**OPTIMIZING NUMERICAL WEATHER PREDICTION MODEL PERFORMANCE USING MACHINE LEARNING TECHNIQUES**

**ABSTRACT**

Weather forecasting primarily uses numerical weather prediction models that use weather observation data, including temperature and humidity, to predict future weather. The Korea Meteorological Administration (KMA) has adopted the GloSea6 numerical weather prediction model from the UK for weather forecasting. Besides utilizing these models for real-time weather forecasts, supercomputers are essential for running them for research purposes. However, owing to the limited supercomputer resources, many researchers have faced difficulties running the models. To address this issue, the KMA has developed a low-resolution model called Low GloSea6, which can be run on small and medium-sized servers in research institutions, but Low GloSea6 still uses numerous computer resources, especially in the I/O load. As I/O load can cause performance degradation for models with high data I/O, model I/O optimization is essential, but trial-and-error optimization by users is inefficient. Therefore, this study presents a machine learning-based approach to optimize the hardware and software parameters of the Low GloSea6 research environment. The proposed method comprised two steps. First, performance data were collected using profiling tools to obtain hardware platform parameters and Low GloSea6 internal parameters under various settings. Second, a machine learning model was trained using the collected data to determine the optimal hardware platform parameters and Low GloSea6 internal parameters for new research environments. The machine-learning model successfully predicted the optimal parameter combinations in different research environments, exhibiting a high degree of accuracy compared to the actual parameter combinations. In particular, the predicted model execution time based on the parameter combination showed a significant outcome with an error rate of only 16% compared to the actual execution time. Overall, this optimization method holds the potential to improve the performance of other high-performance computing scientific applications.

**CHAPTER-1**

**INTRODUCTION**

Significant advancements in computing performance have facilitated the emergence of numerical weather prediction (NWP) [1] models that use large-scale numerical computations for weather forecasting. Since 1999, the Korea Meteorological Administration (KMA) has been using a global data assimilation and prediction system based on the global spectral model, which is based on the global spectrum model from the Japan Meteorological Agency. The KMA introduced the global NWP model GloSea6 [2] from the UK Met Office in 2022 and has since used it for weather forecasting.

GloSea6 comprises two main models: ATMOS and OCEAN. The ATMOS model comprises atmospheric (UM) and land surface (JULES) models, while the OCEAN model comprises ocean (NEMO) and sea ice (CICE) models. Model execution begins after a preprocessing stage, during which the Earth is divided into grids, and initial and auxiliary data called analysis fields are collected for each grid. Subsequently, the analysis fields are used to prepare input fields for the forecast model, after which numerical model calculation begins.

Owing to its high demand for computing resources,the KMA provides a low-resolution version of GloSea6 called Low GloSea6 for researchers who lack access to supercomputers. However, even Low GloSea6 requires significant computing resources, and as the model has a high data input/output (I/O) nature, I/O optimization is essential. Notably, general users, who are atmospheric science researchers and not computer scientists, may find conducting performance optimization through trial-and-error inefficient. This paper presents a machine learning-based approach to optimize the hardware and software parameters of the Low GloSea6 research environment.

This study proposes a new cross-inference optimizationmethod for the NWP model Low GloSea6 using machine learning and benchmark tools. Specifically, the following are detailed:

• We defined the entire workflow for performancecross-validation and validated it through experiments.

• Necessary data for cross-inference were categorized into two types: execution hardware platform parameters and internal software parameters of Low GloSea6,and important parameters among them were extracted through model/data validation.

• We used Darshan to collect detailed data on I/O characteristics and verified the final results using runtime data to perform I/O performance cross-validation.

• This study demonstrates the applicability of various machine-learning techniques to explain the complex interactions between the execution hardware platform parameters and the Low GloSea6 internal software parameters, thereby making it feasible to cross-infer performance on a new execution hardware platform.

• The proposed method has been generalized throughout the workflow, demonstrating that it is a general methodology that is not limited to Low GloSea6, which is the subject of this paper.

This paper is structured as follows: Section II describes related research, while Section III provides a detaileddescription of GloSea6, a numerical model for weather prediction, and the profiling tool used for performance data collection. Section IV explains the hardware/software optimization methodology in the research environment, including the dataset and model used. In Section V, the experiments conducted using the optimization methodology after the model and data verification are described and analyzed. Section VI presents the conclusion and future plans.

**CHAPTER-2**

**LITERATURE SURVEY**

**‘‘A large ensemble seasonal forecasting system: GloSea6,’’**

**P. Davis, C. Ruth, A. A. Scaife, and J. Kettleborough,**

The Global Seasonal forecasting system (GloSea) is the Met Office's monthly to seasonal ensemble prediction system. The current version, GloSea5, which has been operational since 2014, has undergone a significant scientific upgrade. This includes an update to the coupled model, which contains a treatment of convective entrainment, improvements to sea-ice physics and a scheme for iceberg advection. In addition, we include a more realistic treatment of land-surface initialization, using a forced land model to initialize soil moisture instead of a climatology. We also use CMIP6 forcing data to replace a) a constant Solar forcing with variable fluxes that represent the Solar Cycle and b) zonal mean climatology of Ozone concentrations with three-dimensional, time-varying fields. We describe the scientific performance of GloSea6 using re-forecasts covering the period 1993-2016, with Winter and Summer start dates and 100 members per season. Results are compared with GloSea5 as a baseline. Changes in skill are generally positive or neutral, and we also investigate the impact of the larger ensemble on the skill. GloSea6 better reproduces both the near-surface temperature and Z500 anomalies during the 2003 European heatwave, although there is no significant change for the 2010 Russian event. Additionally, we see reduced sea-surface temperature (SST) biases for both El Nino and La Nina during December-January-February (DJF). Furthermore, the SST ensemble mean standard deviation for El Nino during DJF is closer to the observed value. The NAO skill is similar to that of GloSea5. We note, however, that the September sea-ice extent bias is larger than found in GloSea5. This could be a consequence of the treatment of melt-ponds and sea-ice drag.

**‘‘Tuning HDF5 for Lustre file systems,’’**

**M. Howison, Q. Koziol, D. Knaak, J. Mainzer, and J. Shalf,**

HDF5 is a cross-platform parallel I/O library that is used by a wide variety of HPC applications for the flexibility of its hierarchical object-database representation of scientific data. We describe our recent work to optimize the performance of the HDF5 and MPI-IO libraries for the Lustre parallel file system. We selected three different HPC applications to represent the diverse range of I/O requirements, and measured their performance on three different systems to demonstrate the robustness of our optimizations across different file system configurations and to validate our optimization strategy. We demonstrate that the combined optimizations improve HDF5 parallel I/O performance by up to 33 times in some cases – running close to the achievable peak performance of the underlying file system – and demonstrate scalable performance up to 40,960-way concurrency.

**‘‘Taming parallel I/O complexity with auto-tuning,’’**

**B. Behzad et al.,**

We present an auto-tuning system for optimizing I/O performance of HDF5 applications and demonstrate its value across platforms, applications, and at scale. The system uses a genetic algorithm to search a large space of tunable parameters and to identify effective settings at all layers of the parallel I/O stack. The parameter settings are applied transparently by the auto-tuning system via dynamically intercepted HDF5 calls. To validate our auto-tuning system, we applied it to three I/O benchmarks (VPIC, VORPAL, and GCRM) that replicate the I/O activity of their respective applications. We tested the system with different weak-scaling configurations (128, 2048, and 4096 CPU cores) that generate 30 GB to 1 TB of data, and executed these configurations on diverse HPC platforms (Cray XE6, IBM BG/P, and Dell Cluster). In all cases, the auto-tuning framework identified tunable parameters that substantially improved write performance over default system settings. We consistently demonstrate I/O write speedups between 2× and 100× for test configurations.

**‘‘Optimizing I/O performance of HPC applications with autotuning,’’**

**B. Behzad, S. Byna, Prabhat, and M. Snir,**

To improve parallel I/O performance, it is imperative to optimize the adjustable parameters across the different layers of the I/O software stack. Finding an optimal configuration for different scenarios is hampered by the complex interaction dynamics between these parameters and the large parameter space. Previous research efforts have focused on tuning these parameters using independent algorithms; however, these approaches exhibit certain shortcomings such as unstable performance results and delayed convergence rates.This paper introduces OPRAEL, an auto-tuning approach on parallel I/O tasks by ensembles and performance modeling using regression analysis. To test its effectiveness, we applied this approach on the Tianhe-II supercomputer using one well-known I/O benchmark(IOR) and two I/O kernels(S3D-I/O, BT-I/O). Leveraging our experience in predictive modeling, we optimized the tuning of the I/O stack parameters. Our experimental results show a remarkable 10.2X improvement in write performance speedup for the optimization task with BT-I/O and a 500x500x500 input. We also compared the potential of using a single search algorithm versus using reinforcement learning search in the I/O parameter auto-optimization task. Our results show that OPRAEL outperforms the traditional approach, resulting in a maximum 8.4X improvement in write performance for the 128-process IOR optimization.

**‘‘Auto-tuning of IO accelerators using black-box optimization,’’**

**S. Robert, S. Zertal, and G. Goret,**

High Performance Computing (HPC) applications' performance and behavior rely on software and hardWare environments Which are often highly configurable. Finding their optimal parametrization is a very complex task. The size of the parametric space and the non-linear relationship between the parameters and the delivered performance make hand-tuning, theoretical modeling or exhaustive sampling unsuitable for most cases. In this paper, We propose an auto-tuning loop that uses black-box optimization to Find the optimal parametrization of IO accelerators for a given HPC application in a limited number of iterations, Without making any assumption on the performance function. After a literature review of the selected methods for tuning the accelerators, We describe their implementation and experimentation in our HPC context using two IO accelerators developed by Atos. We also define several metrics to evaluate the quality of our optimization, as our criteria of success go further than finding the optimal parameters. The obtained results show that this framework successfully improves the execution time of two applications used conjointly With a pure software accelerator and a mixed hardWare-software one. We indeed observe possible time gains of respectively 38% and 20% for each accelerator compared to launching the same application accelerated With the default parameters.

**‘‘Improving the I/O performance of applications with predictive modeling based auto-tuning,’’**

**A. Bağbaba, X. Wang, C. Niethammer, and J. Gracia,**

Parallel I/O is an essential part of scientific applications running on high performance computing systems. Typically, parallel I/O stacks offer many parameters that need to be tuned to achieve the best possible I/O performance. Unfortunately, there is no default best configuration of parameters; in practice, these differ not only between systems but often also from one application use-case to the other. However, scientific users often do not have the time nor the experience to explore the parameter space sensibly and choose the proper configuration for each application use-case. This paper proposes an auto-tuning approach based on I/O monitoring and predictive modeling, which can find a good set of I/O parameter values on a given system and application use-case. We demonstrate the feasibility to auto-tune parameters related to the Lustre file system and the MPI-IO ROMIO library transparently to the user. In particular, the model predicts for a given I/O pattern the best configuration from a history of I/O usages. We have validated the model with two I/O benchmarks, namely IOR and MPI-Tile-IO, and a real Molecular Dynamics code, namely ls1 Mardyn. We achieve an increase of I/O bandwidth by a factor of up to 18 over the default parameters for collective I/O in the IOR and a factor of up to 5 for the non-contiguous I/O in the MPI-Tile-IO. Finally, we obtain an improvement of check-point writing time over the default parameters of up to 32 in ls1 Mardyn.

**‘‘24/7 characterization of petascale I/O workloads,’’**

**P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley,**

Developing and tuning computational science applications to run on extreme scale systems are increasingly complicated processes. Challenges such as managing memory access and tuning message-passing behavior are made easier by tools designed specifically to aid in these processes. Tools that can help users better understand the behavior of their application with respect to I/O have not yet reached the level of utility necessary to play a central role in application development and tuning. This deficiency in the tool set means that we have a poor understanding of how specific applications interact with storage. Worse, the community has little knowledge of what sorts of access patterns are common in today's applications, leading to confusion in the storage research community as to the pressing needs of the computational science community. This paper describes the Darshan I/O characterization tool. Darshan is designed to capture an accurate picture of application I/O behavior, including properties such as patterns of access within files, with the minimum possible overhead. This characterization can shed important light on the I/O behavior of applications at extreme scale. Darshan also can enable researchers to gain greater insight into the overall patterns of access exhibited by such applications, helping the storage community to understand how to best serve current computational science applications and better predict the needs of future applications. In this work we demonstrate Darshan's ability to characterize the I/O behavior of four scientific applications and show that it induces negligible overhead for I/O intensive jobs with as many as 65,536 processes.

**‘‘A case study in application I/O on Linux clusters,’’**

**R. Ross, D. Nurmi, A. Cheng, and M. Zingale,**

A critical but often ignored component of system performance is the I/O system. Today’s applications demand a great deal from underlying storage systems and software, and both high-performance distributed storage and high level interfaces have been developed to fill these needs. In this paper we discuss the I/O performance of a parallel scientific application on a Linux cluster, the FLASH astrophysics code. This application relies on three I/O software components to provide high-performance parallel I/O on Linux clusters: the Parallel Virtual File System, the ROMIO MPI-IO implementation, and the Hierarchical Data Format library. Through instrumentation of both the application and underlying system software code we discover the location of major software bottlenecks. We work around the most inhibiting of these bottlenecks, showing substantial performance improvement. We point out similarities between the inefficiencies found here and those found in message passing systems, indicating that research in the message passing field could be leveraged to solve similar problems in high-level I/O interfaces.

**CHAPTER-3**

**SYSTEM ANALYSIS AND DESIGN**

**EXISTING SYSTEM**

Optimization studies for applications running in real-world or research environments have been conducted in various fields. One such approach is the modification of I/O library codes to achieve I/O optimization of applications. Howison et al. [3] demonstrated performance improvements for high-performance computing (HPC) applications through code modifications and optimizations of HDF5 and MPI-IO libraries, considering the file system characteristics. Another research method is to achieve I/O optimization by deriving optimal file systems and I/O library parameters. In addition, Behzad et al. [4], [5] used a genetic algorithm to optimize the I/O performance of an application. They created a set of parameters by exploring the file system and I/O library parameter space, measured the I/O performance of the benchmark tool using the parameter set, and iteratively optimized the parameter set based on the measurements until the best I/O performance was achieved.

Robert et al. [6] optimized an I/O accelerator using black-box optimization techniques that find input parameters with maximum and minimum performance metrics without considering internal mechanisms. They optimized three input parameters (I/O throughput, I/O latency, and I/O memory usage) of the Atos Flash Accelerator, an I/O accelerator that accelerates I/O operations of various HPC applications using NAND flash memory technology, and used basic metrics, such as I/O operation processing time, as performance indicators.

Finally, they validated that the I/O accelerator performance can be improved by applying black-box optimization. Bağbaba et al. [7] implemented an automated tuning solution for the optimal parameters of Lustre parallel file system and MPI-IO ROMIO library, a high-performance implementation of MPI-IO, using I/O monitoring and performance prediction. The solution employed a random forest-based machinelearning algorithm and was validated using two benchmarking tools (IOR-IO and MPI-Tile-IO) and a molecular dynamics model (ls1 Mardyn.’’). Our research differs from previous studies in two ways.

First, our study enables easy optimization, even without prior I/O optimization knowledge. While Howison et al. [3] achieved I/O performance optimization by modifying the I/O library code, this approach requires a developer’s expertise and is not easily accessible to general users. In contrast, our research focuses on machine learning-based performance optimization that is easily modifiable and accessible by considering the hardware and software parameters of the research environment. Second, our study simultaneouslyconsiders hardware platform parameters and internal software parameters.

Behzad et al. [4], [5] optimized I/O using adjustable parameters in the parallel I/O stack, specifically related to file systems, HDF5, and MPI-IO libraries. However, the research did not consider benchmark tool parameter optimization. Robert et al. [6] used the parameters of I/O throughput, I/O latency, and I/O memory usage of the Atos Flash Accelerator I/O accelerator for its optimization. These parameters are internal software parameters mentioned in this paper, and hardware platform parameters were not considered.

Our research has broad applicability. Bağbaba et al. [7] study focused on the MPI-IO ROMIO library and Lustre parallel file system in a single research environment, which limits its generalizability. In contrast, we collected data in two different research environments and conducted validation on different hardware platform environments using Low GloSea6. In addition, we used MPICH, an MPIIO implementation with high accessibility that can be applied regardless of the specific implementation version of MPICH. To verify this, we conducted experiments in research environments using different versions of MPICH.

**Disadvantages**

* The gradient boosting model not combines weak models into strong models using weights and adds a sequential characteristic to the traditional bagging method.
* The MLR model lacks hyperparameters, as it is a characteristic of linear regression estimation. The random forest model has a hyperparameter called ‘‘mtry,’’ which determines the number of features used for each tree.

**PROPOSED SYSTEM**

The system proposes a new cross-inference optimization method for the NWP model Low GloSea6 using machine learning and benchmark tools. Specifically, the following are detailed:

• We defined the entire workflow for performance cross-validation and validated it through experiments. • Necessary data for cross-inference were categorized into two types: execution hardware platform parameters and internal software parameters of Low GloSea6, and important parameters among them were extracted through model/data validation.

• We used Darshan to collect detailed data on I/O characteristics and verified the final results using runtime data to perform I/O performance cross-validation.

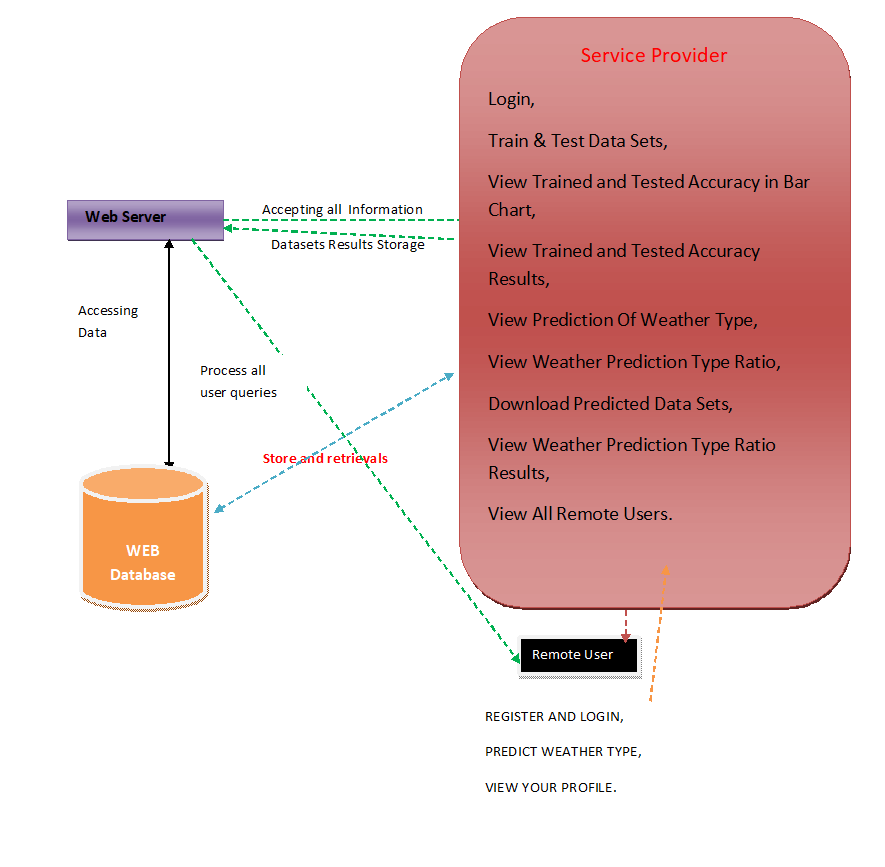
• This study demonstrates the applicability of various machine-learning techniques to explain the complex interactions between the execution hardware platform parameters and the Low GloSea6 internal software parameters, thereby making it feasible to cross-infer performance on a new execution hardware platform.

• The proposed method has been generalized throughout the workflow, demonstrating that it is a general methodology that is not limited to Low GloSea6, which is the subject of this paper.

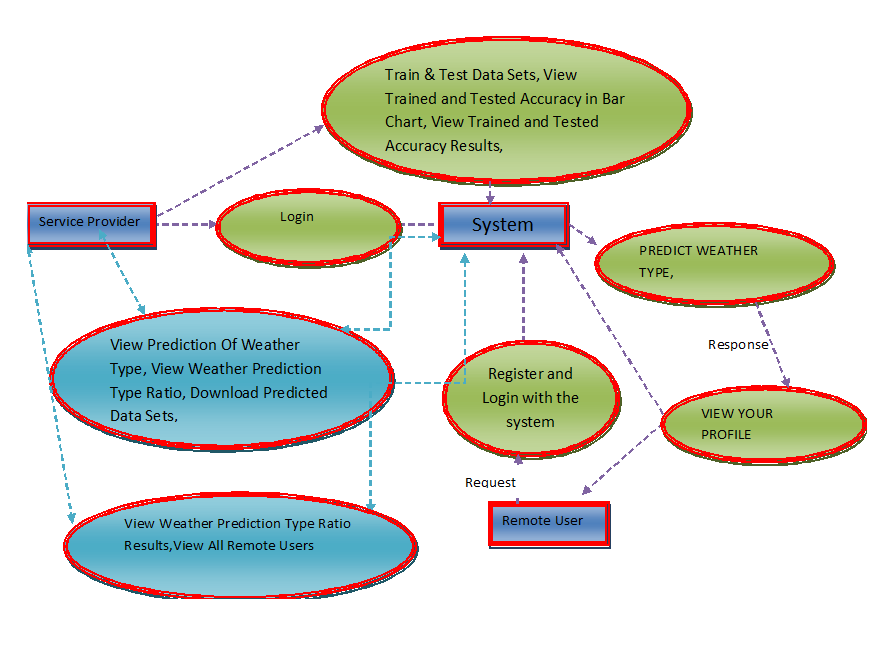
**Advantages**

* We propose a new cross-inference optimization method that considers both the hardware platform and internal parameters of the application program using machine learning and benchmark tools to improve the performance of Low GloSea6.
* MLR is a method for predicting the dependent variable through independent variables, assuming a linear relationship between them. Random forest and gradient boosting are ensemble models based on decision trees. The ensemble is a technique used to compensate for the instability of decision trees by combining weak models to create a strong model.

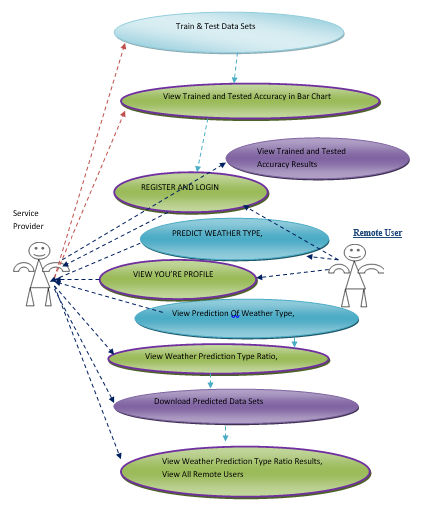
**SYSTEM ARCHITECTURE**



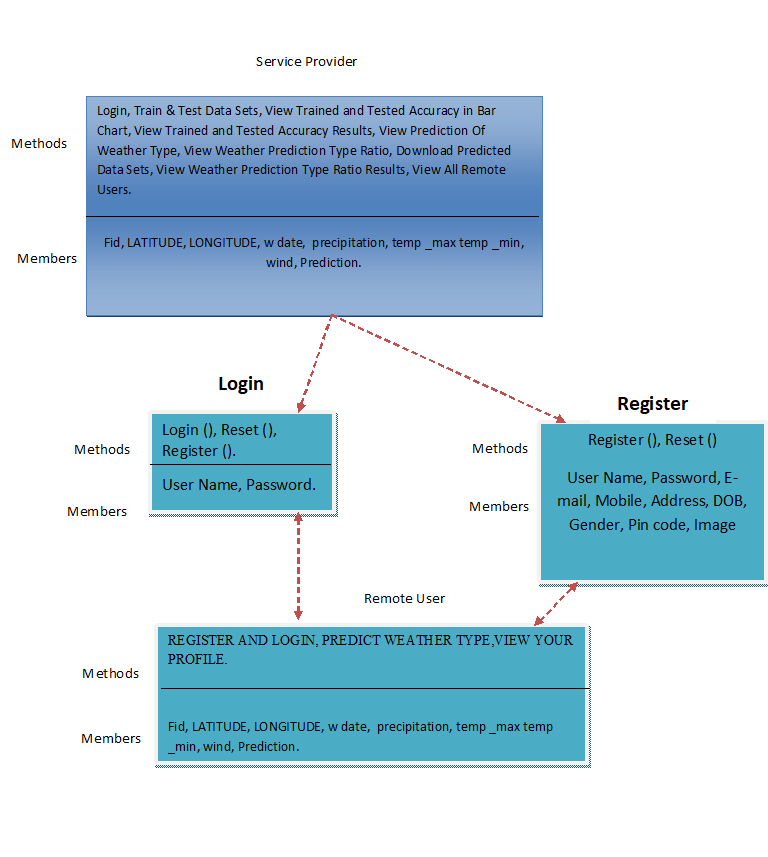
**DATA FLOW DIAGRAM**



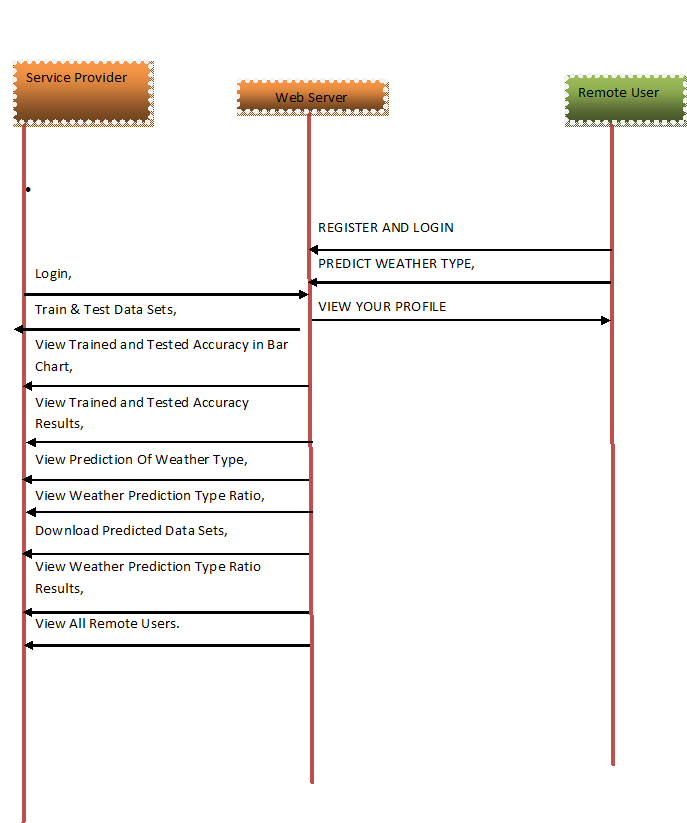
**USE CASE DIAGRAM**



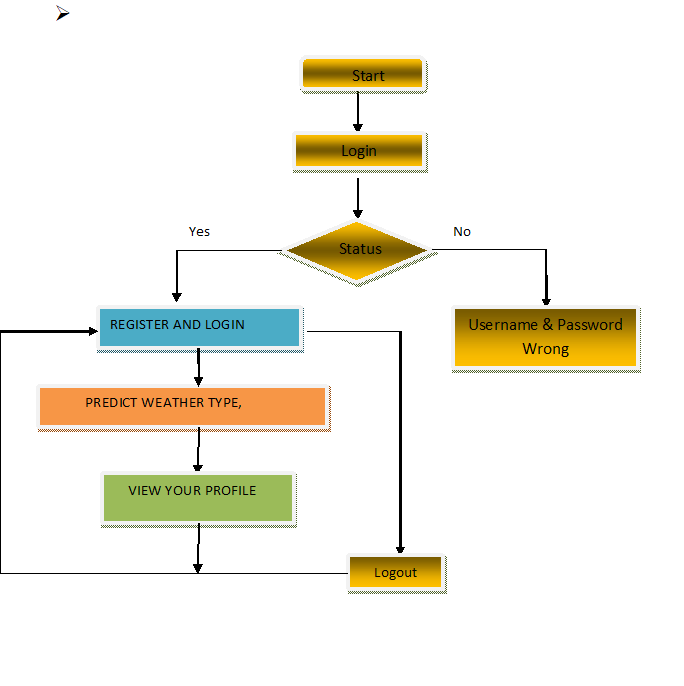
**CLASS DIAGRAM**



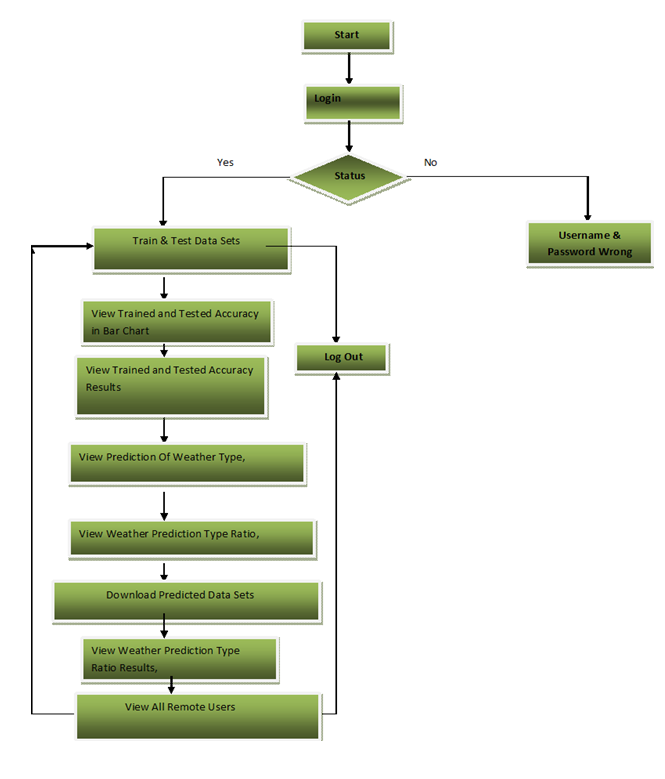
**SEQUENCE DIAGRAM**

****

* **Flow Chart : Remote User**

****

* **Flow Chart : Service Provider**

****

**SYSTEM REQUIREMENTS**

➢ **H/W System Configuration:-**

➢ Processor - Pentium –IV

➢ RAM - 4 GB (min)

➢ Hard Disk - 20 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* **Operating system :** Windows 7 Ultimate.
* **Coding Language :** Python.
* **Front-End :** Python.
* **Back-End :** Django-ORM
* **Designing :** Html, css, javascript.
* **Data Base :** MySQL (XAMP Server).

**CHAPTER-4**

**IMPLEMENTATION**

**Modules**

**Service Provider**

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as Browse Datasets and Train & Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Predicted Job Title Identification Type, View Job Title Identification Type Ratio, Download Predicted Data Sets, View Job Title Identification Type Ratio Results, View All Remote Users.

**View and Authorize Users**

In this module, the admin can view the list of users who all registered. In this, the admin can view the user’s details such as, user name, email, address and admin authorizes the users.

**Remote User**

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN, Predict Job Title Identification Type, VIEW YOUR PROFILE.

**CHAPTER-5**

**SOFTWARE ENVIRONMENT**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability)and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



**Create a Project**

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure −

myproject/

manage.py

myproject/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements −

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

\_\_init\_\_.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

**Introduction to Python**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. The Python 2 language, i.e. Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it.[32][33] With Python 2's end-of-life, only Python 3.5.x and later are supported. Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source[35] reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

**SYNTAX AND SEMANTICS**

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation.

Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

**Indentation**

Main article: Python syntax and semantics § Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation doesn't have any semantic meaning.

**Statements and control flow**

Python's statements include (among others):

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound.

Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.

* The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The while statement, which executes a block of code as long as its condition is true.
* The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
* The raise statement, used to raise a specified exception or re-raise a caught exception.
* The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
* The def statement, which defines a function or method.
* The with statement, from Python 2.5 released in September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior and replaces a common try/finally idiom.
* The break statement, exits from the loop.
* The continue statement, skips this iteration and continues with the next item.
* The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
* The assert statement, used during debugging to check for conditions that ought to apply.
* The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import \* or from <module name> import <definition 1> [as <alias 1>], <definition 2> [as <alias 2>],

The print statement was changed to the print() function in Python 3.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

**EXPRESSIONS**

Some Python expressions are similar to languages such as C and Java, while some are not:

Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating point/division. Python also added the \*\* operator for exponentiation.

From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.

From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.

In Python, == compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.

Python uses the words and, or, not for its boolean operators rather than the symbolic &&, ||, ! used in Java and C.

Python has a type of expression termed a list comprehension. Python 2.4 extended list comprehensions into a more general expression termed a generator expression.

Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y operator common to many other languages).

Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.

Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

Python has a "string format" operator %. This functions analogous to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2".

In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. "spam={0} eggs={1}".format("blah", 2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.

**Python has various kinds of string literals:**

Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".

Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.

Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.

Python has array index and array slicing expressions on lists, denoted as a[key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

List comprehensions vs. for-loops

Conditional expressions vs. if blocks

The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python.

**METHODS**

Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).

**APPLICATIONS OF PYTHON**

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

**Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

**Easy-to-read** − Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain** − Python's source code is fairly easy-to-maintain.

**A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

**Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases** − Python provides interfaces to all major commercial databases.

**GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable** − Python provides a better structure and support for large programs than shell scripting.

INSTALLATION STEPS OF PYTHON

Installing and using Python on Windows 10 is very simple. The installation procedure involves just three steps:

* Download the binaries
* Run the Executable installer
* Add Python to PATH environmental variables

To install Python, you need to download the official Python executable installer. Next, you need to run this installer and complete the installation steps. Finally, you can configure the PATH variable to use python from the command line.

**Step 1**: Download the Python Installer binaries

* Open the official Python website in your web browser. Navigate to the Downloads tab for Windows.
* Choose the latest Python 3 release. In our example, we choose the latest Python 3.7.3 version. Click on the link to download Windows x86 executable installer if you are using a 32-bit installer.
* In case your Windows installation is a 64-bit system, then download Windows x86-64 executable installer.

**Step 2:** Run the Executable Installer

1. Once the installer is downloaded, run the Python installer.
2. Check the Install launcher for all users check box. Further, you may check the Add Python 3.7 to path check box to include the interpreter in the execution path.



1. Select **Customize installation**.

Choose the optional features by checking the following check boxes:

1. Documentation
2. pip
3. tcl/tk and IDLE (to install tkinter and IDLE)
4. Python test suite (to install the standard library test suite of Python)
5. Install the global launcher for `.py` files. This makes it easier to start Python
6. Install for all users.



Click Next.

1. This takes you to Advanced Options available while installing Python. Here, select the Install for all users and Add Python to environment variables check boxes.

Optionally, you can select the Associate files with Python, Create shortcuts for installed applications and other advanced options. Make note of the python installation directory displayed in this step. You would need it for the next step.

After selecting the Advanced options, click Install to start installation.



1. Once the installation is over, you will see a Python Setup Successful window.



**Step 3:** Add Python to environmental variables

The last (optional) step in the installation process is to add Python Path to the System Environment variables. This step is done to access Python through the command line. In case you have added Python to environment variables while setting the Advanced options during the installation procedure, you can avoid this step. Else, this step is done manually as follows.

In the Start menu, search for “advanced system settings”. Select “View advanced system settings”. In the “System Properties” window, click on the “Advanced” tab and then click on the “Environment Variables” button.

Locate the Python installation directory on your system. If you followed the steps exactly as above, python will be installed in below locations:

* C:\Program Files (x86)\Python37-32: for 32-bit installation
* C:\Program Files\Python37-32: for 64-bit installation

The folder name may be different from “Python37-32” if you installed a different version. Look for a folder whose name starts with Python.

Append the following entries to PATH variable as shown below:





**Step 4:** Verify the Python Installation

You have now successfully installed Python 3.7.3 on Windows 10. You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation. Search for the command prompt and type “python”. You can see that Python 3.7.3 is successfully installed.



An alternate way to reach python is to search for “Python” in the start menu and clicking on IDLE (Python 3.7 64-bit). You can start coding in Python using the Integrated Development Environment(IDLE).



**Uses**

Since 2003, Python has consistently ranked in the top ten most popular programming languages in the TIOBE Programming Community Index where, as of February 2020, it is the third most popular language (behind Java, and C). It was selected Programming Language of the Year in 2007, 2010, and 2018.

* An empirical study found that scripting languages, such as Python, are more productive than conventional languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++".
* Large organizations that use Python include Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify and some smaller entities like ILM and ITA. The social news networking site Reddit is written entirely in Python.
* Python can serve as a scripting language for web applications, e.g., via mod\_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications.
* SQLAlchemy can be used as data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.
* Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a mathematical software with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.
* Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.
* Python is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras, Pytorch and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.
* Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.
* Python is used extensively in the information security industry, including in exploit development.
* Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.
* Due to Python's user-friendly conventions and easy-to-understand language, it is commonly used as an intro language into computing sciences with students. This allows students to easily learn computing theories and concepts and then apply them to other programming languages.

**CHAPTER-6**

**SYSTEM STUDY AND TESTING**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### **TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER-7**

**SCREEN SHOTS**

**CHAPTER-8  
CONCLUSION**

A machine learning-based approach for optimizing hardware/ software parameters of scientific applications was demonstrated in this study. The weather forecast scientific application Low GloSea6 was used as a target, and a dataset containing the application’s internal parameters and hardware platform parameters and performance data based on the combination of these two parameters was constructed. Before applying the machine-learning model, the dataset was verified, and the validity of the regression model trained with insufficient data was ensured through the LOOCV technique. The optimal hardware platform parameters and corresponding Low GloSea6 internal parameters were found using the trained machine-learning model in a new research environment and these values agreed with the actual parameter combinations. In particular, the predicted execution time based on the parameter combination showed a 16% error rate compared to the actual execution time, demonstrating a meaningful result in predicting execution time. The proposed optimization method can be applied to improve the performance of other HPC scientific applications. Besides weather and climate modeling, to name a few, there are computational fluid dynamics (CFD) simulations, molecular dynamics (MD) simulations, and quantum chemistry calculations. Frequently, scientists who run such HPC scientific applications used to get help from staff members at supercomputing centers to optimize their applications, and our optimization method will help this manual performance optimization process expedited.

Two directions for future research are outlined in terms of data. First, increasing the absolute amount of data is necessary. In this study, the accurate prediction of execution time was hindered owing to the omission of some hardware platform parameters. Therefore, collecting additional hardware/ software parameters and I/O performance indicators would improve model performance. Second, implementing the benchmark-based cross-inference optimization method proposed in this study’s initial algorithm would be beneficial. This would accelerate data collection and enable the collection of parameter values not collected in this study through alternative parameters, thereby expanding the model performance improvement and application range.

**REFERENCES**

[1] *Concept of a Numerical Forecast Model*. Accessed: Aug. 10, 2023. [Online]. Available: <http://web.kma.go.kr/aboutkma/intro/supercom/> model/model\_concept.jsp

[2] P. Davis, C. Ruth, A. A. Scaife, and J. Kettleborough, ‘‘A large ensemble seasonal forecasting system: GloSea6,’’ Dec. 2020, vol. 2020.

[3] M. Howison, Q. Koziol, D. Knaak, J. Mainzer, and J. Shalf, ‘‘Tuning HDF5 for Lustre file systems,’’ Lawrence Berkeley Nat. Lab., Berkeley, CA, USA, Tech. Rep. LBNL-4803E, 2010. [4] B. Behzad et al., ‘‘Taming parallel I/O complexity with auto-tuning,’’ in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2013, p. 68.

[5] B. Behzad, S. Byna, Prabhat, and M. Snir, ‘‘Optimizing I/O performance of HPC applications with autotuning,’’ *ACM Trans. Parallel Comput.*, vol. 5, no. 4, pp. 1–27, Mar. 2019, doi: 10.1145/3309205.

[6] S. Robert, S. Zertal, and G. Goret, ‘‘Auto-tuning of IO accelerators using black-box optimization,’’ in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2019, pp. 1022–1027, doi: 10.1109/HPCS48598.2019.9188173.

[7] A. Bağbaba, X. Wang, C. Niethammer, and J. Gracia, ‘‘Improving the I/O performance of applications with predictive modeling based auto-tuning,’’ in *Proc. Int. Conf. Eng. Emerg. Technol. (ICEET)*, Oct. 2021, pp. 1–6, doi: 10.1109/ICEET53442.2021.9659711.

[8] S. Valcke and R. Redler, ‘‘The OASIS coupler,’’ in *Earth System Modelling*, vol. 3. Berlin, Germany: Springer, 2012, pp. 23–32, doi: 10.1007/978-3-642-23360-9\_4.

[9] *Analysing UM Outputs*. Accessed: Feb. 14, 2023. [Online]. Available: <http://climate-cms.wikis.unsw.edu.au/Analysing_UM_outputs> [10] *Unidata | NetCDF*. Accessed: Nov. 28, 2022. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/

[11] Icons8. *Free Icons, Clipart Illustrations, Photos, and Music*. Accessed: Jul. 18, 2023. [Online]. Available: https://icons8.com

[12] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, ‘‘24/7characterization of petascale I/O workloads,’’ in *Proc. 2009 Workshop Interfaces Archit. Sci. Data Storage*, Sep. 2009, pp. 1–10.

[13] *Darshan Introduction*. Accessed: Aug. 10, 2023. [Online]. Available: <https://wordpress.cels.anl.gov/darshan/wp-content/uploads/sites/54/> 2014/08/iiswc-2014-darshan-instrumentation.pdf

[14] R. Ross, D. Nurmi, A. Cheng, and M. Zingale, ‘‘A case study in application I/O on Linux clusters,’’ in *Proc. ACM/IEEE Conf. Supercomput.*, New York, NY, USA, Nov. 2001, p. 11, doi: 10.1145/582034.582045.

[15] S. Herbein, D. H. Ahn, D. Lipari, T. R. W. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer, ‘‘Scalable I/Oaware job scheduling for burst buffer enabled HPC clusters,’’ in *Proc. 25th ACM Int. Symp. High-Perform. Parallel Distrib. Comput.*, New York, NY,

USA, May 2016, pp. 69–80, doi: 10.1145/2907294.2907316.

[16] *R Package*. Accessed: Apr. 26, 2023. [Online]. Available: <https://cran.rproject>. org/src/contrib/Archive/

[17] R. Genuer, J.-M. Poggi, and C. Tuleau, ‘‘Random forests: Some methodological insights,’’ 2008, *arXiv:0811.3619*.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2009, doi: 10.1007/978-0-387-84858-7.

[19] *Package RandomForest*. Accessed: Apr. 26, 2023. [Online]. Available: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

[20] *PackageGBM*. Accessed: Apr. 26, 2023. [Online].Available: <https://cran.rproject>. org/web/packages/gbm/gbm.pdf