Reliable Execution and Deployment of Workflows in Cloud Computing

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Monika Bharti
(Roll No. 801032016)

Under Supervision of
Ms.Anju Bala
Assistant Professor,CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

Certificate

I hereby certify that the work which is being presented in the thesis report titled, **"Reliable Execution and Deployment of Workflows in Cloud Computing "**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Comp.Sci & Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mrs. Anju Bala* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Monika Bharti

Signature:

(Monika Bharti)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Anju Bala

(Mrs. Anju Bala)

Computer Science & Engineering Department,
Thapar University
Patiala

Countersigned by

03/07/2011

(Dr. Maninder Singh)
Associate Professor & H.O.D,
Computer Science & Engineering Department,
Thapar University,
Patiala

(Dr. S.K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all, I thank God almighty for guiding me the right direction. With his mercy, it has been made possible for me to reach so far.

It is a great privilege to express my gratitude and admiration towards my respected supervisors Mrs. Anju Bala. She has been an esteemed guide and a great support behind achieving the task. Without her able guidance, kind efforts and encouragement, the work wouldn't have been what it is. I am truly grateful to her for extending their total co-operation and understanding whenever I needed help and guidance from her.

I am extremely grateful to Dr. Maninder Singh, Head, Department of Computer Science & Engineering, Thapar University, Patiala for providing best facilities and atmosphere for the creative work guidance and encouragement.

I would like to thank all those who have contributed to the completion of my thesis work and helped me with valuable suggestions for improvement. Above all I am grateful to my parents and grandparents who soulfully provided me their constant support, and encouraging attitude to undertake the challenge of this proportion.

I am also very thankful to the entire faculty and staff members of Computer Science & Engineering Department for their direct and indirect help, cooperation, love and affection which made my stay at Thapar University memorable.

Monika Bharti

(801032016)

Abstract

Cloud computing is a paradigm that provides demand service resources like software, hardware, platform, and infrastructure. Due to the advantages of cost-effectiveness, on-demand provision, easy for sharing, scalability, reliability, cloud computing has grown in popularity with research community for deploying scientific applications such as workflows. The underlying system in the workflows is Workflow Management System, which provides the end user with the required data and appropriate application program for their tasks. Workflow management is a fast evolving technology which is increasingly being exploited by businesses in a variety of industries. Its primary characteristic is the automation of processes involving combinations of human and machine-based activities, particularly those involving interaction with IT applications and tools. The main purpose of a workflow management system (WfMS) is to support the definition, execution, registration and control of business processes.

Workflow scheduling is one of the key issues in the workflow management that maps and manages the execution of inter-dependent tasks on the distributed resources. It allocates suitable resources to workflow tasks such that the execution can be completed to satisfy objective functions imposed by users. Proper scheduling can have significant impact on the performance of the system. Fault tolerance is another major concern to guarantee availability and reliability of critical services as well as application execution. In order to minimize failure impact on the system and application execution, failures should be anticipated and proactively handled.

So, there is need to implement reliable execution of workflows in cloud environment. Since in workflow, there is lot of data and compute nodes to process data, therefore execution time is required to minimize. Therefore, a tool is required to design and specify of task parallelism and task dependency ordering, coordinate execution of the tasks over large computer resources, facilitate workflow reuse over multiple datasets. The thesis discusses the various tools for generating workflow and these tools have been compared on the basis of operating system, databases, architecture and environment. Pegasus bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto distributed resources. It automatically locates the necessary input data and computational resources necessary for workflow execution.

Condor Scheduler manages individual workflow tasks and supervises their execution on local and remote resources. So, the application on workflow is designed and implemented with Pegasus. The designed workflow is analyzed, monitored and scheduled. Then the designed workflow can be deployed on Nimbus. The workflow type application is also designed using Oozie. The generated workflow is implemented in hadoop environment. Hadoop is an open-source JAVA based software platform developed by Apache Software Foundation. It lets one easy to write and run their applications on clusters to process huge amount of data. Then the designed workflows in oozie and Pegasus have been compared on the basis of mapper, enviorment, scheduling, and reliability with compatible cloud environments like Nimbus and Hadoop.

Keywords: *Cloud Computing, Workflow, Workflow Management System, Workflow Engine*

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter includes background, architecture, framework, examples, advantages and research issues for cloud computing. The framework model includes its deployment models, service models, essential characteristics and common characteristics.

1.1 Background of Cloud Computing

In the early 1980s, stand-alone computers were used by most of the people. Information and the applications used to work with these computers were abandoned on the computer's desktop. By the end of that decade, more and more offices were networking their computers, allowing information to be shared easily through servers from companies. In the 1990s, the Internet made its transition from the domain of research and academia to the general public. Information among offices and businesses was fairly shared. By the end of the 1990s, the Internet also began to generate Application Service Providers which untied the IT department to maintain servers. ASPs also provided knowledge fine tuned to the applications they hosted. This was a real benefit to anyone who tried to use web mapping platforms. At the turn of the 21st century, Salesforce.com, instead of hosting applications formed specifically for Application Service Providers began charging for access to the applications in succession on their servers. At the same time, Amazon utilized virtualization technology that allows a computer to imitate one or more computers in software. Like Amazon, Google also leveraged a positive cash flow after the dot-com bubble to purchase up data centers and network capacity throughout the country. The desire to generate business insights and competitive advantage in an age of information and globalization leads to cloud computing. The distributed computing paradigm that mixes aspects of Grid computing, Internet computing, Utility computing, Autonomic computing and Green computing is Cloud Computing.

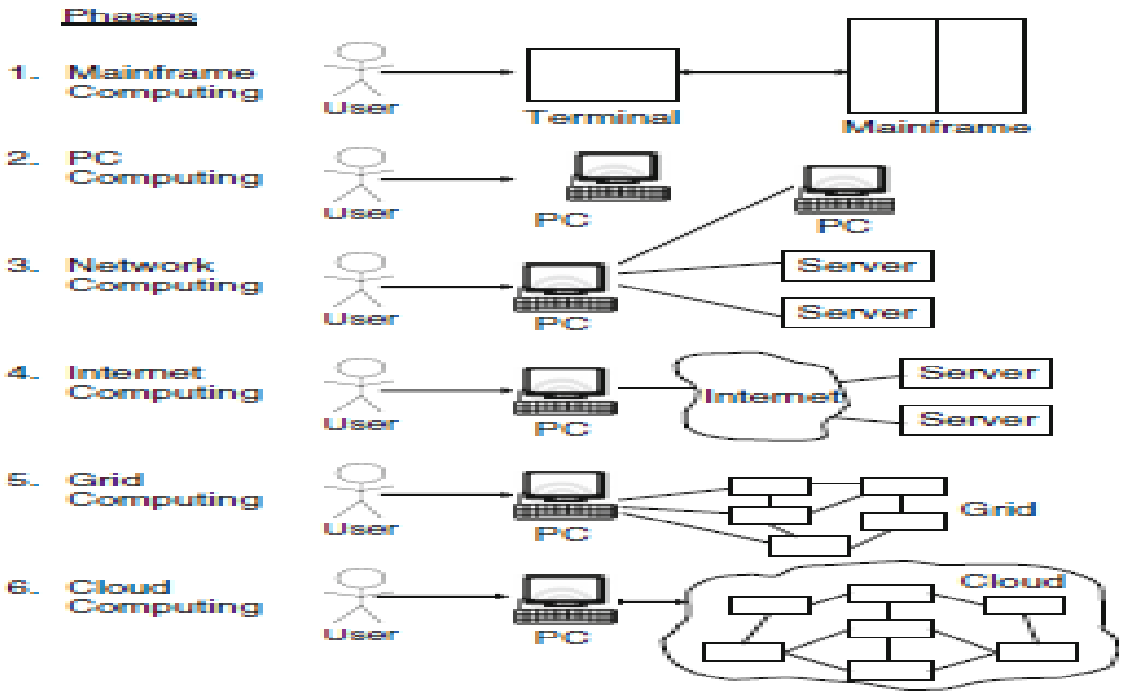


Figure 1.1[1]: Evolution of Cloud Computing

1.1.1 Cloud Computing Cloud computing is a technology which makes use of internet and central remote servers to maintain data and applications. One can use applications without installation and access their personal files at any computer with internet access. Its main features are centralizing storage, memory, processing and bandwidth.

Figure 1.2[2]: Cloud Computing

The development and the success of Cloud are due to the maturity reached by hardware and software virtualization, data transportation, dynamic extendibility and distributional. It promises to provide on-demand computing power with quick implementation, little maintenance, less IT staff, and consequently lower cost. It aims to share data, calculations and services transparently among users of a massive grid.

A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers [3]. It provides computing resources in the pool for users through internet. Integrated cloud computing is a whole dynamic computing system. It provides a mandatory application program environment [4]. It can deploy, allocate or reallocate computing resource dynamically and monitor the usage of resources at all times. In general, cloud computing has a distributed foundation concern, and scrutinize the distributed system, to achieve the purpose of proficient use of the system [5]. Cloud computing collects and manage all the computing resources automatically through software. It integrates the history data and present data to formulate the collected information more accurate and provide supplementary intelligent service for users and enterprises in the process of data analysis [6].

1.2 Architecture of Cloud Computing

The architecture of cloud computing is categorized in to Front End and Back End where Front End is end user/client or any application (i.e. web browser etc.), using cloud services and back End is the network of servers with any computer program and data storage system. Cloud has centralized server administration system which administers the system, balances client supply, adjusts demands, monitors traffic and avoids congestion. The centralized server follows protocols, called middleware. It reins the communication of cloud network among them. The demand for resources on cloud network is not always constant from client to cloud. Hence the servers of cloud are not capable to run at their full capacity. In order to overcome this negative aspect, server virtualization technique is applied in which all physical servers are virtualized and they run multiple servers with either same or different application. As one physical server acts as multiple physical servers, it curtails the need for more physical machines.

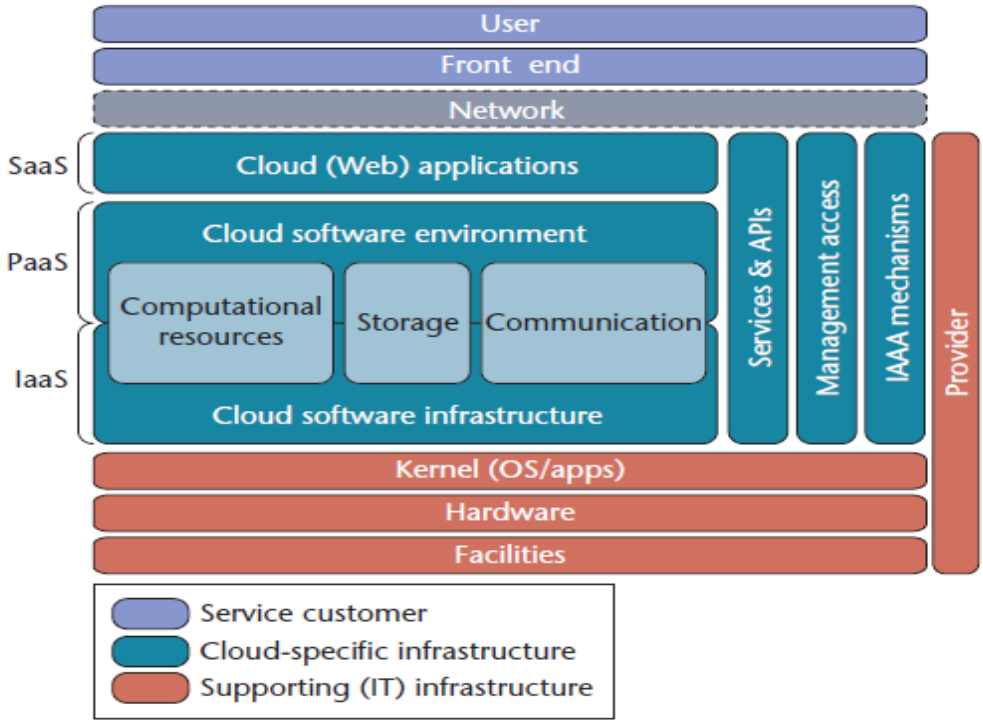


Figure 1.3[7]: Cloud Computing Architecture

1.2.1. Cloud Services

A physical web server typically has three tiers to it: The physical infrastructure, the operating system platform, and the web application software being run. A cloud container may contain one, two or all of these layers. The services provided by cloud computing are [8]:

a) **SAAS (Software as a service):** The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. Example: Rackspace Mosso, Web Fusion.

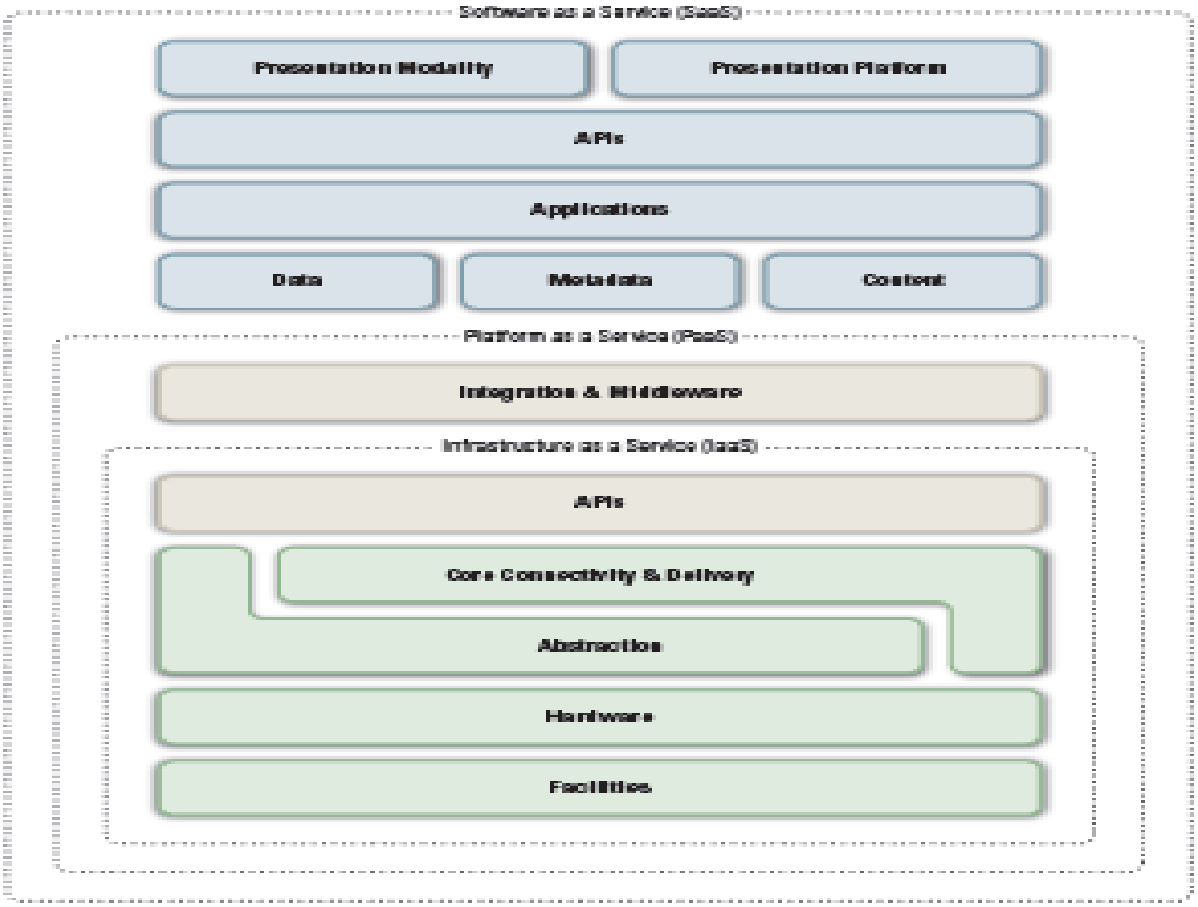


Figure 1.4[9]: Cloud Computing Services

b) PAAS (Platform as a service): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating system, or storage, but has control over the deployed applications and possibly application hosting environment configurations. Example: Google App Engine, Microsoft Windows Azure.

c) IAAS (Infrastructure as a Service): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed

applications, and possibly limited control of select networking components. Example: Amazon EC2.

1.3. The Framework of Cloud Computing

Clouds are a large pool of easily usable and accessible virtualized resources which can be dynamically re-configured to adjust to a variable load allowing also for optimum resource utilization.

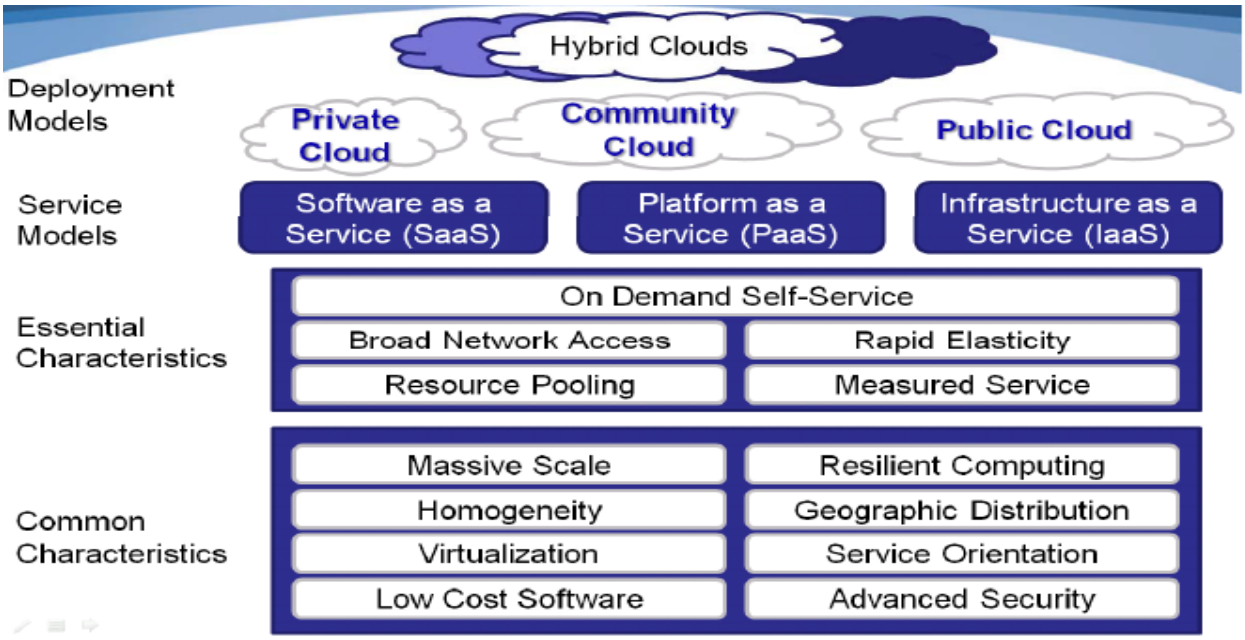


Figure 1.5[10]: Framework of Cloud Computing

1.3.1 Deployment Models

a) **Private cloud:** The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.

b) **Community cloud:** The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

c) **Public cloud:** The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

d) **Hybrid cloud:** The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

1.3.2 Essential characteristics of Cloud Computing

a) **Virtualization**: We can virtualizes many factors such as IT resource, software, hardware, operating system and net storage, and manage them in the cloud computing platform; every environment has nothing to do with the physical platform.

b) **Distributional**: It refers to the physical node which the computation uses is distributed.

c) **Ultra large-scale:** The scale of cloud is large. The cloud of Google has owned more than one million servers. Even in Amazon, IBM, Microsoft, Yahoo, they have more than hundreds of thousand servers. There are hundreds of servers in an enterprise. Cloud enlarges the user's computing power.

d) **High reliability:** Cloud uses data multi-transcript fault tolerant, the computation node isomorphism exchangeable and so on to ensure the high reliability of the service. Using cloud computing is more reliable than local computer.

e) **Versatility:** Cloud computing doesn't aim at certain special application. It can produce various applications supported by cloud, and one cloud can support different applications running it at the same time.

f) **High extendibility:** The scale of cloud can extend dynamically to meet the increasingly requirement.

g) **On demand service:** Cloud is a large resource pool that you can buy according to your need; cloud is just like running water, electric, and gas that can be charged by the amount that you used.

h) **Extremely inexpensive :** Because the cloud's special fault tolerance can be built by very inexpensive nodes, the centered management of cloud make the enterprise needn't undertake the management cost of data center that increase very fast. The versatility can increase the utilization rate of the available resources compared with traditional system, so users can fully enjoy the low cost advantage.

1.3.3 Examples of Cloud Computing

a) "Gmail": It is a web-based e-mail service that allows an organization to run its e-mail system using Google's systems and provides the capability to access an End User's inbox from a supported web browser, read mail, compose, reply to, and forward mail, search mail, and manage mail through labels.

b) "Google Talk": It provides the ability for real time communication between End Users through instant messaging. Additionally it provides the ability for one-to-one communication via video and voice. Google Talk is integrated into the Gmail web interface, and also has an XMPP programmatic interface.

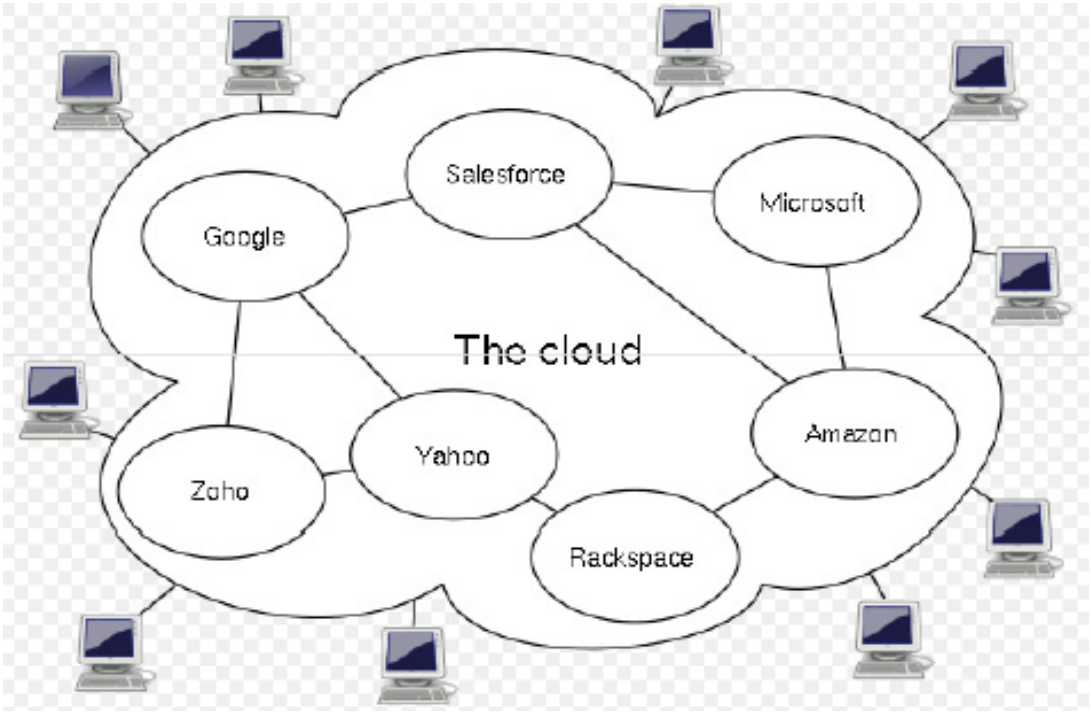


Figure 1.6 [11]: Examples of Cloud Computing

c) "Google Docs": It provides web-based tools for creating and collaborating on documents, spreadsheets, and presentations. An End User can create these documents through a web browser or upload documents from other supported systems, and can share documents with other End Users to jointly collaborate and edit.

d) **"Microsoft 365"**: its familiar Microsoft Office collaboration and productivity tools delivered through the cloud. Everyone can work together easily with anywhere access to email, web conferencing, documents, and calendars. It includes business-class security and is backed by Microsoft. Whether you are a small business or multinational enterprise, Office 365 offers plans designed to fit your organization's unique needs. ITS products are Microsoft Exchange Online, Microsoft Share Point, Microsoft Lync Online, Office Professional Plus, and Office Web Apps.

e) **"Social networking":** Perhaps the most famous use of cloud computing, which does not strike people as "cloud computing" at first glance is social networking Websites, including Facebook, LinkedIn, MySpace, Twitter, and many, many others.

1.3.4 Advantages of Cloud Computing [12]

a) **Scalability on Demand**: All organizations have to deal with changes in their environments. The ability of cloud computing solutions to scale up and down is a major benefit. If an organization has periods of time in which their computing resource needs are much higher or lower than normal, cloud technologies (both private and public) can deal with those changes. The organization pays for the IT resources it actually uses; it does not have to maintain multiple sets of artificially high levels of resources to handle peak demands.

b) **Streamlining the Data Center**: An organization of any size will have a substantial investment in its data center. That includes buying and maintaining the hardware and software, providing the facilities in which the hardware is housed and hiring the personnel who keep the data center running. An organization can streamline its data center by taking advantage of cloud technologies internally or by offloading workload into the public.

c) **Improving Business Processes:** The cloud provides an infrastructure for improving business processes. An organization and its suppliers and partners can share data and applications in the cloud, allowing everyone involved to focus on the business process instead of the infrastructure that hosts it.

d) **Minimizing Startup Costs:** For companies that are just starting out, organizations in emerging markets, or even "Skunk Works" groups in larger organizations, cloud

computing greatly reduces startup costs. The new organization starts with an infrastructure already in place, so the time and other resources that would be spent on building a data center are borne by the cloud provider, whether the cloud is private or public.

1.4. Research issues of Cloud Computing

a) **Workflow Scheduling [13]**: Scheduling refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. Workflow scheduling is a process that maps and manages the execution of inter-dependent tasks on the distributed resources.

b) **Load Balancing [14]:** The goal of a cloud-based architecture is to provide some form of elasticity, the ability to expand and contract capacity on-demand. The implication is that at some point additional instances of an application will be needed in order for the architecture to scale and meet demand. That means there needs to be some mechanism in place to balance requests between two or more instances of that application. The mechanism most likely to be successful in performing such a task is a load balancer. The challenges of attempting to build such architecture without a load balancer are staggering.

c) **Workflows in Cloud Computing [15]**: A workflow models a process as consisting of a series of steps that simplifies the complexity of execution and management of applications. It enables the structuring of applications in a directed acyclic graph form.

d) **Fault Tolerance in Cloud Computing [16]**: It is designed to provide reliable and continuous services for wired and wireless distributed networks despite the failures of some of their components.

e) **Fault Detection in Cloud Computing [17]**: It is an essential building block for reliability, performance, load balancing and scheduling where mobile users engage in computing using data-rich environments and multi-cloud solutions.

f) **Security [18]:** Data security is another important research topic in cloud computing. Since service providers typically do not have access to the physical security system of data centers, they must rely on the infrastructure provider to achieve full data security by achieving confidentiality and auditability.

1.5. Organization of Thesis

Chapter 1 has introduction. Literature survey is presented in chapter 2. Chapter 3 contains the problem statements taken up in the thesis. Design of the solution is presented in chapter 4. Chapter 5 has experimental results. Conclusion and future scope is discussed in Chapter 6.

Chapter 2

Literature Review

This chapter gives a brief introduction to workflow, workflow lifecycle, workflow management system, its various research issues, tells about the need for workflows in cloud computing, comparison of various workflow tools. It also discuss about Pegasus workflow tool, Oozie workflow engine and Hadoop cloud platform in detail.

2.1 Workflow

A workflow expresses an automation of procedures wherein files and data are passed between procedures applications according to a defined set of rules, to achieve an overall goal [19]. "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [20]." It consists of a sequence of steps that simplifies the complexity of execution and management of applications. Operational aspects of workflow are:

- How tasks are structured,
- Who performs them
- What their relative order is
- How they are synchronized
- How information flows to support the tasks
- How tasks are being tracked

2.1.1. Workflow Model/Specification: Workflow Model or workflow specification defines a workflow together with its task definition and structure definition. In Figure 2.2, different types of workflow models are shown, namely abstract and concrete. They are also referred to as abstract workflows and concrete workflows [21] [22]. Concrete models are referred to as executable workflows.

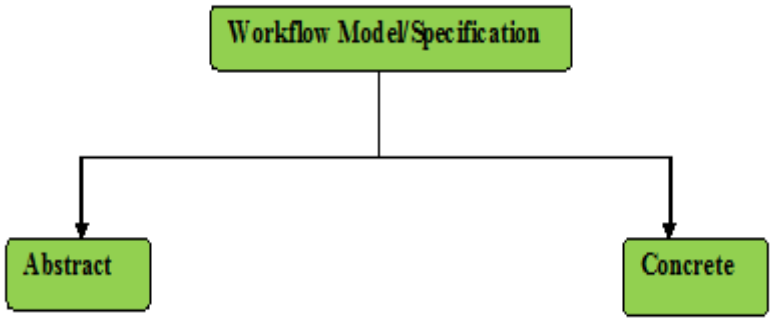
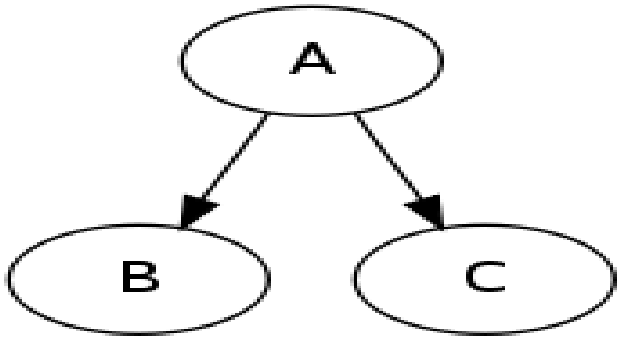


Figure 2.1: Workflow Mode/Specification

In an abstract model, a workflow is described in an abstract form and provides a elastic way for users to define workflows without being fretful about low-level implementation details. Tasks in an abstract model are portable and can be mapped onto any apt Grid services at run-time by using suitable discovery and mapping mechanisms. In contrast, a concrete model binds workflow tasks to specific resources and may include tasks in some cases, acting as data movement to transfer data in and out of the computation and data publication to circulate newly derived data.

2.1.2. Different structures of workflows in cloud computing [23]

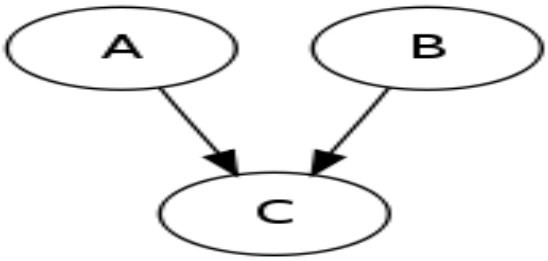
a) Fan out: Completion of a task needs to notify all its successors.



- A; B; C
- A; C; B

Figure 2.2[23]: Fan out

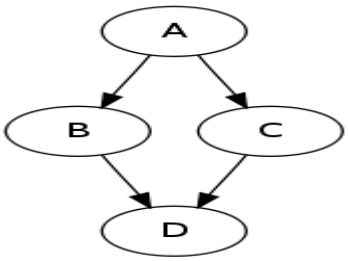
b) Fan in: A task cannot start until all its predecessors have completed.



- A; B; C
- B; A; C

Figure 2.3[23]: Fan in

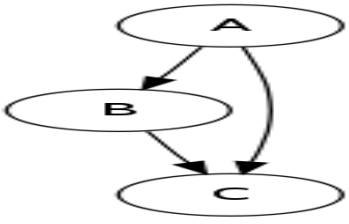
c) Diamond: Combination of \fan in" and \fan out". Need to ensure that D is not run twice.



- A; B; C; D
- A; C; B; D

Figure 2.4[23]: Diamond

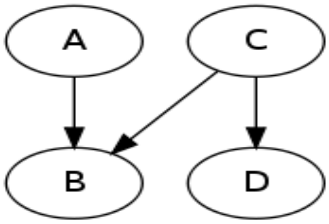
d) Intermediary: Another variation of combination \fan in" and \fan out". Need to ensure that C is not run twice.



- A; B; C

Figure 2.5[23]: Intermediary

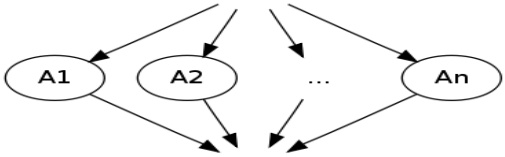
e) N: Another variation of combination \fan in" and \fan out". Computational linguistics theory: N structures in a pomset grammar increase the degree of context sensitivity of the grammar.



- A; C; B; D
- A; C; D; B
- C; A; B; D
- C; A; D; B
- C; D; A; B

Figure 2.6[23]: N

f) Task partitioning (Parameter sweep, Map Reduce): Issues are dynamic generation of task partitions and combination of task partitioning with previous structure types.



- A1; A2; ...; An

Figure 2.7[23]: Task partitioning

2.2 Workflow Lifecycle [24]

a) Design: One of the most critical challenges facing organizations when deploying process automation solutions is rapidly designing new processes and cost effectively modifying those processes as the business needs change. Adobe lifecycle workflow reduces development costs by providing a highly intuitive environment for business and IT professionals to visualize design end-to-end workflows without programming knowledge.

b) Integrate: When deploying process automation solutions, the existing IT environment may not be easy to scale or extend, making it difficult to integrate new processes. Adobe

lifecycle workflow ensures automated processes can be extended and scaled through a component-based architecture.

c) **Deploy :** Automated processes may need to be deployed in an environment with many different operating system , database technologies and application servers and may need to be integrated with existing Enterprise Resource Planning system or Customer Relationship Management systems. Adobe lifecycle workflow allows large enterprise deployment covering multiple users, multiple departments and multiple geographies by supporting many different operating systems, database technologies and application servers.

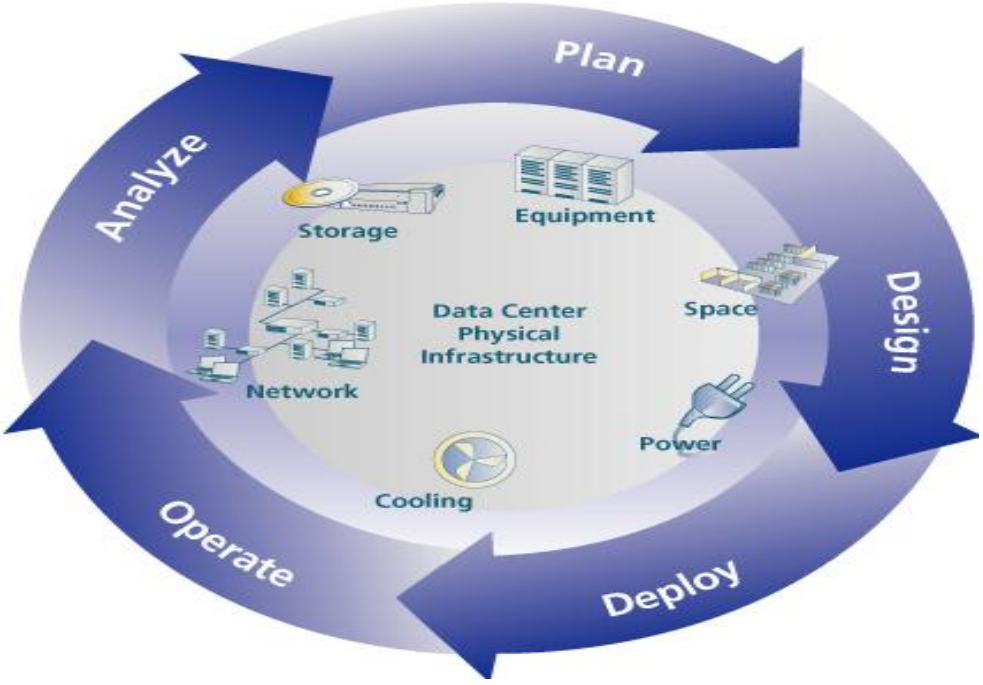


Figure 2.8[24]: Workflow Lifecycle stages

d) **Manage:** Managing automated processes can be difficult if the management tools are non-intuitive, not powerful enough or inflexible. Adobe lifecycle workflow provides its own tools and leverages its integration with other Lifecycle products to ensure effective management.

e) **Optimize:** Process management optimization can be hampered by incomplete or stale process data. Adobe lifecycle workflow provides out-of-the-box monitoring of generic

metrics and complete access to all critical real-time and historical process data for analysis.

2.3 Workflow Management System

In workflow, there is lot of data and compute nodes to process data, therefore execution time is required to minimize. Hence, a tool is required to design and specify of task parallelism and task dependency ordering, coordinate execution of the tasks over large computer resources, facilitate workflow reuse over multiple datasets. Hence, workflow management system comes. Workflow management is a fast evolving technology which is increasingly being exploited by businesses in a variety of industries. Its primary characteristic is the automation of processes involving combinations of human and machine-based activities, particularly those involving interaction with IT applications and tools [25]. The main purpose of a workflow management system (WfMS) is to support the definition, execution, registration and control of business processes.

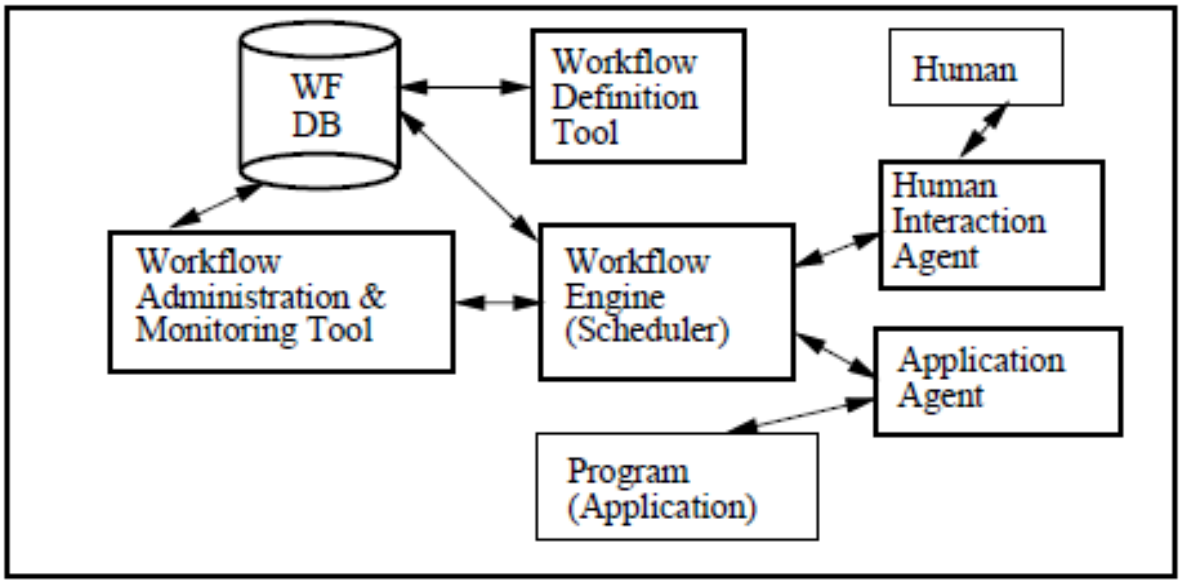


Figure 2.9[26]: Components of a Workflow System

A workflow schema is basically a directed graph with nodes representing the steps to be performed by typically executing a program that accesses a database/document. The arcs connecting the steps are of two types: data and control arcs. A data arc denotes the flow of data between steps while a control arc specifies the ordering sequence among two steps. The second

may also have a condition coupled with it to specify that the following step is to be executed only if the condition evaluates to true. The program linked with a step and the data that is accessed by the step are not known to the WFMS and therefore the steps themselves are 'black boxes' as far as the WFMS is fretful. A WFMS does not have access to step related information entrenched in resource managers accessed by a program executed to carry out a step. Hence, it needs additional information to determine for example if two steps from different workflows could have accessed the same resources in a conflicting manner. Using a workflow schema as a outline, several instances of the workflow can be executed. The workflow engine is responsible for managing all the instances of all schemas, i.e.; it has to schedule the steps of the instances when they are eligible for execution. It provides the agent with the information required to execute a step. The agent is accountable for executing the step and communicating back the outcome of the step to the engine. The workflow database (WFDB) provides the persistence required to smooth the progress of forward recovery in case of failure of the workflow engine. Failures can occur in workflow system components and they have to be handled efficiently to improve WFMS availability.

It can be seen that scheduling is a function module of the Workflow Engine(s), thus it is a significant part of workflow management systems. The need for workflow management system:

- describing complex scientific procedures
- automating data derivation processes
- high-performance computing (HPC) to improve throughput and performance
- Provenance management and query (persistence components).

Workflow Scheduling: Workflow scheduling is one of the key issues in the workflow management. The cloud computing environment is composed of services scattered on the Internet where the communication overheads can no longer be ignored. In this scenario, dynamic adaptation to network bandwidth for communication should be reflected, workflow scheduling came. It is the important module for the workflow management system that maps and manages the execution of inter-dependent tasks on the distributed resources. It allocates suitable resources to workflow tasks such that the execution can be completed to satisfy objective functions imposed by users. Proper scheduling can have significant impact on the

performance of the system. Example: In a bank cheque processing scenario, there are millions of concurrent cheque processing transactions per day, while each of them is a rather simple workflow instance with only a few steps. This cheque-processing procedure can be modeled as an instance-intensive workflow. If cheque-processing needs image transmission for authentication, it can be further modeled as instance-intensive workflows involving significant communication overhead. Moreover, if the cheque-processing procedure is executed on rented services within a given budget, they can be modeled as cost constrained instance-intensive workflows.

Fault Tolerance: Fault tolerance is a major concern to guarantee availability and reliability of critical services as well as application execution. In order to minimize failure impact on the system and application execution, failures should be anticipated and proactively handled. Fault tolerance techniques are used to predict these failures and take an appropriate action before failures actually occur.

2.4 Comparison of various Workflow tools in Cloud Computing

a) **UGENE [27]:** Unipro UGENE is a free cross-platform genome analysis suite that integrates number of biological tools and algorithms, provides both graphical user and command line interfaces. It is distributed under the terms of the GNU General Public License. It works on Windows, Mac OS X or Linux and requires only some clicks to install.

b) **Bonita Open Solution 281]:** Bonita is made of 3 foremost components: Bonita Studio which allows the user to design graphically the forms that will be exposed to the end user in order to act together with the process using standards and technologies such as XPDL or jBPM. It relies on Eclipse. Bonita BPM Engine is a JAVA API that allows you to interact programmatically with your processes. It is available under LGPL. It relies on Hibernate. Bonita User Experience is a portal that allows each end-user to manage all the tasks in a webmail-like interface in which he or she is involved. The portal also permits the owner of a process to administrate and get reports about processes. It relies on GWT.

c) **OrangeScape [29]:** It is browser based development environment and provides 4 design perspectives: Model design, Form design, Process design and Action design. OrangeScape Cloud uses GAE, offering a shared-everything or a shared-Processing

multitenancy models which is achieved by deploying the application into different Google app engine accounts. OrangeScape Enterprise runs on JEE application servers and supports standard databases including Oracle, Microsoft SQL Server, MySQL and IBM DB2.

d) **Google App Engine [30]:** Google App Engine, GAE is a platform as a service (PaaS) cloud computing platform used for developing and hosting web applications in Google-managed data centers. It is free up to a certain level of consumed resources. Fees are charged for additional storage, bandwidth, or instance hours required by the application. Currently, the supported programming languages are Python, Java and Go. Google App Engine supports many Java standards and frameworks. Core to this is the servlet2.5 technology using the open-source Jetty Web Server, along with accompanying technologies such as JSP.

e) **YAWL [31]:** Based on a rigorous analysis of existing workflow management systems and workflow languages, a new workflow language called YAWL (Yet Another Workflow Language) was developed by Wil van derAalst. This language was based on the one hand on Petri nets, a well-established concurrency theory with a graphical representation, and on the other hand on the well-known Workflow Patterns. It offers comprehensive support for the control-flow patterns and has a proper formal foundation.

f) **Oozie [32]:** Oozie is a Java Web-Application that runs in a Java servlet-container. Oozie workflow is a collection of actions arranged in a control dependency DAG (Direct Acyclic Graph), specifying a sequence of actions execution, specified in hPDL (a XML Process Definition Language). It also has server based Workflow Engine specialized in running workflow jobs with actions that run Hadoop Map/Reduce and Pig jobs. Oozie workflows contain control flow nodes and action nodes.

g) **Kaavo [33]:** It provides a framework to mechanize the deployment and run-time management of applications and workloads on multiple clouds (Infrastructure-as-a-Service Layer). It takes a top-down application-centric approach for deploying and managing applications in the cloud. One of the key innovations by Kaavo is the ability to capture the deployment and run-time management behavior of any complex application or workload in a single XML document.

Table 2.1: Comparison of various Workflow tools in Cloud Computing

Name of tool	OS	Language	Founder	Description	Architecture	Database	Companies
UGENE	Cross platform	C++, QtScript	Unipro	Integrates tools and algorithms	Client-server	NCBI, PDB, UniprotKB/Swiss-Prot	IBM
Bonita Open Solution	Cross platform	JAVA	French National Institute For Research in CS	Creates high-tech workflows and spreadsheets	Client-server	ERP, ECM	VENTECH
Orange Scape	JEE Application server	JAVA	Google	shared-Processing multitenancy models	Client-server	Oracle, MYSQL,IBM DB2	WIPRO
Google App Engine	Windows	Python, APIs, URL fetch	Google	Allows user to run web application	Client-server	Python, java	IBM, MICROSOFT
YAWL	Cross platform	XML Schema, XPath and XQuery.	Wil van der Aalst, Arthur ter Hofstede	based on the Petri nets and Workflow Patterns.	Client-server	MY SQL	GECKO
Oozie	Cross platform	Hpdl	Team of Yahoo Developers	Java Web-Application that runs in a Java servlet-container	Client-server	Apache Tomcat	Nintex
Kaavo	Cross platform	Java, PHP	JAMAL MAZHER	Application centric approach to the management of cloud	Application-centric	Amazon, Rackspace	IBM
Pegasus	LINUX , WINDOWS	JAVA, PERL, PYTHON	Ewa Deelman	translate complex computational tasks into workflows	Client-server	XML, JAVA	AMAZON EC2

h) **Pegasus [34]:** Pegasus consists of a set of components like The Pegasus Mapper, Execution Engine and Task Manager that run and manage workflow-based applications in different environment, including desktops, clusters, grids, and now clouds. It bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto distributed resources. It automatically locates the necessary input data and computational resources necessary for workflow execution.

2.5 Pegasus Workflow Management System

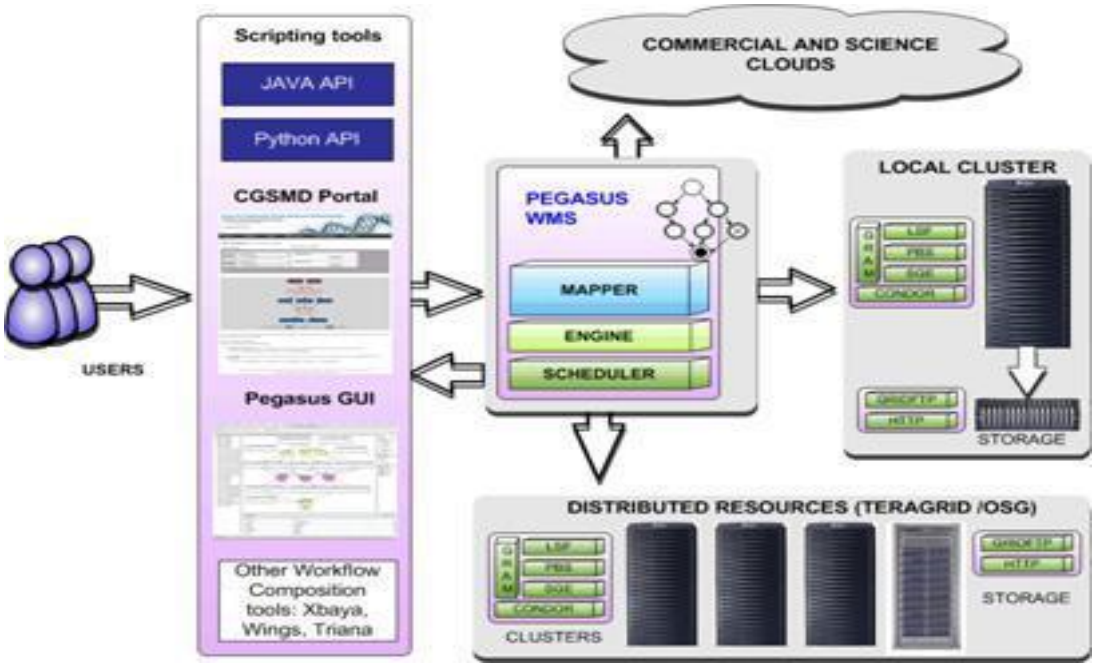


Figure 2.10[35]: Architecture of Pegasus

Pegasus WMS [36] which automatically converts abstract workflow descriptions into concrete execution plans containing resource-specific information. In order to generate an executable workflow, Pegasus analyzes the abstract workflow, adds auxiliary data transfer and cleanup jobs, performs workflow optimizations such as task clustering and workflow reduction, and generates resource-specific job descriptions. The concrete workflow is passed to DAGMan [37] and Condor [38] for execution on distributed resources. After the workflow finishes execution, this workflow can be deployed on its compatible cloud platforms like Nimbus [39], Eucalyptus [40], FutureGrid [41].

2.6. Oozie Workflow Engine

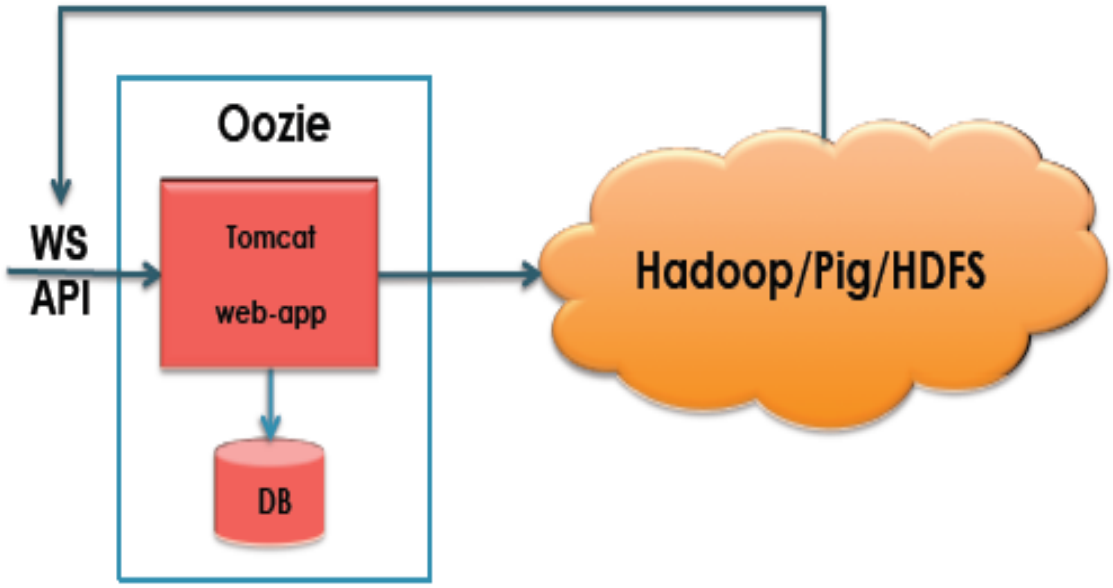


Figure 2.11[42]: Oozie Workflow Engine

Oozie[43] is a Java Web-Application based Workflow Engine specialized in running workflow jobs with actions (i.e. Hadoop Map/Reduce jobs, Pig jobs). A collection of actions defines workflow written in hPDL (a XML Process Definition Language), arranged in a control dependency DAG (Direct Acyclic Graph) where "control dependency" from one action to another means that the second action can't run until the first action has completed. This workflow actions start jobs in remote systems (i.e. Hadoop, Pig) which upon action completion, callback Oozie to notify the action completion. At this point Oozie proceeds to the next action in the workflow. Oozie workflows contain control flow nodes and action nodes.

- Control flow nodes define the beginning and the end of a workflow (start, end and fail nodes) and provide a mechanism to control the workflow execution path (decision, fork and join nodes).
- Action nodes are the mechanism by which a workflow triggers the execution of a computation/processing task.

2.7. Hadoop Cloud Environment

Hadoop Platform [44]: Hadoop is a framework written in Java for running applications on large clusters built of commodity hardware. It is composed of Hadoop distributed file system (HDFS), computational paradigm named Map/Reduce and distributed database –Hbase.

HDFS: HDFS is a highly fault-tolerant distributed file system, designed to be deployed on low-cost hardware that provides high throughput access to application data. The name node and the data node are two important parts of the storage foundation of distributed computing

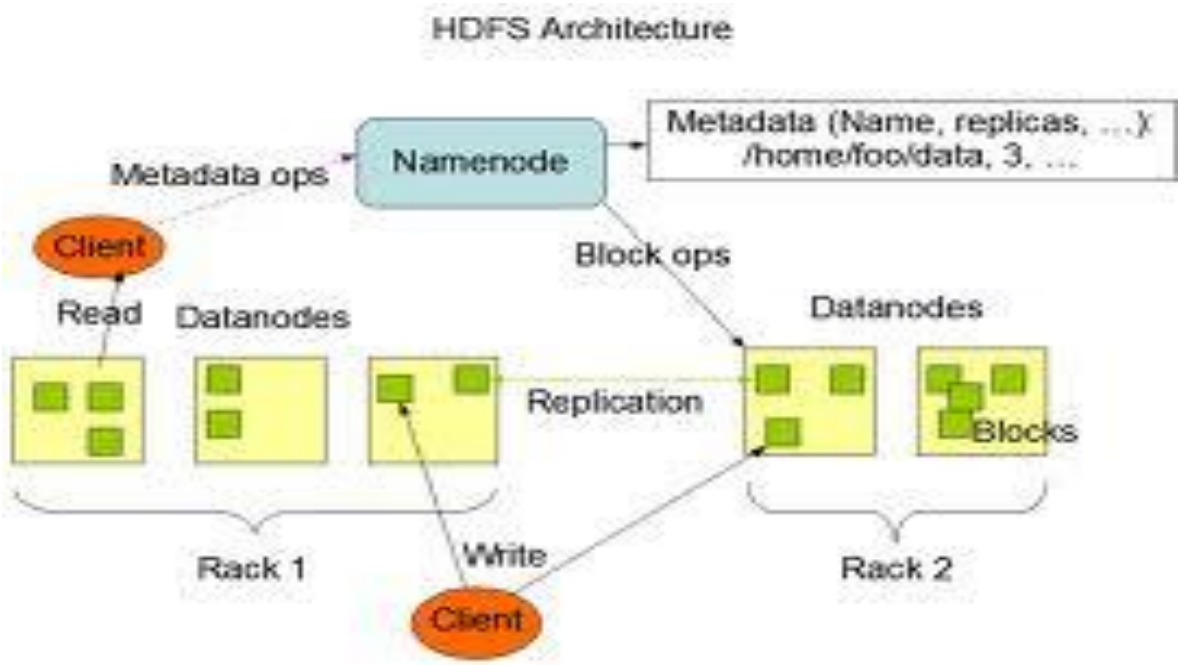


Figure 2.12[45]: HDFS Architecture

The name node, Manager of HDFS, manages the file system namespace and maintains the file system tree and the metadata for all the files and directories in the tree. Data nodes are the work horses of the file system which store and retrieve blocks when they are told to and they report back to the name node periodically with lists of blocks that they are storing.

Map/Reduce Model: it is a programming model which works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key-value pairs as input and output. Execute a Map / Reduce process requires five steps: an input file, a worker, the buffered pairs are written to local Disk, multiple Reduce worker Parallel execution, Output the final results. It has a simple model of data processing: inputs and outputs for the map and reduce

functions are key-value pairs. Users only need to specify the map and reduce functions to write parallel programs.

Distributed storage system-hbase: Hbase is a distributed storage system for managing structured data that is designed to scale to a very large size. HBase is a distributed column-oriented database built on top of HDFS. HBase tables are like those in an RDBMS, only cells are versioned, rows are sorted, and columns can be added. It is characterized with an HBase master node orchestrating a cluster of one or more region server slaves which reduce the relative importance of each node so do not worry about a single node failure and to ensure system reliability.

MultiNode Configuration: The multi-node cluster is built using two Ubuntu boxes. First install, configure and test a "local" Hadoop setup for each of the two Ubuntu boxes, and then "merge" these two single-node clusters into one multi-node cluster in which one Ubuntu box will become the designated master and the other box will become only a slave.

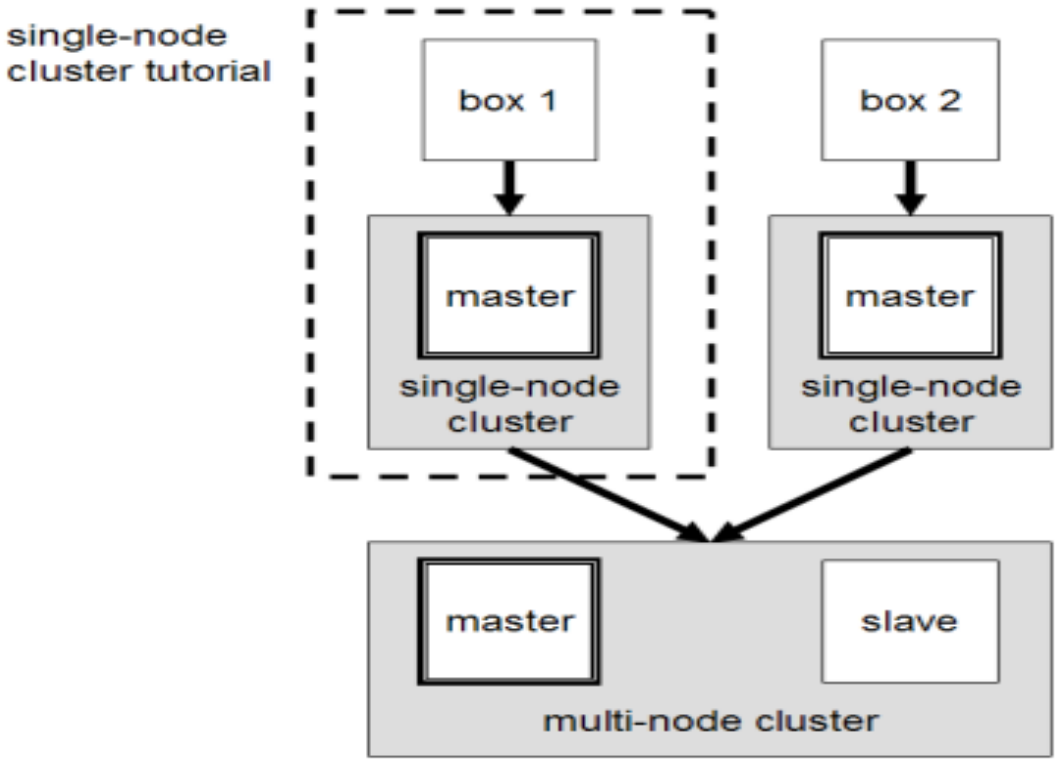


Figure 2.13[46]: MultiNode Cluster

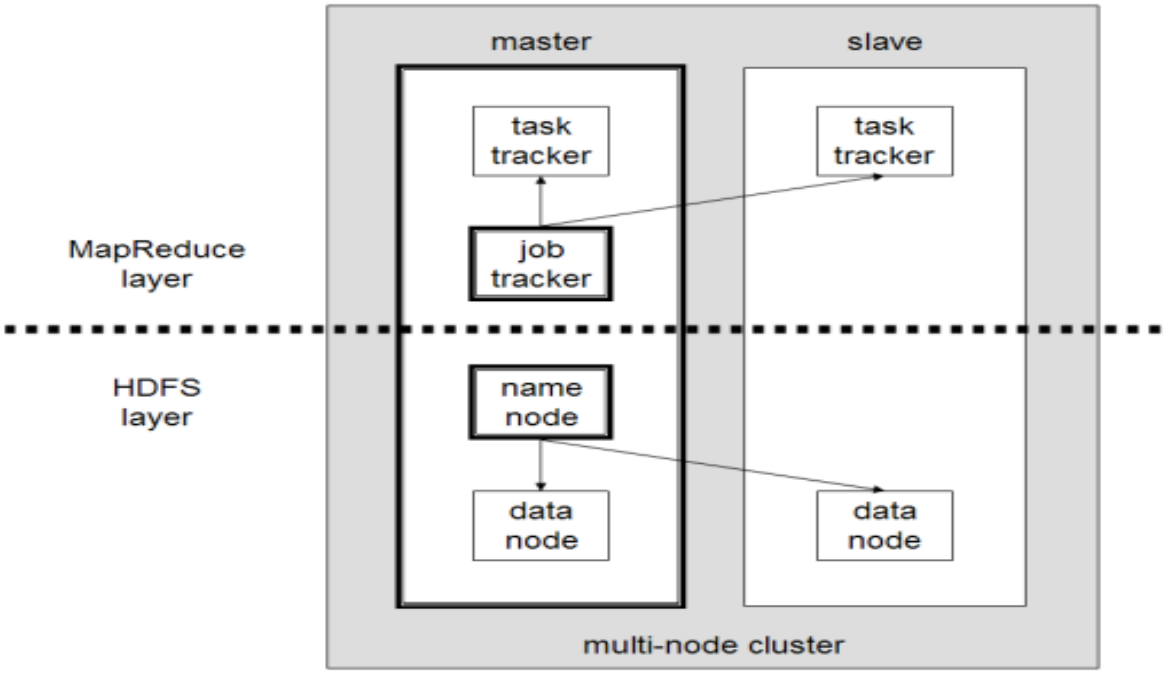


Figure 2.14[46]: Internal working of MultiNode Cluster

2.8. Different Workflow tools compatible with Hadoop

Hamake [47]: Hamake is a lightweight utility and workflow engine for Hadoop that helps you to organize your Hadoop map-reduce jobs or Pig scripts in a workflow that will be launched and controlled from a single machine. The order of execution of Hadoop map-reduce jobs or Pig scripts in hamake is based on dataflow programming model and is controlled by data, not by tasks themselves. It is very simple in installation and configuration. In fact all you need to install Hamake is to copy a single jar file to the directory of your choise.

Kepler [48]: it is a user-friendly open source scientific workflow system; users can easily utilize MapReduce in their domain-specific problems and connect them with other tasks in a workflow through a graphical user interface. Besides application composition and execution, the architecture also supports easy reuse and sharing of domain-specific MapReduce components through the Kepler infrastructure..

Azkaban [49]: Azkaban is a workflow scheduler that allows the independent pieces to be declaratively assembled into a single workflow, and for that workflow to be scheduled to run periodically. Other functionality available from Azkaban can then be declaratively layered on top of the job without having to add any code. This includes things like email notifications of success or failure, resource locking, retry on failure, log collection, historical job runtime information, and so on.

Cascading [50]: Cascading is an API for defining, sharing, and executing data processing workflows on a distributed data grid or cluster. Cascading relies on Apache Hadoop. To use Cascading, Hadoop must be installed locally for development and testing, and a Hadoop cluster must be deployed for production applications. Cascading greatly simplifies the complexities with Hadoop application development, job creation, and job scheduling.

Table 2.2: Different Workflow tools compatible with Hadoop

Name of tool	Workflow Description Languague	Requires Servlet/JSP container	Execution Model	Dependencies Mechanisms	Allows to run PIG Latin scripts	Advantages
Oozie	hpdl	Yes	Daemon	Explicit	Yes	Workflow scheduling , manage data processing jobs
Hamake	XML	No	Command line utility	Data-driven	Yes	Lightweight utility, organize Hadoop Map Reduce jobs, Pig script and local programs
Kepler	JAVA	Yes	Graphical user interface	Data - intensive	Yes	Provides a graphical user interface (GUI) for designing scientific workflows,
Azkaban	Text file	Yes	Daemon	Explicit	Yes	Simple batch scheduler for constructing and running Hadoop jobs or other offline processes
Cascading	JAVA API	No	API	explicit	Yes	Develop robust Data Analytics and Data Management applications.

Chapter 3

Problem Statement

3.1. Gap Analysis

In the beginning, workflows were being implemented in grids. The need to implement workflows in cloud was due to the reduced performance faced in grids. The application scalability is the prime benefit of moving from Grids to Clouds as it allows real-time provisioning of resources to meet application requirements. Hence, it enables workflow management systems to readily meet Quality of- Service (QoS) requirements of applications unlike traditional approach that required advance reservation of resources in global multi-user Grid environments.

Workflow applications often require very complex execution environments that are difficult to create on grid resources. Besides, each grid site has a different configuration, which outcome in extra effort each time an application needs to be ported to a new site. Virtual machines allow the application developer to create a fully customized, portable execution environment configured specifically for their application. There is need to design and implement workflows in various cloud platforms depending upon compatibility of workflow tools with cloud platform.

3.2. Need of Workflows in Cloud Computing

Cloud computing has grown in popularity with research community for deploying scientific applications such as workflows due to its advantages of cost-effectiveness, on-demand provision and easy for sharing. Due to continues growth , workflows are widely performed in collaborative cloud environments that consist of a number of datacenters, hence, an urgent need for exploiting strategies which can place the application data across globally distributed datacenters and schedule tasks according to the data layout to reduce both the latency and make span for workflow execution.

Workflows applications often very complex execution environments in grid computing that include specific operating systems, libraries, file system structures ,application programs, configuration files .So there is a need to designed and implement the workflows with different resources which are situated at different places of the world. Since it's difficult to maintain versions of application modules, one can use cloud to deploy its application. Cloud provides the skill to access shared resources and common infrastructure, offering services on demand over the network to perform operations. The end user is usually not aware of location of physical resources and devices being accessed. End user can develop, deploy and manage their applications 'on the cloud', which entails virtualization of resources.

There is a need to implement the workflows which can be analyzed, debugged, monitored and scheduled. These workflows can be designed and executed in any workflow engines. Fault tolerance, scheduling, load balancing, and energy efficiency and so on are the issues faced. These workflows can be deployed in their compatible cloud platforms. The various cloud platforms can be compared based on their characteristics.

Chapter 4

Design of the solution

This chapter discusses about how the problem stated in previous chapter can be solved with the help of Data Flow Diagrams and UML Diagrams.

4.1. Data Flow diagram and UML diagram for Pegasus

The solution to the problem has been designed through Data Flow Diagrams and UML Diagrams (use case, activity, sequence) for Pegasus.

4.1.1. Data flow diagram for working of Pegasus: The user input the workflow code to the Pegasus WMS where processing is done by the system using its own database and finally the html link is generated which can be copied to web browser and then graphs and plots can be viewed.

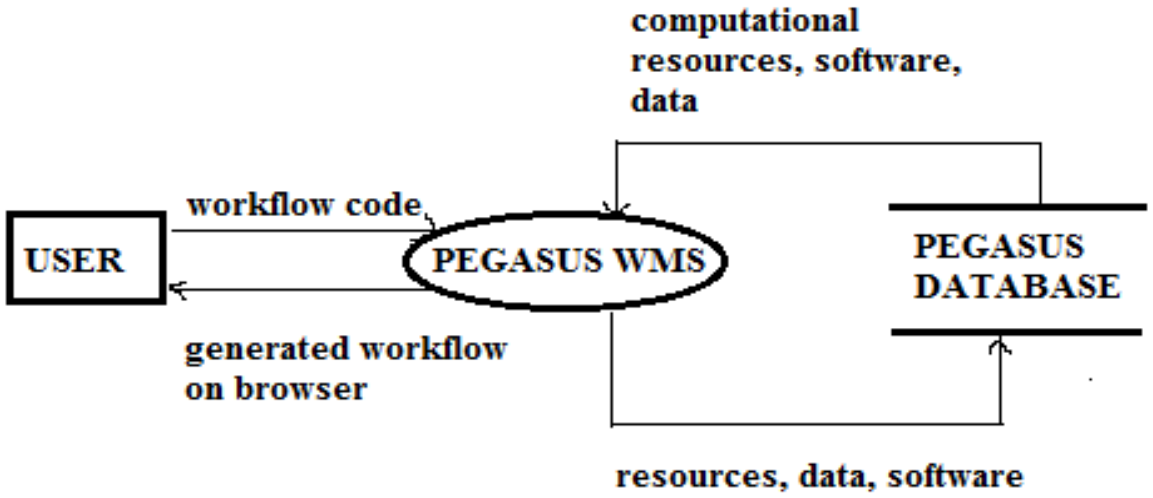


Figure 4.1: DFD of Pegasus

4.1.2. Use-case diagram for Pegasus: The workflow code given by user is submitted to Pegasus WMS. At the same time, user can choose the environment like cloud, grid,

desktop and campus cluster, where he can deploy its workflow depending upon Pegasus tool compatibility with the environment.

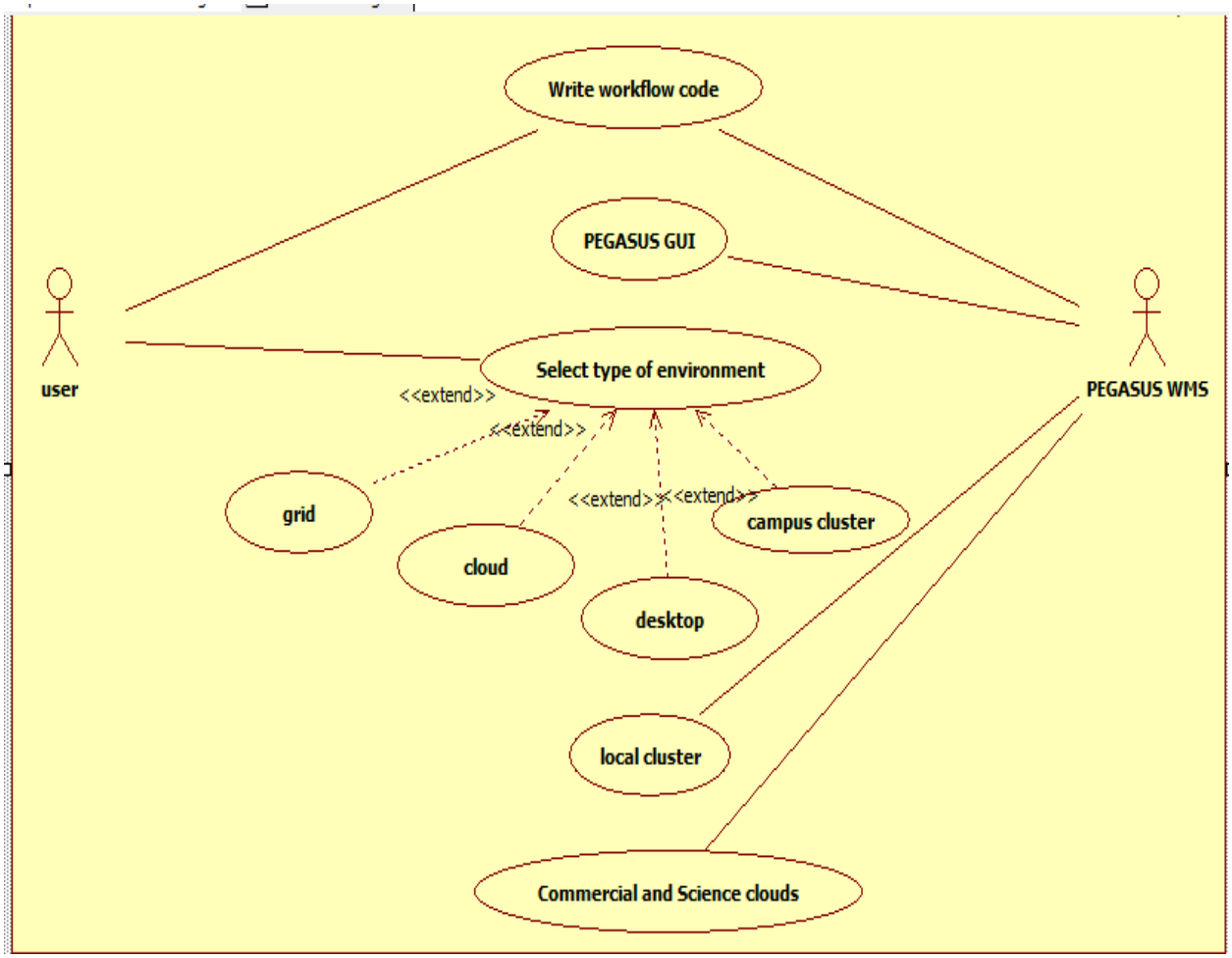


Figure 4.2: Use-case diagram of Pegasus

4.1.3. Design of Workflow: The design of the workflow is generated using Pegasus Workflow Management system. The design shows the various sections of Thapar University. It is broadly divided into school, academics and centres. Academics further tell about various departments of engineering like computer science, chemical, electronics communication, mechanical and biotechnology environmental and science. Central library comes under centres whereas L.M. Thapar School of management comes under school section.

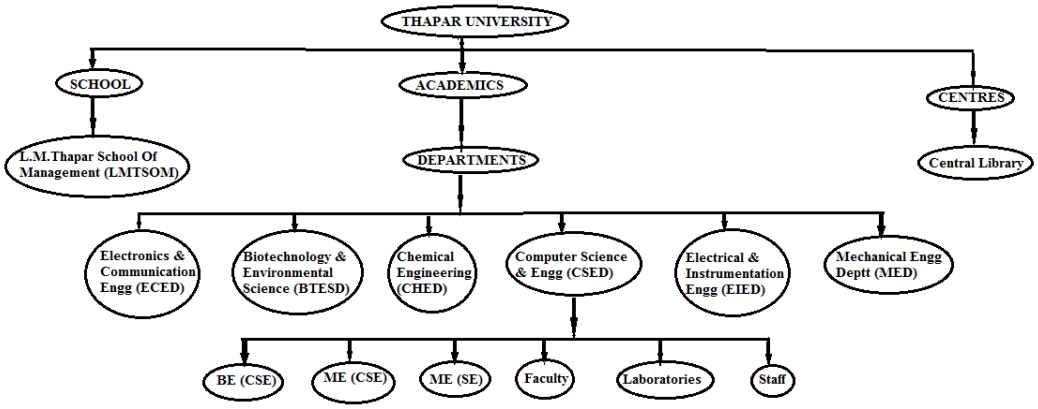


Fig 4.3: Design of Workflow

4.2. Data flow diagram and UML diagram for Hadoop

The solution to the problem has been designed through Data Flow Diagrams for the Hadoop environment and UML Diagrams (use case, activity, sequence).

4.2.1. Data flow diagram for Hadoop Environment: Once the workflow is submitted by the user, Oozie workflow engine that is installed on the Hadoop cloud platform processes it and generates the output as workflow. Hadoop has its database called HBase, from where it gets details related to all the nodes connected to it.

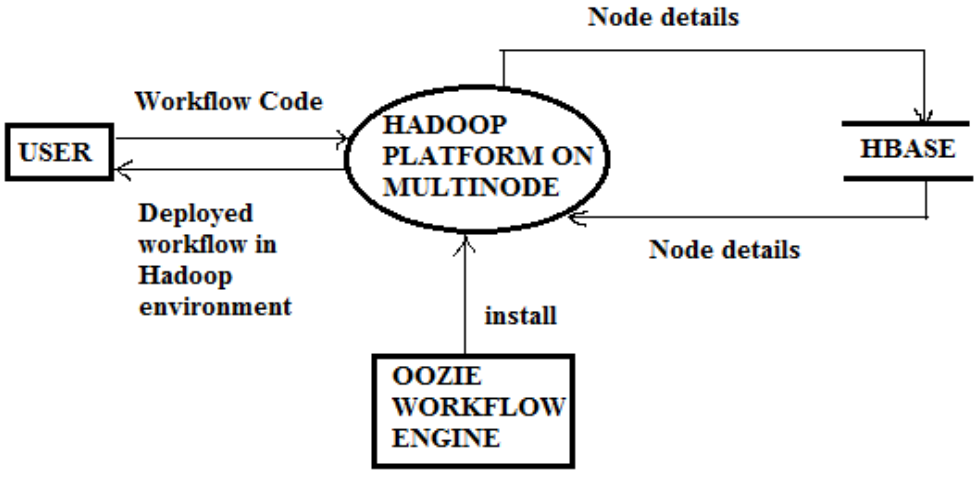


Figure 4.4: DFD of Hadoop Environment

4.2.2. Sequence diagram: The MultiNode Hadoop is installed on both machines. Now Oozie workflow is also installed on both the nodes. Once Oozie is installed, the user submits its code, processing is done on the master machine. Master machine has job tracker, which keeps track of jobs running on slave machine whereas slave machine has task tracker, by which slave machine can track jobs on master machine. Finally, workflow as output is generated and given to the user.

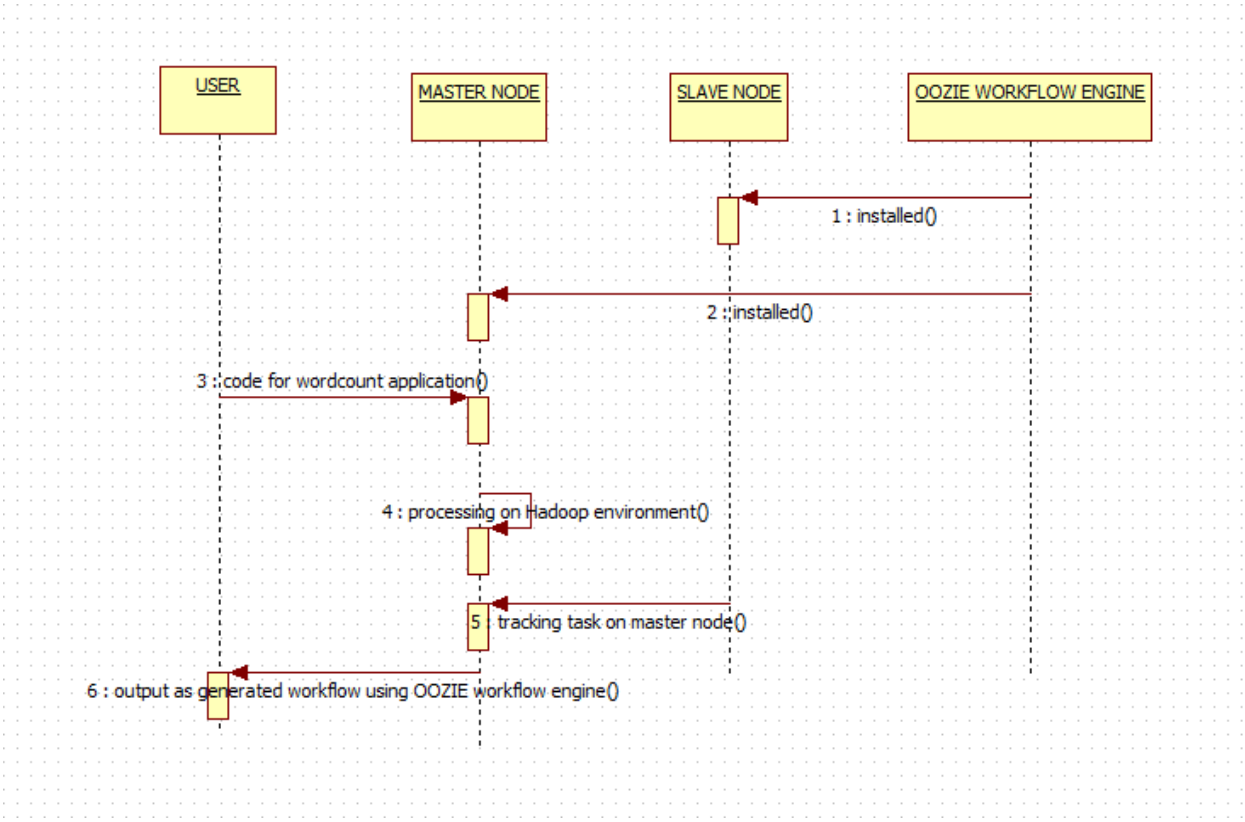


Figure 4.5: Sequence Diagram of working of Hadoop

4.2.3. Activity Diagram for Word Count Application: The code for the word count application is submitted by the user. As soon as the sentence enters by the user, it first counts the total number of words in the sentence. Then a word is entered and it is checked whether this word has occurred previously or not. If yes, word count is set to zero then compare that word with each word of the sentence. If word matches, word count increments to 1 else it goes to the next word of the sentence and follows the same procedure till the next word of the sentence becomes null.

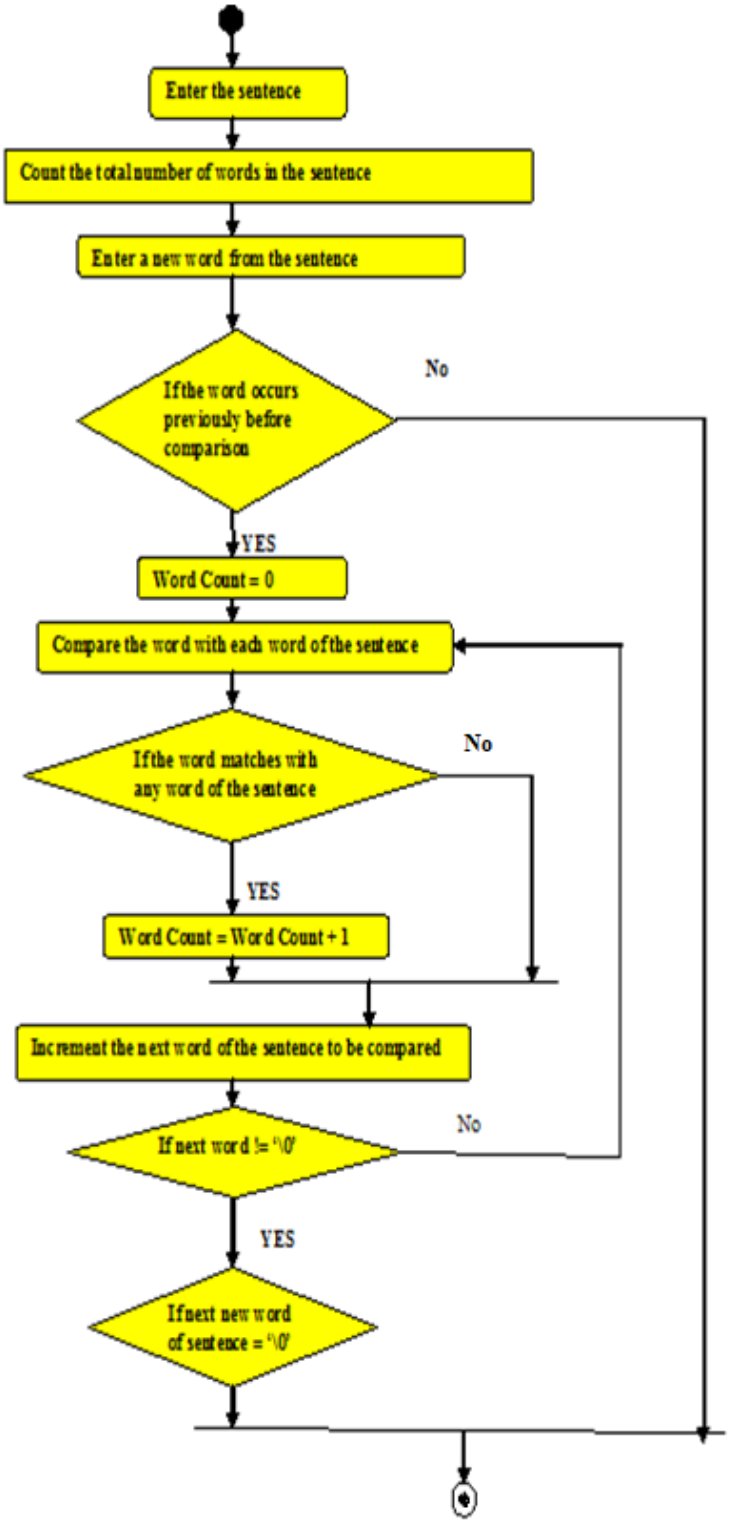


Figure 4.6: Activity diagram for Word Count Application

Chapter 5

Experimental Results

5.1. Generation of Workflow using Pegasus

Pegasus is a configurable system for mapping and executing abstract application workflows over a wide range of execution environment including a laptop, a campus cluster, a Grid, or a commercial or academic cloud. Today, Pegasus runs workflows on Amazon EC2, Nimbus, Open Science Grid, the TeraGrid, and many campus clusters.

5.1.1. Steps to install Pegasus: Pegasus can be installed on both windows and linux. For windows, Oracle Virtual Box is directly download and installed whereas for linux, its version e.g. Ubuntu 10.04 needs to be installed and on that machine, Oracle Virtual Box is downloaded and installed. Now, on Virtual Box, Pegasus virtual image needs to launch and its vmdk files are added to create virtual machine inside virtual box. We need to configure virtual machine by giving its name, operating system, version, base memory and finally virtual hard disk. Hence, virtual machine is installed.

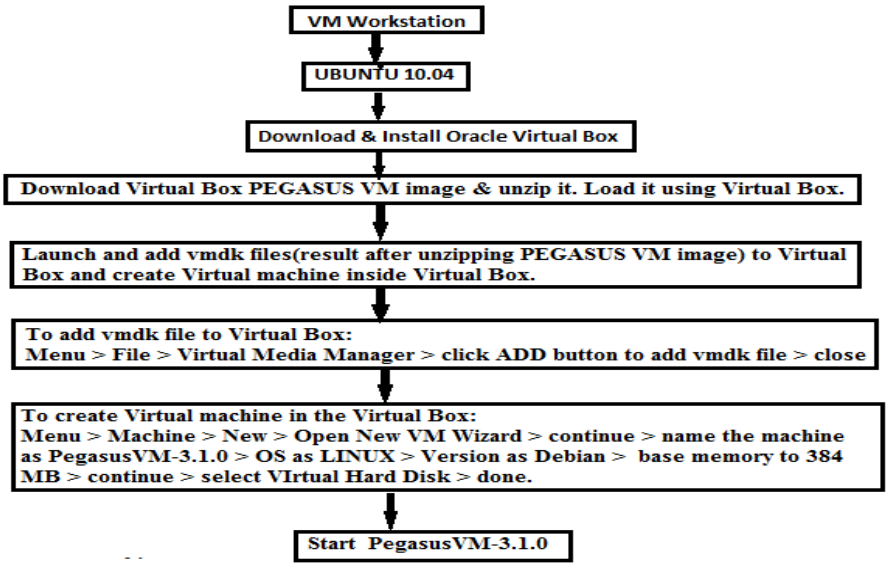


Figure 5.1: Pegasus installation steps

Here, virtual machine is configured and named as Rohit. Its operating system is chosen as Linux, version as Debian, base memory to 512MB, vmdk file is selected as virtual hard disk and click done. Fig 5.2 shows the installed virtual machine, rohit.

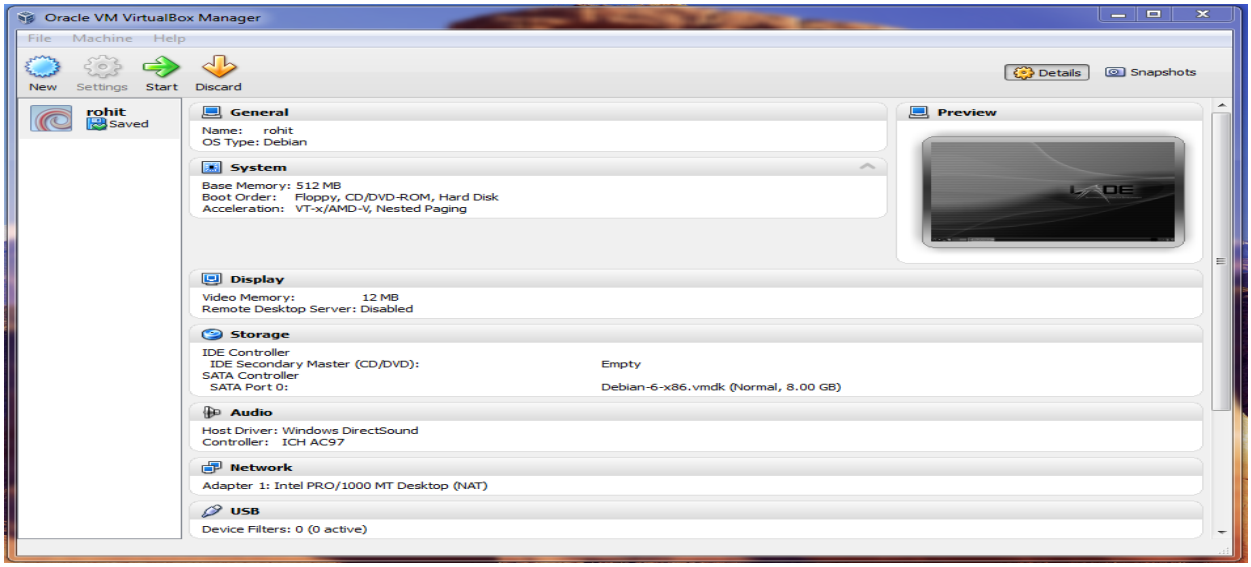


Figure 5.2: Pegasus virtual machine named as Rohit

5.1.2. LX terminal of Pegasus VM: Once the virtual machine, rohit is installed, its LX terminal can be opened and it appears like shown in fig 5.3.

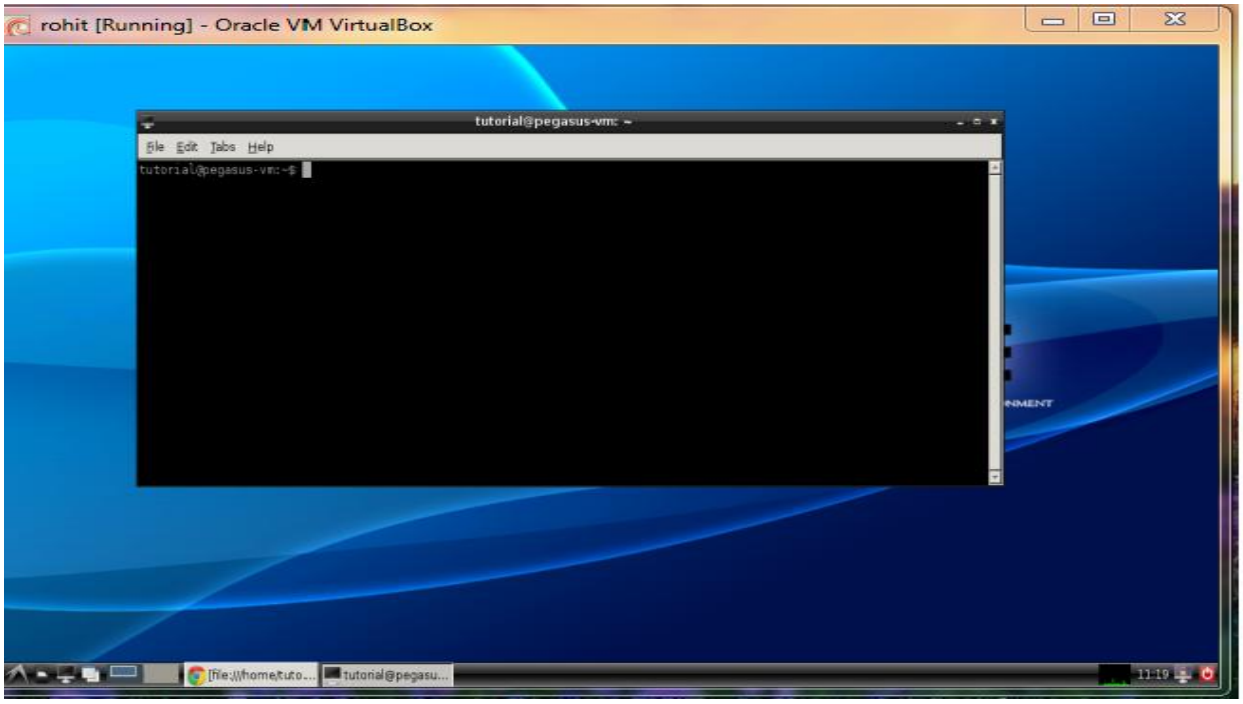


Figure 5.3: LX terminal of Rohit VM

5.1.3. The Grid Credentials: The Grid Credentials needs to be obtained to run workflows on the Grid. The VM already has a user certificate installed for the Pegasus user. To generate the proxy, run the grid-proxy-init command.

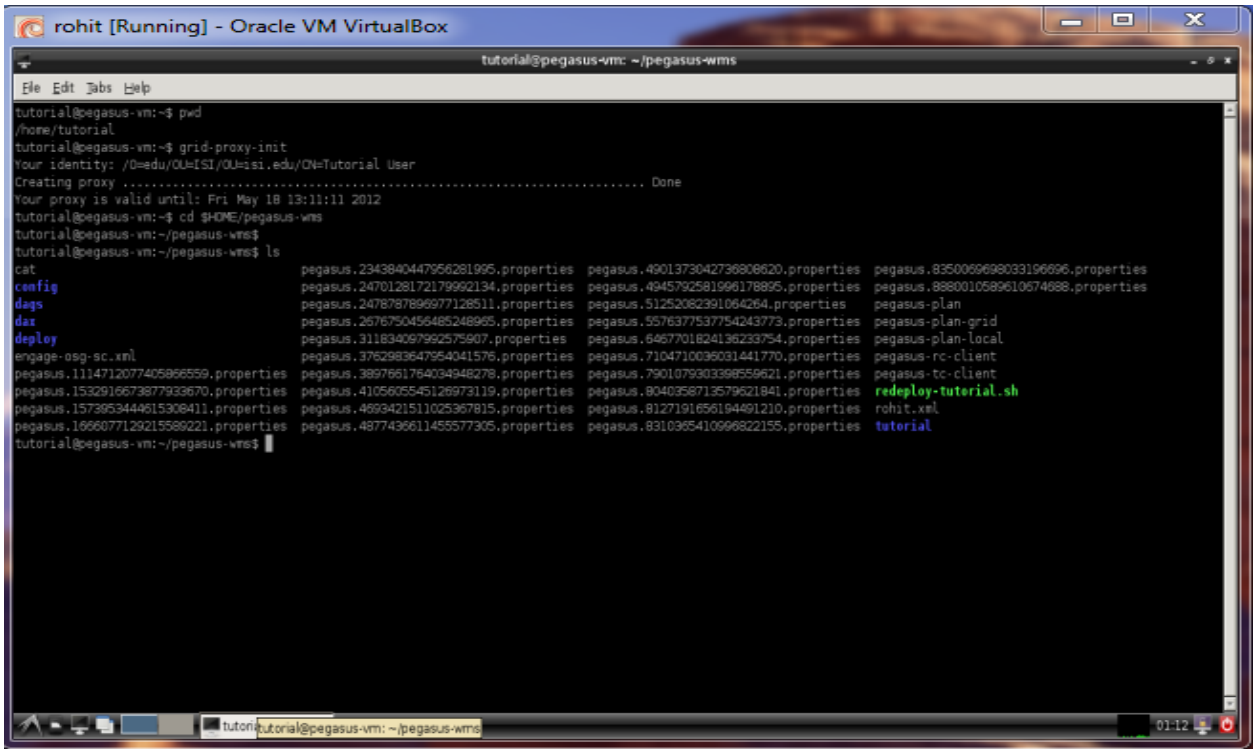


Figure 5.4: The Grid Credentials

5.1.4. Analysis of jobs: Pegasus-analyzer is a command-line utility for parsing several files in the workflow directory and summarizing useful information to the user. It should be used after the workflow has already finished execution. Pegasus-analyzer quickly goes through the jobstate.log file, and isolates jobs that did not complete successfully. It then parses their submit, and kickstart output files, printing to the user detailed information for helping the user debug what happened to his/her workflow. Once command is given, it will gives the details about the total number of jobs that are running, number of jobs that are succeded or failed and unsubmitted. Pegasus-analyzer also works on the failed workflow submit directory to see what job failed.

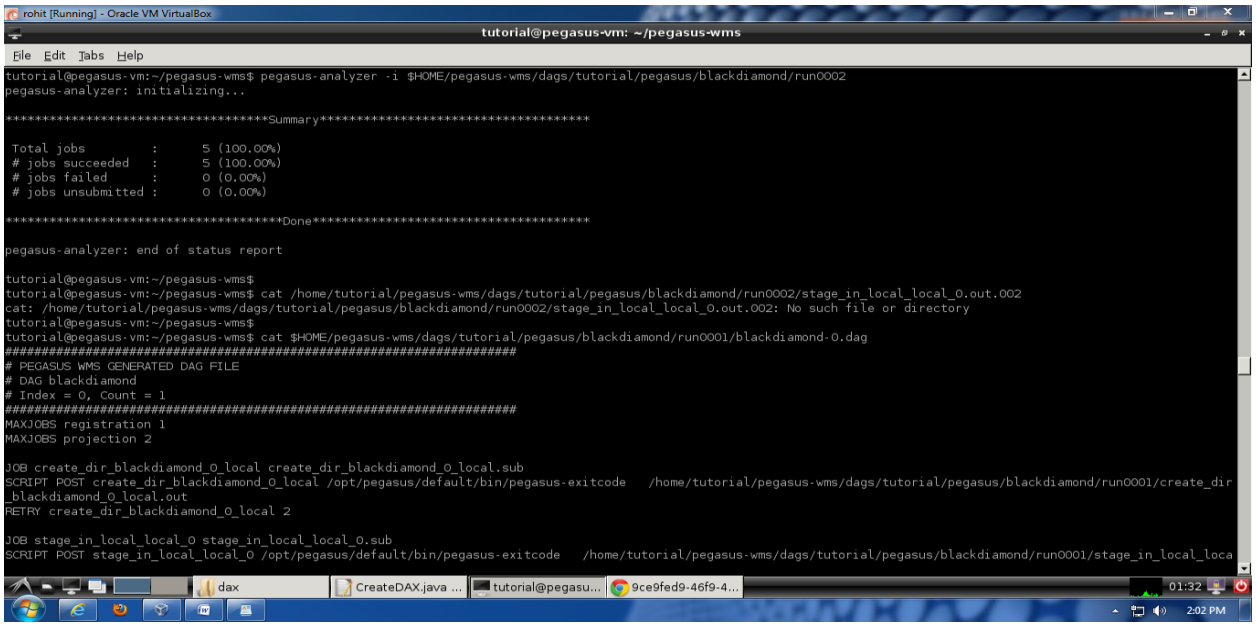


Figure 5.5: Analysis of jobs

5.1.5. Jobs monitoring and scheduling: Pegasus comes bundled with useful tools that help users debug workflows and generate useful statistics and plots about their workflow runs. These tools internally parse the Condor log files and have a similar interface.

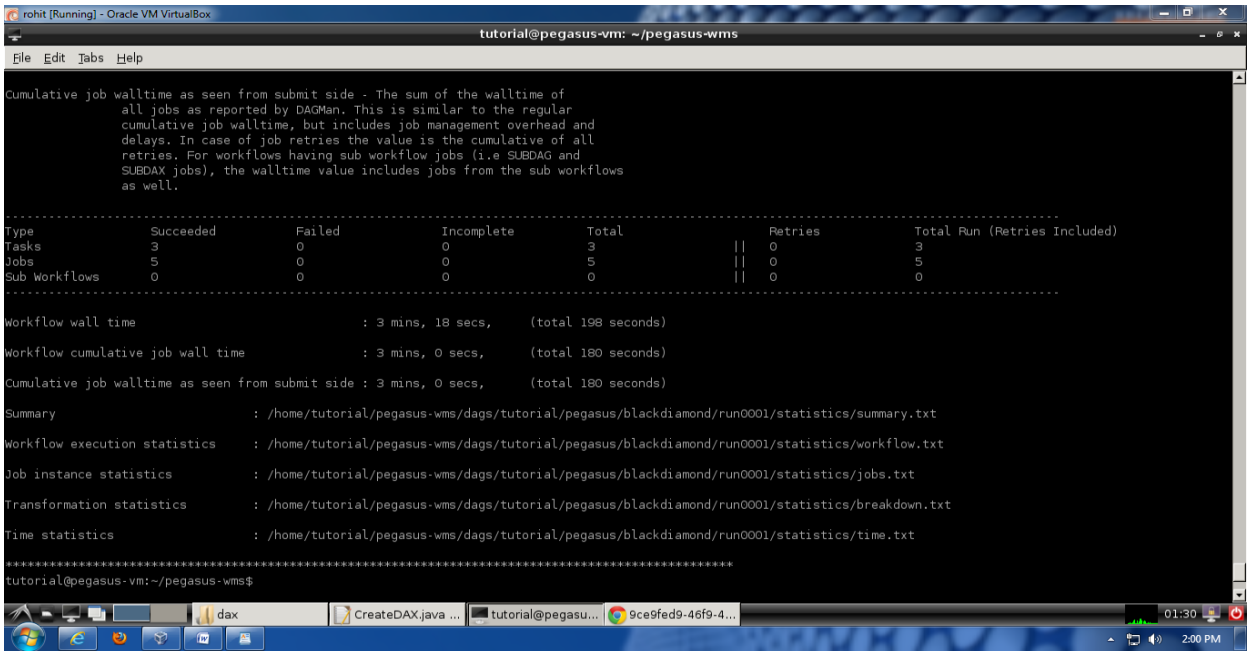


Figure 5.6: Jobs monitoring and scheduling

Pegasus-monitored is used to follow workflows, parsing the output of DAGMan's dagman.out file. In addition to generating the jobstate.log file, which contains the various states that a job goes through during the workflow execution, Pegasus-monitored can also be used to mine information from jobs' submit and output files, and either populate a database. It is automatically invoked by Pegasus-run, and tracks workflows in real-time.

5.1.6. HTML link: Graphs and charts generated by pegasus-plots can be viewed by opening the generated html file in the web browser.

Figure 5.7: HTML link

5.1.7. Pegasus plots: Pegasus-plots generate graphs and charts to visualize workflow execution. By default the output gets generated to plots folder inside the submit directory.

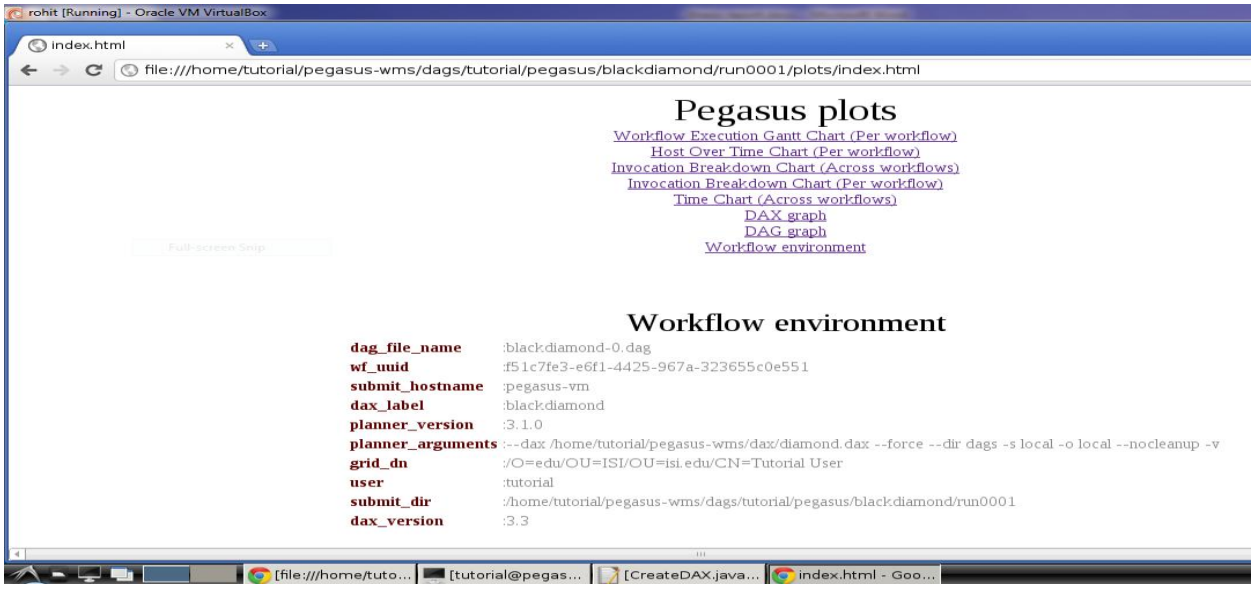


Figure 5.8: Pegasus Plots

5.1.8. DAX graph: The graph generated for the input given as shown in fig 5.9

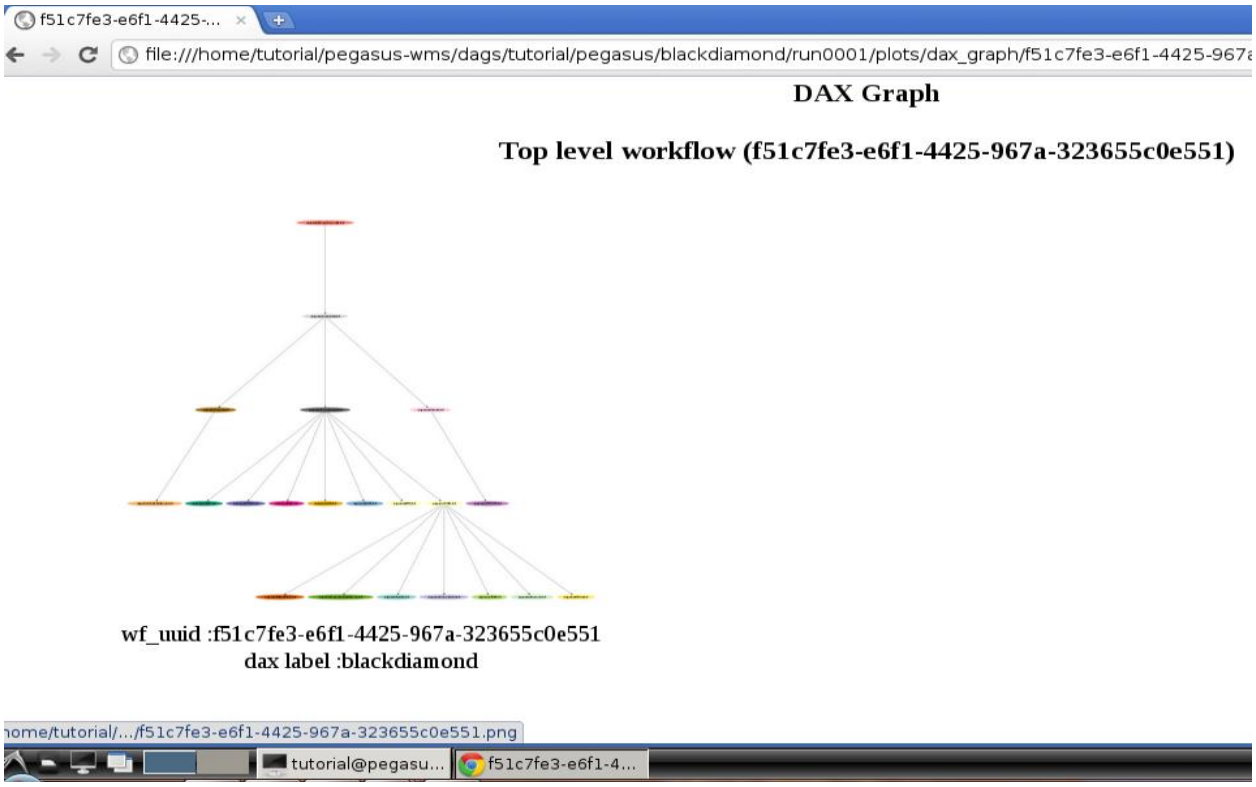


Figure 5.9: DAX graph

5.1.9. Nimbus Cloud: In order to get credentials for Nimbus Cloud via FutureGrid, it asks to have a valid portal account, member of valid project and SSH key submission.

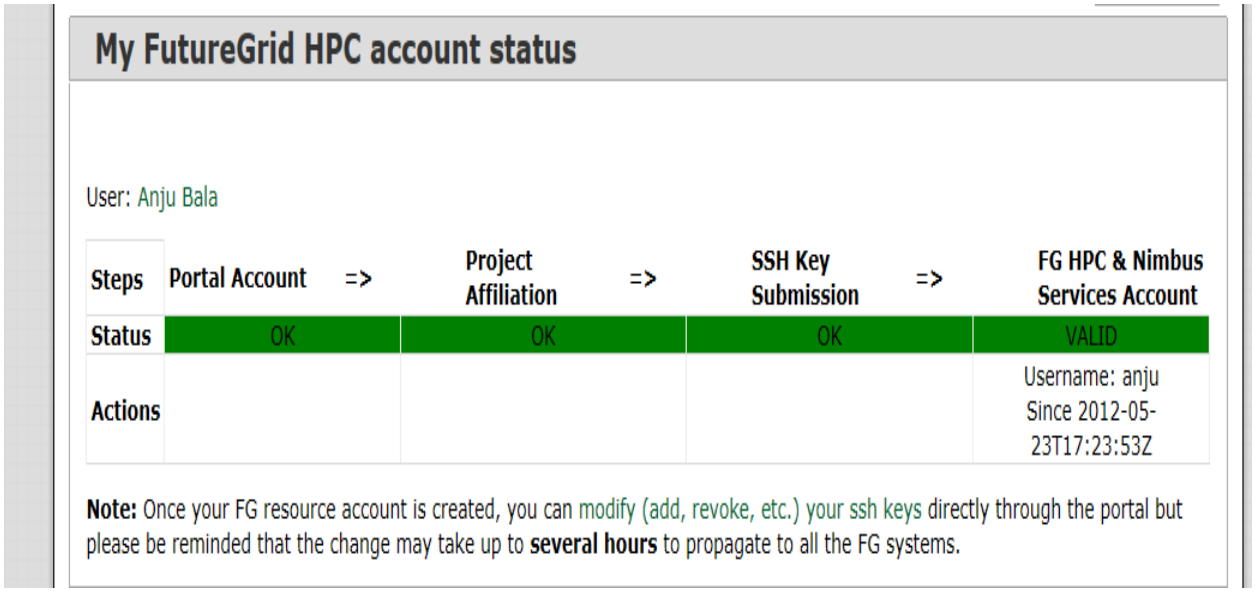


Figure 5.10: Nimbus Cloud

5.2. Generation of Workflow using Oozie

5.2.1. Installation of Hadoop MultiNode: Hadoop is a framework for running applications on large clusters built of commodity hardware. The Hadoop framework includes three cores: Hadoop distributed file system (HDFS) computational paradigm named Map/Reduce, distributed database –Hbase.

a) **SSH key generation:** Once command is given, public key-pair is generated which is saved to location as defined by user. The key fingerprint and key's randomart image is also generated and shown.

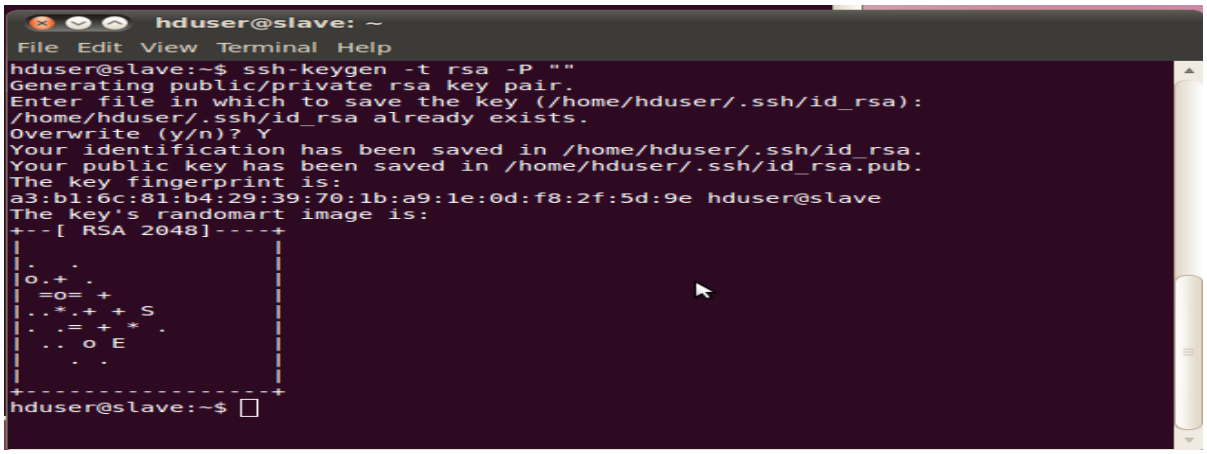


Figure 5.11: SSH key generation

b) **Hadoop extraction:** After downloading Hadoop from Apache Mirrors, it is extracted. The folders like bin, conf, lib, src and so on are extracted folders of the hadoop platform.

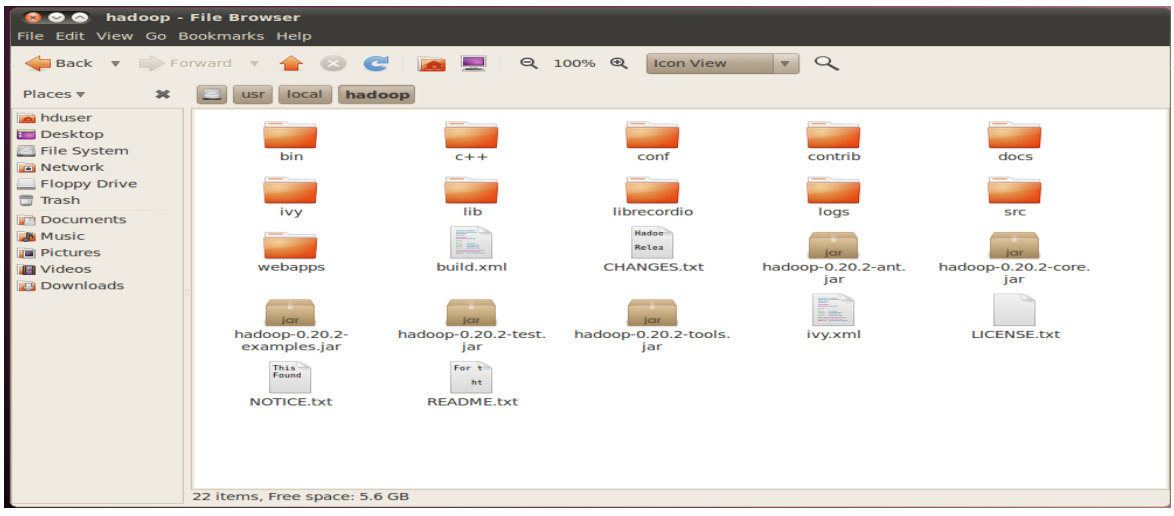


Figure 5.12: Hadoop extraction

c) **Hadoop installation:** Once Hadoop is extracted, it needs to install on the machine. Run the command – ls –l. The output is shown in Fig 5.12.

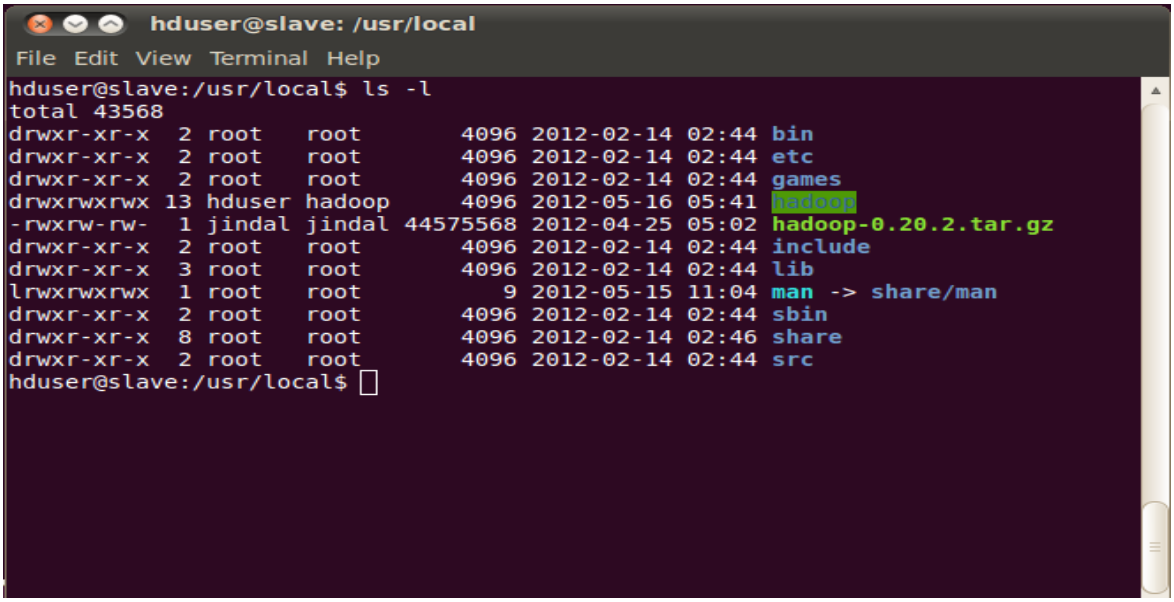


Figure 5.13: Hadoop installation

d) **Running Word Count Application:** The word count example is inbuilt application in Hadoop platform. Run the command on master machine as shown in Fig 5.13 and it will output as shown.

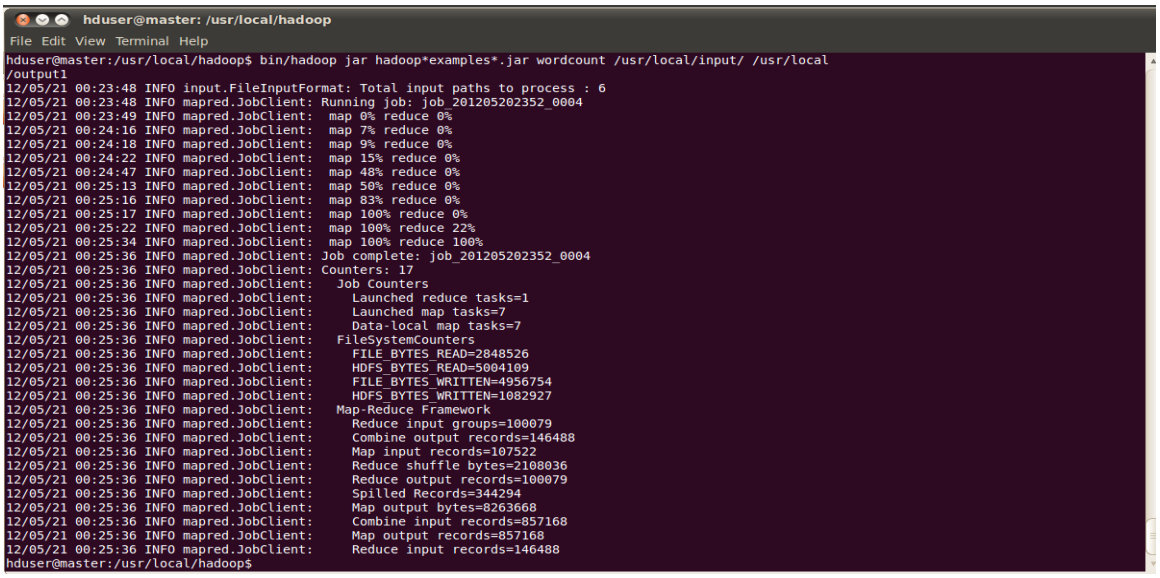


Figure 5.14: Running Word Count Example on master machine

e) **On slave machine for its data node:** Since slave node has task tracker, it can track word count job on master machine.

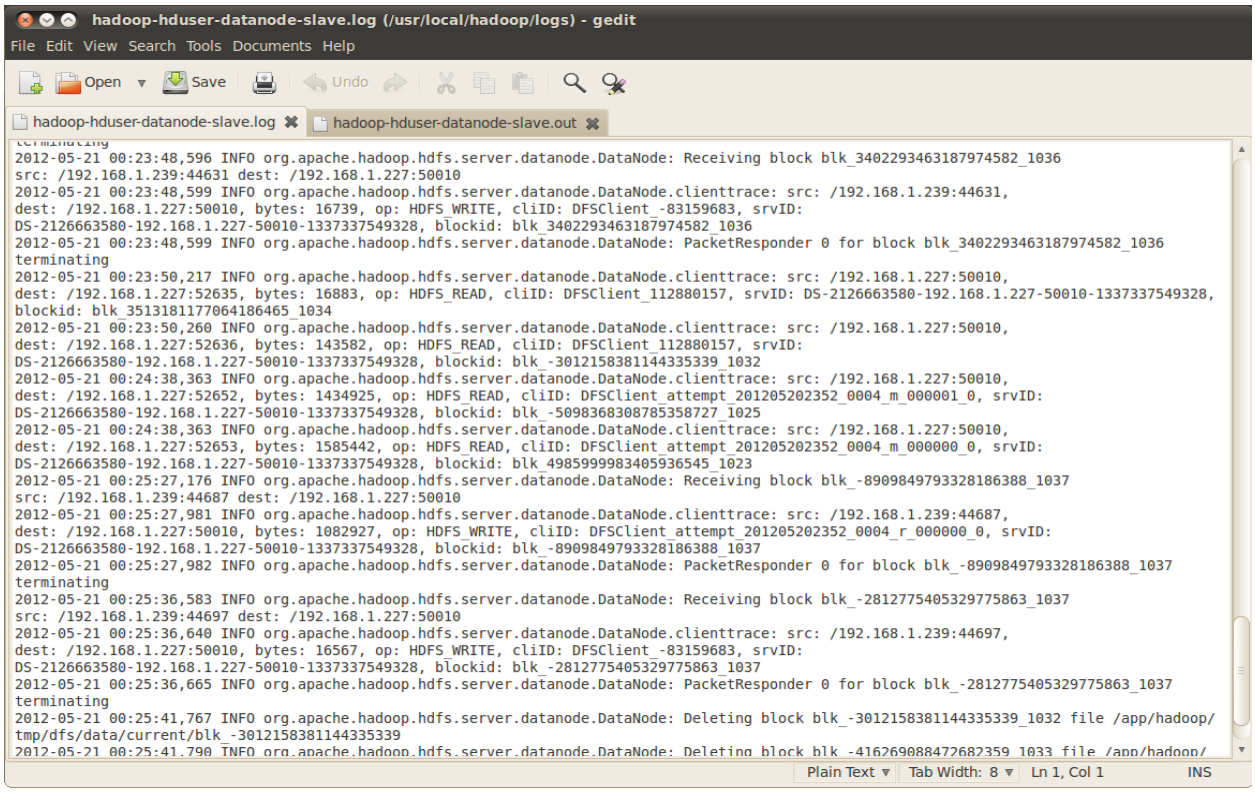


Figure 5.15: On slave machine

5.2.2. Oozie installation

System requirements:

- Unix
- Java 1.6+
- Apache Hadoop-0.20.2
- ExtJS library
- Download or build an Oozie binary distribution
- Download a Hadoop binary distribution
- Download ExtJS library

a)Download Oozie distribution tar.gz and untar it to the location /usr/local.

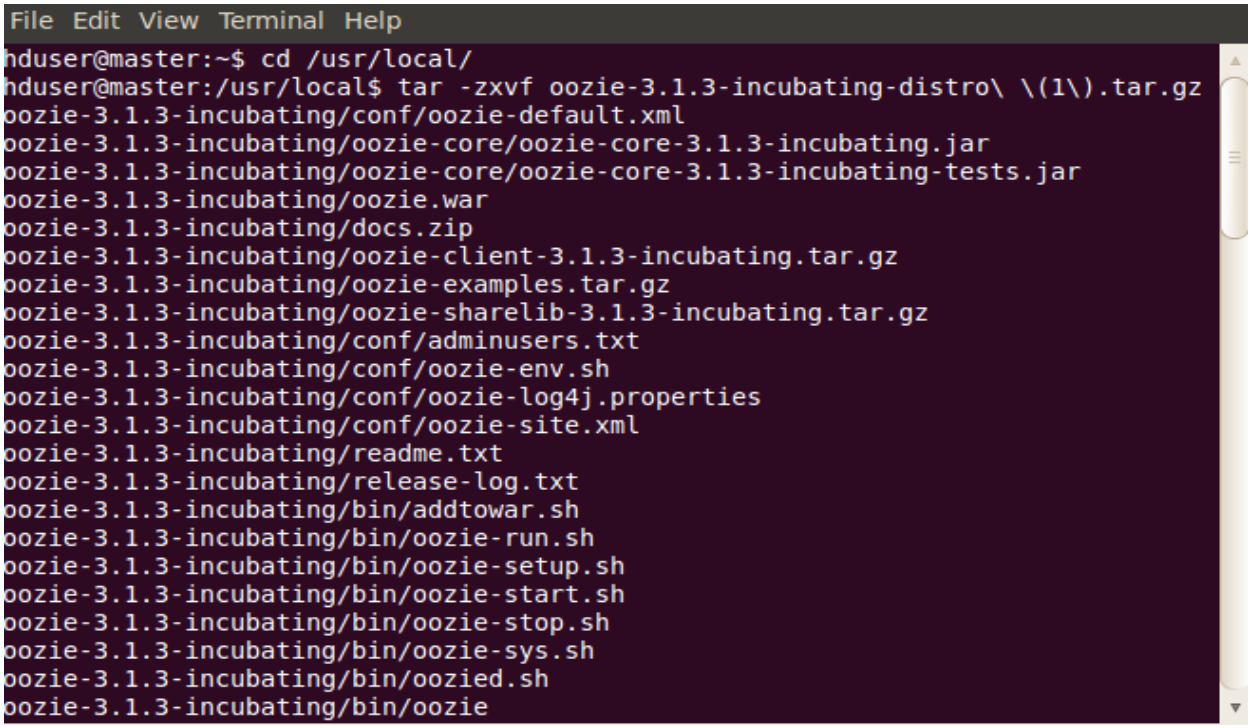


Figure 5.16: Oozie distribution tar.gz

b) Use the oozie-setup.sh script to add the Hadoop JARs and the ExtJS library to Oozie.

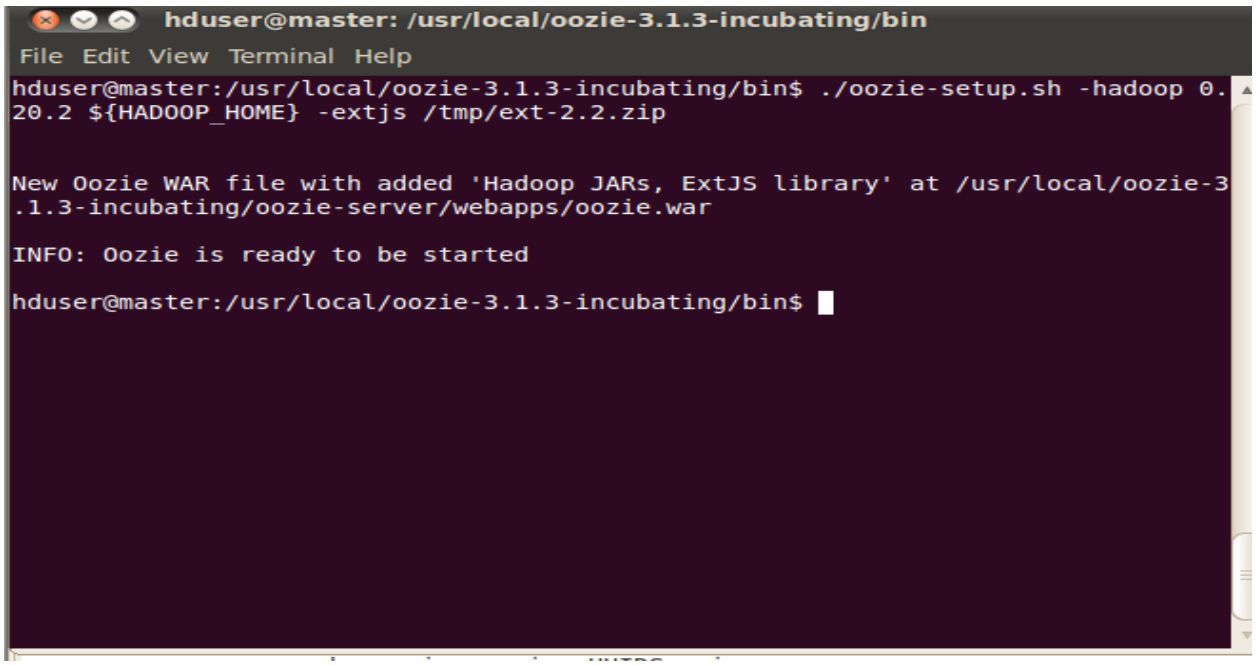


Figure 5.17: Add Hadoop JARs and ExtJS library

c) Starting Hadoop

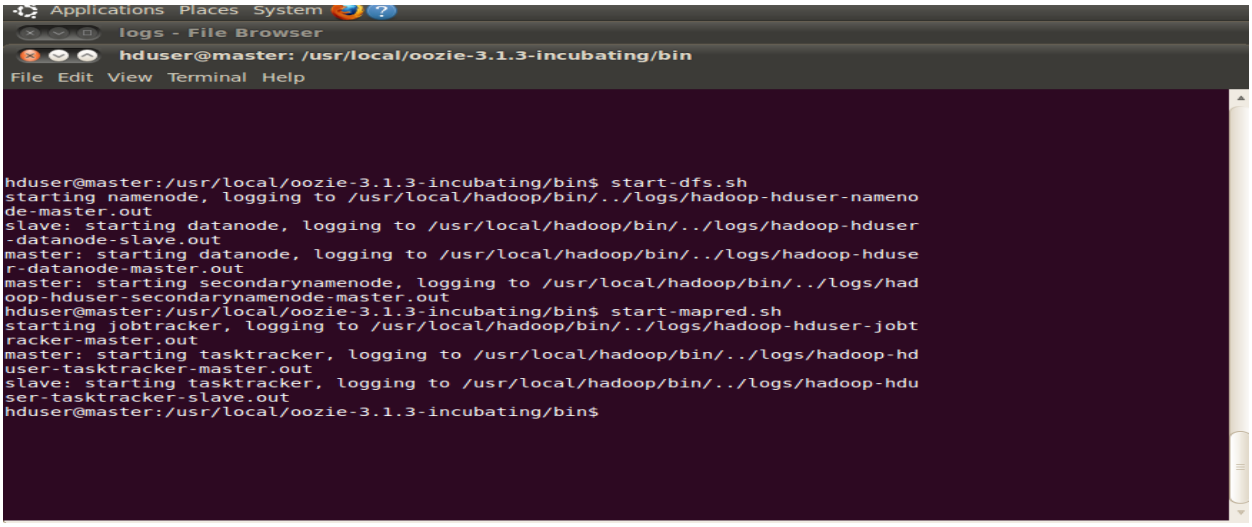


Figure 5.18: Starting Hadoop

d) Job properties

```
1  nameNode=hdfs://master:54310
2  jobTracker=master:54311
3  queueName=default
4  examplesRoot=mapReduce
5  oozie.wf.application.path=${nameNode}/usr/local/${examplesRoot}
6  inputDir=/usr/local/input1
7  outputDir=/usr/local/mapReduceOut
8
```

Figure 5.19: Job Properties

e) **Copy the workflow directory to HDFS**.

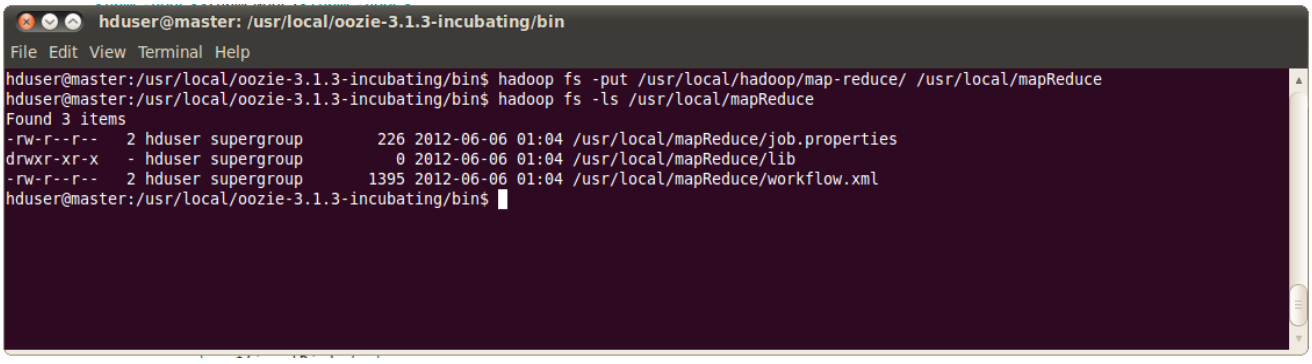


Figure 5.20: Copy the workflow directory to HDFS

f) Copy the input directory to HDFS

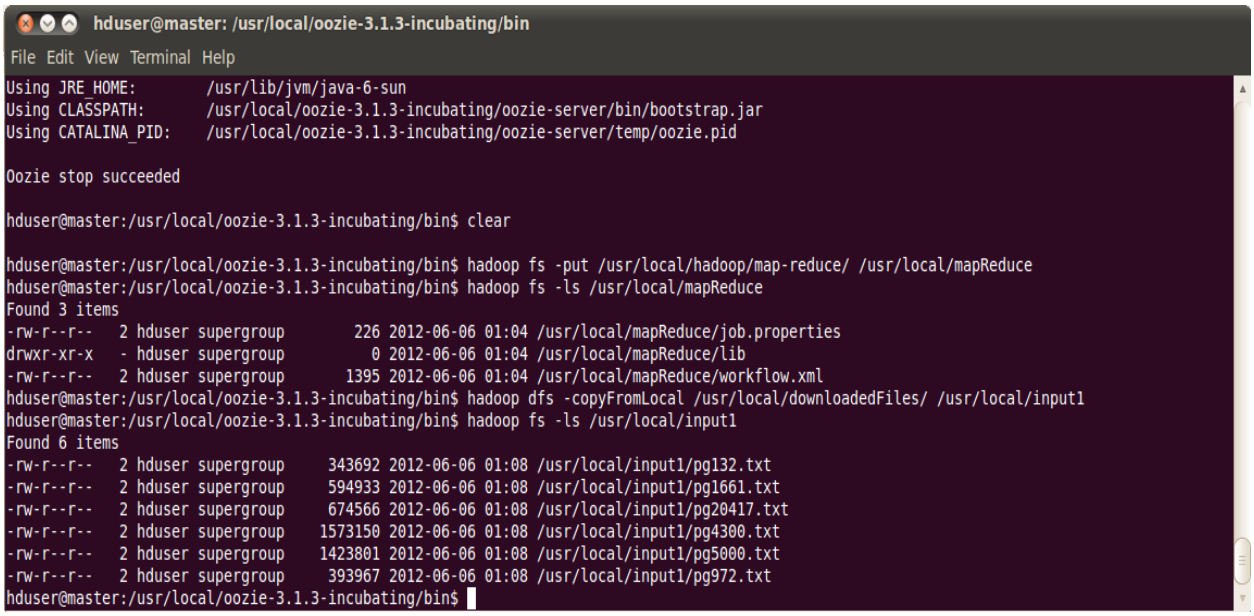


Figure 5.21: Copy input directory to HDFS

g) To start Oozie as a daemon process, run:

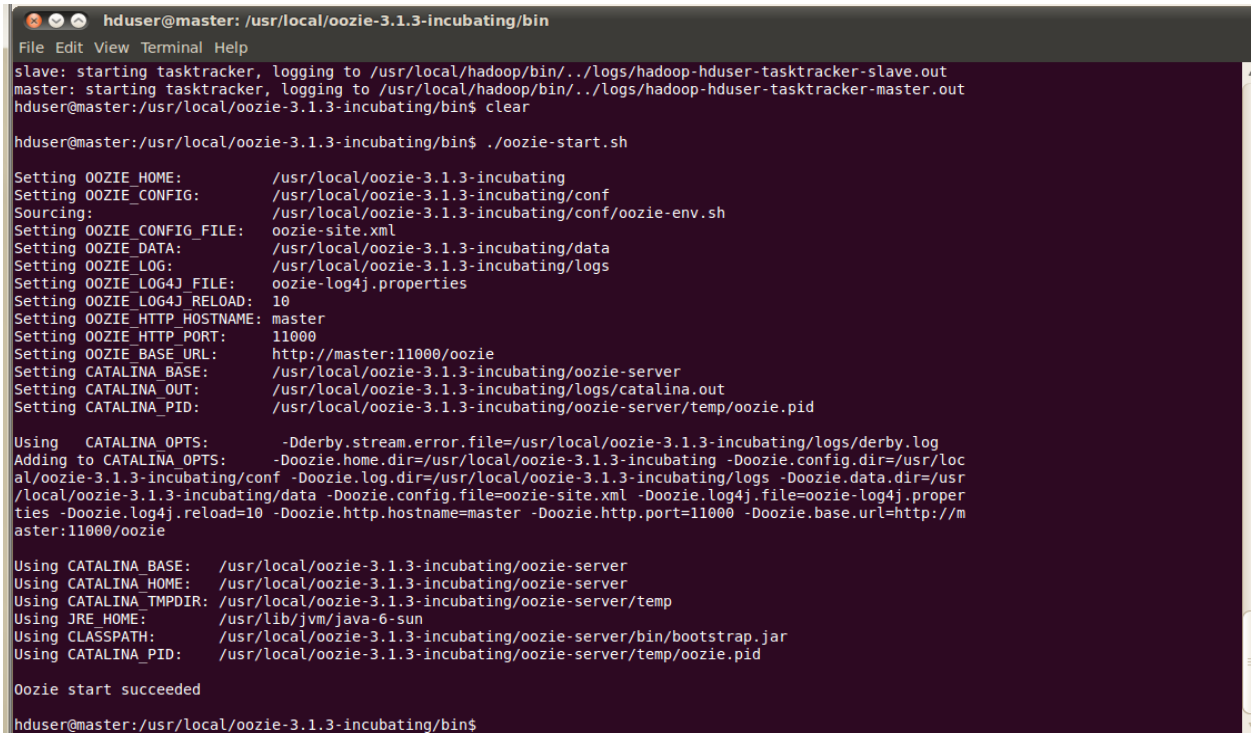


Figure 5.22: Start Oozie as a Daemon process

h) **Using the Oozie command line tool check the status of Oozie**

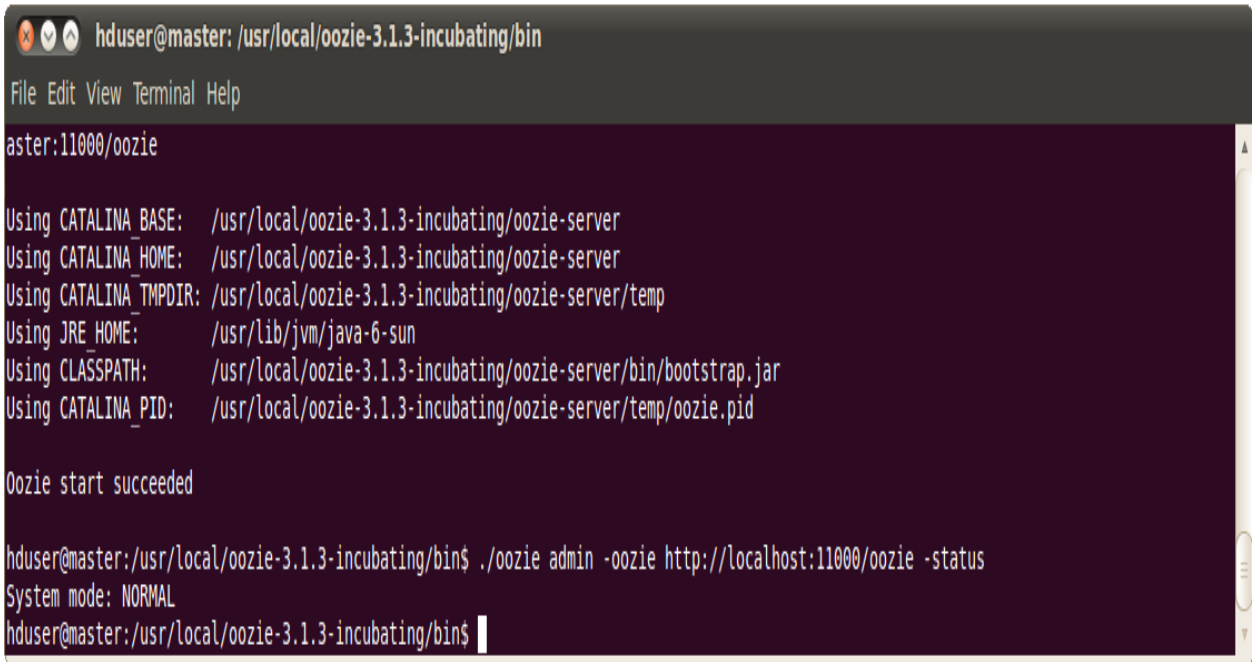


Figure 5.23: Checking status of Oozie

i) **Oozie web Console in browser**

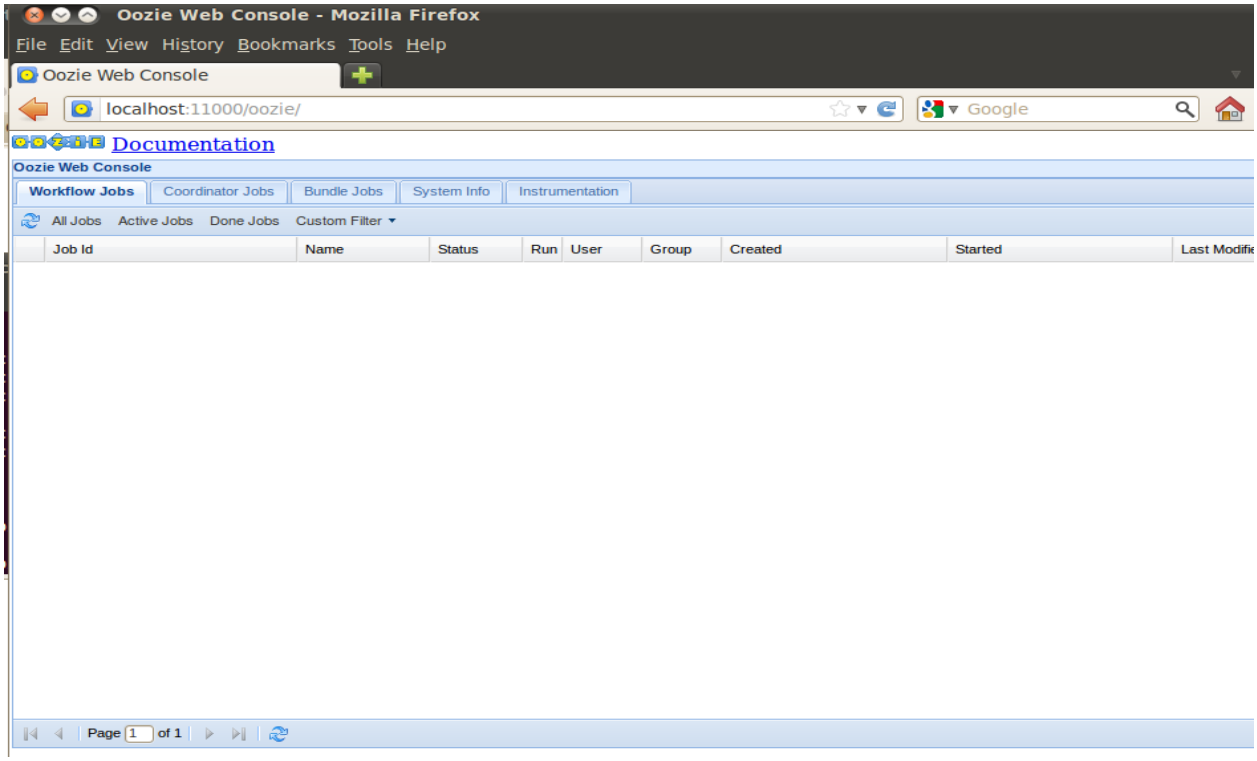


Figure 5.24: Oozie Web Console

j) Run the job and check status

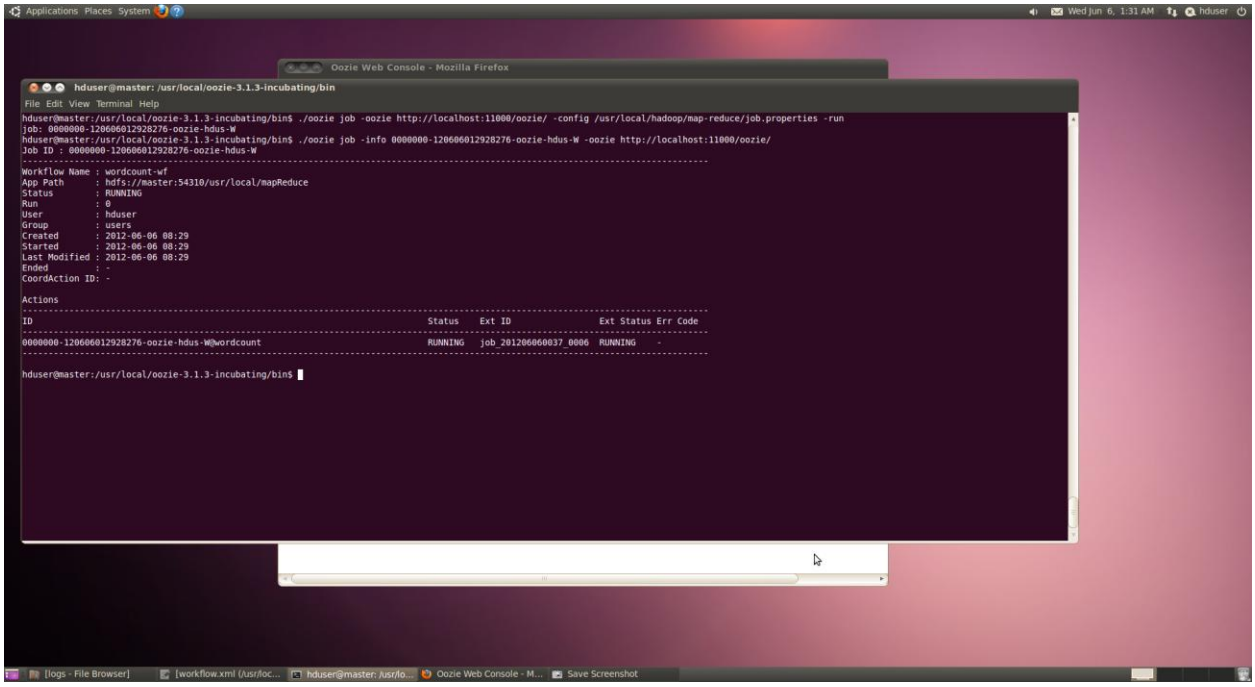


Figure 5.25: Running job and checking status

5.3 Comparison between Pegasus and Oozie: The comparison is done between Pegasus and Oozie on the basis of description, environment, language, mapper, reliability and so on.

Table 5.1: Comparison between Oozie and Pegasus

	OOZIE	PEGASUS
Description	Server based Workflow Engine	Workflow management system
Environment	Hadoop	Nimbus, Eucalyptus, Open Stack and Amazon EC2
Language	hpdl	XML, JAVA, PERL
Mapper	Launcher- Mapper	Pegasus Mapper
Initiation	Action node	Kickstart
Reliability	Unique callback URL	Check pointing, Pegasus-Analyzer.
Scheduling	No inbuilt scheduler.	Condor
Database	HSQLDB, MySQL or Oracle databases.	MySQL.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The Cloud computing is based on "a large pool of easily usable and accessible virtualized resources" that has promised many technological and sociological benefits. The workflow is designed and implemented on Pegasus workflow management system which can be deployed on cloud platforms like Nimbus. The workflow type application is also designed and implemented on Oozie workflow engine which is installed on Hadoop platform. Then a comparison is done between Pegasus and Oozie on the basis of reliability, scheduling, mapper and so on. On the basis of comparison, Pegasus workflow management system is found better as it has features to overcome the challenges like scheduling and fault tolerance.

6.2 Contribution to the thesis

- The workflow is generated using Pegasus workflow management system, which can be deployed on Nimbus cloud environment.
- The Hadoop platform has been installed on multimode using Ubuntu 10.04. Oozie, a workflow engine is also installed on existing Hadoop platform.
- The wordcount application has been designed in Oozie and deployed on Hadoop platform.
- Pegasus and Oozie have been compared. On the basis of comparison, Pegasus has been found better as it overcomes the challenges like scheduling and fault tolerance.

6.3 Future Scope

- The workflow application implemented on Pegasus workflow management system can be deployed on Nimbus cloud, which can be accessed through FutureGrid.
- The database can be managed using Pegasus
- The scheduling algorithms can be implemented on Hadoop.

- The fault tolerance techniques can be designed and further it can be implemented in Hadoop environment.

References

[1] Handbook of Cloud Computing, Furht, Borko; Escalante, Armando (Eds.), 1st ed., XIX, pp.634, 230 illus., Springer, 2010.

[2] http://cloudcomputingcompaniesnow.com/

[3] X. Wang,B. Wang and J. Huang,"Cloud Computing and its Key Techniques",in Proc. of IEEE International Conference,pp.404-410,June 10-12,2011.

[4] http://www.boingboing.net/2009/09/02/cloud-computing-skep.html

[5] N. Carr and F.Y. Yu, "IT is no longer important: the Internet great change of the high ground - Cloud Computing", CITIC Publishing House, October 2008.

[6] Ya-Qin Zhang, "The Future of Computing: 'Client + Cloud' ".September 4, 2008.

[7] B. Groubauer ,T. Walloschek and E. Stocker, "Understanding Cloud Computing Vulnerabilities", Security & Privacy IEEE, vol. 9,no.2,pp. 50-57,March-April,2011.

[8] N. Turner, "Cloud Computing: A Brief Summary", Lucid Communications Limited, Vers. 1.0, September 2009.

[9] "Cloud Computing:Silver Lining or Storm Ahead ?",The Newsletter for Information Assurance Technology Professionals,vol.13,no. 2,Spring 2010.

[10] http://www.mccandless.com/docs/McCandless-Jan-20-2010-CloudComputingHDI.pdf

[11] http://princetonacm.acm.org/downloads/Cloud_Computing.pdf

[12] http://www.opencloudmanifesto.org/opencloudmanifesto1.htm

[13] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing", vol. 34,no.3, pp.171-200, September 2005.

[14] R.P. Padhy, "Load Balancing in Cloud Computing Systems", B.Tech Thesis, NIT, Rourkela, Computer Science and Engineering Department, Rourkela, 2011.

[15] R. Buyya,J. Broberg and A. M. Goscinski, "Cloud Computing:Principles and Paradigms", pp. 664,February, 2011. ISBN: 978-0-470-88799-8.

[16] S. Haider et.al., "Fault Tolerance in Distributed Paradigms", in Proc. of Fifth International Conference on Computer Communication and Management, IACSIT Press, Singapore,2011.

[17] http://nsfcloud2011.cs.ucsb.edu/papers/Pan_Paper.pdf

[18] Q. Zhang,L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", J Internet Serv Appl ,vol.1,no.1 pp. 7–18 ,May 2010.

[19] D. Hollingsworth, "Workflow Management Coalition The Workflow Reference Model", Document Number TC00-1003, Document Status-Issue 1.1, 19th January, 1995.

[20] Indiana university extreme! Lab, "Introduction to Workflows", 7th October, 2003.

[21] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman, "Workflow Management in GriPhyN, The Grid Resource Management", Kluwer, The Netherlands, 2003.

[22] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi and M. Livny, "Pegasus: Mapping Scientific Workflow onto the Grid", in Proceedings of the Across Grids Conference , Nicosia, Cyprus, 2004.

[23] M. J. Pan, "Pomsets : Workflow Management for your cloud", 20th April, 2010, http://cloud.nchc.org.tw/20100420/slides/pomsets.pdf

[24] San Jose, "Adobe Workflow Lifecycle Overview ", 2005.

[25] Workflow Management Coalition, "Workflow Management Coalition Terminology & Glossary", February 1999.

[26] M. Kamath and K. Ramamritham,"Failure Handling and Coordinated Execution of Concurrent Workflows", in Proc. of the Fourteenth International Conference on Data Engineering, pp.334-341, February 23-27,1998.

[27] Unipro UGENE User Manual,Vers. 1.10, December 29, 2011.

[28] http://en.wikipedia.org/wiki/Bonita_Open_Solution

[29] http://en.wikipedia.org/wiki/OrangeScape

[30] http://en.wikipedia.org/wiki/Google_App_Engine

[31] YAWL-USER MANUAL, The YAWL Foundation, 2009.

[32] http://archive.cloudera.com/cdh/3/oozie/DG_Overview.html

[33] http://en.wikipedia.org/wiki/Kaavo

[34]Pegasus-user-guide.pdf,http://pegasus.isi.edu/wms/docs/3.1/pegasus-user-guide.pdf

[35] http://astrocompute.wordpress.com/2011/12/18/the-architecture-of-the-pegasus-workflow-manager/

[36] J.S. Vöckler, G. Juve, E. Deelman, M. Rynge and G. B. Berriman, "Experiences Using Cloud Computing for A Scientific Workflow Application", ScienceCloud'11, June 8, 2011, San Jose, California, USA.

[37] G. Mehta, K. Vahi and Kent Wenger, "Pegasus and DAGMan Generating and running workflows on the Grid",pegasus.isi.edu/tutorial/tg07/PegasusDAGManTG07.pdf

[38] D. Thain and C. Moretti, "Abstractions for Cloud Computing with Condor", CRC Press / Taylor and Francis Books, 2009.

[39] K. Keahey, "Cloud Computing with Nimbus", XtreemOS Summer School 2009 Oxford, September 2009.

[40] D. Johnson, K. Murari, R. Raju, R.B. Suseendran,Y. Girikumar, "Eucalyptus Beginner's Guide – UEC Edition (Ubuntu Server 10.04 - Lucid Lynx)", Vers. v1.0, 25th May 2010.

[41] https://portal.futuregrid.org/tutorials/nimbus

[42] Oozie Specification, a Hadoop Workflow System (PROPOSAL v1.0-2009MAY21).

[43]http://archive.cloudera.com/cdh/3/oozie-2.3.0cdh3u0/WorkflowFunctionalSpec.html

[44] MA Juan, LU Jian-jun, "The Architecture of Tender and Bidding System of Enterprisesbased on Hadoop Cloud Platform", in Proc. of International Conference on Electric Information and Control Engineering,vol.1,pp.6234,2011.

[45] http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

[46] http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

[47] http://code.google.com/p/hamake/wiki/HamakeManual

[48] B. Ludäscher et.al. "Scientific Workflow Management and the Kepler System", Concurrency and Computation: Practice and Experience, vol.18, no.10, pp.1039-1065, 2006.

[49] http://twit88.com/blog/2011/05/27/hadoop-batch-job-scheduler/

[50] "Cascading - User Guide Concurrent", Inc, Vers. 1.0 Copyright © 2007-2009 Concurrent, Inc Published August, 2009.

Appendix

Installing Ubuntu 10.04 on VM Workstation: Ubuntu 10.04 is Debian GNU/LINUX distribution which is served as free and open source software. Ubuntu 10.04 is installed on two machines using iso image, ubuntu-10.04.1-desktop-i386.iso, downloaded from Google and installed on workstation.

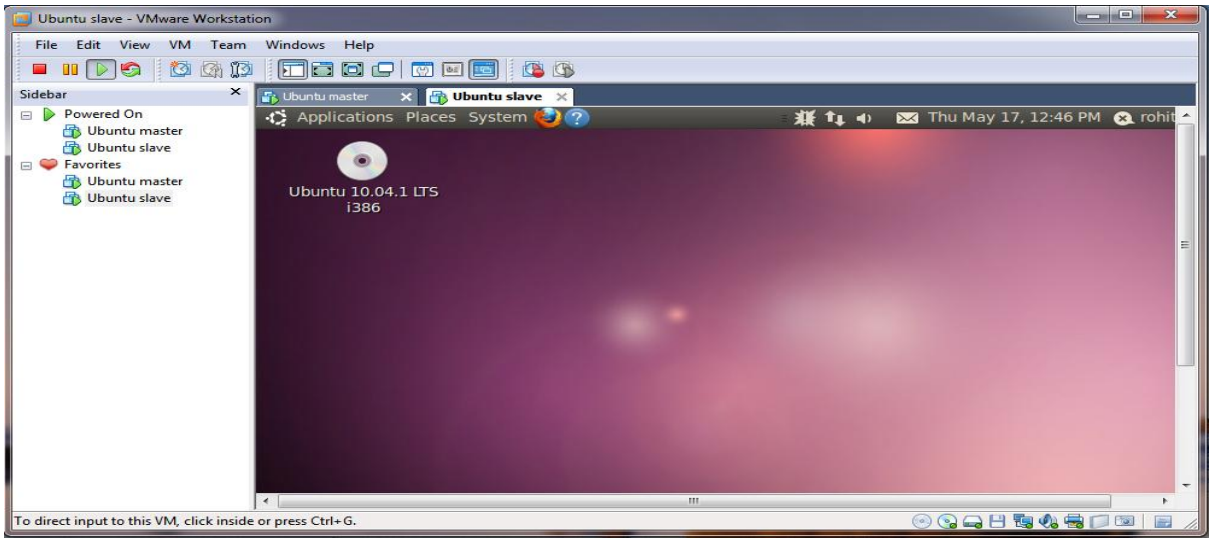


Figure 1: Ubuntu 10.04 Machine

JDK Installation: Hadoop requires a working Java 1.5.x or higher. JDK can be installed by executing the following commands.

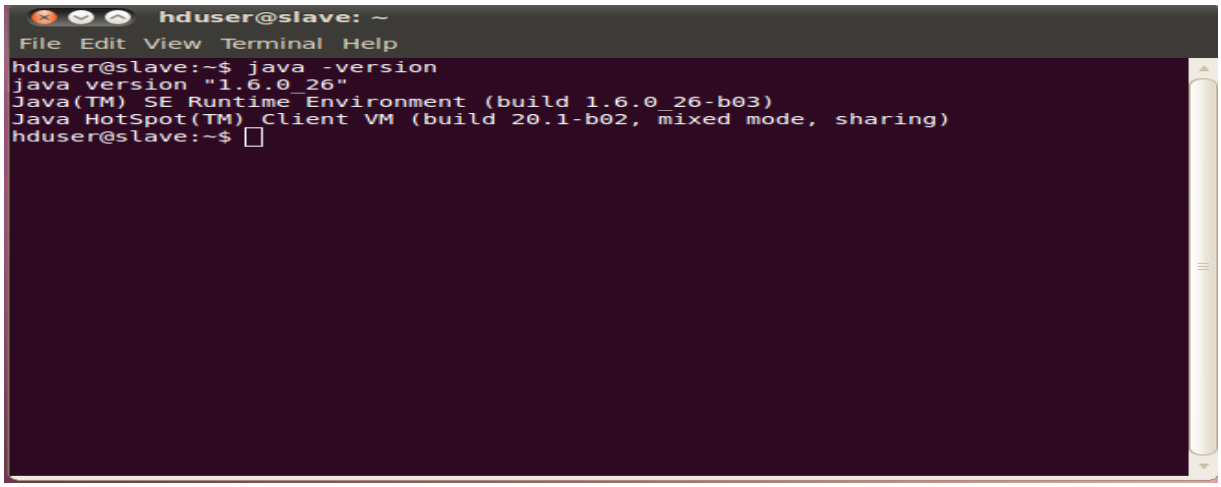


Figure 2: JDK installation

Adding a dedicated Hadoop system user: A dedicated Hadoop user account is required for running Hadoop which helps to separate the Hadoop installation from other software applications and user accounts running on the same machine. User and user group can be added by running following commands.

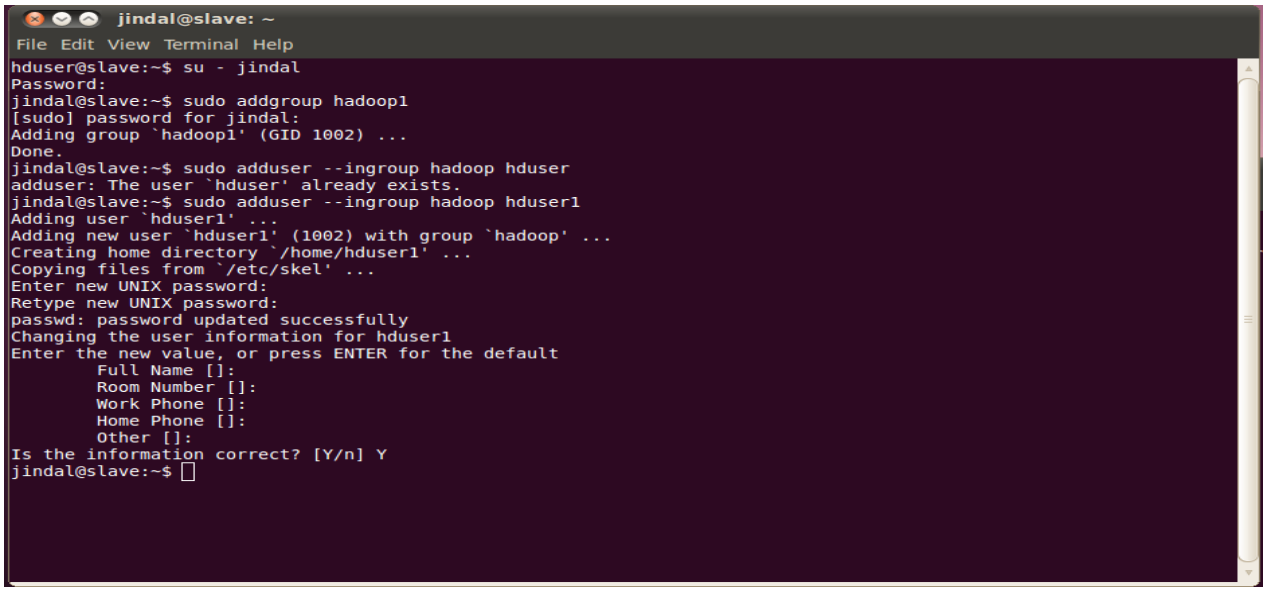


Figure 3: Adding Hadoop user and Hadoop group

Configuring SSH: Hadoop requires SSH access to manage its nodes, i.e. remote machines plus local machine.

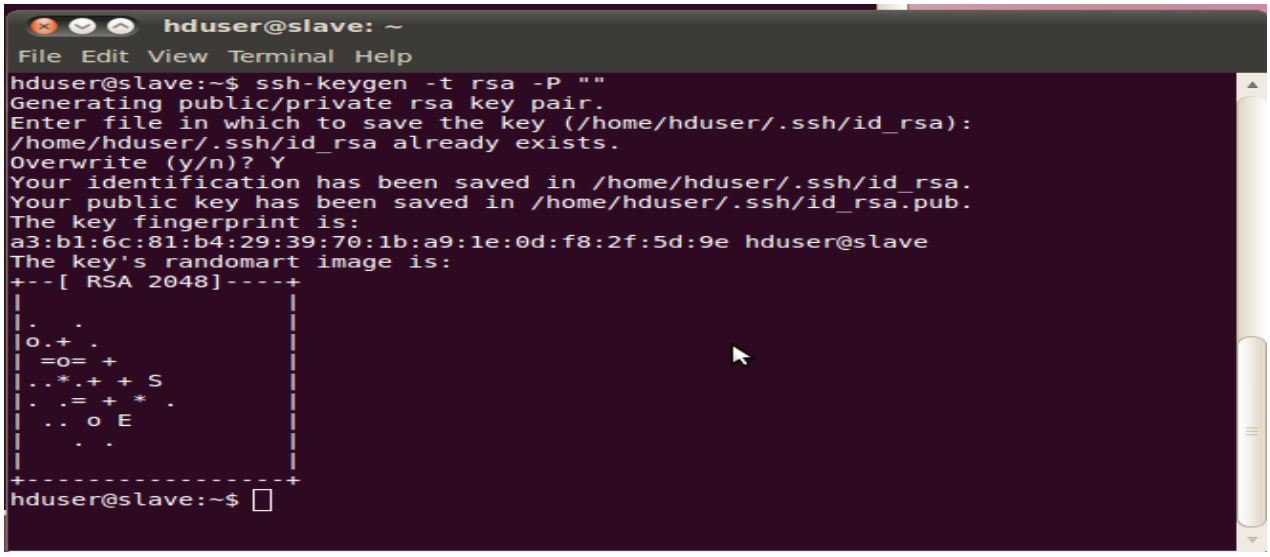


Figure 4: Configuring SSH

Install and configure each node in the cluster.

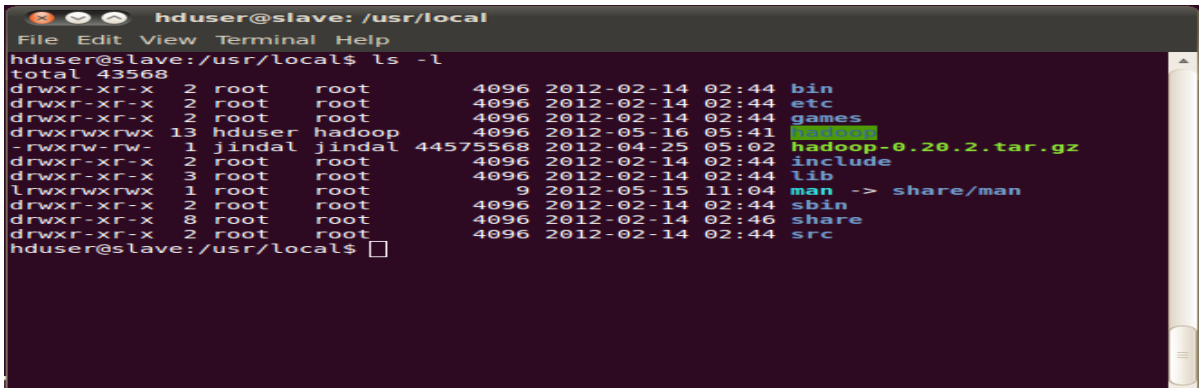


Figure 5: Install and configure

We will call the designated master machine just the master from now on and the slave-only machine the slave**.** We will also give the two machines these respective hostnames in their networking setup, most notably in /etc/hosts.

On Master Machine: Update the hosts file in /etc/hosts with the ip of both slave and master machine.

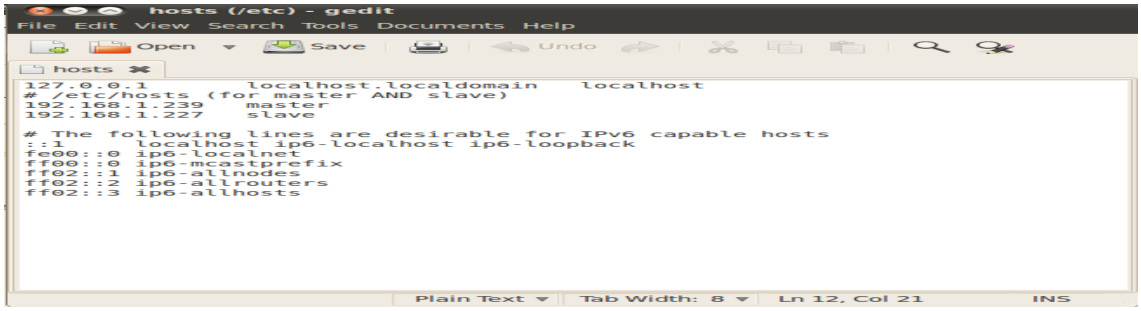


Figure 6: Update hosts file

Update Hostname file in etc/hostname as follows

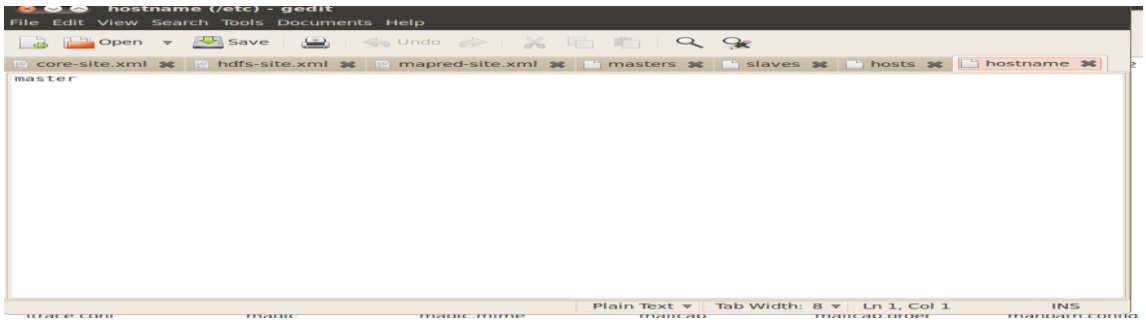


Figure 7: Update Hostname file in etc/hostname

Masters file

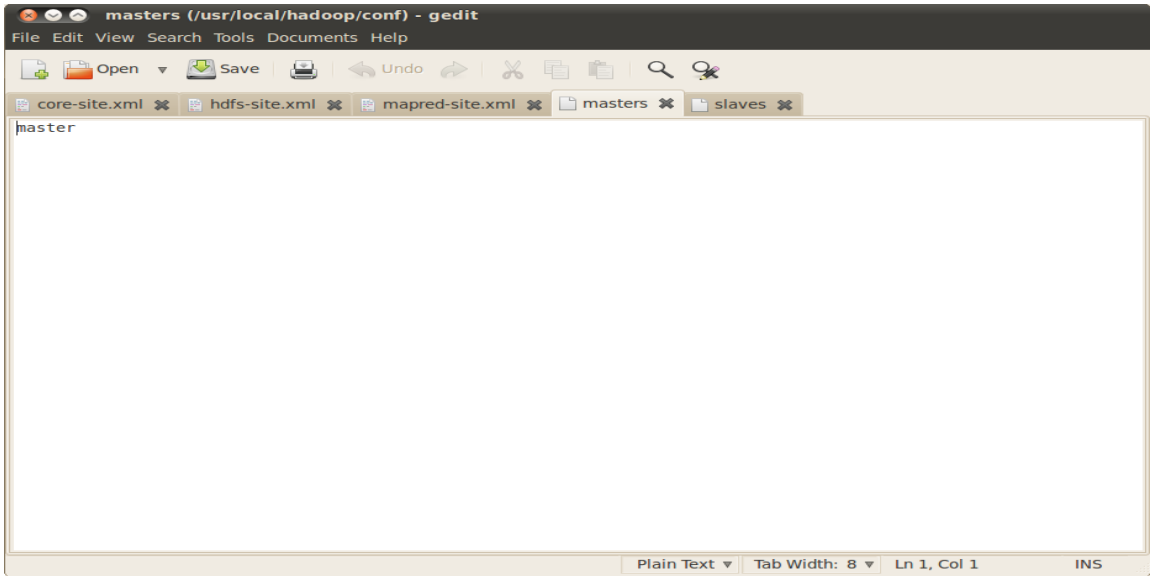


Figure 8: Master file

Slaves file

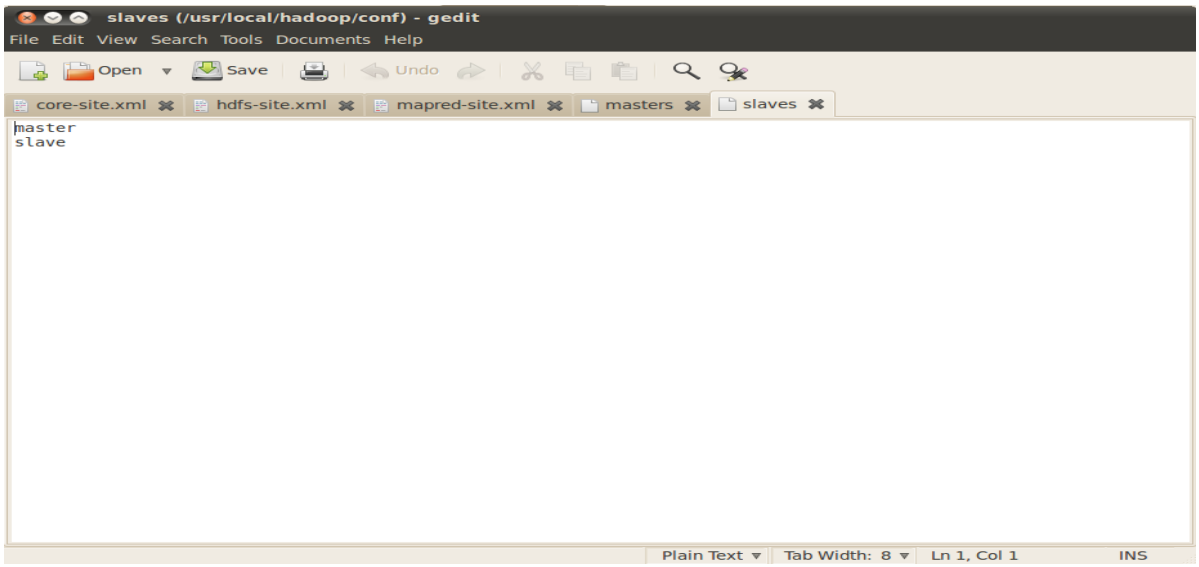


Figure 9: Slave file

Change the configuration files conf/core-site.xml**,** conf/mapred-site.xml and conf/hdfs-site.xml on ALL machines as follows:

First, we have to change the fs.default.name variable (in conf/core-site.xml) which specifies the NameNode (the HDFS master) host and port. In our case, this is the master machine.

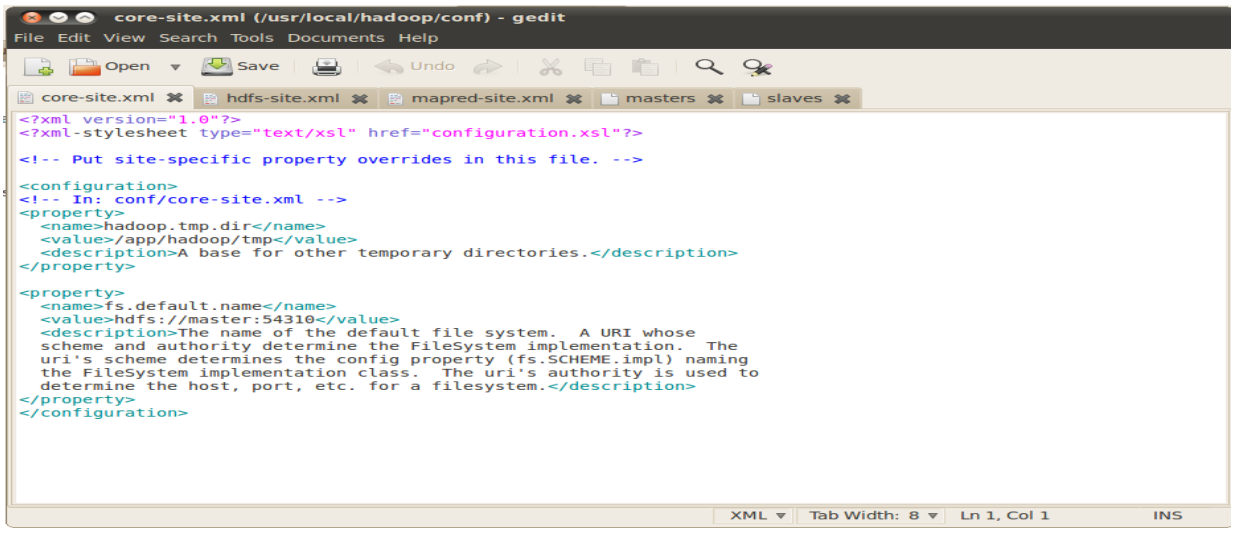


Figure 10: Output of core-site-xml

Second, we change the dfs.replication variable (in conf/hdfs-site.xml) which specifies the default block replication. It defines how many machines a single file should be replicated to before it becomes available.

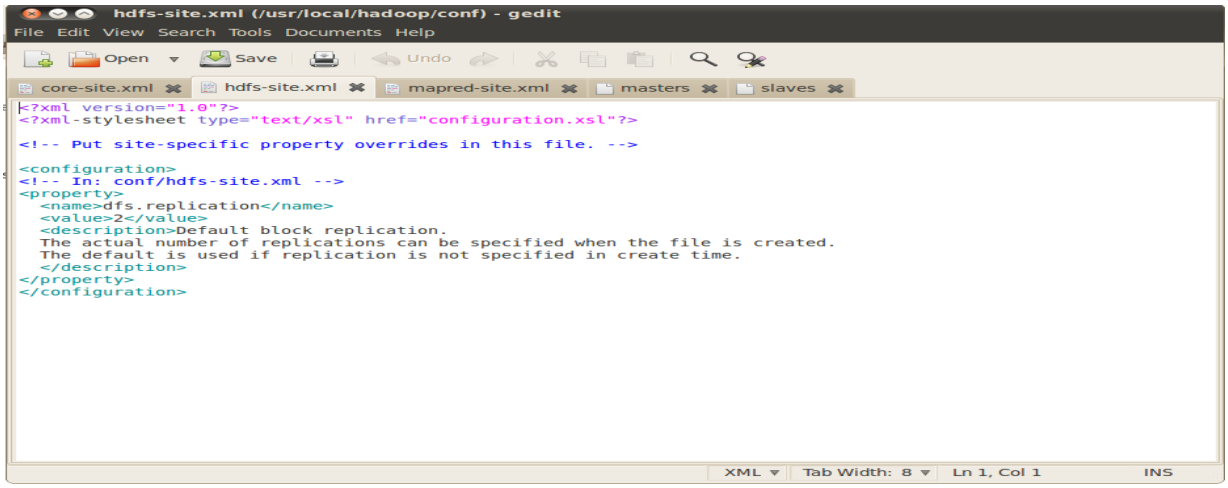


Figure 11: Output of hdfs-site-xml

Third, we have to change the mapred.job.tracker variable (in conf/mapred-site.xml) which specifies the JobTracker (MapReduce master) host and port. Again, this is the master in our case.

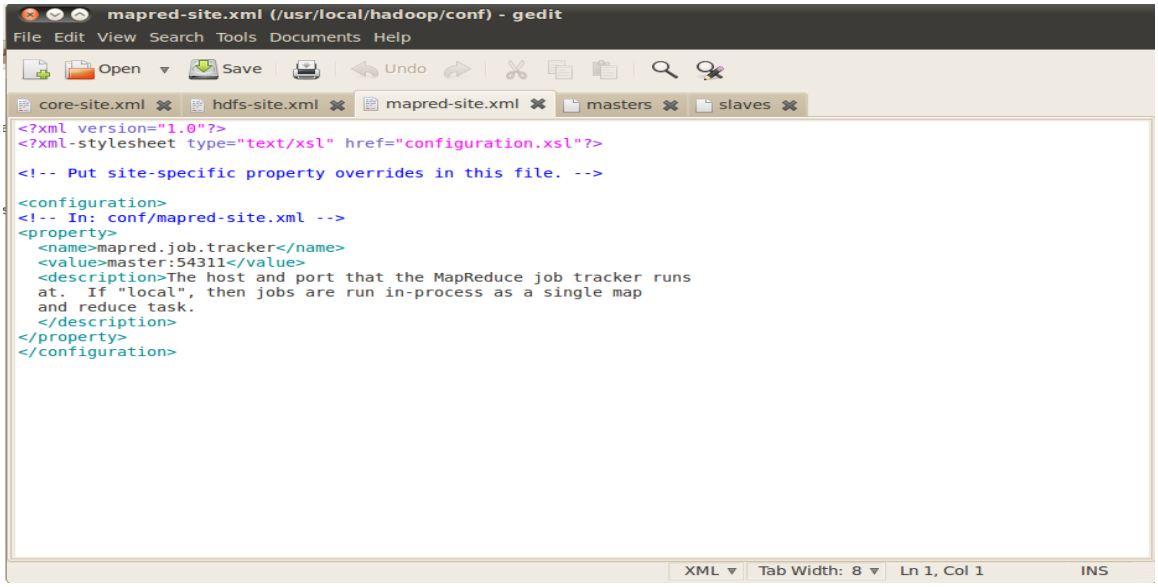


Figure 12: Output of MAPRed-site-xml

On Slave Machines: Update the hosts file in /etc/hosts with the ip of both slave and master machine

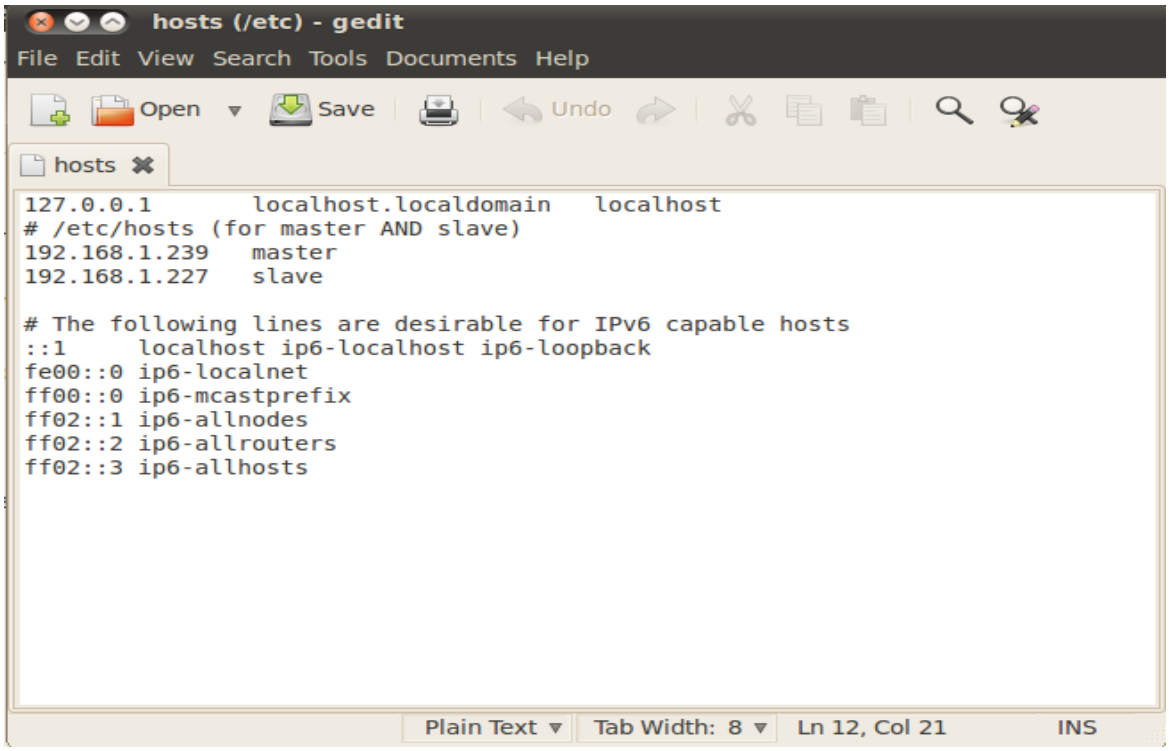


Figure 13: Update Hosts File in etc/hosts

Update the hostname file with the hostname as following.

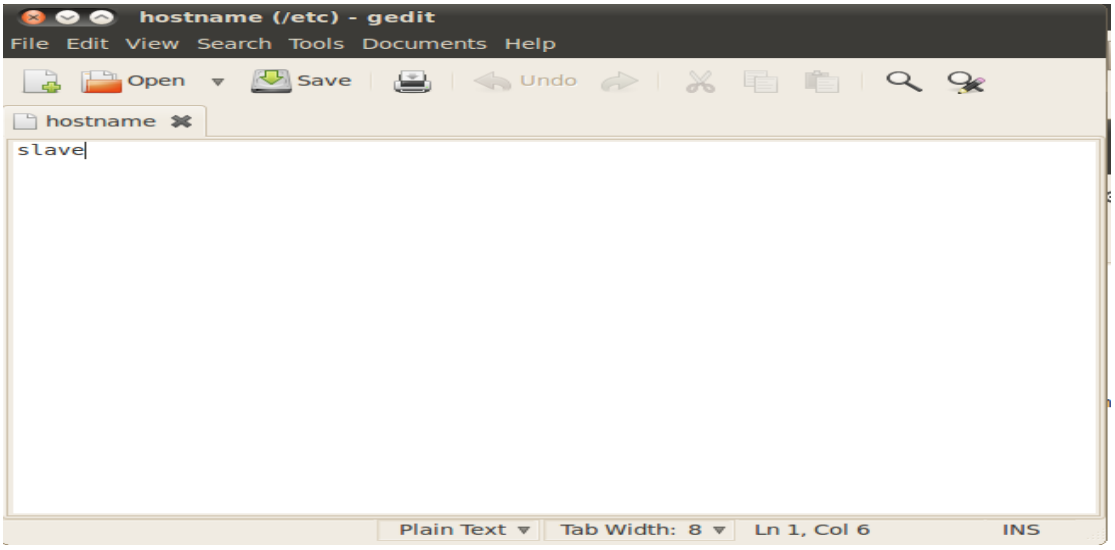


Figure 14: Update Hostname file with the hostname

First, we have to change the fs.default.name variable (in conf/core-site.xml) which specifies the NameNode (the HDFS master) host and port. In our case, this is the master machine.

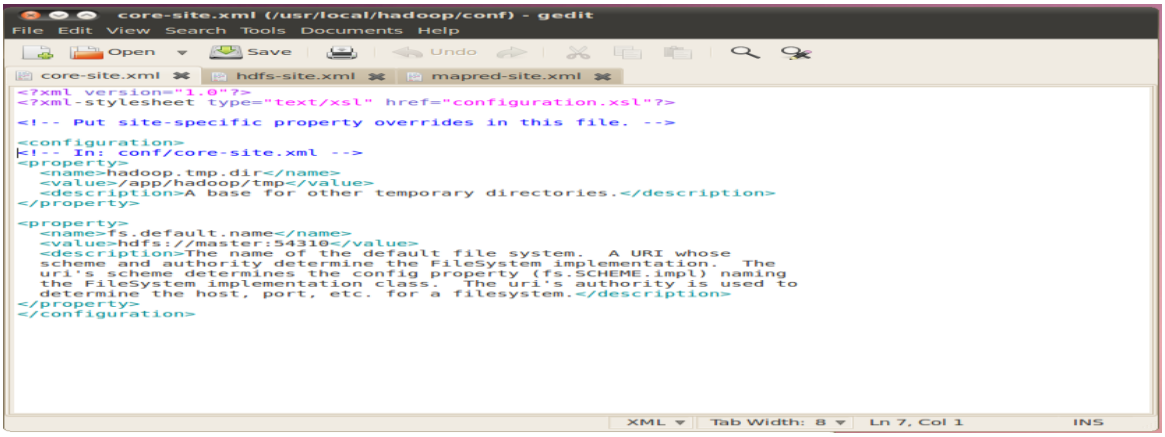


Figure 14: Output of core-site-xml file

Second, we change the dfs.replication variable (in conf/hdfs-site.xml) which specifies the default block replication. It defines how many machines a single file should be replicated to before it becomes available.

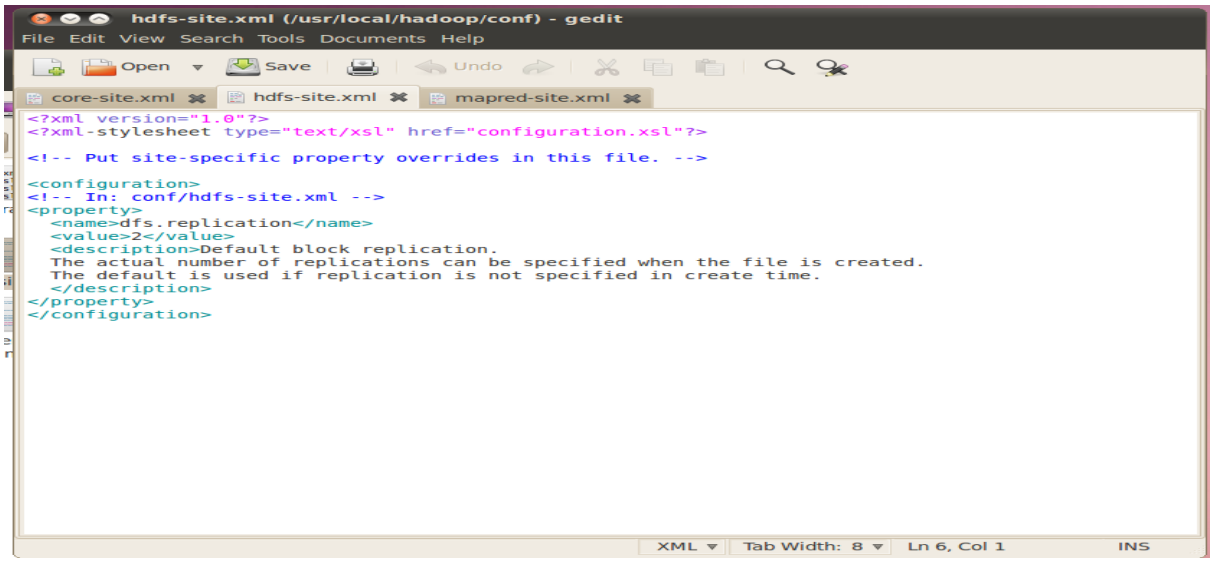


Figure 15: Output of hdfs-site-xml

Third, we have to change the mapred.job.tracker variable (in conf/mapred-site.xml) which specifies the JobTracker (MapReduce master) host and port. Again, this is the master in our case.

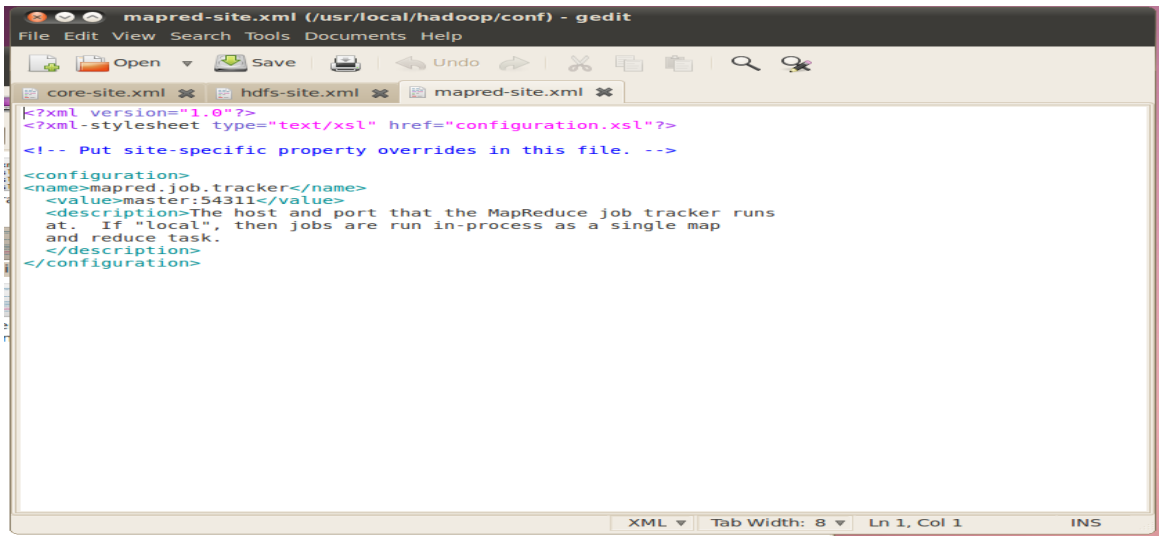


Figure 16: Output of MapRed-site-xml

Starting MultiNode cluster: Starting the cluster is done in two steps. First, the HDFS daemons are started: the NameNode daemon is started on master, and DataNode daemons are started on all slaves (here: master and slave). Second, the MapReduce daemons are started: the JobTracker is started on master, and TaskTracker daemons are started on all slaves.

HDFS Daemons

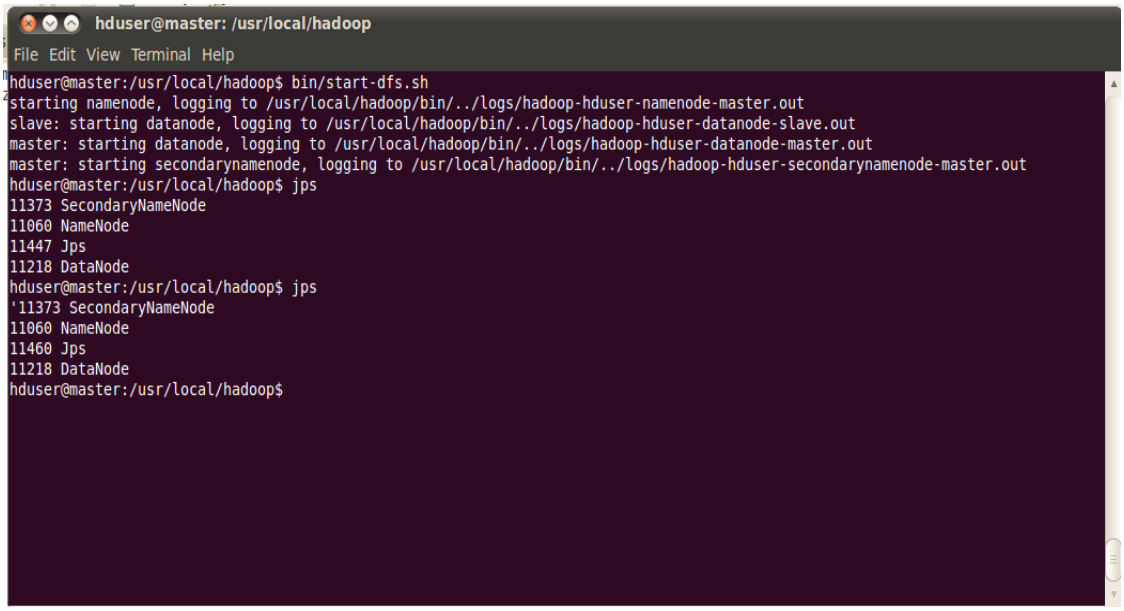


Figure 17: HDFS Daemons

At this time, following java processes will run on slave

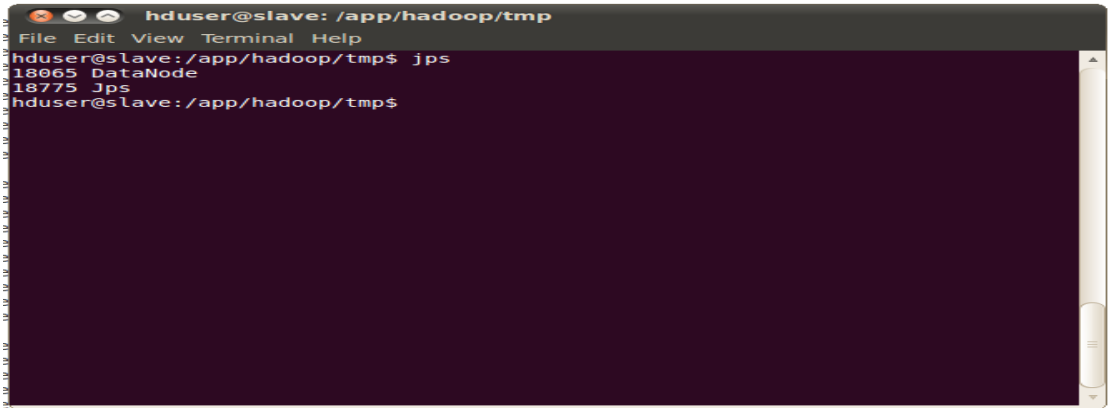


Figure 18: JAVA processes on slave machine

MAPREDUCE Daemons

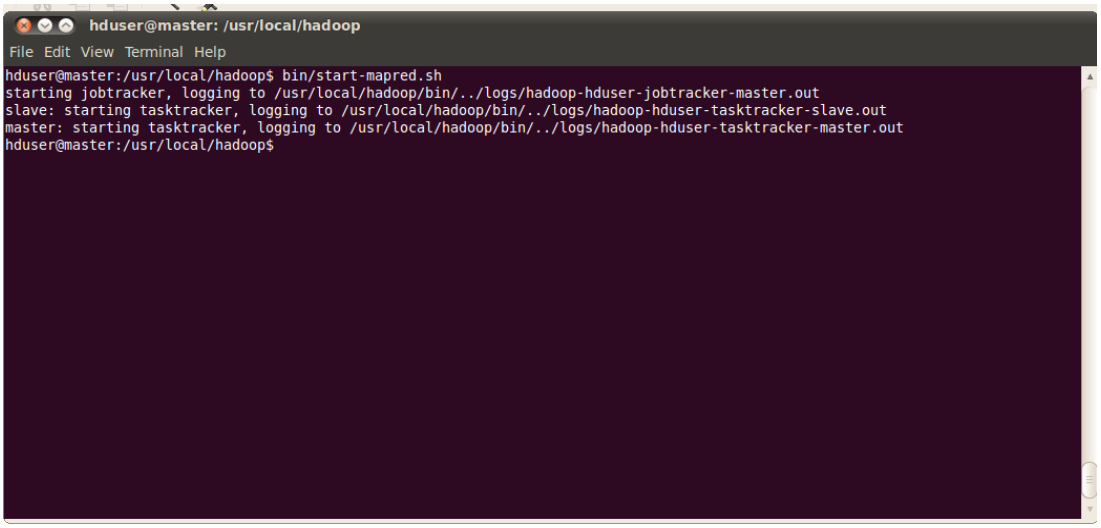


Figure 19: MapReduce Daemons

At this point following JAVA processes will run on master

Figure 20: JAVA processes running on master machine

And Following java processes on the slave

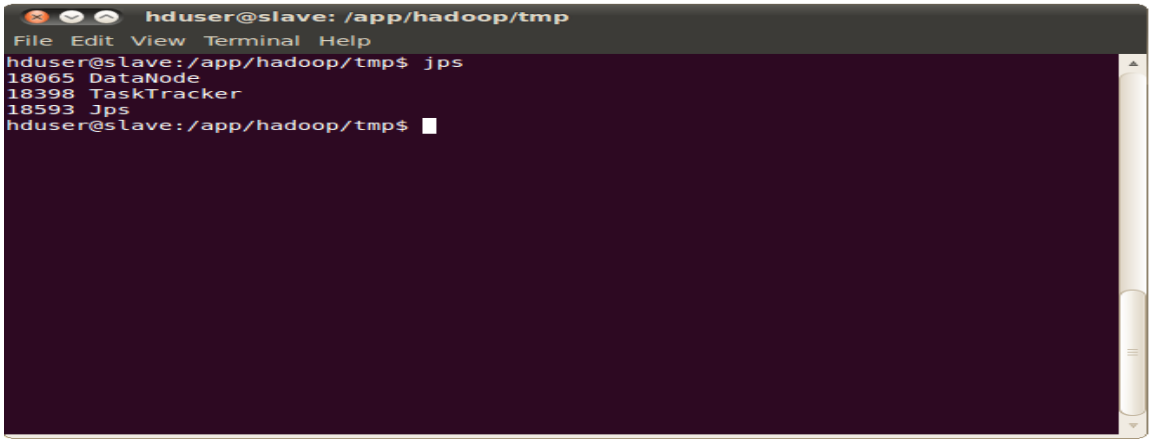


Figure 21: Java processes running on slave machine

List of Publications

Monika Bharti and Anju Bala, "Workflow Management in Cloud Computing". [Accepted]