유자 변경: 명령을 허용하지 않음" 등이 발생할 경우

A. "logs" directory 또는 그 안의 로그 파일들의 소유가 root 로 되어 있을 경우 발생하는 문제.

B. $ sudo chown [log-in ID] logs

C. $ sudo chown [log-in ID] logs/*

D. 다시 $bin/start-all.sh 실행.

(3) Error message로 "jobtracker running as process 2030. Stop it first" 등이 발생할 경우

A. $ bin/stop-all.sh

B. $ jps

i. 만약 "???? Child"가 1 개 이상 표시되면 모두 사라질 때까지 jps 명령어로 확인한 후 다음으로 진행.

C. 다시 $bin/start-all.sh 실행.

(4) If completed, the next result will be shown after running "$ jps" command.

???? NameNode

???? JobTracker

???? Jps

???? SecondaryNameNode

???? TaskTracker

???? DataNode

- 포함되어 있는 Example 파일 실행 테스트

1. work directory 내에 hd_example directory 를 만들고 이동.

   Hadoop-0.20.2$ mkdir ../work/hd_example

   Hadoop-0.20.2$ cd ../work/hd_example

2. input 용 사용을 위해 hadoop 의 conf 디렉토리 내 파일들을 HDFS(Hadoop Filesystem)의 Input 에 복사

   work/hd_example$ ../../hadoop-0.20.2/bin/hadoop fs –put ../../hadoop-0.20.2/conf input

3. example 실행 (HDFP 의 input 내의 파일 중 'dfs 로 시작하고 다음 글자가 'a~z'와 '.'으로 시작하는 문장을 찾아냄)

   work/hd_example$ ../../hadoop-20.2/bin/hadoop jar ../../hadoop-0.20.2/hadoop-0.20.2-examples.jar grep input output 'dfs[a-z.]+'

4. 출력

   // work/hd_example 에 output 디렉토리가 이미 있으면 지우고 진행

work/hd_example$ ../../hadoop-0.20.2/bin/hadoop fs –get output .

5. 출력 확인

work/hd_example$ cat output/*

현재까지의 실행 화면 (SSH 연결 설정은 제외. 처음 실행 시 제외하면 안됨) (디렉토리는 다를 수 있음)

```
psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$ bin/hadoop namenode -format
11/09/19 12:08:27 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = psc-System-Product-Name/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.2
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20 -
r 911707; compiled by 'chrisdo' on Fri Feb 19 08:07:34 UTC 2010
************************************************************/
11/09/19 12:08:27 INFO namenode.FSNamesystem:
fsOwner=psc,psc,adm,dialout,cdrom,plugdev,lpadmin,admin,sambashare
11/09/19 12:08:27 INFO namenode.FSNamesystem: supergroup=supergroup
11/09/19 12:08:27 INFO namenode.FSNamesystem: isPermissionEnabled=true
11/09/19 12:08:27 INFO common.Storage: Image file of size 93 saved in 0 seconds.
11/09/19 12:08:27 INFO common.Storage: Storage directory /tmp/hadoop-psc/dfs/name has been
successfully formatted.
11/09/19 12:08:27 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at psc-System-Product-Name/127.0.1.1
************************************************************/

psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$ bin/start-all.sh
starting namenode, logging to /home/psc/devel/hadoop/hadoop-0.20.2/bin/../logs/hadoop-psc-
namenode-psc-System-Product-Name.out
localhost: starting datanode, logging to /home/psc/devel/hadoop/hadoop-
0.20.2/bin/../logs/hadoop-psc-datanode-psc-System-Product-Name.out
localhost: starting secondarynamenode, logging to /home/psc/devel/hadoop/hadoop-
0.20.2/bin/../logs/hadoop-psc-secondarynamenode-psc-System-Product-Name.out
starting jobtracker, logging to /home/psc/devel/hadoop/hadoop-0.20.2/bin/../logs/hadoop-psc-
jobtracker-psc-System-Product-Name.out
localhost: starting tasktracker, logging to /home/psc/devel/hadoop/hadoop-
0.20.2/bin/../logs/hadoop-psc-tasktracker-psc-System-Product-Name.out

psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$ bin/hadoop fs -put conf input

psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$ bin/hadoop jar hadoop-0.20.2-
examples.jar grep input output 'dfs[a-z.]+'
11/09/19 13:37:42 INFO mapred.FileInputFormat: Total input paths to process : 13
11/09/19 13:37:43 INFO mapred.JobClient: Running job: job_201109191215_0002
11/09/19 13:37:44 INFO mapred.JobClient:  map 0% reduce 0%
11/09/19 13:37:53 INFO mapred.JobClient:  map 15% reduce 0%
11/09/19 13:37:56 INFO mapred.JobClient:  map 30% reduce 0%
11/09/19 13:37:59 INFO mapred.JobClient:  map 46% reduce 0%
11/09/19 13:38:02 INFO mapred.JobClient:  map 61% reduce 10%
11/09/19 13:38:05 INFO mapred.JobClient:  map 76% reduce 10%
11/09/19 13:38:08 INFO mapred.JobClient:  map 92% reduce 10%
11/09/19 13:38:11 INFO mapred.JobClient:  map 100% reduce 20%
```

```
11/09/19 13:38:20 INFO mapred.JobClient:  map 100% reduce 100%
11/09/19 13:38:22 INFO mapred.JobClient: Job complete: job_201109191215_0002
11/09/19 13:38:22 INFO mapred.JobClient: Counters: 18
11/09/19 13:38:22 INFO mapred.JobClient:   Job Counters
11/09/19 13:38:22 INFO mapred.JobClient:     Launched reduce tasks=1
11/09/19 13:38:22 INFO mapred.JobClient:     Launched map tasks=13
11/09/19 13:38:22 INFO mapred.JobClient:     Data-local map tasks=13
11/09/19 13:38:22 INFO mapred.JobClient:   FileSystemCounters
11/09/19 13:38:22 INFO mapred.JobClient:     FILE_BYTES_READ=158
11/09/19 13:38:22 INFO mapred.JobClient:     HDFS_BYTES_READ=18375
11/09/19 13:38:22 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=804
11/09/19 13:38:22 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=280
11/09/19 13:38:22 INFO mapred.JobClient:   Map-Reduce Framework
11/09/19 13:38:22 INFO mapred.JobClient:     Reduce input groups=7
11/09/19 13:38:22 INFO mapred.JobClient:     Combine output records=7
11/09/19 13:38:22 INFO mapred.JobClient:     Map input records=556
11/09/19 13:38:22 INFO mapred.JobClient:     Reduce shuffle bytes=224
11/09/19 13:38:22 INFO mapred.JobClient:     Reduce output records=7
11/09/19 13:38:22 INFO mapred.JobClient:     Spilled Records=14
11/09/19 13:38:22 INFO mapred.JobClient:     Map output bytes=193
11/09/19 13:38:22 INFO mapred.JobClient:     Map input bytes=18375
11/09/19 13:38:22 INFO mapred.JobClient:     Combine input records=10
11/09/19 13:38:22 INFO mapred.JobClient:     Map output records=10
11/09/19 13:38:22 INFO mapred.JobClient:     Reduce input records=7
11/09/19 13:38:22 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments.
Applications should implement Tool for the same.
11/09/19 13:38:22 INFO mapred.FileInputFormat: Total input paths to process : 1
11/09/19 13:38:22 INFO mapred.JobClient: Running job: job_201109191215_0003
11/09/19 13:38:23 INFO mapred.JobClient:  map 0% reduce 0%
11/09/19 13:38:31 INFO mapred.JobClient:  map 100% reduce 0%
11/09/19 13:38:43 INFO mapred.JobClient:  map 100% reduce 100%
11/09/19 13:38:45 INFO mapred.JobClient: Job complete: job_201109191215_0003
11/09/19 13:38:45 INFO mapred.JobClient: Counters: 18
11/09/19 13:38:45 INFO mapred.JobClient:   Job Counters
11/09/19 13:38:45 INFO mapred.JobClient:     Launched reduce tasks=1
11/09/19 13:38:45 INFO mapred.JobClient:     Launched map tasks=1
11/09/19 13:38:45 INFO mapred.JobClient:     Data-local map tasks=1
11/09/19 13:38:45 INFO mapred.JobClient:   FileSystemCounters
11/09/19 13:38:45 INFO mapred.JobClient:     FILE_BYTES_READ=158
11/09/19 13:38:45 INFO mapred.JobClient:     HDFS_BYTES_READ=280
11/09/19 13:38:45 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=348
11/09/19 13:38:45 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=96
11/09/19 13:38:45 INFO mapred.JobClient:   Map-Reduce Framework
11/09/19 13:38:45 INFO mapred.JobClient:     Reduce input groups=3
11/09/19 13:38:45 INFO mapred.JobClient:     Combine output records=0
11/09/19 13:38:45 INFO mapred.JobClient:     Map input records=7
11/09/19 13:38:45 INFO mapred.JobClient:     Reduce shuffle bytes=0
11/09/19 13:38:45 INFO mapred.JobClient:     Reduce output records=7
11/09/19 13:38:45 INFO mapred.JobClient:     Spilled Records=14
11/09/19 13:38:45 INFO mapred.JobClient:     Map output bytes=138
11/09/19 13:38:45 INFO mapred.JobClient:     Map input bytes=194
11/09/19 13:38:45 INFO mapred.JobClient:     Combine input records=0
11/09/19 13:38:45 INFO mapred.JobClient:     Map output records=7
11/09/19 13:38:45 INFO mapred.JobClient:     Reduce input records=7

psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$ bin/hadoop fs -get output .

psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$ cat output/*
cat: output/_logs: 디렉터리입니다
3     dfs.class
2     dfs.period
1     dfs.file
1     dfs.replication
1     dfs.servers
1     dfsadmin
1     dfsmetrics.log
```

```
psc@psc-System-Product-Name:~/devel/hadoop/hadoop-0.20.2$
```

⑧ Fully-Distributed Operation

- Hadoop 설정파일에는 두 종류가 있음

1. Read-only default Configuration
    1. src/core/core-default.xml
    2. src/hdfs/hdfs-default.xml
    3. src/mapred/mapred-default.xml
2. Site-specific Configuration
    1. conf/core-site.xml
    2. conf/hdfs-site.xml
    3. conf/mapred-site.xml

- Master Setting

1. conf/core-site.xml

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://psc1:9000</value>
    </property>
</configuration>
```

2. conf/hdfs-site.xml

```
<configuration>
        <property>
```

```
            <name>dfs.replication</name>
            <value>2</value>
        </property>
    </configuration>
```

3. conf/mapred-site.xml

```
<configuration>
    <property>
        <name>mapred.job.tracker</name>
        <value>hdfs://psc1:9001</value>
    </property>
</configuration>
```

4. conf/master 에 Master 컴퓨터 이름을 등록

   psc1

5. conf/slave 에 각 컴퓨터 이름을 등록

   psc1

   psc2

- Slave Setting 은 Master Setting 과 같음
- psc1 와 psc2 를 신뢰관계로 등록(이 작업 이전에 Pseudo Distributed Operation 의 'SSH 연결 설정'과 동일하게 ~/.ssh 디렉토리가 생성되어 있어야 함. 참조 요망.)

  1. psc1 serve 에 있는 ~/.ssh/authorized_keys 라는 키 파일을 psc2 server 의 ~/.ssh/authorized_key 로 Overwrite 시킴. (아래와 같이 scp 명령어로 현 서버에서 다른 서버로 copy 가능)

     $ scp ~/.ssh/authorized_key psc@psc2:~/.ssh

- /etc/hostname 에 각자의 컴퓨터 이름을 써줌

  1. 예를들면 Master 는 psc1, Slave 는 psc2

- /etc/hosts 에 아이피와 컴퓨터 이름을 써주고 쓰지 않는 아이피, 즉 로컬호스트 아이피는 주석처리

  #127.0.0.1 localhost

  #127.0.1.1 XXXXXX

  172.30.1.100 psc1

  172.30.1.101 psc2

- Master 에서 HDFS 포맷

  1. bin/hadoop namenode –format

- Master 데몬 프로그램 실행

  1. bin/start-all.sh

  <주의> Haoop 이 시작된 후에 http://localhost:50030 으로 확인해 보면 Nodes 수가 0 -> 1 -> 2 식으로 몇 십 초의 시간차가 걸릴 수 있음. F5 키를 누르는 식으로 확인을 하면서 잠시 기다릴 필요가 있음.

- 실행시킨 뒤 jps 로 확인

  1. psc1$ jps

     7620 Jps

     7558 TaskTracker

     7378 JobTracker

     7282 SecondaryNameNode

7073 DataNode

6885 NameNode

// 위와 같이 NameNode 와 DataNode 를 포함하여 6 개 이상의 Processors 가 동작 중에 있어야 함.

2. 만약 DataNode 등이 보이지 않을 경우 Hadoop 을 종료시킨 후 HDFS 를 Fomat 시키고(경우에 따라 Format 을 시키지 않아도 됨) 다시 Hadoop 시작.

   $ bin/stop-all.sh

   $ bin/hadoop namenode –format

   $bin/start-all.sh

3. 위와 같이 새로 시작해도 jps 명령어에 대해 필요한 processor 가 생성되지 않으면 Hadoop 종료 후 시스템을 Rebooting 하고 hdfs format 부터 다시 시작.

4. psc2$ jps // slave 에서는 다음 processor

   2819 Jps

   2579 DataNode

   2739 TaskTracker

- 실행 후 Master 에서
  bin/hadoop dfsadmin –refreshNodes
  bin/hadoop dfsadmin –report
  로 노드 확인

- 웹에서 http://psc1:50070 에서 SafeMode is On 이라고 적혀있을 시
  bin/hadoop dfsadmin –safemode leave

v. Sample

① 새로 디렉토리를 만듦.

work$ mkdir WordCount

work$ cd WordCount

② 소스는
http://hadoop.apache.org/mapreduce/docs/r0.21.0/mapred_tutorial.html
여기서 받아쓰기로 함 (웹에서 콘트롤 키를 누른 상태에서 마우스로 긁어서 copy 후 WordCount.java 파일을 새로 만듦)

③ 그 후 컴파일을 하도록 하는데 폴더를 생성해 주는 것이 좋음
work/WordCount$ mkdir word
work/WordCount$ javac –cp ../../hadoop-0.20.2/hadoop-0.20.2-core.jar –d word WordCount.java

(javac 가 없을 경우 본 문서 앞쪽에 설명되어 있듯이 JDK 를 먼저 설치 해야 함)

④ 그리고 jar 파일로 만들어 줌
work/WordCount$ jar cvf Word.jar –C word .

⑤ input 파일들을 만들어 주는데 사용자가 편한 쪽으로 만듬

work/WordCount$ nano input/input1
//내용은 Hello World Bye World
work/WordCount$ nano input/input2
//내용은 Hello Hadoop Goodbye Hadoop

⑥ HDFS 에 input 파일을 등록

work/WordCount$ ../../hadoop-0.20.2/bin/hadoop fs –put input /user/psc/word/input

⑦ 등록됐는지 확인

work/WordCount$ ../../hadoop-0.20.2/bin/hadoop fs –ls /user/psc/word/input

⑧ 실행

work/WordCount$ ../../hadoop-0.20.2/bin/hadoop jar Word.jar org.myorg.WordCount
/user/psc/word/input /user/psc/word/output

⑨ 결과 확인

work/WordCount$ ../../hadoop-0.20.2/bin/hadoop fs –cat /user/psc/word/output/part-r-00000

PSC Platform 구축 : Hadoop에서 GPGPU를 사용하기 위한 작업

1 목적 : Hadoop+GPGPU를 구현하기 위해 Pipe 구조를 통해 C++ 프로그래밍을 하고 컴파일 후 실행하려면 Shared library(GPGPU, libhdfs File System)들을 Call 해야 하는데 Ubuntu Linux 또는 Hadoop의 에러로 보이는 문제 때문에 라이브러리 Link를 해 주어야 함.

2 다음과 같은 명령어를 "반드시" Master/Slave node의 서버에서 실행하여야 함.(Directory 이름은 각각의 서버 시스템에 맞게 변경될 수 있음)

```
$ sudo ln -s /usr/local/cuda/lib64/libnpp.so.4.0.17 /usr/lib/libnpp.so.4

$ sudo ln -s /usr/local/cuda/lib64/libcusparse.so.4.0.17
/usr/lib/libcusparse.so.4

$ sudo ln -s /usr/local/cuda/lib64/libcurand.so.4.0.17
```

```
/usr/lib/libcurand.so.4

$ sudo ln -s /usr/local/cuda/lib64/libcufft.so.4.0.17 /usr/lib/libcufft.so.4

$ sudo ln -s /usr/local/cuda/lib64/libcublas.so.4.0.17
/usr/lib/libcublas.so.4

$ sudo ln -s /usr/local/cuda/lib64/libcudart.so.4.0.17
/usr/lib/libcudart.so.4

$ sudo ln -s /home/psc/pscf_01/hadoop-0.20.2/c++/Linux-amd64-
64/lib/libhdfs.so.0.0.0 /usr/lib/libhdfs.so.0

$ sudo ln -s /usr/local/jdk1.7.0_01/jre/lib/amd64/server/libjvm.so
/usr/lib/libjvm.so
```

3 위 명령 실행 후 /usr/lib에는 다음 link 파일들이 생성됨.

libnpp.so.4, libcusparse.so.4, libcurand.so.4, libcufft.so.4, libcublas.so.4, libcudart.so.4, libhdfs.so.0, libjvm.so

PSC Platform 구축 : Hadoop source 빌드 방법 (참조)

4 목적 : Hadoop은 Binary로 제공되지만 직접 Source를 컴파일 해서 사용하고 싶거나 추가적인 라이브러리 source들을 Build 하고 싶을 때 사용. Binary를 직접 download 받아서 사용할 경우에는 필요 없는 작업임.

5 Ant를 설치

A. sudo apt-get install ant

6 Ant가 잘 설치 되었는지 확인

A. ant –version

B. 만약 "Unable to locate tools.jar. Expected to find it in /usr/local/jre1.6.0_26/lib tools.jar 라는 메시지가 나온다면, tools.jar 파일이 다른곳에 있을 가능성이 크므로 찾아서 옮김

C. find –type f –name tools.jar

D. 필자의 경우는 /usr/lib/jvm/java-6-openjdk/jre/lib 에 위치했음

E. mv /usr/lib/jvm/java-6-openjdk/jre/lib /tools.jar $JAVA_HOME/lib

7 Build

A. 설치되어있는 hadoop 폴더에 build.xml파일을 빌드

B. ant mvn-install

8 Build Success라는 메시지가 나왔다면 build 폴더에 hadoop-0.20.203.1-SNAPSHOT이라는 폴더가 생성되었을 것이고, bin/hadoop 을 실행했을 때 돌아가는 것을 확인 하면 성공

PSC Platform 구축 : Mars(MapReduce + CUDA) 빌드

1 http://www.cse.ust.hk/gpuqp/Mars.html 에서 Mars를 다운
The latest version을 클릭하면 받을 수 있음

2 다운받은 파일을 SDK code samples가 설치되어 있는 폴더/C/에 압축 해제
/home/psc/NVIDIA_GPU_Computing_SDK/C/

3 폴더 중 sample_apps라고 있는데 이것들을 실행시키려면
/home/psc/NVIDIA_GPU_
Computing_SDK/C/bin/linux/release 에 해당프로그램 폴더가 있어야 함

ex) /home/psc/NVIDIA_GPU_Computing_SDK/C/bin/linux/release/MatrixMul

4 해당 폴더에 가서 샘플 프로그램 빌드
/home/psc/NVIDIA_GPU_Computing_SDK/C/sample/MatrixMul 에가서
sudo make 해줌

5 다시 릴리즈에 만든 폴더로 가서 작동 확인
/home/psc/NVIDIA_GPU_Computing_SDK/C/bin/linux/release/MatrixMul/MatrixMul

PSC Platform 구축 : JCUDA(java를 이용한 CUDA) 설치 및 예제

1 JCUDA를 자신의 컴퓨터에 맞게 다운을 받음
http://www.jcuda.de/downloads/downloads.html

2 받은 압축 파일을 해제함
/psc/JCuda-All-0.4.0-beta1-bin-linux-x68_64

3 ~/.bashrc에 등록

A. LD_LIBRARY_PATH에 Path 추가

Export LD_LIBRARY_PATH=[기존 등록된 Path]:/home/psc/pscf_01/JCuda-All-0.4.0-beta1-bin-linux-x86_64

B. 적용 (중복되어서 등록되는 것이 싫으면 새로 booting)

$source ~/.bashrc

$sudo ldconfig

4 예제파일을 받음(예제는 디바이스쿼리, 런타임 드라이버 믹스 샘플)
http://www.jcuda.de/samples/samples.html

5 예제파일 중 JCufftSample.java를 컴파일, 실행하기 위해서는 다음 jar 파일을 JCuda-All-0.4.0-beta1-bin-linux-x68_64 디렉토리에 download

A. JTransforms (http://sites.google.com/site/piotrwendykier/software/jtransforms)

i. jtransforms-2.4.jar 파일을 확보

6 예제파일 중 Jcudpp-x.x.x-xxxxx.jar 파일을 확보하기 위하여 준비할 것

A. cmake 준비

$ sudo apt-get cmake

B. **(2.0 버전은 JCudpp를 생성할 수 없으므로 이 부분은 Skip. 즉, C번부터 진행)**Cudpp 의 소스 파일(cudpp_src_2.0.zip)을 다운로드 후 압축해제 및 Makefile 생성.(다음 사이트에서 소스를 받음 http://code.google.com/p/cudpp/downloads/detail?name=cudpp_src_2.0.zip&can=2&q=)

```
cudpp_src_2.0$ cmake -i .
Would you like to see advanced options? [No]:
Please wait while cmake processes CMakeLists.txt files....



Variable Name: BUILD_APPLICATIONS
Description: If on, builds the sample applications.
Current Value: OFF
New Value (Enter to keep current value): ON

Variable Name: BUILD_SHARED_LIBS
Description: On to build shared libraries, off for static libraries.
Current Value: OFF
New Value (Enter to keep current value): ON

Variable Name: CMAKE_BUILD_TYPE
```

Description: Choose the type of build, options are: None(CMAKE_CXX_FLAGS or CMAKE_C_FLAGS used) Debug Release RelWithDebInfo MinSizeRel.
Current Value:
New Value (Enter to keep current value):

Variable Name: CMAKE_INSTALL_PREFIX
Description: Install path prefix, prepended onto install directories.
Current Value: /usr/local
New Value (Enter to keep current value): /usr/local/cudpp

Variable Name: CUDA_BUILD_CUBIN
Description: Generate and parse .cubin files in Device mode.
Current Value: OFF
New Value (Enter to keep current value):

Variable Name: CUDA_BUILD_EMULATION
Description: Build in Emulation mode
Current Value: OFF
New Value (Enter to keep current value):

Variable Name: CUDA_SDK_ROOT_DIR
Description: Path to a file.
Current Value: CUDA_SDK_ROOT_DIR-NOTFOUND
New Value (Enter to keep current value): /usr/local/cuda

Variable Name: CUDA_TOOLKIT_ROOT_DIR
Description: Toolkit location.
Current Value: /usr/local/cuda
New Value (Enter to keep current value): /usr/local/cuda

Variable Name: CUDA_VERBOSE_BUILD
Description: Print out the commands run while compiling the CUDA source file. With the Makefile generator this defaults to VERBOSE variable specified on the command line, but can be forced on with this option.
Current Value: OFF
New Value (Enter to keep current value): ON

Variable Name: CUDA_VERBOSE_PTXAS
Description: On to enable verbose output from the PTXAS assembler.

```
Current Value: OFF
New Value (Enter to keep current value): ON

Variable Name: EXECUTABLE_OUTPUT_PATH
Description: Directory where all executables will be stored
Current Value: /home/psc/devel/cudpp_src_2.0/bin
New Value (Enter to keep current value):

Variable Name: LIBRARY_OUTPUT_PATH
Description: Directory where all the libraries will be stored
Current Value: /home/psc/devel/cudpp_src_2.0/lib
New Value (Enter to keep current value):

Please wait while cmake processes CMakeLists.txt files....

CMake complete, run make to build project.

cudpp_src_2.0$ make
Linking CXX shared library ../../lib/libcudpp.so
[ 28%] Built target cudpp
Linking CXX shared library ../../lib/libcudpp_hash.so
[ 57%] Built target cudpp_hash
Linking CXX executable ../../bin/cudpp_testrig
[ 86%] Built target cudpp_testrig
Linking CXX executable ../../bin/cudpp_hash_testrig
[ 94%] Built target cudpp_hash_testrig
Linking CXX executable ../../bin/simpleCUDPP
[100%] Built target simpleCUDPP
```

C. Cudpp 의 소스 파일(cudpp_src_1.1.1.zip)을 다운로드 후 압축해제 및 컴파일(다음 사이트에서 소스를 받음. http://code.google.com/p/cudpp/downloads/detail?name=cudpp_src_1.1.1.zip&can=1&q=)

```
// 1.1.1 버전은 Makefile이 이미 있음. 그대로 컴파일 진행.
cudpp_src_1.1.1/cudpp$ make
```

D. JCudpp의 소스파일을 다운로드 후 압축 해제 및 makefile 생성 (http://www.jcuda.de/downloads/downloads.html에서 최근 버전의 "Source code of all libraries"를 클릭해서 다운로드)

```
JCuda-All-0.4.0-beta1-src$ sudo vi CMakeLists.txt
    // 다음을 마지막 부분에 추가 (Directory는 적절히 조정)
    set (CUDA_CUDPP_INCLUDE_DIR "/home/psc/devel/cudpp_src_1.1.1/cudpp/include/")
    set (CUDA_CUDPP_LIBRARY "/home/psc/devel/cudpp_src_1.1.1/lib/")
    add_subdirectory(JCudppJNI)

// cmake로 Makefile 생성
JCuda-All-0.4.0-beta1-src $ cmake -i .
Would you like to see advanced options? [No]:
Please wait while cmake processes CMakeLists.txt files....

Variable Name: CMAKE_BUILD_TYPE
Description: Choose the type of build, options are: None(CMAKE_CXX_FLAGS or CMAKE_C_FLAGS
used) Debug Release RthDebInfo MinSizeRel.
Current Value:
New Value (Enter to keep current value):

Variable Name: CMAKE_INSTALL_PREFIX
Description: Install path prefix, prepended onto install directories.
Current Value: /usr/local
New Value (Enter to keep current value):

Variable Name: CUDA_BUILD_CUBIN
Description: Generate and parse .cubin files in Device mode.
Current Value: OFF
New Value (Enter to keep current value):

Variable Name: CUDA_BUILD_EMULATION
Description: Build in Emulation mode
Current Value: OFF
New Value (Enter to keep current value):
```

```
Variable Name: CUDA_SDK_ROOT_DIR
Description: Path to a file.
Current Value: CUDA_SDK_ROOT_DIR-NOTFOUND
New Value (Enter to keep current value): /usr/local/cuda

Variable Name: CUDA_TOOLKIT_ROOT_DIR
Description: Toolkit location.
Current Value: /usr/local/cuda
New Value (Enter to keep current value): /usr/local/cuda

Variable Name: CUDA_VERBOSE_BUILD
Description: Print out the commands run while compiling the CUDA source file.  With the
Makefile generator this defaults to VERBOSE variable specified on the command line, but can be
forced on with this option.
Current Value: OFF
New Value (Enter to keep current value): ON

Please wait while cmake processes CMakeLists.txt files....

CMake complete, run make to build project.

JCuda-All-0.4.0-beta1-src$ make
[ 10%] Building CXX object CommonJNI/CMakeFiles/CommonJNI.dir/src/JNIUtils.cpp.o
[ 20%] Building CXX object CommonJNI/CMakeFiles/CommonJNI.dir/src/Logger.cpp.o
[ 30%] Building CXX object CommonJNI/CMakeFiles/CommonJNI.dir/src/PointerUtils.cpp.o
/home/psc/devel/JCuda-All-0.4.0-beta1-src/CommonJNI/src/PointerUtils.cpp: In function
'PointerData* initPointerData(JNIEnv*, _jobject*)':
/home/psc/devel/JCuda-All-0.4.0-beta1-src/CommonJNI/src/PointerUtils.cpp:173:23: warning: NULL
used in arithmetic
/home/psc/devel/JCuda-All-0.4.0-beta1-src/CommonJNI/src/PointerUtils.cpp: In function 'bool
isPointerBackedByNativeMemory(JNIEnv*, _jobject*)':
/home/psc/devel/JCuda-All-0.4.0-beta1-src/CommonJNI/src/PointerUtils.cpp:299:23: warning: NULL
used in arithmetic
Linking CXX static library ../lib/libCommonJNI.a
[ 30%] Built target CommonJNI
[ 40%] Building CXX object JCudaDriverJNI/CMakeFiles/JCudaDriver-linux-
x86_64.dir/src/JCudaDriver.cpp.o
Linking CXX shared library ../lib/libJCudaDriver-linux-x86_64.so
[ 40%] Built target JCudaDriver-linux-x86_64
```

```
[ 50%] Building CXX object JCudaRuntimeJNI/CMakeFiles/JCudaRuntime-linux-
x86_64.dir/src/JCudaRuntime.cpp.o
Linking CXX shared library ../lib/libJCudaRuntime-linux-x86_64.so
[ 50%] Built target JCudaRuntime-linux-x86_64
[ 60%] Building CXX object JCublasJNI/CMakeFiles/JCublas-linux-x86_64.dir/src/JCublas.cpp.o
Linking CXX shared library ../lib/libJCublas-linux-x86_64.so
[ 60%] Built target JCublas-linux-x86_64
[ 70%] Building CXX object JCufftJNI/CMakeFiles/JCufft-linux-x86_64.dir/src/JCufft.cpp.o
Linking CXX shared library ../lib/libJCufft-linux-x86_64.so
[ 70%] Built target JCufft-linux-x86_64
[ 80%] Building CXX object JCurandJNI/CMakeFiles/JCurand-linux-x86_64.dir/src/JCurand.cpp.o
Linking CXX shared library ../lib/libJCurand-linux-x86_64.so
[ 80%] Built target JCurand-linux-x86_64
[ 90%] Building CXX object JCusparseJNI/CMakeFiles/JCusparse-linux-
x86_64.dir/src/JCusparse.cpp.o
Linking CXX shared library ../lib/libJCusparse-linux-x86_64.so
[ 90%] Built target JCusparse-linux-x86_64
[100%] Building CXX object JCudppJNI/CMakeFiles/JCudpp-linux-x86_64.dir/src/JCudpp.cpp.o
Linking CXX shared library ../lib/libJCudpp-linux-x86_64.so
[100%] Built target JCudpp-linux-x86_64
```

E. Jcudpp-0.4.0-beta1.jar 생성

```
// JCuda *********
JCuda-All-0.4.0-beta1-src/JCudaJava/src$ mkdir ../classes

// JCuda : Java compilation
JCuda-All-0.4.0-beta1-src/JCudaJava/src$ javac –d ../classes jcuda/*.java
JCuda-All-0.4.0-beta1-src/JCudaJava/src$ javac –d ../classes jcuda/driver/*.java
JCuda-All-0.4.0-beta1-src/JCudaJava/src$ javac –d ../classes jcuda/runtime/*.java

// JCuda : jar file creation
JCuda-All-0.4.0-beta1-src/JCudaJava/classes$ jar cf jcuda-0.4.0-beta1.jar jcuda/*.class
jcuda/driver/*.class jcuda/runtime/*.class

// JCuda : copy a jar and a C-library file to JCuda binary directory
JCuda-All-0.4.0-beta1-src/JCudaJava/classes$ cp jcuda-0.4.0-beta1.jar ../../../JCuda-All-0.4.0-beta1-
bin-linux-x86_64/
```

```
$ cp /home/psc/devel/JCuda-All-0.4.0-beta1-src/lib/libJCudaDriver-linux-x86_64.so
/home/psc/devel/JCuda-All-0.4.0-beta1-bin-linux-x86_64/
$ cp /home/psc/devel/JCuda-All-0.4.0-beta1-src/lib/libJCudaRuntime-linux-x86_64.so
/home/psc/devel/JCuda-All-0.4.0-beta1-bin-linux-x86_64/

// JCudpp *********
JCuda-All-0.4.0-beta1-src/JCudppJava/src$ mkdir ../classes

// JCudpp : Java compilation
JCuda-All-0.4.0-beta1-src/JCudppJava/src$ javac -d ../classes -cp ../../JCudaJava/classes/jcuda-0.4.0-
beta1.jar jcuda/jcudpp/*.java

// JCudpp : jar file creation
JCuda-All-0.4.0-beta1-src/JCudppJava/classes$ jar cf jcudpp-0.4.0-beta1.jar jcuda/jcudpp/*.class

// JCudpp : copy a jar and a C-library file to JCuda binary directory
JCuda-All-0.4.0-beta1-src/JCudppJava/classes$ cp jcudpp-0.4.0-beta1.jar ../../../JCuda-All-0.4.0-
beta1-bin-linux-x86_64/
$ cp /home/psc/devel/JCuda-All-0.4.0-beta1-src/lib/libJCudpp-linux-x86_64.so
/home/psc/devel/JCuda-All-0.4.0-beta1-bin-linux-x86_64/
```

7 예제파일 중 "jcuda.utils"를 위한 JcudaUtils-x.x.x.jar 파일 준비

A. jcudaUtils-0.0.3.jar download 하고 JCuda 디렉토리에 copy.

i. http://www.jcuda.de/utilities/utilities.html

General JCuda utilities
jcudaUtils-0.0.3.jar
	This archive contains some utility classes for JCuda, together with the source code and the API documentation.
	
	One of these classes is the "KernelLauncher" class, which simplifies the setup and launching of kernels using the JCuda Driver API. It allows creating CUBIN files from inlined source code that is given as a String or from existing CUDA source files. CUBIN files can be loaded and the kernels can be called more conveniently due to automatic setup of the kernel arguments.
	
	With the KernelLauncher, calling a kernel function is almost as simple as with the CUDA Runtime API: A kernel call like
	
	`kernel<<<gridDim, blockDim>>>(arg0, arg1);`
	
	may be executed with the KernelLauncher by calling
	
	`kernelLauncher.setup(gridDim, blockDim).call(arg0, arg1);`
	
	The KernelLauncher will automatically set up the configuration parameters and arguments for the kernel call, taking into account the alingment requirements for the given parameters.
	
	Here is an example showing how the KernelLauncher may be used to execute a kernel that was compiled from inlined source code.
	
	Additionally, the archive contains some classes that offer functionalities which are similar to the "CUTIL" functions of the NVIDIA CUDA SDK, such as
	
	• Parsing command line arguments
	• Comparing arrays
	• Simple file I/O
	• Timer functions
	
	These classes are mainly intended for simplifying the process of porting the existing NVIDIA CUDA code samples to Java. They may also be helpful for debugging or creating unit tests.

B. jcudaUtils-0.0.3.jar에 대한 Example Test

* example test // download a example at http://www.jcuda.org/utilities/KernelLauncherSample.java

```
// compilation

JCuda-All-0.4.0-beta1-bin-linux-x86_64$ javac –cp .:jcuda-0.4.0-beta1.jar:jcudaUtiles-0.0.3.jar KernelLauncherSample.java

// run

JCuda-All-0.4.0-beta1-bin-linux-x86_64$ java –cp .:jcuda-0.4.0-beta1.jar:jcudaUtiles-0.0.3.jar KernelLauncherSample

// result : Error caution
```

8 예제파일 중 JOGL 사용의 경우 준비

A. jogl을 자신의 환경에 맞게 다운

http://jogamp.org/deployment/webstart/archive/

jogl-2.0-b23-20110303-linux-amd64.7z (jogl-2.0-b41-20110916-linux-amd64.7z 이후 버전은 실행이 안되므로 권장하지 않음)

B. 다운받은 파일을 적당한 디렉토리에 압축을 풀어 놓음(여기에서는 "~/devel/jogl-2.0-b23-20110303-linux-amd64")

C. .bashrc 에서 LD_LIBRARY_PATH와 CLASSPATH를 추가 또는 수정

cd

nano .bashrc (편의에 따라 vi .bashrc)

기존에 LD_LIBRARY_PATH와 CLASSPATH가 적혀 있다면 추가로 적어 줌.(JOGL의 저장된 위치가 "/home/psc/pscf_01/jogl-2.0-b23-20110303-linux-amd64"일 경우)

```
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${CUDA_HOME}/lib64:/home/psc/pscf_01/jogl-2.0-b23-20110303-linux-amd64/lib"

export CLASSPATH="${CLASSPATH}:/home/psc/pscf_01/jogl-2.0-b23-20110303-linux-amd64/jar"
```

다 수정하고 저장한 뒤 적용

$ source .bashrc

$ sudo ldconfig

D. 예제를 다운로드 (JCudaDriverGLSample3.java, simpleGL_kernel.cu)
http://www.jcuda.de/samples/samples.html

E. 다운받은 파일들을 JCUDA가 있는 폴더로 옮김(여기에서는 "~/devel/JCuda-All-0.4.0-beta1-bin-linux-x86_64")

9 예제 실행 방법

A. cu파일이 필요 없는 경우(디바이스 쿼리)

i. 바로 컴파일 및 실행하면 됨

javac –cp .:jcuda-0.4.0-beta1.jar[:jcublas-0.4.0-beta1.jar] JCudaDeviceQuery.java

java –cp .:jcuda-0.4.0-beta1.jar[:jcublas-0.4.0-beta1.jar] JCudaDeviceQuery

B. cu파일이 필요 한 경우(런타임 디바이스 믹스 샘플)

i. cu파일이 있는경우는 cubin 파일 또는 ptx로 cu파일을 컴파일을 해야 함

① cubin

nvcc –cubin –m64 –arch sm_20 invertVectorElements.cu –o invertVectorElements.cubin

② ptx

nvcc –m64 -ptx invertVectorElements.cu –o invertVectorElements.ptx

ii. 그 후 컴파일 및 실행

javac –cp .:jcuda-0.4.0-beta1.jar[:jcublas-0.4.0-beta1.jar] JCudaRuntimeDriverMixSample.java

java –cp .:jcuda-0.4.0-beta1.jar[:jcublas-0.4.0-beta1.jar] JCudaRuntimeDriverMixSample

C. 폴더내에 있는 jar파일 classpath에 추가하는 방법

Javac –cp $(for i in lib/*.jar ; do echo –n $i: ; done). My.main.class

Example files with compiling & running	Description

JCublasSample.java
$ javac -cp .:jcuda-0.4.0-beta1.jar:jcublas-0.4.0-beta1.jar JCublasSample.java
$ java -cp .:jcuda-0.4.0-beta1.jar:jcublas-0.4.0-beta1.jar JCublasSample
	A JCublas sample, which performs a 'sgemm' operation, once in plain Java and once in JCublas, and verifies the result.

JCufftSample.java
// jtransforms-2.4.jar 미리 download 후(위 설명 참조)
$ javac -cp .:jcuda-0.4.0-beta1.jar:jcufft-0.4.0-beta1.jar:jtransforms-2.4.jar JCufftSample.java
$ java -cp .:jcuda-0.4.0-beta1.jar:jcufft-0.4.0-beta1.jar:jtransforms-2.4.jar JCufftSample
	Shows how to perform an in-place 1D real-to-complex transform using JCufft, and compares the result to a reference solution that is computed with JTransforms.

JCurandSample.java
$ javac -cp .:jcuda-0.4.0-beta1.jar:jcurand-0.4.0-beta1.jar JCurandSample.java
$ java -cp .:jcuda-0.4.0-beta1.jar:jcurand-0.4.0-beta1.jar JCurandSample
	A simple example of how to use JCurand.

JCusparseSample.java
$ javac -cp .:jcuda-0.4.0-beta1.jar:jcusparse-0.4.0-beta1.jar JCusparseSample.java
$ java -cp .:jcuda-0.4.0-beta1.jar:jcusparse-0.4.0-beta1.jar JCusparseSample
	An example showing how to use JCusparse. This example is a direct port of the example from the CUSPARSE documentation.

JCudppSample.java
$ javac -cp .:jcuda-0.4.0-beta1.jar:jcudpp-0.4.0-beta1.jar JCudppSample.java
$ java -cp .:jcuda-0.4.0-beta1.jar:jcudpp-0.4.0-beta1.jar JCudppSample
Creating input data
Performing sort with Java...
Performing sort with JCudpp...
Error caution
	A sample that uses JCudpp to sort an array of integers and verifies the result.

JCudaVectorAdd.java JCudaVectorAddKernel.cu $ javac –cp .:jcuda-0.4.0-beta1.jar JCudaVectorAdd.java ~~$ nvcc –ptx JCudaVetorAddKernel.cu –m64 –o JCudaVectorAddKernel.ptx~~ // automatically executed with the next	The example that is used in the Tutorial: This sample shows how to load and execute a simple

line. $ java –cp .:jcuda-0.4.0-beta1.jar JCudaVectorAdd	vector addtion kernel using the JCuda driver bindings. The CUDA source file is compiled into a PTX file at runtime using the NVCC. The PTX file is loaded as a module, and the kernel function is executed.

JCudaReduction.java
reduction.cu
$ javac –cp .:jcuda-0.4.0-beta1.jar JCudaReduction.java
~~$ nvcc –m64 –ptx reduction.cu –o reduction.~~ptx // automatically executed with the next line.
$ java –cp .:jcuda-0.4.0-beta1.jar JCudaReduction
	The example that performs a reduction, using a kernel that is based on the reduction example from the CUDA SDK.

JCudaDriverSample.java
JCudaSampleKernel.cu
$ javac –cp .:jcuda-0.4.0-beta1.jar JCudaDriverSample.java
$ java –cp .:jcuda-0.4.0-beta1.jar JCudaDriverSample
	This sample is similar to the vector addition example, but also shows how to to pass a 2D array (i.e. an array of pointers) to a kernel function.

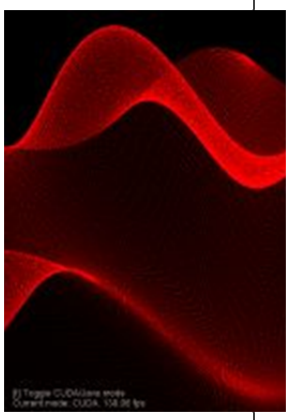
JCudaDeviceQuery.java
$ javac –cp .:jcuda-0.4.0-beta1.jar JCudaDeviceQuery.java
$ java –cp .:jcuda-0.4.0-beta1.jar JCudaDeviceQuery
	A program that queries and prints all attributes of all available devices.

JCudaRuntimeDriverMixSample.java
 	
The kernel that is executed: invertVectorElements.cu
invertVectorElements.cubin (for 32 bit)
 	
$ javac -cp .:jcuda-0.4.0-beta1.jar:jcublas-0.4.0-beta1.jar JCudaRuntimeDriverMixSample.java
$ nvcc -cubin -m64 -arch sm_20 invertVectorElements.cu -o invertVectorElements.cubin
$ java -cp .:jcuda-0.4.0-beta1.jar:jcublas-0.4.0-beta1.jar JCudaRuntimeDriverMixSample
	With CUDA 3.0 it is possible to mix runtime- and driver API calls. This is a simple example that shows how data may be allocated and modified with a mixed sequence of runtime- and driver operations.
	
	Please refer to the CUDA Programming Guide for more information about how driver and runtime calls may be mixed.

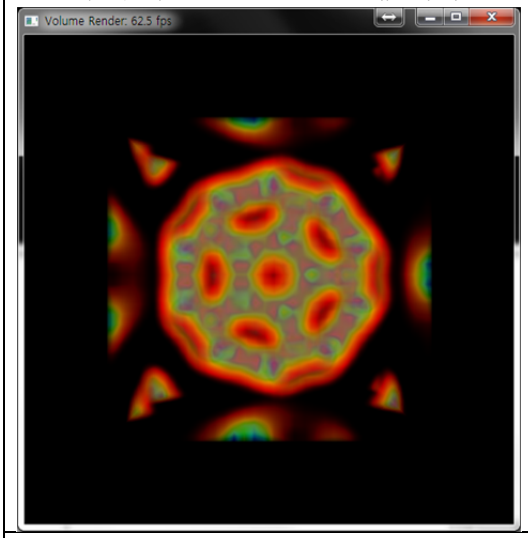
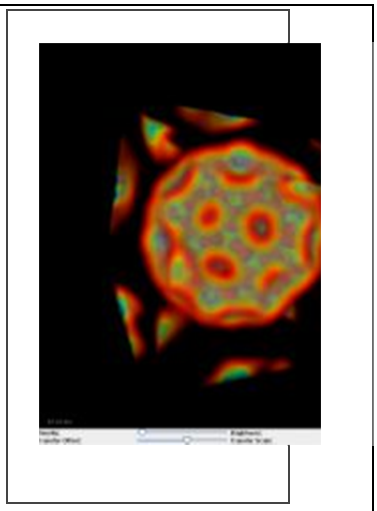
JCublasMatrixInvert.java
$ javac –cp .:jcuda-0.4.0-beta1.jar:jcubras-0.4.0-beta1.jar JCublasMatrixInvert.java
$ java –cp .:jcuda-0.4.0-beta1.jar:jcubras-0.4.0-beta1.jar JCublasMatrixInvert
	This is an example showing how a matrix may be inverted using JCublas.
	
	(based on the code from this forum post)

Matrix Inversion 32bit CUBIN files 64bit CUBIN files * 표 아래에 실행 방법 자세히 설명	This is an example of a matrix inversion. It inverts a matrix by calling several kernels that perform the individual steps of a Gauss elemination. To load and launch the kernels, it uses the KernelLauncher class from the Utilities package.

	The first archive contains the Java source files, and the source files of the CUDA kernels. The other archives contain the precompiled CUBIN files for 32 and 64 bit, respectively. If you have a C compiler installed (for example, Visual Studio or GCC) then you do not need the CUBIN files, since they will be compiled automatically from the source files at program startup. Acknowledgements: The original kernels and the host implementation have been developed by Christoph Wagner (Hochschule Mannheim) in his diploma thesis in the ZAFH-AMSER project, based on a presentation (PDF file) by Christian Heinrich (Fraunhofer SCAI). The source code of the kernels has first been published in this NVIDIA forum thread, and is redistributed here with permission of the original author.
JCudaDriverGLSample3.java simpleGL_kernel.sm_10.cubin (for 32 bit) // JCudaDriverGLSample3.java & simpleGL_kernel.cu with JOGL JCuda-All-0.4.0-beta1-bin-linux-x86_64$ javac -cp .:jcuda-0.4.0-beta1.jar:$(for i in ~/devel/jogl-2.0-b23-20110303-linux-amd64/jar/*.jar; do echo -n $i: ; done). JCudaDriverGLSample3.java ~~JCuda-All-0.4.0-beta1-bin-linux-x86_64$ nvcc -m64 -ptx simpleGL_kernel.cu -o simpleGL_kernel.ptx~~ // automatically executed with the next line JCuda-All-0.4.0-beta1-bin-linux-x86_64$ java -cp .:jcuda-0.4.0-beta1.jar:$(for i in ~/devel/jogl-2.0-b23-20110303-linux-amd64/jar/*.jar; do echo -n $i: ; done). JCudaDriverGLSample3	A sample application demonstrating basic JCuda/JOGL interoperability. It puts a simple, animated sine wave pattern onto a grid of 512x512 points which are stored in a vertex buffer object. The CUDA kernel function is called during each rendering pass to update the vertex positions inside the vertex buffer object, and then the vertex buffer object is rendered using JOGL.

	 In order to compile and run this sample, you will have to download JOGL from JogAmp.org. The CUBIN file for this sample is created from the Simple OpenGL sample from the NVIDIA CUDA samples web site. It is compiled for 32 bit architectures. For 64 bit architectures, you may have to compile your own CUBIN file. See the notes on creating CUBIN files.

JCudaDriverTextureSample.java
volumeRender_kernel.sm_10.cubin (for 32 bit)
 	
The volume data set that is loaded:
Bucky.raw
 	
// JCudaDriverTextureSample.java & volumeRender_kernel.sm_10.cubin & Bucky.raw
JCuda-All-0.4.0-beta1-bin-linux-x86_64$ javac -cp .:jcuda-0.4.0-beta1.jar:$(for i in ~/devel/jogl-2.0-b23-20110303-linux-amd64/jar/*.jar; do echo -n $i: ; done). JCudaDriverTextureSample.java
 	
JCuda-All-0.4.0-beta1-bin-linux-x86_64$ java -cp .:jcuda-0.4.0-beta1.jar:$(for i in ~/devel/jogl-2.0-b23-20110303-linux-amd64/jar/*.jar; do echo -n $i: ; done). JCudaDriverTextureSample
 	
// Result : Error caution
	A sample application demonstrating how to use textures with JCuda. The application loads a RAW volume data file, stores the volume data in a 3D texture, uses a CUDA kernel to render the volume data into a pixel buffer object and displays the pixel buffer object using JOGL.

//********* C++ Windows version에서의 실행 참조(x86 기준)
// 오른쪽의 Volume rendering에서 http://developer.download.nvidia.com/compute/DevZone/C/Projects/volumeRender.zip 을 클릭해서 x86용 volumeRender.zip을 download 후 압축 해제.
// Visual Studio 2008 기준으로 Build 시에 "cutil32D.lib"와 "shrUtils32D.lib"를 필요로 할 경우(CUDA SDK 4.0 RC 이후에 제공되지 않음) 각각 %NVSDKCOMPUTE_ROOT%/c/common에서 cutil_vs2008.sln, $(NVSDKCOMPUTE_ROOT)/shared/shrUtils_vs2008.sln을 build 해서 두 lib 파일들을 확보함. 그리고 "속성"->"링커"->"일반"->"추가 라이브러리 디렉터리"에 ;"$(NVSDKCOMPUTE_ROOT)/c/common/lib/$(PlatformName)";"$(NVSDKCOMPUTE_ROOT)/shared/lib/$(PlatformName)"를 추가
// 실행 시 cudart32_32_16.dll, cutil32D.dll, glew32.dll을 찾으면 "시스템 속성"->"고급"->"환경변수"->"Path"에 ;%CUDA_BIN_PATH%;%NVSDKCOMPUTE_ROOT%/c/common/lib/Win32;%NVSDKCOMPUTE_ROOT%/c/bin/Win32/Release 추가 후 Visual Studio 재 시작
 
	
	In order to compile and run this sample, you will have to download JOGL from JogAmp.org.
	
	The CUBIN file for this sample is created from the Volume rendering sample from the NVIDIA CUDA samples web site.It is compiled for 32 bit architectures. For 64 bit architectures, you may have to compile your own CUBIN file. See the notes on creating CUBIN files.
	
	Additional data sets may be obtained from http://volvis.org/

JCudaDriverTextureTest.java JCudaDriverTextureTestKernels.cu JCudaDriverTextureTestKernels.cubin (for 32bit)	This is a sample- and test class for texture handling. It shows how 1D, 2D and 3D arrays of float and float4 values may be accessed via texture references.

* Matrix Inversion 실행 방법 상세 설명

```
JCuda-All-0.4.0-beta1-bin-linux-x86_64/matrixInvert_0.0.1/src$ javac -d .. -cp ../..:../../jcuda-0.4.0-beta1.jar:../../jcublas-0.4.0-beta1.jar:../../jcudaUtils-0.0.3.jar org/jcuda/samples/matrix/*.java
Note: org/jcuda/samples/matrix/MatrixInvert.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

```
// ***실행 방식 1 또는,
JCuda-All-0.4.0-beta1-bin-linux-x86_64/matrixInvert_0.0.1$ vi manifest.txt // 파일에 Main-Class: org.jcuda.samples.matrix.MatrixInvertSample 추가

JCuda-All-0.4.0-beta1-bin-linux-x86_64/matrixInvert_0.0.1$ jar cvmf manifest.txt matrixInvert_0.0.1.jar org

JCuda-All-0.4.0-beta1-bin-linux-x86_64/matrixInvert_0.0.1$ java -jar matrixInvert_0.0.1.jar

// ***실행 방식 2
JCuda-All-0.4.0-beta1-bin-linux-x86_64/matrixInvert_0.0.1$ java org.jcuda.samples.matrix.MatrixInvertSample

// 결과 : Error caution
```

PSC Platform 구축 : Hadoop C++ 예제 컴파일 및 실행

예제에 들어가기 앞서 현재 hadoop-0.20.203 버전에서는 안된다고 합니다. 그래서 이전 버전인 hadoop-0.20.2를 설치해주시기 바랍니다. 설치방법은 앞과 동일합니다

1 다음 명령을 실행

$ sudo ln -s /usr/local/cuda/lib64/libcudart.so.4.0.17 /usr/lib/libcudart.so.4

// 단 /usr/local/cuda/lib64/libcudart.so.4.0.17에서 4.0.17은 존재하는 최근 버전으로 변경

2 Hadoop이 설치되어 있는 폴더로 이동
cd hadoop/hadoop-0.20.2

3 max_temperature.cpp 파일을 만듬
nano max_temperature.cpp
내용은 아래 참조

```
#include <algorithm>
#include <limits.h>
#include <string>
#include "stdint.h"

#include "hadoop/Pipes.hh"
#include "hadoop/TemplateFactory.hh"
#include "hadoop/StringUtils.hh"


class MaxTemperatureMapper : public HadoopPipes::Mapper {
public :
    MaxTemperatureMapper(HadoopPipes::TaskContext& context) {
    }
    void map(HadoopPipes::MapContext& context) {
            std::string line = context.getInputValue();
            std::string year = line.substr(15, 4);
            std::string airTemperature = line.substr(87, 5);
            std::string q = line.substr(92, 1);
            if(airTemperature !="+9999" && (q=="0" || q== "1" || q=="4" || q=="5"
|| q=="9" )) {
                    context.emit(year, airTemperature);
            }
    }
};

class MapTemperatureReducer : public HadoopPipes::Reducer {
public :
    MapTemperatureReducer(HadoopPipes::TaskContext& context) {
    }
    void reduce(HadoopPipes::ReduceContext& context) {
            int maxValue = INT_MIN;
            while(context.nextValue()){
                    maxValue = std::max(maxValue,
HadoopUtils::toInt(context.getInputValue()));
            }
    context.emit(context.getInputKey(), HadoopUtils::toString(maxValue));
    }
};

int main(int argc, char *argv[]) {
    return HadoopPipes::runTask(HadoopPipes::TemplateFactory<MaxTemperatureMapper,
MapTemperatureReducer>());
}
```

4 Makefile을 만듬

nano Makefile

내용은 아래 참조

```
CC = g++
```

```
PLATFORM = Linux-amd64-64 //자신의 플랫폼에 맞게 설정 32bit는 Linux-i386-32
CPPFLAGS = -m64 –I (하둡이 설치되어 있는 위치)/c++/$(PLATFORM)/include

max_temperature : max_temperature.cpp
$(CC) $(CPPLAGS) $< -Wall –L$(하둡이 설치되어 있는 위치)/c++/$(PLATFORM)/lib –
lhadooppipes –lhadooputils –lpthread –g –O2 –o $@
```

5 컴파일

$ make max_temperature

6 Hadoop이 실행되고 있지 않으면 먼저 하둡을 실행 시킴

Hadoop-0.20.2$ bin/hadoop namenode –format

Hadoop-0.20.2$ source conf/hadoop-env.sh

Hadoop-0.20.2$ bin/start-all.sh

7 예제파일을 넣음(예제는 온도가 있던 sample.txt)

Hadoop-0.20.2$ bin/hadoop dfs –mkdir ncdc

Hadoop-0.20.2$ bin/hadoop dfs –put sample.txt ncdc

8 실행

Hadoop-0.20.2$ bin/hadoop dfs –mkdir bin

Hadoop-0.20.2$ bin/hadoop dfs –put max_temperature bin/max_temperature

Hadoop-0.20.2$ bin/hadoop pipes ₩

–D hadoop.pipes.java.recordreader=true ₩

–D hadoop.pipes.java.recordwriter = true ₩

-input ncdc/sample.txt ₩

–output ncdc-out ₩

–program bin/max_temperature

9 실행값 확인

bin/hadoop dfs –text ncdc-out/part-00000

PSC Platform 구축 : 멀티 유저 계정에서 Hadoop 시스템 사용

1 목적 : 한 Master Server에서 여러 유저들이 각자 계정을 만들고 구축된 PSC Platform 을 공유해서 사용하기 위한 방법

2 새 계정을 만듦

adduser 또는 GUI 환경(좌상의 전원 버튼 클릭 후 '시스템 설정' -> '사용자와 그룹')에서 계정을 생성.

여기서는 원래 계정은 psc

새로 생성된 계정은 pscuser01 이라고 하겠음

계정을 만들었으면 작업하기 편하게 루트권한을 쓸 수 있는 psc로 로그인한 뒤, 터미널 창을 두 개 열어서 하나는 가만히 두고, 또 하나는 "sudo login" 으로 pscuser01을 접속함

3 PSC 터미털 창에서 Hadoop 파일을 /usr/local/에 copy

/usr/local 은 루트권한 밖에 수정을 못하므로 psc에서 파일을 옮기도록 함 파일의 위치는 psc에 있다는 것을 가정함

psc$ sudo cp -r hadoop /usr/local/

4 권한 수정

hadoop에서는 logs 파일을 새로 생성하거나, 불러와서 수정하는 일이 있기 때문에 권한에 대한 수정이 필요. conf파일도 마찬가지로 읽고 실행 하거나 수정이 있을 수도 있기 때문에 모든 권한을 주도록 함

psc$ cd /usr/local/hadoop

psc$ sudo chmod 777 logs

psc$ sudo chmod 777 conf

5 PATH 지정

pscuser01 터미널 창에서 PATH를 등록해줌으로써 /usr/local/hadoop 을 직접 들어가지 않고 실행 할 수 있도록 함

pscuser01$ cd ~

pscuser01$ nano .bashrc

파일 끝에 아래 내용 추가

```
export HADOOP_HOME=/usr/local/hadoop
export PATH=${PATH}:${HADOOP_HOME}/bin
```

저장 후 나가기

pscuser01$ source .bashrc

6 실행

PATH 지정을 했기 때문에 바로 실행 할 수 있음

pscuser01$ start-all.sh

실행을 하면 .ssh 파일이 없어서 계속 진행하냐고 묻는데 답을 yes로 하면 됨

PSC Platform 구축 : 사용상 주의점

1 Hadoop 실행 중 문제가 발생하여 중단, Hadoop 실행이 끝나지 않아서 control+C로 중지 시킬 경우

A. $ jps 명령어를 실행하여 "???? Child" Processor가 있으면 정상적으로 종료된 것이 아니므로 stop-all.sh를 실행한 후 "???? Child"가 없어질 때까지 기다린 후 "hadoop namenode –format" 실행 후 다시 "start-all.sh"를 실행

2 2개 이상의 node(server)에서 Hadoop을 시작할 때 Slave Node가 연결이 안되어서 ssh psc2 등으로 slave node에 연결해 보면,

A. 'ssh: connect to host psc2 port 22: No route to host' 와 같은 메시지가 나오면서 연결이 되지 않는 경우가 있음.

B. 이 때는 psc2의 eth0 등이 연결이 되어 있지 않거나 고정 IP가 동적 IP로 변경이 되어 있지 않은지 확인해 보고(Ubuntu의 경우 상위 메뉴 오른쪽에 있는 '상하 화살표' 모양의 아이콘 클릭) network cable을 서버의 다른 쪽 network port에 연결 해 보면 고정 IP가 살아날 수 있음.