

INTELL

(INTEgrated code generation and paraLLeLization)



By

Raja Hamza Iqbal

M. Umar Farooq

Muneeb Ullah Mashhood

Submitted to the Faculty of Computer Software Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment

for the requirements of a B.E. Degree in Computer Software Engineering

JUNE 2014

CERTIFICATE

Certified that the contents and form of project report entitled “**INTELL**” submitted by

1) NC Hamza Iqbal, 2) NC Umar Farooq, and 3) NC Muneeb Ullah have been found satisfactory for the requirement of the degree.

Supervisor: _____

Asst. Prof. Dr. Hammad Afzal

ABSTRACT

The advancement in GPU technology can be attributed mostly due to heavy investment in gaming industry. This has led to a need to use GPUs for high performance computing, for example GPUs have become an irreplaceable component of the world's most powerful super computers. The use of NVIDIA's GP-GPU by commercial products such as Adobe Photoshop CS6 adds to the validity of this argument. One important issue using GPU technology is that the advancement in architecture is progressing very rapidly which results in a wide variety of different architectures. To utilize these GPUs to their optimal potential, the software must be optimized according to these varying architectures. This can be a tedious and time consuming process.

INTELL is a benchmarking tool that provides users with the ability to test and evaluate new and expensive hardware without having to purchase it. With the rapid development in GPU technology it has become increasingly difficult for people to evaluate the true capabilities of the new systems based on specifications data alone because actual performance depends on a number of different variables. INTELL allows developers and researchers to utilize the capabilities of GPU's through a web interface. It provides users with the ability to test different kernels on different architectures by actually performing the tests. These results are then provided to the user in the form of graphs for evaluation. This allows researchers and developers to reduce the cost of their projects.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of

Any other award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents, without whose unflinching support and cooperation,

a work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

There is no success without the will of ALLAH Almighty. We are grateful to ALLAH, who has given us guidance, strength and enabled us to accomplish this task. Whatever we have achieved, we owe it to Him, in totality. We are also grateful to our parents and family and well-wishers for their admirable support and their critical reviews. We would like to thank our supervisors Dr. Hammad Afzal & Dr. Zaki Murtaza, for their continuous guidance and motivation throughout the course of our project. Without their help we would have not been able to accomplish anything.

Table of Contents

1	Introduction.....	4
1.1	Purpose	4
1.2	Problem domain.....	4
1.2.1	State-of-the-art in GPU Programming Tools.....	4
1.2.2	Manually Tuned Libraries.....	5
1.3	Motivation.....	5
1.4	Goals and objectives	6
1.4.1	Goals	6
1.4.2	Objective	6
1.5	Deliverables.....	6
1.6	System overview	7
2	Introduction.....	9
2.1	Limitations	10
2.1.1	Limitations of Manually-Tuned Libraries	10
3	Purpose.....	12
3.1	Project Scope	12
3.2	Overall Description	12
3.2.1	Product Perspective	12
3.2.2	Product Features.....	13
3.2.3	User Classes and Characteristics.....	13
3.2.4	Operating Environment	14
3.2.5	Design and Implementation Constraints	15
3.2.6	Assumption and Dependencies	15
3.3	System Features.....	15
3.3.1	Command Line Interface.....	15
3.3.2	Web Interface	16
3.3.3	CUDA Kernels	17
3.3.4	Graphs	18
3.3.5	Serial to Parallel Code	18
3.4	External Interface Requirements.....	19

3.4.1	User Interfaces	19
3.4.2	Hardware Interfaces	20
3.4.3	Software Interfaces.....	21
3.4.4	Communications Interfaces	21
3.5	Nonfunctional Requirements.....	22
3.5.1	Performance Requirements.....	22
3.5.2	Safety Requirements.....	22
3.5.3	Security Requirements.....	22
3.5.4	Software Quality Attributes	23
4	Architectural Representation	26
4.1.1	Design Rationale	26
4.1.2	Basic Architecture	27
4.2	Architectural Goals and Constraints	28
4.2.1	Throughput	28
4.2.2	Hardware	28
4.3	Use Case View	29
4.4	Use Case Specification	29
4.4.1	Use Cases	29
4.5	Logical View	38
4.6	Responsibilities	38
4.6.1	PHP-Controller	38
4.6.2	PHP-Model	38
4.6.3	PHP-View.....	39
4.6.4	Java Communication module.....	39
4.6.5	JavaServer Module.....	39
4.6.6	JavaExecution Module	39
4.7	Data Flow	40
5	System Implementation.....	42
5.1.1	Programming Language:	42
5.1.2	Development Tools:.....	42
5.1.3	Database:	42
5.1.4	Operating System:.....	42
5.1.5	Complete System Implementation:	42

6	System Testing.....	45
6.1	Test Case 1	45
6.2	Test Case 2	46
6.3	Test Case 3	47
6.4	Test Case 4	47
6.5	Test Case 5	49
6.6	Test Case 6	50
6.7	Test Case 7	51
7	Future Work Conclusion	54

Table of Figures

Figure 1:Homepage.....	20
Figure 2: GTX-770.....	21
Figure 3: System Use Case	29
Figure 4: Logical view.....	38
Figure 5: Data Flow	40
Figure 6: Register	57
Figure 7: Login.....	58
Figure 8:Submit Parameters	58
Figure 9: View Results.....	59

Table of Tables

Table 1: GPU Engine Specs.....	20
Table 2:GPU Memory Specs.....	21
Table 3: Login UseCase	31
Table 4: Sign Up Use Case	32
Table 5 : GetXML Use case.....	34
Table 6: Display Graph Use Case.....	34
Table 7: Submit Test Data Use Case.....	35
Table 8: View Previous Results Use Case	36
Table 9:View Benchmark Info Use Case.....	37
Table 10: Edit Profile use Case	38
Table 11: Test Case 1	45
Table 12:Test Case 2	46
Table 13:Test Case 3	47
Table 14:Test Case 4	49
Table 15:Test Case 5	49
Table 16:Test Case 6	51
Table 17:Test Case 7	52

CHAPTER 1

INTRODUCTION

1 Introduction

Peak performance of graphics processors (GPUs) is now in the Teraflop range for a single device, and general purpose code on GPUs has demonstrated up to two orders of magnitude performance gains as compared to conventional CPUs. In spite of this enormous potential, it is very difficult to develop GPU applications that make efficient use of all the architectural features and achieve high performance, particularly given the significant architectural changes across GPU generations. For example, consider Nvidia GPUs.

1.1 Purpose

The purpose of this document is to present a detailed description of INTELL. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external input.

1.2 Problem domain

1.2.1 State-of-the-art in GPU Programming Tools

Nvidia has introduced their parallel programming extension called CUDA (Compute United Device Architecture), which has demonstrated high performance for many scientific applications. The CUDA parallel programming model tries to provide programmers with a modest set of language extensions to exploit the parallelism and effectively use the memory hierarchy of the Nvidia GPU devices. As previously stated, multiple aspects of programming Nvidia GPUs must be explicitly managed to maximize the performance gain. Researchers have reported days of effort to program high performing

solutions for well understood problems like matrix-matrix multiplication. In addition to that, the portability of solutions on devices from different vendors or different generation of devices from the same vendor is also a challenging issue.

1.2.2 Manually Tuned Libraries

One of the solutions to the above problem is that the vendors of the GPUs create their own manually tuned libraries and provide access to them to the programmers. This would mean that they have to update these libraries regularly and the developers would still be unable to utilize the hardware until the vendors update these libraries. This particular approach is being used by Nvidia in their CUBLAS library.

1.3 Motivation

Researchers need a lot of funds from their respective institutions in order to carry out the much needed research work, whether it be medical field or simulations. We have already discussed that it costs heavily on budget of organizations to have an instance of latest hardware for each of the researchers. The HPC community came up with motivation for acquiring a single instance of hardware such as GPU and then allowing shared use of it over the web to interested researchers. There was no already existing solution to such a problem so we wanted to contribute in HPC community by building a solution which allows for use of GPU over internet for performing various research tests as well as ensuring that GPU is used for one test at a time.

1.4 Goals and objectives

1.4.1 Goals

Provide a solution to use GPU over internet for benchmarking and research purposes by executing tests and providing performance results.

1.4.2 Objective

Provide a web based interface through which users can register for GPU utilization. Request for tests on GPU and acquire results after test being performed. The registered users are expected to be authenticated with mobile phones to avoid misuse of GPU.

1.5 Deliverables

Deliverable Name	Deliverable Summary Description
Software Requirements Specification(SRS) Document	Complete Description of WHAT system will do, who will use it. Detailed description of functional and non-functional requirements and system features.
Analysis Document	Detailed requirement analysis and analysis models are included.
Design Document	Complete description of How the system will do. Design models are included.
Code	Complete code with the API.
Testing Document	Whole system is tested corresponding to the specifications. System is tested at all levels of Software Development Life Cycle (SDLC).

Complete System	Complete working system.
-----------------	--------------------------

1.6 System overview

INTELL shall provide a web interface that would be used by researchers to optimize and test different benchmarks for different GPU architectures. It shall provide the researchers with test results of the optimization achieved under different configurations. This shall lead to better optimizations for code recipes used in software development.

Chapter 2

Literature Reviews

2 Introduction

Nvidia GPU chips are partitioned into multiple streaming multiprocessors (SMs), each of which have multiple cores. Details of how a computation gets mapped to these SMs affects the performance enormously.

- Explicit data movement from the CPU address space to the GPU memory hierarchy is needed.
- The GPU memory hierarchy must be managed explicitly in software to hide memory latency.
- Improving memory bandwidth must also be managed explicitly using memory operations supported in software, and dedicated hardware mechanisms.
- Maximum performance on a GPU may depend on tuning multiple parameters in the application.

Thus the programmer must explicitly manage available parallelism and the heterogeneous memory hierarchy. Failing to address any of these aspects appropriately may change the mapping of the computation onto the physical resources and thus may severely affect performance. Tools to simplify the programming and performance tuning process have the potential to increase the accessibility of this important technology and improve performance of the resulting code.

2.1 Limitations

2.1.1 Limitations of Manually-Tuned Libraries

Manually tuned libraries provide high performance routines for a set of widely used computations, which speed up parts of the application, improving the overall performance of the applications. There are some limitations with such libraries.

- Many problems require dedicated and exclusive programming effort to devise solutions for GPUs. This approach cannot be extended to cater to the needs of the scientific community to generate general application code. Every different application needs the same amount of time and effort on the part of developers to achieve a high performance solution.
- There are only so many different solution strategies that can be explored if done manually. There is a good chance that for a certain problem, the programmer may only explore a few possible solutions in a potentially large search space of possibilities, as it is too hard to evaluate all of them. Thus, developers tend to settle for local maxima, ignoring the large picture due to inability to look through a large search space.
- Different versions of these libraries are needed for different platforms and may need to be modified with every new generation of a specific architecture. Thus portability is a major concern in developing GPU specific applications.

Chapter 3

Software Requirement Specification

Document

3 Purpose

Intell is a basically a benchmarking tool that allows its users to evaluate and benchmark the performance of expensive GPUs without having to buy the actual hardware by performing tests with user specified parameters.

3.1 Project Scope

The final product enables the users to utilize the GPU on the server for research purposes by allowing them to test generated code of commonly used benchmarks on specified GPUs without obtaining the expensive hardware. This will provide the users the ability to conduct their research through a Web Interface thus providing portability.

The product will also provide a test bed to the users which will allow them to compare the parallelization of code and give them a measure of optimization through parallelization.

3.2 Overall Description

3.2.1 Product Perspective

The advancement in GPU technology can be attributed mostly due to heavy investment in gaming industry. This has led to a need to use GPUs for high performance computing, for example GPUs have become an irreplaceable component of the world's most powerful super computers. The use of NVIDIA's GP-GPU by commercial products such as Adobe Photoshop CS6 adds to the validity of this argument. One important issue using GPU technology is that the advancement in architecture is progressing very rapidly which results in a wide variety of different architectures. To utilize these GPUs to their optimal potential, the software must be optimized according to these varying architectures. This can be a

tedious and time consuming process. A solution to this problem is to use manually tuned libraries provided by the GPU vendors.

INTELL shall provide a web interface that would be used by researchers to optimize and test different benchmarks for different GPU architectures. It shall provide the researchers with test results of the optimization achieved under different configurations. This shall lead to better optimizations for code recipes used in software development.

3.2.2 Product Features

This product shall allow the users of this product to perform the following functions.

- The System should be accessible via web interface.
- The System shall calculate the results (running times of different benchmarks with custom parameters provided by the user) and provide formatted results in the form of graphs and charts.
- The system shall provide the user with downloadable results.
- The system shall allow the users to create user accounts and login with those accounts.
- The system shall generate optimized parallel code recipes for each benchmark custom generated to each GPU on the server.
- The system shall run the test calculations for a particular benchmark with user provided parameters on a specified GPU on the server.

3.2.3 User Classes and Characteristics

Users shall include

- Students
- Evaluators

- Scientists and Researchers

3.2.3.1 Students

Students researching in GPU programming will be able to understand the architecture of GPUs and GPU programming by understanding and working with INTELL. They will also be able to understand the proper application of parallel and serial code.

3.2.3.2 Evaluators

Evaluators are intended to verify and validate the product and its functionalities with respect to the required specifications. This product will initially be evaluated by the project supervisor. Functionalities and required specifications shall be verified and validated by the Panel of Evaluators.

3.2.3.3 Scientists and Researchers

This software is meant for the Scientists and Researches. They will be able to use this product to convert their Serial C Code into an Optimized Parallel Code. The efficiency of Optimized Parallel Code will then be compared with that of Serial C code to help the scientists understand the difference.

3.2.4 Operating Environment

The server machine is expected to be Intel i5 or i7 processor working along with Nvidia GPU. Apache server will offer the web environment for serving user requests.

The client machine however just requires a web browser and an internet connection. It requires client to have a compatible browser preferably Mozilla Firefox 11.0 and any operating system is supported. The project does not require client to have any advanced computing machine.

3.2.5 Design and Implementation Constraints

The customer's organization will be responsible for maintaining the delivered software.

Since INTELL is a web based system it shall require a client server based architecture design.

Also due to limited number of GPUs on the server the number of concurrent users at any given time shall be small.

This software demands a client server architecture due to the requirement of a web based application.

3.2.6 Assumption and Dependencies

Our System shall be assuming and depending upon the following facts:

- CuBLAS is installed and working on Server.
- Windows 7 or above is installed and working on Server.

3.3 System Features

The major features of the software end product shall include following:

3.3.1 Command Line Interface

3.3.1.1 Description and Priority

A stand-alone command line interface shall allow access to a script which is able to communicate with already existing CUBLAS scripts and as well as easily accessible through a web programming language. This command line interface (CLI) utility is most critical since it will not only allow web interfaces to interact with scripts on Server machine but also an abstract layer which allows for adding more features or scripts in later years of research.

Priority: High

3.3.1.2 Stimulus/Response Sequences

Whenever user initiates a process which involves interacting or sending commands to CUDA-CHILL scripts, this CLI utility will be called via an “exec” function in PHP. This utility will output the results in a specified directory and typically requires kernel name/type, problem size and output directory which can be set to default.

3.3.1.3 Functional Requirements

REQ-1: The System should be accessible via web interface may it be a mobile phone or desktop PC

REQ-2: The System must be able to adapt any future changes in kernels (which are nine at present) if required

REQ-3: Output values for displaying graphs

REQ-4: Results available as download to user.

3.3.2 Web Interface

3.3.2.1 Description and Priority

A web interface which is served to each client requesting use of the services. The web interface is expected to be responsive to accommodate the mobile browsers as well as desktop browsers.

Priority: Medium

3.3.2.2 Stimulus/Response Sequences

User will request browser for web interface by entering the URL of website and Apache server will reply back with web page which is then displayed by web browser.

3.3.2.3 Functional Requirements

REQ-5: They system should be accessible via mobile or desktop web browser

REQ-6: Displaying the graphs to user.

REQ-7: Displaying comparisons of graphs with benchmarks.

3.3.3 CUDA Kernels

3.3.3.1 Description and Priority

This category includes the currently present nine CUBLAS kernels which are available to user for testing and comparing results based on different problem sizes. These kernels range from mathematics to biological sciences. There are two kernels for the scope of this project.

Priority: High

3.3.3.2 Stimulus/Response Sequences

Problem or sample size are given as input to these kernels via CLI utility and these kernels after running and giving numerical values give control back to CLI. The numerical values are then used to give graphical representations.

3.3.3.3 Functional Requirements

REQ-8: Serve for the Test Bed purposes.

REQ-9: Graphs based on actual data obtained from CUBLAS

The kernels that shall be available on INTELL are

REQ-12: Matrix Matrix Multiplication

REQ-13: Matrix Vector Multiplication

3.3.4 Graphs

3.3.4.1 Description and Priority

The system must be able to represent all data whether it be obtained from running a kernel with CUDA-CHILL or benchmark data, in a graphical manner so the researcher is able to make a decision based on it.

Priority: Medium

3.3.4.2 Stimulus/Response Sequences

The user requests for graphical output of problem with some sample size. These graphs are then shown to user via web interface as well as available to be downloaded by user so he is able to interpret results by other methods if required (which might involve custom graphs).

3.3.4.3 Functional Requirements

REQ-16: Outputs are presented as graphs

REQ-17: Outputs are available as downloads

REQ-18: Benchmark comparisons and test bed outputs

3.3.5 Serial to Parallel Code

3.3.5.1 Description and Priority

This feature is involved with conversion of serial code into a parallel code. The user will be provided with fix set of nine kernels and their codes but user can modify the sample problem sizes. The parallelized c++ code for a kernel will be shown as output to user.

Priority: Medium

3.3.5.2 Stimulus/Response Sequences

User requests a specific kernel to be run along with problem size and its corresponding parallel code will be generated. This parallel code is accessible via web interface.

3.3.5.3 Functional Requirements

REQ-19: Code Optimization through CUBLAS framework.

REQ-20: Support for heterogeneous platforms.

3.4 External Interface Requirements

3.4.1 User Interfaces

There shall be a tutorial available on the website as part of the web interface that shall guide the user on basic interaction scenarios with the system.

The user interfaces shall be designed while keeping in mind the user goals and other Human Computer Interaction guidelines as specified in ISO 9241-11. The user interfaces shall be evaluated in context to user goals on the basis of effectiveness and efficiency in achieving those goals.

The system shall prompt the user with error messages only on activities that might cause the system to fail. In other situations the system shall only give warning messages because the expected users of the system are researches and the need to be allowed to conduct their research unhindered.

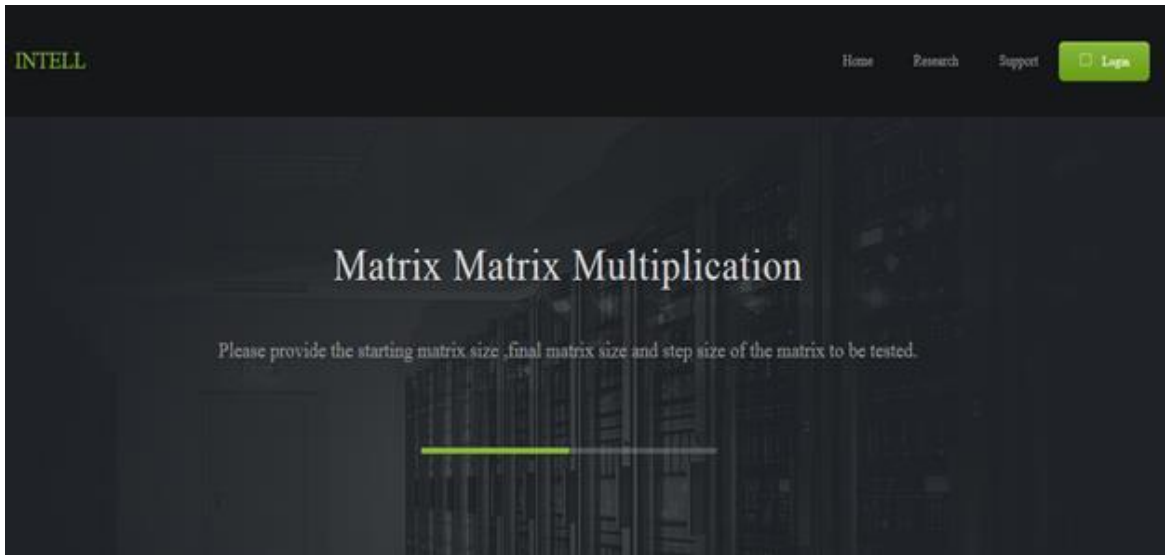


Figure 1:Homepage

3.4.2 Hardware Interfaces

3.4.2.1 Graphical Processing Unit

High-performance GeForce GTX 770 graphics card is designed from the ground up to deliver high-speed, smooth gaming.

GTX 770 GPU ENGINE SPECS:	
• CUDA Cores	1536
• Base Clock (MHz)	1046
• Boost Clock (MHz)	1085
• Texture Fill Rate (billion/sec)	134

Table 1: GPU Engine Specs

GTX 770 MEMORY SPECS:	
Memory Speed	7.0 Gbps
Standard Memory Config	2048 MB
Memory Interface	GDDR5

Memory Interface Width	256-bit
Memory Bandwidth (GB/sec)	224.3

Table 2: GPU Memory Specs



Figure 2: GTX-770

3.4.3 Software Interfaces

3.4.3.1 Client End minimum configuration

Scripting language: JavaScript, PHP

Browser: All browsers from IE 11 and onwards to Firefox and Google Chrome.

3.4.3.2 Server end minimum Configuration

Database: MySQL

Scripting Language: PHP

3.4.4 Communications Interfaces

The client shall access the web interface (website) through the HTTP protocol and interact with this application only through the said web-interface.

3.5 Nonfunctional Requirements

3.5.1 Performance Requirements

The system should ensure that each GPU is tasked with one request at a time so that the results are as reliable as possible.

The response time of the system without including the time for the actual test should be under 1 minute with a 1mbps connection. Since the time for the actual test is highly correlated to the given problem parameters so that time cannot be included in calculation of the systems general response time.

3.5.2 Safety Requirements

Backup power for the hardware must be ensured because sudden power interrupts might cause the costly hardware to malfunction as well as damage the results. The server must be properly cooled because if the hardware gets overheated the results might be affected. The hardware should be properly cooled at the server. The passwords must be kept safe and secret. A user may submit one request at a time for processing since the wait queue is limited to maximum 20 requests. Passwords of the users must be must be kept safe.

3.5.3 Security Requirements

The passwords will be stored in form of hashes using MD5 hashing, and will not be transferred while communication with user. The http protocol will be used for communication as the data transferred is not critical. Once user has given us his personal info it will be protected using password salts. The database contains all of the user logins/passwords and other information that must be protected from hackers who would try to infiltrate the system and steal any personal or user information and try to login under a stolen name. In addition, the modules that users load onto the site and the site's codes

are all protected from outside hackers who may want to negatively alter the code that's present for current games on the site without logging in or registering. Moreover, the modules that are loaded into the system are scanned for viruses, Trojans, or other attachments that can weaken the security of the system.

As users can log onto our system with a password and login, it is important that we guarantee their security. Hence, Secure Socket Layer (SSL) encryption is used as well as a Digital Certificate, which would provide complete security for all parties involved in transactions. Secure Socket Layer is a World Wide Web service that authenticates and encrypts the communication between clients and servers. Thus, all user and platform connections can be protected.

In addition, a firewall is used to enhance the security of the system by checking all request message content and filtering all the information that is accepted from users. Thus, data privacy is guaranteed for all parties.

The security requirements of the system will be implemented in the actual design stages of the system during the integration of components and front end designing. This will guarantee that the security components are integrated correctly into the system and that there will be less vulnerability in the testing stages of the production process.

3.5.4 Software Quality Attributes

3.5.4.1 Reliability

- Each GPU must not process more than one test code recipe at a time in order to ensure maximum reliability of the results.
- The results should be repeatable on different hardware with the same specifications.

3.5.4.2 Availability

- The system should be designed in multi-tier web architecture to increase availability.

3.5.4.3 Maintainability

- The mean time to change should be less than 2 weeks.

3.5.4.4 Robustness

- The system should be able to detect and recover from errors like invalid parameters for benchmarks within a 30 min time limit.

3.5.4.5 Usability

- A user with at least a bachelor's degree in a computer related field should be able to operate the system with less than 10 errors a day after a 2 hour training session.

Chapter 4

Architecture and Design

4 Architectural Representation

4.1.1 Design Rationale

The different design models that were considered are mentioned below.

The primary problem was the dual nature of the INTELL that is, it offers all the functionality of a basic website plus it also needs to actually execute code on hardware. In order to achieve this the problem was to bridge different technologies. The options available for this were

4.1.1.1 Bridge php with C++

For this we had two options

- To call an 'executable request sending module', this approach was rejected because allowing an executable to run on the server has too many security risks such as code injection into a process.
- To write a COM enabled custom dll and use it for IPC between a c++ module and php server. This approach had the factor of unnecessarily increasing the complexity of the system, also it still wouldn't solve the scalability problem because the c++ module would need to manage the execution of job request as well as communication with php server.

4.1.1.2 Bridge php with java

For this the first option was to use a php/java bridge but this meant that the core kernel must be written in php but php is not meant for such complex tasks and the code becomes complicated and hard to maintain. Secondly it would mean that the php controller would be performing far too many tasks which means absolutely no scalability thus this approach was also rejected.

So the best option was to use a multi-tier web application that is divided into 3 major modules. Namely a web interface including a 3-tier web server, a java web service available only to the web server, a java job execution module which provides service to the Java Server.

4.1.2 Basic Architecture

4.1.2.1 Architectural Design

The basic requirement for INTELL is to provide a web interface that allows its users to benchmark different GPUs easily. In order to achieve these goals the design depends upon a multi-tier client server architecture pattern.

INTELL is a multi-tier web application that is divided into 3 major modules.

- A web interface including a 3-tier web server.
- A java web service available only to the web server.
- A java job execution module which provides service to the Java Server.

4.1.2.2 Description

4.1.2.2.1 A web interface including a 3-tier web server

This shall provide a website interface through which the users shall interact with the system. It shall be a 3-tier web server in php utilizing the MVC(Model View Controller) architectural pattern. Thus it shall comprise of 3 subsystems or layers i.e Model, view and controller. Underneath the model shall be an SQL database for storing user information.

4.1.2.2.2 A java web service available only to the web server

This shall be a backend java server that shall be responsible for handling the workload management for the GPUs available to the system. The web server would interact with this server through invoking a small module that shall send a job request to this server. The

java server shall then designate a GPU to carry out the job and send a 'job order' to the java job execution module.

This server allows the system to separate the web and application portion of INTELL and thus allows the web server to operate independently.

4.1.2.2.3 Java job execution module

This shall be a small server module that shall provide the service of executing a job to the java server. It shall be deployed on each machine that has a GPU available for INTELL . Each of the instance of this module shall register itself with the java server on startup. This module is responsible for executing a job using CUBLAS and generating results in xml format.

4.2 Architectural Goals and Constraints

- a separate back end capable of executing code upon hardware
- a web interface as a front end that remains available to the users for access
- displaying results to the user in a meaningful format
- provide user management
- A reliable link between the backend while keeping the dependencies to a minimum.

4.2.1 Throughput

Throughput is to be limited by the number of available GPUs because a every job request needs to executed exclusively in order to ensure the accuracy of the results.

4.2.2 Hardware

The hardware required is nvidia GPUs, we used the Nvidia GTX 770 and Nvidia GTX 660 for testing.

4.3 Use Case View

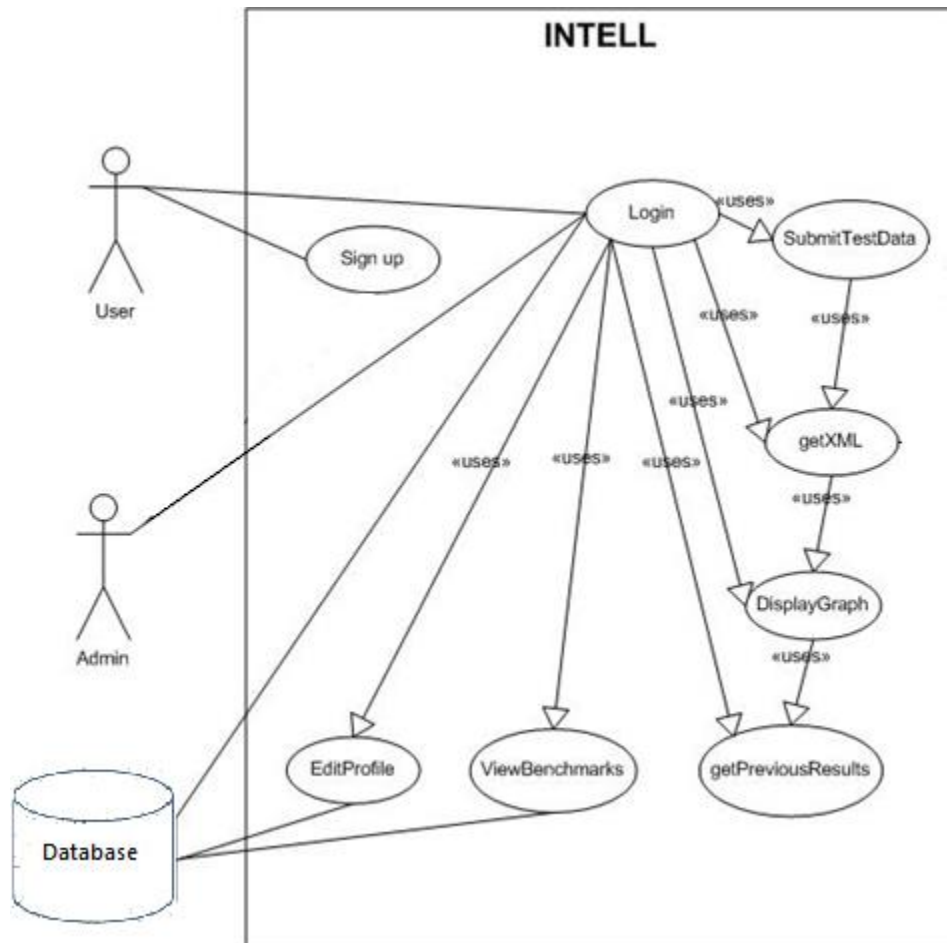


Figure 3: System Use Case

4.4 Use Case Specification

4.4.1 Use Cases

4.4.1.1 Login

Name	LOGIN
Brief Description	This Use Case describes the process by which users log into. It also sets up access permissions for various categories of users.
Actors	1. System User

	2. Administrator
Pre-conditions	The user must not be already logged in.
Normal Flow	<ol style="list-style-type: none"> 1. The user request website by typing in URL of webpage. 2. User clicks login button which redirects him to login page. 3. The user enters his username and password and clicks submit. 4. The system gets username and password combination from database. 5. The system matches username and password combination with that from database. 6. If match is positive 7. Allow user to use other page by storing a session variable with his identity. 8. The system will set access permission. 9. The system will display main homepage to user.
Alternative flow	<ol style="list-style-type: none"> 1. Redirect user to login page. 2. User will re-enter credentials for login scenario or leave web page by closing window.
Post-condition	<ol style="list-style-type: none"> 1. The user id session variable must not be destroyed before logout. 2. The user must be able to use services accessible to him. Including, 3. getXML() 4. DisplayGraph() 5. submitTestData()

	<ol style="list-style-type: none"> 6. getPreviousResults() 7. viewBenchamrksInfo() 8. editProfile()
Assumptions	Internet is accessible.

Table 3: Login UseCase

4.4.1.2 Sign Up

Name	Sign Up
Brief Description	This Use Case describes the process by which users sign up with our system. It also sets up access permissions for various categories of users.
Actors	<ol style="list-style-type: none"> 1. System User 2. Administrator
Pre-conditions	The user must not be logged in already.
Normal Flow	<ol style="list-style-type: none"> 1. The user request website by typing in URL of webpage. 2. User clicks on sign up link which redirects him to sing up page. 3. Sign up page displays a message to user that it is 2 step authentication and they will be required to verify email and mobile number within same session. 4. The user enters his details including name, mobile, email, password, institute and clicks submit button. 5. User is redirected to signup page with error message displayed.

	<ol style="list-style-type: none"> 6. User is sent a verification email. 7. User clicks on link in email. 8. A code is sent to user mobile. 9. User is presented with a form to enter the code sent on their mobile phone. 10. Their profile is activated and they are redirected to login page
Alternative flow	<ol style="list-style-type: none"> 1. The user is redirected to sign up page with message of email already exists. 2. The user is asked to sign up again and not allowed to login. 3. The user is asked to sign up again and not allowed to login
Post-condition	<ol style="list-style-type: none"> 1. The user is allowed to login using his username and password. 2. The user must be able to use services. Including, <ol style="list-style-type: none"> a. getXML() b. DisplayGraph() c. submitTestData() d. getPreviousResults() e. viewBenchamrksInfo() f. editProfile()
Assumptions	User is not already Logged in.

Table 4: Sign Up Use Case

4.4.1.3 GetXML

Name	GetXML
Brief Description	In this use case, user requests for an XML version of results. The results are from tests performed on actual GPU based on inputs from user.
Actors	1. User
Pre-conditions	2. The user is already logged in. 3. The user has submitted test data.
Normal Flow	1. The user submits request for performing tests on GPU with benchmark and its required inputs to our system. 2. The user requests results of test by clicking on link in top menu. 3. The user is returned with result data in XML format which can be used for DisplayGraph()
Alternative flow	1. The user is displayed a message saying you must request a test before requesting results. 2. The user is notified that test is in waiting queue and will be notified of results by email when it's completed.
Post-condition	1. The user is already logged in 2. The user request for results of tests he requested on a GPU. 3. The database is looked up if the results are available or not. 4. If results are available then they are returned to user in XML structure.

	5. Else the user is informed of the about their tests status that it is currently in wait queue for jobs.
--	---

Table 5 : GetXML Use case

4.4.1.4 Display Graph

Name	DisplayGraph
Brief Description	This use case implies displaying the data from getXML in graphical form.
Actors	User
Pre-conditions	<ol style="list-style-type: none"> 1. User is Logged in already. 2. The user has requested test and received the test results.
Normal Flow	<ol style="list-style-type: none"> 1. The user requests to show graphs of data by clicking on link which is along with test results received. 2. The user XML data is used to create graphs with JS. 3. The graphs are displayed to user.
Post-condition	<ol style="list-style-type: none"> 1. User can view the graphs later

Table 6: Display Graph Use Case

4.4.1.5 Submit Test Data

Name	Submit Test Data
Brief Description	This use case implies the user request for executing test on GPU by giving the benchmark requirements along with compulsory inputs for that benchmark.
Actors	User
Pre-conditions	<ol style="list-style-type: none"> 2. User is already logged in. 3. User has knowledge of Benchamrks and range of test data.

Normal Flow	<ol style="list-style-type: none"> 1. User is logged in to the system. 2. The user is presented with hyperlink on main menu to perform a test on GPU. 3. The clicks on link and is presented with a web page. 4. User selects a benchmark from drop down. 5. User selects input range for selected benchmark. 6. The user request is submitted and they are notified with results complete message as soon as these are completely executed and ready in XML form.
Alternative flow	<ol style="list-style-type: none"> 1. The user is requested to submit benchmark and input range again for performing test.
Post-condition	<ol style="list-style-type: none"> 1. The users are able to view results submitted later. 2. The users are able to check on status of tests submitted.

Table 7: Submit Test Data Use Case

4.4.1.6 View Previous Results

Name	View Previous Results
Brief Description	This use case is for viewing results of previously submitted tests.
Actors	User
Pre-conditions	<ol style="list-style-type: none"> 1. User has already logged in. 2. User has submitted test request with benchmark and respective inputs. 3. The test results are available to user.
Normal Flow	<ol style="list-style-type: none"> 1. The user is logged in.

	<ol style="list-style-type: none"> 2. The user request to view previous results by clicking on hyperlink on main web page. 3. The users test results are returned from system in XML form. 4. The XML returned is used to display results in graphs.
Alternative flow	<ol style="list-style-type: none"> 1. The user is asked to submit a test request using SubmitTestData(). 2. The user is notified of current state that results are pending execution on actual hardware and they will be notified of results by email as they are available.
Post-condition	User is not allowed to delete any result

Table 8: View Previous Results Use Case

4.4.1.7 View Benchmark Info

Name	View Benchmark Info
Brief Description	This use case allows for viewing information about available benchmarks.
Actors	User
Pre-conditions	User must be logged in.
Normal Flow	<ol style="list-style-type: none"> 1. The user logs in to the website using their username and password. 2. The user clicks on link to view benchmarks details from main menu. 3. The user is presented with dropdown to seletec a specific benchmark. 4. The user slected a benchmark.

	5. The user is presented with details about the benchmark and input requirements for it.
Alternative flow	1. The user is not allowed to view benchmarks and details. They are redirected to login page.
Post-condition	The user is allowed to SubmitTestData()

Table 9:View Benchmark Info Use Case

4.4.1.8 Edit Profile

Name	Edit Profile
Brief Description	This use case allows for user to update their profiles.
Actors	<ol style="list-style-type: none"> 1. Users 2. Admin 3. Use Cases Used: 4. Login
Pre-conditions	The user must be logged in already.
Normal Flow	<ol style="list-style-type: none"> 1. The user selects to edit profile by clicking on link from main menu. 2. The user is presented with similar input form as of signup to edit their profile information. 3. The user can opt in to update email or not. 4. On update of email, user is required to re verify their email by clicking on link in email and then entering the code send on mobile via input form. 5. The user profile is updated as code is entered. 6. The user is notified of change via older email as well.

Alternative flow	If admin wants to access the official list of doctors and staff then it can change or add any doctors.
Post-condition	<ol style="list-style-type: none"> 1. The user profile is not updated. 2. The user profile is not updated and notified of this particular activity via older email.

Table 10: Edit Profile use Case

4.5 Logical View

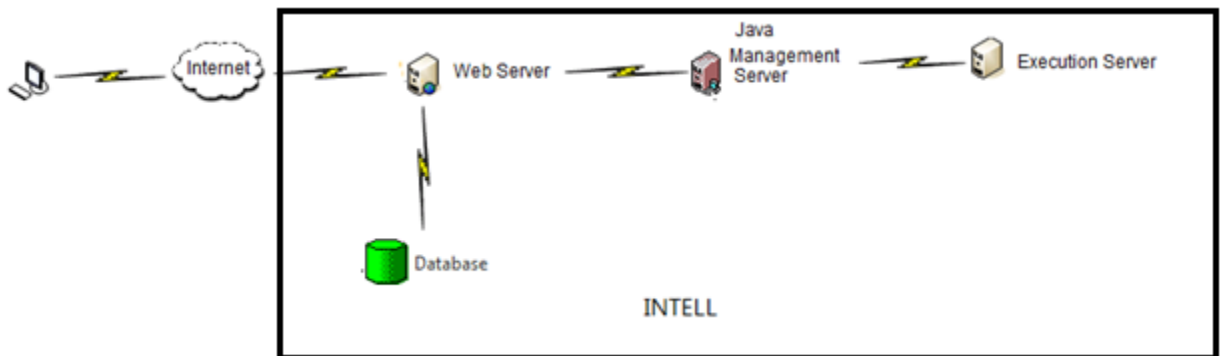


Figure 4: Logical view

4.6 Responsibilities

4.6.1 PHP-Controller

This is the business logic component of the web server that utilizes the MVC architectural pattern. It is responsible for managing the user management constraints as well as forwarding the user's job requests to the JavaServer Module.

4.6.2 PHP-Model

This is the data management component of the web server that utilizes the MVC architectural pattern. It is responsible for providing an interface for the Database to the controller and hiding the lower level detailed queries.

4.6.3 PHP-View

This is the view and display management component of the web server that utilizes the MVC architectural pattern. It comprises of a set of views that the controller can demand as required and then can be displayed to the user.

4.6.4 Java Communication module

This module is to allow the php-controller module to send a job request to the java server module.

4.6.5 JavaServer Module

The java server module separates the request management details from the web server. It allows multiple instances of javaExecution Module to register themselves. Then registered servers are given requests to execute and return the results.

4.6.6 JavaExecution Module

It is responsible for executing the job requests via cuda-chill through invoking a python script. Also it must acquire the details of the on board GPU upon start up and then send a registration request to the java server module.

4.7 Data Flow

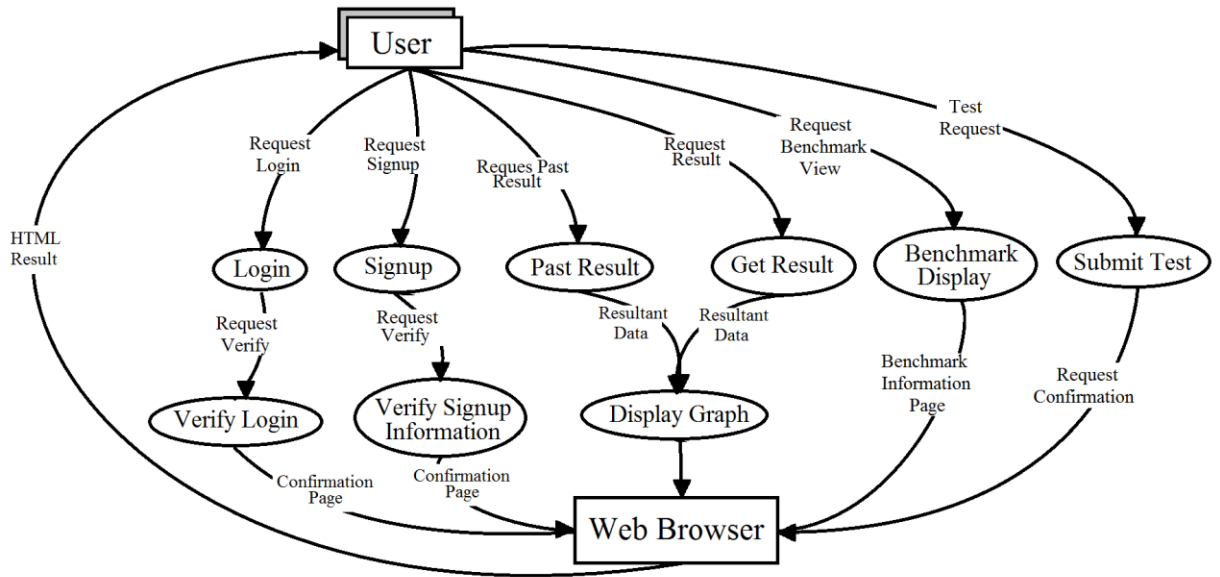


Figure 5: Data Flow

Chapter 5

Implementation

5 System Implementation

5.1.1 Programming Language:

Java is used as programming language to develop the application. Php ,html used for web development and for maintaining the database SQL language is used.

Cuda c is used for utilizing the gpu functionality.

5.1.2 Development Tools:

Netbeans and visual studio with cuda toolkit 6 is used for the development of the application and dreamweaver is used for website development.

5.1.3 Database:

Databases were developed and managed in MS Sql.

5.1.4 Operating System:

On server side, apache is used while the server backend has been tested with windows as an operating environment, while website application is tested on all browsers of Microsoft windows.

5.1.5 Complete System Implementation:

5.1.5.1 JavaServer Module

Acts as a bridge between the GPU client module and the webserver module and controls which job is to be executed on which GPU. This is a critical module since it co-ordinates all the backend functionality of the system. It also handles uploading the result file to the webserver.

5.1.5.2 GPU Client Module

This module is responsible for registering itself with the Java Server. Each instance of this module is responsible for one GPU device. It receives a Job Request from the Java Server and is responsible for handling the execution of the kernel on the GPU and transferring the result file back to the Java Server.

Chapter 6

TESTING AND EVALUATION

6 System Testing

6.1 Test Case 1

Test Case Name	Start the communication of Java Server with Website
Test case no	1
Description	This module will start communication channel with the website. This communication channel is to accept job requests from the website, sent to Java Server.
Testing Technique Used	Black Box testing
Preconditions	The Java Server Executable jar file is running.
Input Values	-Button Click.
Valid Inputs	-Button is Clicked.
Steps	-Press the “Communicate with Website” button.
Expected Output	Java Server will start listening from the Website.
Actual Output	Java Server is waiting for requests from Website.
Status	Pass.

Table 11: Test Case 1

6.2 Test Case 2

Test Case Name	Start the communication of Java Server with GPU Client.
Test case no	2
Description	This module will start a communication channel with the GPU Client. A dedicated port is opened by the Java Server for this channel.
Testing Technique Used	Black Box testing
Preconditions	The Java Server Executable jar file is running.
Input Values	-Button Click.
Valid Inputs	-Button is Clicked.
Steps	-Press the “Communicate with GPU Client” button.
Expected Output	Java Server will start communication channel with GPU Client.
Actual Output	Java Server started communication channel with GPU Client.
Status	Pass.

Table 12:Test Case 2

6.3 Test Case 3

Test Case Name	Subscribe at Java Server.
Test case no	3
Description	GPU Client will subscribe available GPU at Java Server to be used for a job request.
Testing Technique Used	Black Box testing
Preconditions	The GPU Client Executable jar file is running.
Input Values	-Button Click.
Valid Inputs	-Button is Clicked.
Steps	-Press the “Subscribe at Java Server” button.
Expected Output	Java Server will subscribe GPU.
Actual Output	GPU is subscribed.
Status	Pass.

Table 13:Test Case 3

6.4 Test Case 4

Test Case Name	Register on INTELL Website.
Test case no	4

Description	This feature allows a user to register on ITNELL. To use INTELL features a user must register himself on INTELL Website.
Testing Technique Used	Black Box testing
Preconditions	The INTELL Website is running.
Input Values	-Username -Email -Password -Contact Number -Date of Birth -Click Register Button
Valid Inputs	- Alphabets only for Username - Email format - Alphanumeric Password - Numerics for Contact Number - Date format for Date of Birth - Register Button is Clicked.
Steps	- Fill all the fields - Click "Register" button.
Expected Output	User is registered on INTELL and receives a message.
Actual Output	User is registered and message is received..

Status	Pass.
--------	-------

Table 14:Test Case 4

6.5 Test Case 5

Test Case Name	Login on INTELL Website.
Test case no	5
Description	This feature allows a user to login to INTELL. A user can access INTELL features only if he is registered and login.
Testing Technique Used	Black Box testing
Preconditions	The INTELL Website is running and user is registered.
Input Values	-Username -Password
Valid Inputs	- Alphabets only for Username - Alphanumeric Password
Steps	- Fill all the fields - Click “Login” button.
Expected Output	User is Logged into INTELL.
Actual Output	User is Logged into INTELL.
Status	Pass.

Table 15:Test Case 5

6.6 Test Case 6

Test Case Name	Request for a Job
Test case no	6
Description	This feature allows a user to request for a job
Testing Technique Used	Black Box testing
Preconditions	The INTELL Website is running and user is Logged into INTELL.
Input Values	<ul style="list-style-type: none">- Select Benchmark (Kernal Code)- Select GPU.- Give parameters for Benchmark.- Press “Send Request” Button to send Request.
Valid Inputs	<ul style="list-style-type: none">- Benchmark is Selected.- Gpu is Selected.- Parameters must be filled.- Button is Clicked.
Steps	<ul style="list-style-type: none">- Fill all the fields- Click “Send Request” button.
Expected Output	Request is sent to the Java Server and Request is accepted for processing.

Actual Output	Java Server is processing the request after receipt.
Status	Pass.

Table 16: Test Case 6

6.7 Test Case 7

Test Case Name	Request for a Job
Test case no	7
Description	This feature allows a user to request for a job
Testing Technique Used	Black Box testing
Preconditions	The INTELL Website is running and user is Logged into INTELL.
Input Values	<ul style="list-style-type: none"> - Select Benchmark (Kernel Code) - Select GPU. - Give parameters for Benchmark. - Press "Send Request" Button to send Request.
Valid Inputs	<ul style="list-style-type: none"> - Benchmark is Selected. - Gpu is Selected. - Parameters must be filled. - Button is Clicked.
Steps	<ul style="list-style-type: none"> - Fill all the fields

	- Click "Send Request" button.
Expected Output	Request is sent to the Java Server and Request is accepted for processing.
Actual Output	Java Server is processing the request after receipt.
Status	Pass.

Table 17: Test Case 7

Chapter 7

Conclusion and Future Work

7 Future Work Conclusion

Future work can include adding support for new kernels and making them available to the users also support for different libraries can be added apart from CuBLAS with GPU's of other hardware vendors, Since INTELL has been designed as a module based distributed application, all this can be achieved with relatively little overhead.

In short INTELL is based on concept of distributed processing i.e. cloud computing by allowing the user to perform processing from the browser but it is also different from cloud computing in that the users of INTELL are only providing test parameters instead of providing data on which the processing is performed. This means that INTELL is to be used as a Benchmarking and evaluation tool instead of a processing resource which is true in case of a cloud.

APPENDIX A-1

USER MANUAL

1. Reading Instructions

This user manual will describe the way in which INTELL can be used.

It contains the instructions needed to utilize this software.

This system provides a user friendly interface which allows you to efficiently interact with the system.

This Manual should be read in the order given.

2. Installation

The front end of this software is a website. Therefore, no particular Operating System is required.

Internet Explorer 9 or above, Mozilla Firefox or Google Chrome are required for the Website to work properly.

3. How to use the system

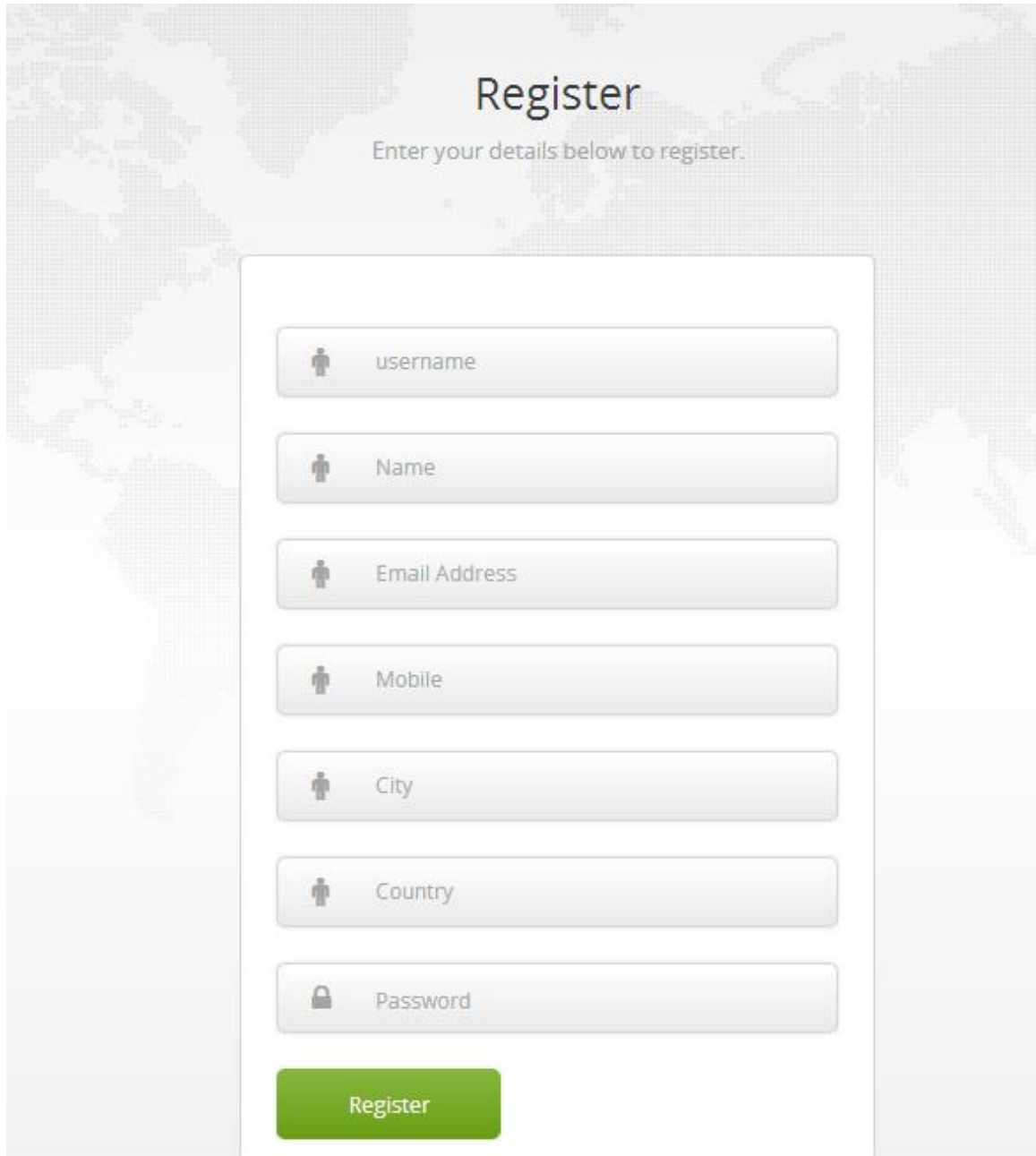
3.1 Hardware

Intell does not require its users to use specific hardware. System with an Internet connection is enough to carry out the job.

3.2 Dashboard

Operation of Dashboard of INTELL comprises of following steps:

1. Register with INTELL



The image shows a registration form titled "Register" with the instruction "Enter your details below to register." The form is set against a background of a world map. It contains seven input fields, each with a person icon on the left and a label on the right: "username", "Name", "Email Address", "Mobile", "City", "Country", and "Password". The "Password" field has a lock icon on the left. Below the input fields is a green "Register" button.

Figure 6: Register

2. Login to your Account

Home GPU Info Submit Test View Graph [login](#)

Log in please

Log in with your username or create an [account](#).

Stay signed in? [Login](#)

About INTELL
INTELL stands for INTEgrated tool to paraLLelize and evaluate serial code. It is part of a project that is intended for the automated conversion of serial code to the optimized parallel code.
INTELL utilizes an underlying polyhedral loop transformation and code generation framework called CUBLAS.

Syndicate
Raja Hamza Iqbal
Muhammad Umar Farooq
Muneeb Ullah Mashhood

Figure 7: Login

3. Select Benchmark, Give proper Parameters and Submit it (i.e. starting matrix size, Ending matrix size, Steps in matrix size for matrix matrix multiplication)

Home GPU Info Submit Test View Graph [Logout](#)

Matrix Matrix Multiplication Kernel - Arguments

- Matrix Matrix Multiplication Kernel - Arguments
- Matrix Vector Multiplication Kernel - Arguments

[Submit Data](#)

About INTELL
INTELL stands for INTEgrated tool to paraLLelize and evaluate serial code. It is part of a project that is intended for the automated conversion of serial code to the optimized parallel code.
INTELL utilizes an underlying polyhedral loop transformation and code generation framework called CUBLAS.

Syndicate
Raja Hamza Iqbal
Muhammad Umar Farooq
Muneeb Ullah Mashhood

Figure 8: Submit Parameters

4. View Results

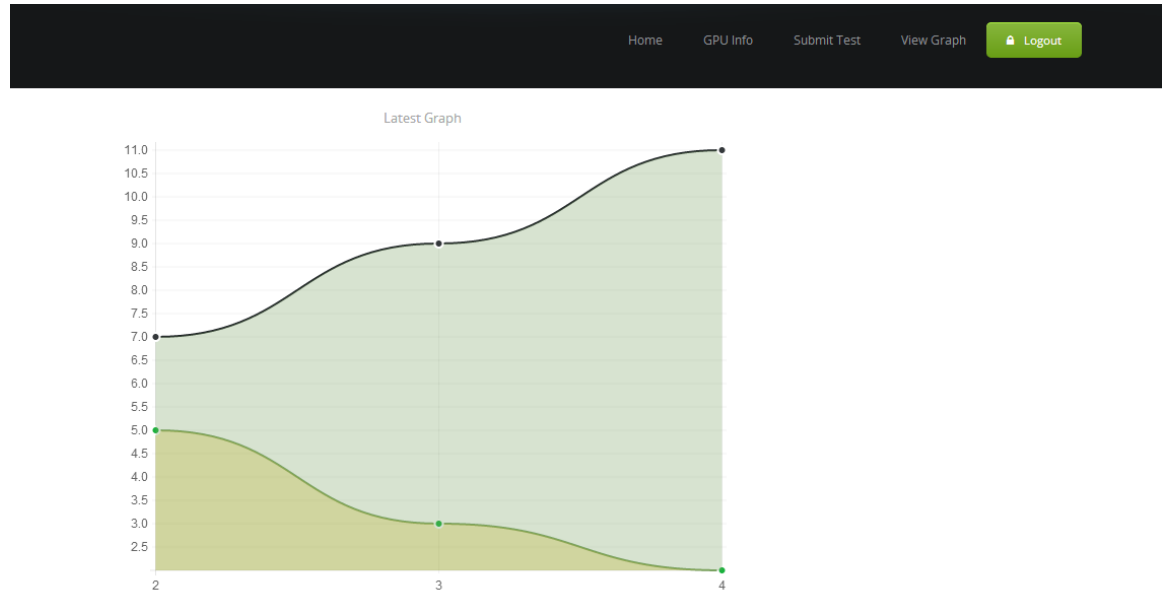


Figure 9: View Results

5. Logout.

These steps are in order and they encapsulate the application logic.