CRISP: A JavaScript strategy for cloud application development

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

Master of Engineering

in

Software Engineering

Submitted By

Ayush Sahu

(Roll No. 801431004)

Under the supervision of:

Ms. Ashima Singh

Asst. Professor

Thapar University, Patiala

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2016

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*CRISP: A JavaScript strategy for cloud application development*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software 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 *Ms. Ashima Singh* and refers other researcher's work which are duly listed in the reference section.

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

Signature

(Ayush Sahu)

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

(Ms. Ashima Singh)

Asst. Professor, CSED

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

Abstract

The application development in cloud is increasing day by day. Various strategies are being employed for providing service over cloud. There are organizations which develop native application and want to provide and access the service through cloud. Such organizations need cost effective methods to deploy their service as application in cloud environment. The traditional methodology involves writing efficient code in a language for a particular platform. This code provides high efficiency during execution while keeping the deployment cost low. But development time and development cost always increases when we follow such methodology. Also the code is not portable to be executed on other platform. Thus these complex methodologies of cloud development are not suitable for beginners, small organizations and startups as they have low budget and less experience to implement such methodologies.

In this thesis, CRISP (Conversion, Reformat code, Isolate module, Sandbox, Partition) strategy has been proposed for refined conversion of native application to JavaScript for cloud application development. JavaScript is chosen as medium for writing application because it is mostly used language among developers and provides rich API (Application Programming Interface) for writing applications. It can be used both on client and server side hence it is easy to learn one language and implement services using it. However, hand written code and machine converted code are not efficient enough to be used for application development. If hand written code or machine converted code is used directly then it would lead to less efficiency and more development.

The CRISP strategy allows to improve efficiency of code and optimize it for cloud application development. Various steps in this strategy removes bugs of converted code in each step. The code is divided into libraries to allowing code reuse for other applications. Finally the code is sandboxed to provide security and then it is hosted on cloud. The developed application can be deployed on cloud efficiently. Thus proposed strategy provides economic method for cloud application development.

Acknowledgement

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life. It is a great privilege to express my gratitude and admiration towards my respected supervisor **Ms. Ashima Singh** Assistant Professor Computer Science & Engineering Department. She has been an esteemed guide and great support behind achieving this task. This work would not have been possible without the encouragement and able guidance of her. I also thank my supervisor for her time, patience, discussions and valuable comments. Her enthusiasm and optimism made this experience both rewarding and enjoyable. I am truly grateful to her for extending her total co-operation and understanding whenever I needed help and guidance from her. I am also heartily thankful to **Dr. Maninder Singh**, Associate Professor and Head, Computer Science & Engineering Department and **Rupali Bhardwaj**, PG coordinator, for motivation and providing uncanny guidance and support throughout the preparation of the thesis report.

I will be failing in my duty if I do not express my gratitude to **Dr. S. S. Bhatia**, Senior Professor and Dean of Academic Affairs, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field.

I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable. Last but not least, I would like to thank my family for their wonderful love and encouragement, without their blessings none of this would have been possible.

Ayush Sahu
(**801431004)**

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

In this chapter, a brief introduction of cloud computing, deployment models and cloud security is given.

1.1 Cloud Computing

Cloud computing is an emerging paradigm to deliver services through internet. As network bandwidth and internet services are growing, internet has become essential in everyday life facilitating the functioning of cloud services. Customizable services in terms of storage space, hardware availability, network bandwidth, platform variation etc. are characteristics of cloud platform [4]. Cloud services provide lease based services that can be released on-demand whenever needed. Email, social networking, banking solution, e-commerce, stock exchange etc. are various services being provided through cloud. Infrastructure providers and service providers work together to setup infrastructure and provide services to the end users [5]. Cloud shares characteristics with autonomic computing, ubiquitous computing, client-server model and grid computing. There has been sharp rise in this field due to its capability to provide services in various fields of human life.

The relative growth of cloud computing as compared to other technology fields is shown in Fig. 1-1. The cloud computing has made it possible to overcome the barriers of providing service through client-server paradigm in cost, complexity, maintainability etc. The cloud service makes it possible to access flexible services irrespective of location, thus it has been widely accepted by end users in different parts of the world [7]. This flexibility of cloud service also allows an individual to customize services for end users according to their need. End users acceptability to cloud services has led to wide range implementation in different fields like health, sports, money exchange services etc. Cloud provides reliable services through its parallel running infrastructure. The infrastructure is setup such that no interruption occurs in case of hardware failure or security attack. Multiple systems executing in parallel allows to make a system that is reliable to work continuously without needing

effort from customer end [45]. This feature of cloud has increased the trust of customers on services provided through cloud

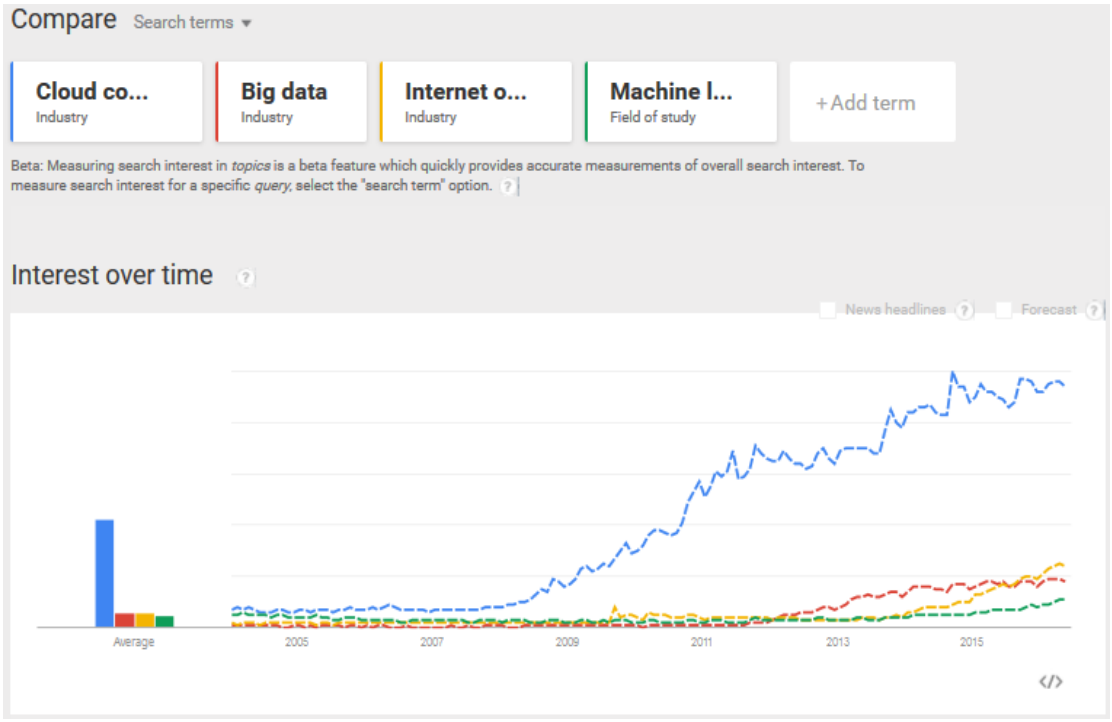


Fig.1-1: Comparative growth of different technologies

To study cloud computing there is a need to understand different perspectives given by researchers [2]. This allows us to lay down the basic motive behind setting up cloud and provide service through it. Following is a commonly used cloud definitions:

NIST definition of cloud computing - *Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources* (*e.g., networks, servers, storage, applications, and services*) *that can be rapidly provisioned and released with minimal management effort or service provider interaction* [43].

1.2 Evolution of Cloud Computing

Cluster and grid computing forms roots of cloud computing. These technologies provide prime paradigm for setting up the IT service through cloud to end users [1]. The main task of these of service providers is not just to provide service but also to provide these services in efficient manner because capital investments, security and

privacy of consumers are involved. The Quality of Service (QoS) must be maintained at all levels of application servicing [57].

Cluster computing and Grid computing is defined differently by various researchers. Buyya *et al.* describes Grid computing as distributed computing in large-scale providing consumers with resources that are present in geographically vivid location [62]. Cluster computing is a parallel and distributed system according to Pfister [3]. Thus there are many characteristics that trace back from cloud computing to Grid and Cluster computing.

Grid computing traces back its roots from electric grids that are used to provide electricity as a service [7]. Electricity consumers are charged through Pay-as-you-go model. The development and scale up of Datacenters, Distributed system, Web 2.0, Supercomputers, Cloud Computing and Grids is shown in Fig. 1-2. These utility systems come under distributed system. Systems coordinate and communicate with each other via intra and inter network to share data whenever there is requirement.

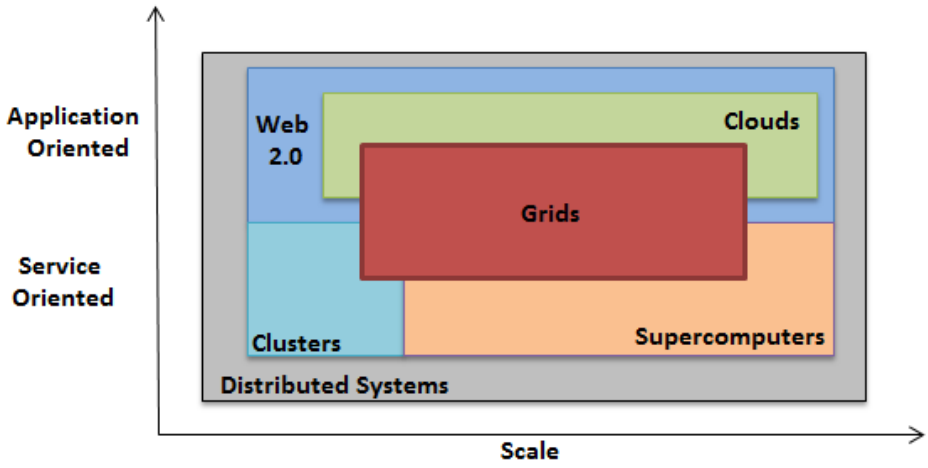


Fig.1-2: Utility Computing Overview

1.3 Characteristics of Cloud Computing

Shared resources, payment flexibility, extensive scalability, elasticity and self-provisioning are the basic attributes that define any cloud computing service [6]. Various important characteristics of cloud are defined below:

1.3.1 Multitenancy

Sharing of resources at different level (network level, application level and host level) is referred as multitenancy [51]. It allows sharing of resources between customers on the cloud rather distinct allocation of resources to each customer.

1.3.2 Pay as you go

The cloud services allow flexible payment by leasing services depending on different factors like functionalities used, active time period etc. within fractional cost compared to setting up whole infrastructure. Customers of cloud services need to only pay for resources used and the time for which it is used [4].

1.3.3 Scalability

Resources available can be scaled up or down as needed, allowing cost effective method for resource utilization. Customer can control scaling from small amount of systems to thousands of system to meet customer requirements. Various scalable resources are systems, network bandwidth, storage space etc.

1.3.4 Elasticity

It refers to the user choice to increase or decrease the computing resources whenever there is need. It also allows to release resources when they are no longer required [8]. Charges of service are applied only for the period the subscribed services are used.

1.3.5 Self-provisioning

Users are provided with capability to control the cloud resources through Graphical User Interface (GUI) panel. They can customize the service being provided to meet their needs and budget [9].

1.3.6 Ubiquitous computing

Cloud computing allows to provide service through different means and in different form. This allows to cater needs of different customer and user.

Various technologies can be used that are efficient for a particular device and provide effective service through it [25].

1.3.7 Merging services

Different services provided online can be merged to create a new service. This is achieved by merging different services through proper API provided for that service. The same process can be used for creating desktop application and smartphone application. This reduces the complexity of applications. It also reduces the budget of organizations by using services that are already available and reliable [6].

1.4 Cloud Service Level Agreements

Consumers should make an agreement with service providers to get the demanded service after subscribing for it. Service Level Agreements are used to provide such agreement between providers and consumers. Cloud services are preferred by customers depending on the facility provided by those service providers. Various service providers have come into play during past few years [10]. The comparative growth of different cloud service providers is shown in Fig. 1-3. This shows that preference of Google cloud service is increasing in recent years. Google is providing facility to its customer by providing free services for few months. After the end of free cloud service usage a customer can opt for continuing its service [11]. There is also provision to buy different service module by subscribing for it. Other cloud services remain prominent by providing cloud service in customer friendly way.

Amazon elastic cloud computing has maintained its leadership in providing cloud services. Many individual, start-ups and organizations are selecting Amazon service as the platform for supporting their homebrew applications, software services and enterprise solution through cloud [12]. However there are only few service providers that allow executing JavaScript code on cloud platform. Google is providing these services through different frameworks. These services are provided through node.js as the core service framework [14]. Node.js allows executing the JavaScript code on cloud server side. In Google cloud, one needs to configure node.js framework to execute JavaScript code.

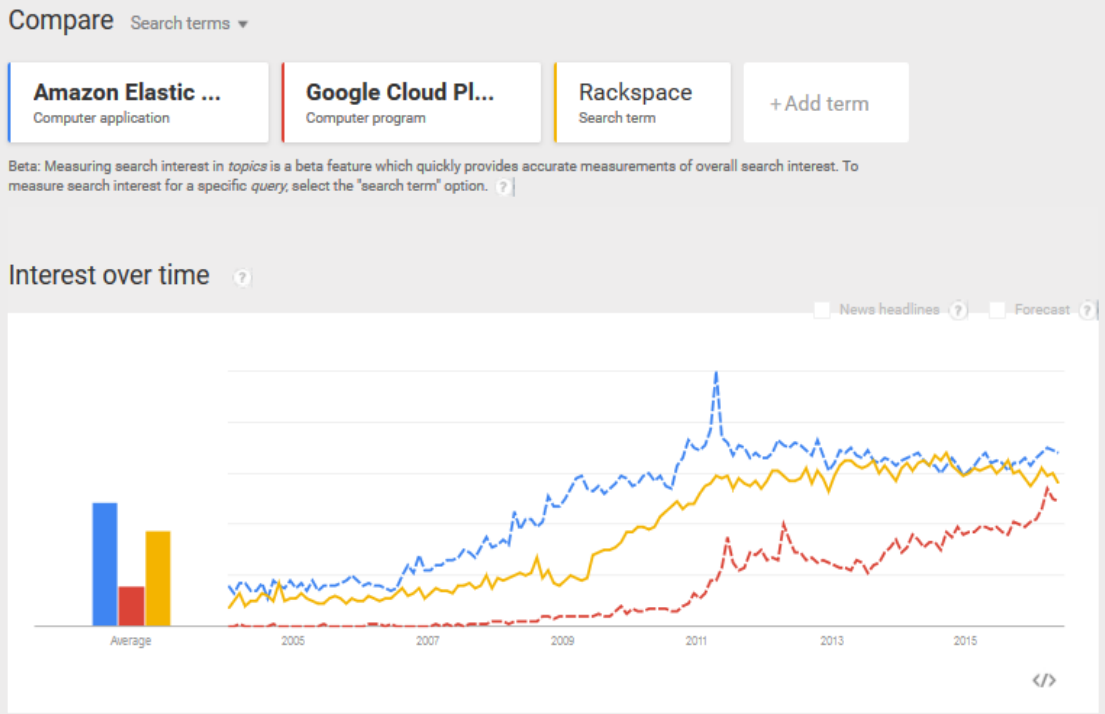


Fig.1-3: Growth of different cloud service providers

1.5 SPI framework for cloud service delivery model

Cloud computing offers high processing capability for all types of business whether big or small. This makes it possible to deploy any application within short period of time and access vast resources for implementing business logic to provide certain services [14]. Cloud computing is described by SPI framework [51] as shown in Fig. 1-4. The SPI acronym refers to software-as-a-service (*S*aaS), platform-as-a-service (*P*aaS), and infrastructure-as-a-service (*I*aaS). This framework provides the basis for deploying infrastructure, system programs and end user application at different levels.

1.5.1 Software as a Service (SaaS)

The end user access software or needed application as a service through internet. It makes it possible for customer to access the service without having to go through routine of installing software in each machine [15]. It reduces the maintenance cost and other effort on part of customer. Middleware software is used to access the cloud services. In most cases it is browser [12]. In other scenario like smartphone or netbook it is custom built application. It enables to execute multiple instances of software on

various machines. Google Docs, zoho and gmail are some of the popular software as a service [57].

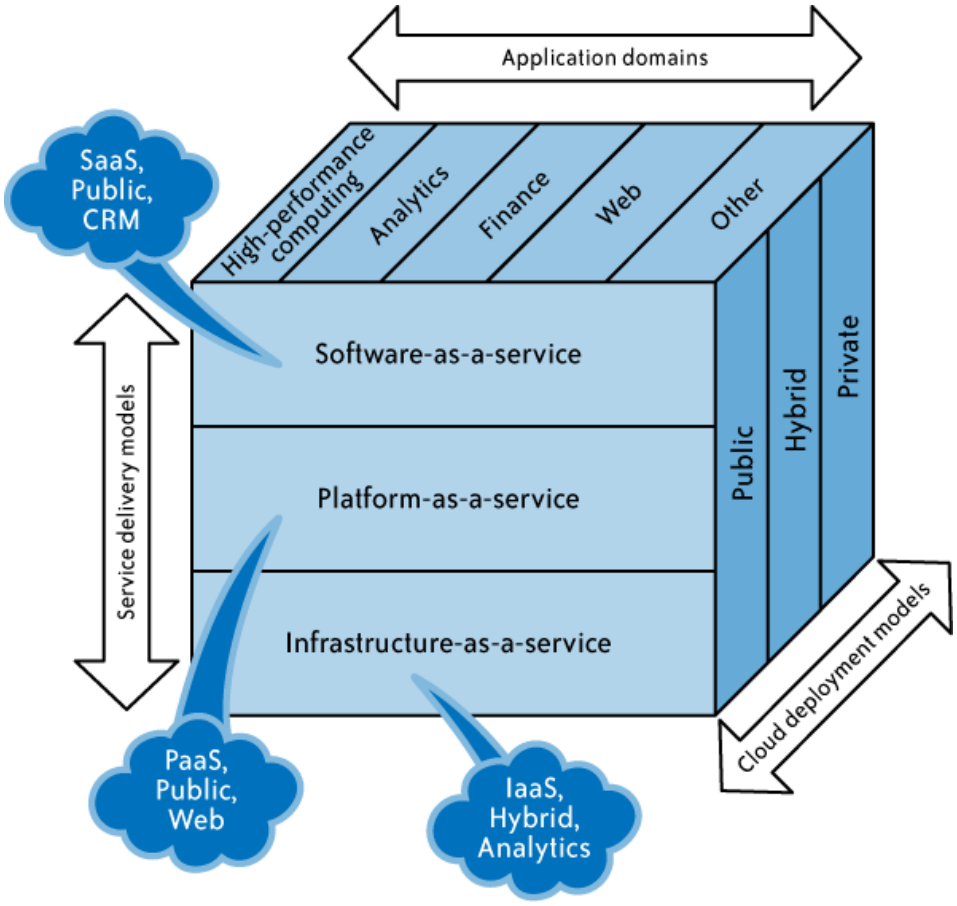


Fig.1-4: SPI framework

1.5.2 Platform as a service (PaaS)

It is a platform that provides resources to deploy application that execute on cloud [7]. This platform provides basis over which multiple services are deployed. It acts as a middleware between infrastructure and software that needs to be hosted on cloud. Microsoft Azure, Salesforce.com, Google App Engine, Rollbase are few of the available platform as a service [14]. Load balancers allow to balance load of executing application over infrastructure.

1.5.3 Infrastructure as a Service (IaaS)

It is the basic foundation of cloud. It refers to the basic hardware like servers , storage devices, network management devices etc. that makes it

possible to deploy platform over which software is serviced. Underlying hardware is abstracted at this level. This abstraction enables to manage the underlying resources in efficient manner [6]. It is built in such a way that it is scalable to meet requirement needs of various customers [19]. Hypervisors like Oracle VM, Xen, VMware ESX/ESxi etc. runs virtual machine over hardware. These virtual machine acts as a closed environment for deploying different platform. This provides feasible solution to manage and deploy services for different customers on cloud over same infrastructure [20]. Various IaaS service providers are Rockspace hosting, Amazon Web Services, Joyent, VMware etc.

Different personnel control different levels of services. The Fig. 1-5 shows the various tasks that needs to be managed at each level. It also depicts the tasks that needs to be managed by 3rd party.

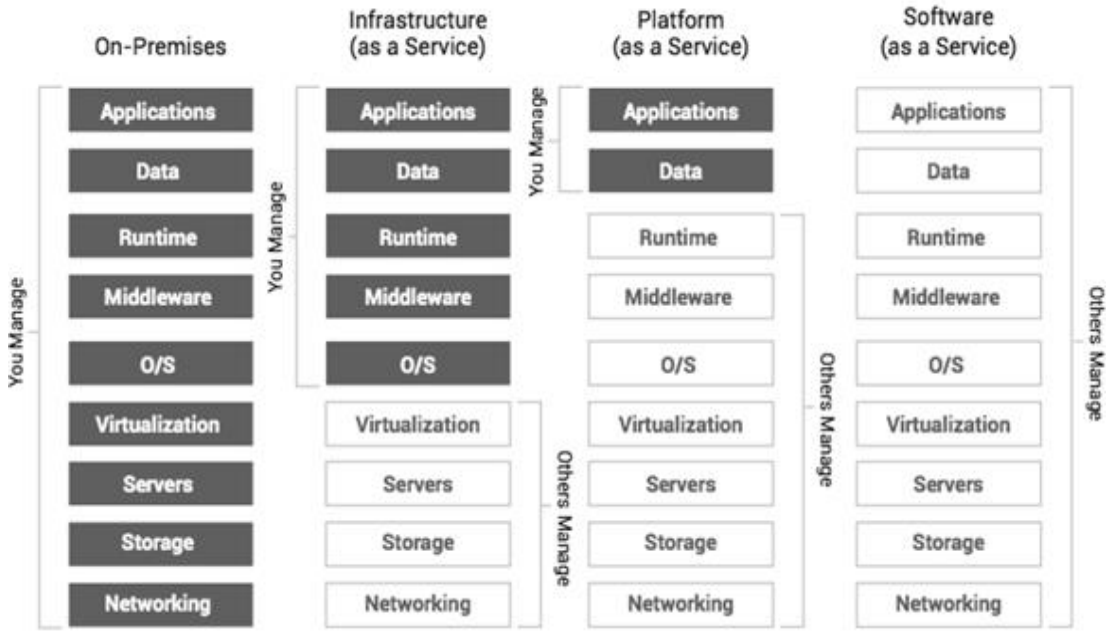


Fig.1-5: Services management at each level

Flexibility of services decreases as we go from IaaS layer to SaaS layer. However agility increases as we go from IaaS layer to SaaS layer. Thus end user application services are provided through SaaS. The level of flexibility, service agility and budget act as deciding factor for selecting the level at which one wants to provide the service.

After deciding all the requirement a cloud deployment model can be selected to provide services.

1.6 Cloud Deployment Models

Cloud services are provided through different models. These models define the different ways of deploying applications over cloud. These deployment models define the relationship between cloud and user. Deployment models are important as they support cloud computing that enables implementation of virtualized resources in dynamic and scalable form over internet connections by service provider [27]. This allows end users to be free from complexity, expertise, knowledge and implementation needed to provide service through cloud [50]. The majority of cloud services are provided through data centers built over hardware with different sets of virtualization technologies [28]. This deployment model abstracts the users over all view of infrastructure being used to construct cloud. In this deployment model open standards are important for growth of cloud computing. Various cloud deployment models are as follows:

1.6.1 Public clouds

It is defined as cloud computing where mainstream services are publically accessible. These deployment models provides self-service basis and fine-grained control over cloud resources through internet, web services or web applications via third-party service provider sharing resources for utility-computing basis.

A public cloud is managed, operated and hosted by a vendor working in data center. Over common infrastructure cloud service is provided to subscribed customers [6]. A third party vendor is responsible for security management and privacy management for day to day work. Hence the public cloud customer has low control over logical security and physical security attributes of public clouds.

1.6.2 Private Clouds

Cloud services being provided by emulating the computation on private network is referred as internal clouds or private clouds. This method of virtualization automation

of products delivers many benefits using cloud computing without backdrop, compromising data security and reliability concerns [35, 48]. Organizations can easily purchase, construct and maintain cloud without much capital cost with less work. The individual or organization is responsible for setting up cloud is responsible for the private cloud operation.

Private clouds are different from public clouds as storage infrastructure, computing resources and network used for private cloud are used by a single organization rather than by other organizations [4]. This has given rise to various patterns of clouds which are as follows:

- Dedicated – It refers to those private clouds that are hosted through a data center owned by customer and maintained by internal departments.
- Community – It refers to those private cloud that are created by a third party. These are operated, owned and managed by a vendor bounded by contractual clause and service level agreements with compliance to security requirements [40].
- Managed – These types of private cloud are maintained by vendor and the infrastructure is customer owned.

In general, third party and internal IT personnel manage hosting and security operation in private cloud model. This demands high degree of logical and physical security to control different aspect of private cloud infrastructure [42].

1.7 Cloud security and privacy

According to market demand and customer need, services of cloud can be availed and used. Security and privacy are important concern over cloud [44]. The data from various sources like social networking sites, government repository, organizations repository are present in cloud [46]. While running different processes there is always probability of data leak, deletion, overwrite etc. Thus there is need to manage all the data, processes, virtual machine, hypervisor etc. to ensure data security and privacy [47]. There is also need of protection barrier between applications as one application can intrude the working of other application and cause data mangling [52].

Virtualization provides security and privacy between different applications running over cloud. Private cloud offers more security as compared to public cloud.

With increasing need of computing services and market demand by customers, it has become a necessity to provide fast deployable software that can be executed over multiple platforms [53]. The software developers and technology companies are finding ways to deploy software over cloud in effective, fast and reliable ways [56]. To reduce application deployment time various strategies have been used like automation of activities, providing native language support directly on cloud [1], native development using C/C++ language to run applications in browser [14], using multi-tier architecture, application migration etc. Each method provides certain benefits over another.

Native development involves using efficient languages like C/C++, Java etc. to build applications. It is used where efficiency is needed and execution time should be less.

Application migration is a technique for executing existing software over cloud. It is one of the important methods for providing service over cloud when the software is available that needs to be serviced over cloud. Migrating application to cloud rather developing application from scratch is a common scenario for small and medium scale industry as it reduces deployment cost, maintenance effort, allows code reusability etc[55,12]. Migration of application is rather a complex process and often involves many steps [21].

Thus through this chapter cloud computing is introduced. The cloud security and privacy issues have also been studied through this chapter.

Chapter 2

Literature Survey

In this chapter, various methods and current trends of cloud computing are analyzed. Abstract of those studies is provided below.

2.1 Application migration and Cloud cost problem

App migration makes it possible to use existing software on cloud. It is achieved by different processes that modify the existing codebase of application to make it execute on cloud framework. Rebuilding, rehosting and revision of available software code are few ways to migrate applications.

Wang *et al.* discussed various work done on application migration from perspectives like data dynamics [59], security analysis [49], pooling of cloud resource etc. Wood *et al.* studies dynamic pooling of cloud resources while setting up cloud infrastructure [60]. These studies present different factors like cost, data privacy and security that affect application migration. Cloud applications run inside virtual machine. Virtual machines are operating system that provides necessary services to utilize cloud platform. Virtual machine based migration of software is a complex process.

Andrikopoulos *et al.* researched and found that complexity of the migration process has been reduced with the advent of business logic runtime environment like Google App Engine and database services like Amazon Relational Database Services [8]. This benefit has encouraged product developers to migrate their software to cloud and provide service through it. However it is not always possible to migrate full application to the cloud due to technical infeasibility, high cost involved and low performance of translated code obtained.

Various frameworks and methods have been proposed to estimate cost of servicing on cloud. Klem *et al.* compared distributed computing with cloud computing with respect to factors like leasing, delivering service, storing data and deploying applications etc. The concluded that distributed computing is limited in comparison to cloud computing to provide service to customers. The author also found that cloud

computing offers better methods to provide service to end users. Although, its acceptance depends on service provider and consumers to setup business model that gives good value [30].

Assunçao *et al.* investigated the benefits the organization can take by using the Cloud computing infrastructure rather using local computing infrastructure. The author evaluated the cost of various cloud service scheduling strategies being implemented in organizations. Investigation involved studying, how virtual machines are initialized when resources are low to serve user requests and change it dynamically. A strategy was devised that achieves optimum performance by balancing scheduling and usage cost [13].

Greenberg *et al.* researched problems in various data centers and their cost factors involved in building up cloud. The researcher found that as network grows the cost factor starts depending on network usage. This is due to the fact that the service usage grows as network grows [20].

Simarro *et al.* proposed a cloud broker architecture to deploy virtualized servers across clouds to make handling of data easier for customers and end users. This architecture implements a scheduling module for obtaining optimal placement of virtual infrastructure while abstracting the complexity involved. He validated his approach by performing numerous test on use cases. The author shown that such brokering mechanisms could be effectively used in dynamic deployment [39].

Often migration cost, hosting cost, infrastructure cost and budget needs to be considered before migrating an application. Thus to migrate an application one must use economic and technically feasible methods.

2.2 Cloud security and privacy problem

Cloud applications run within virtual machine, this provides security between different applications. However, it often happens that customer needs to use modules from different vendor which could be incompatible or insecure. Subashini & Kavitha surveyed the security issues present in cloud computing service delivery models [50].

The authors show concern over safety of cloud environment when data of individuals and companies keep increasing in cloud. Their research shows that customers remain reluctant in deploying applications over cloud. Data protection and data privacy remain top priority for deploying applications over cloud. This survey focuses on many important security threats involved in deploying applications over cloud that need to be considered when migrating the applications.

Rosado et . al. justified the importance of security involved in applicaton migration. The author said that their has been a steady increase in cloud services in recent years and market maturity is also increasing to cater the security needs. This research done by author involved survey of different processes involved in application migration. Various factors have also been documented to show the benefits and opportunity involved in migrating application to cloud. However, this study also shown that legacy application migration is more prone to failure [18]. Certain security specifications have been proposed but they are not enough to meet the complexity of cloud applications. Virtualization provides security to great extent still certain elements of cloud need to be secured through new implementation.

Li *et al.* surveyed recent advances in web and cloud application testing [37]. The transformation of web services to cloud services is studied. The study also shown that cloud security is a major concern compared to security issues present in traditional desktop applications while migrating applications. Cloud applications are unique in many ways. This uniqueness causes new challenges for their testing and quality assurance.

- Client-server complexity – Low *et al.* [38] provided basis for different cloud computing determinants. The author investigated the factors affecting cloud adoption. Cloud applications are multilingual. A cloud application is build-up of cloud server used for hosting files and a client side frontend in form of standalone application accessible through browser. Both server and client codebase being implemented in various programming languages requires expertise in different field. Client applications are usually written in C/C++ and server end applications are written in JavaScript, HTML, Java etc.
- Security attacks – Operating environment of cloud application is more prone to attacks like denial of service (DOS), distributed denial of service (DDOS),

cross site scripting (XSS), domain poisoning etc. This cloud vulnerability is due to its open nature. Public clouds are more vulnerable to attacks as they are more open compared to private clouds. Kosta *et al.* developed benchmark for such attacks to evaluate both complex and simple applications [31].

- Multithreading issues - Cloud application executes multiple instances at the same time for different users. Multithreading support is needed to provide multiple instance of same application. This complexity in architecture to provide multithreaded service often leads to security gaps that are used to bypass security measure during attack on cloud. Levandoski *et al.* studied the complexity of multithreading in cloud for ACID transactions and provided solution for such problems [3].
- Un-implemented standards – Often the security standards are not followed by cloud application developers or not covered extensively, leading to vulnerable applications. Standards need to be followed to overcome such vulnerability [8].
- Vendor limitation – Brodkin found seven specific issue related to security before selecting vendor. They are regulatory compliance, privileged user access, data segregation, data location, recovery, long-term viability and investigative support. These issues need to be checked for implementing cloud [6].

To search these security flaws various testing strategy are used like model and graph based technique, mutation testing, concolic testing, random testing, scanning and crawling, user session based testing [32].

2.3 Strategies of application development and migration

To provide simple strategy to overcome the application development and migration barriers many proposals have been given to migrate legacy applications and new software.

Frey and Hasselbring proposed standards that enable software developers to develop and design cloud services by laying out topologies of typed nodes. These standards allow reusability by using existing solution. Individual components of application are

stored as fragments which are later adapted and reused to develop new application [17].

Mohagheghi and Sæther proposed model-driven migration method for deploying services in service-oriented architecture. To propose this model the authors performed state of the art survey and analysis. In this methodology the application is decomposed into smaller modules and then coupling of module is removed. Finally the application is made scalable by moving it to service-oriented architecture [41].

Li *et al.* researched migration of applications to cloud in energy efficient way. The authors found that available methods for application migration to cloud are not efficient and needs to be optimized for effective application migration. Heuristic algorithm and over-provision approach are used to provide energy efficient application migration [36].

Application migration to cloud requires several steps to provide secure, reliable, budget friendly and feasible solution. Several process used in app migration are code porting, code optimization, reusing code, improving security, hosting application. Each process adds distinct features necessary for cloud application.

2.3.1 Code porting

It is a process of developing new codebase from available codebase in a programming language. It could be done via manual porting, software assisted code conversion or by using middleware.

- Manual porting – In manual porting of application a group of developer reprograms the application into new language. This method provides better performance, however it requires too much effort on programming part. It is often accompanied by bugs that arise due to inefficient coding, human mistakes etc.
- Software assisted code conversion - Many applications are being migrated to web by converting existing C/C++ application like DAW plugins [29], plan.js motion planning library [16], faust audio DSP [34] etc. In this technique a software is used for converting available source code in one

language to another language. Zakai developed Emscripten compiler to translate source code in C/C++ language into JavaScript. The Emscripten tool reconstructs low level language into high level JavaScript language. This porting is done via machine compilation of available C/C++ codebase to LLVM bytecode and then converting this bytecode into JavaScript codebase through Emscripten compiler [61]. This conversion technique has also been proposed as the base for implementing research projects in JavaScript. This technique implements JavaScript in asm.js format that gives increased performance [33]. However this converted JavaScript often have many bugs, large code size, dead code and unoptimized code blocks [38]. Thus direct conversion to JavaScript and using it in cloud platform would lead to increased maintenance effort, more deployment cost, decreased performance and low security. In this paper to overcome this drawback of direct conversion. Various techniques are used in stages to optimize, refactor, secure and deploy the JavaScript converted codebase.

- Using Middleware – Many organization use middleware as a method for application migration. This technique requires a middleware between legacy application and underlying cloud platform [53]. The middleware executes over cloud platform and provides virtual execution environment for legacy application. This is fast deployment strategy but it is not resource efficient. Throughput of migrated application is often dependent on the performance of middleware. This method also requires different implementation for different cloud platforms [14]. Hence this approach is only used if middleware for a given application already exists.

2.3.2 Bug removal

Avaialbity of cloud resources is cost dependent. Unoptimized implementation of application on cloud causes low performance, defect development and requires more resource input. Bug repair is one of the most expensive stages in software develoment. Thus there is need to optimize code before migration to get better services within budget. Funcional requirements, non-functional requirements, new source code, architecture etc. are major defect origin. Joint application development (JAD), quality function deployment (QFD), certified reusable code are major defect prevention methods [28]. Certain tools like Eclipse [26], jetbrains [24], visual studio

[22] etc. are tools for refactoring code during development. Such tool accelerate the refactoring process and migration of application to cloud.

2.3.3 Code reuse

Various vendors provide different modules and code that could be reused for deployment of application in cloud. This allows faster deployment and more reliable cloud services. This also reduces the cloud deployment cost. Cloud subscription follows various payment scheme like pay as you go, charging depending on resource size etc. The customer can benefit by reusing codes and modules that have been efficient and already available [25]. It also reduces the bugs arising due to coupling of modules. Hence an application development strategy for cloud should include code reuse strategy to improve quality of cloud application. Mojica *et al.* surveyed the growing environment of mobile softare reuse [44]. The survey shown that there has been exponential growth of mobile applications. Android code for applications is being reused for different applications and reducing the cost of development.

2.3.4 Security

Cloud contains data of large number of users from different domain. The resources are allshared and needs to be managed effectively to ensure securtiy and privacy. Oren *et al.* explores various methods that are efficient for securing cloud applications. These methods are based on different factors to save energy and provide security in low power [45].

2.3.5 Cloud hosting

A cloud application needs a cloud server to host all the resource files. These cloud servers could be leased through organizations like Google, Amazon, iweb etc. or could be deployed by the customer itself using available infrastructure. These cloud hosting organizations provides needed infrastructure to setup the services that customer wants to provide. There is also provision to get installed available framework, virtual system, libraries etc. on cloud to ease the application deployment. A cloud application development strategy must employ cost effective and feasible hosting service to provide services to its customer [36].

Thus common techniques of cloud application development are discussed in this chapter. This lays down the base for further study of proposed methodology and enables to validate it through evaluation.

Chapter 3
Research Gap Analysis

In this chapter, we have highlighted the existing research gaps. We have formulated the problem considering important aspects like Cloud application development for beginners, immature developers and small scale organizations. Last section of this chapter discusses the objectives for the research problem.

3.1 Gap analysis

After researching various cloud development strategies, it was found that medium and large sized organization are more accustomed to use complex methodologies. The highly efficient cloud application development methodologies are not suitable for immature developers. The complex methodologies could not be carried out by individual in the organization. Thus, there is need for efficient methodology that could be used by individuals and small organization for faster cloud application development in less time.

3.2 Problem statement

Cloud application development is much platform specific approach. The developed applications cannot be migrated easily to another platform. There are many organizations who want to develop cloud applications or migrate their application to cloud. Development time and development cost both increases, when we develop applications and want to deploy services using cloud environment. The code portability i.e. ability of the code to be executed on other platform also falls to zero level. The complex methodologies of cloud development are not suitable for beginners, small organizations and startups as they have low budget and less experience to implement such methodologies. Thus, there is need for a development methodology that enables easy cloud application development and migration. Such organizations and immature developers need cost effective method to deploy their service as application in cloud environment.

3.3 Objective

Various objectives of proposed strategy in this thesis are as follows:

- Objective 1 - To study existing tools and techniques for cloud application development.
- Objective 2 - To compare and analyze various approaches for developing and migrating applications in cloud environment.
- Objective 3 – To propose and design an effective strategy for easier and faster development of cloud applications.
- Objective 4- To test and validate the proposed strategy.

Chapter 4

Design and Development of CRISP

In this chapter, the CRISP (Conversion, Reformat code, Isolate module, Sandboxing, Partition) strategy for cloud application development and migration has been proposed. CRISP is a process specific tool chain approach used for easy and faster deployment of cloud applications. This chapter discusses the thoughtful selection and implementation of processes, related tools and techniques applicable for CRISP. Step by step methodology of integrating selected tools are elaborated in detail.

JavaScript is selected as a medium for cloud application development. To develop cloud applications by coding in JavaScript or using conversion of C/C++ to JavaScript for cloud platform there is need to consider various important factors like development time, bug handling, code reusability, etc.

Cloud application development and migration through refactoring of software is one of the common application development methods. This thesis discusses cloud application development and migration of C/C++ applications by refactoring application to JavaScript. JavaScript is selected as the medium for application development and migration as it is one of the mostly used language, offers standard API for programming in HTML 5 and can be used on both client and server side. The popularity of JavaScript among developers is as shown through Github survey in Fig. 4-1.

In the proposed CRISP strategy, a process specific tool chaining strategy is used for faster and reliable cloud application development. This strategy has been validated for effective resource utilization. The different processes in proposed strategy are as follows:

1. Code development or Code conversion

2. Reformat code

3. Isolate module

4. Sandboxing

5. Partition

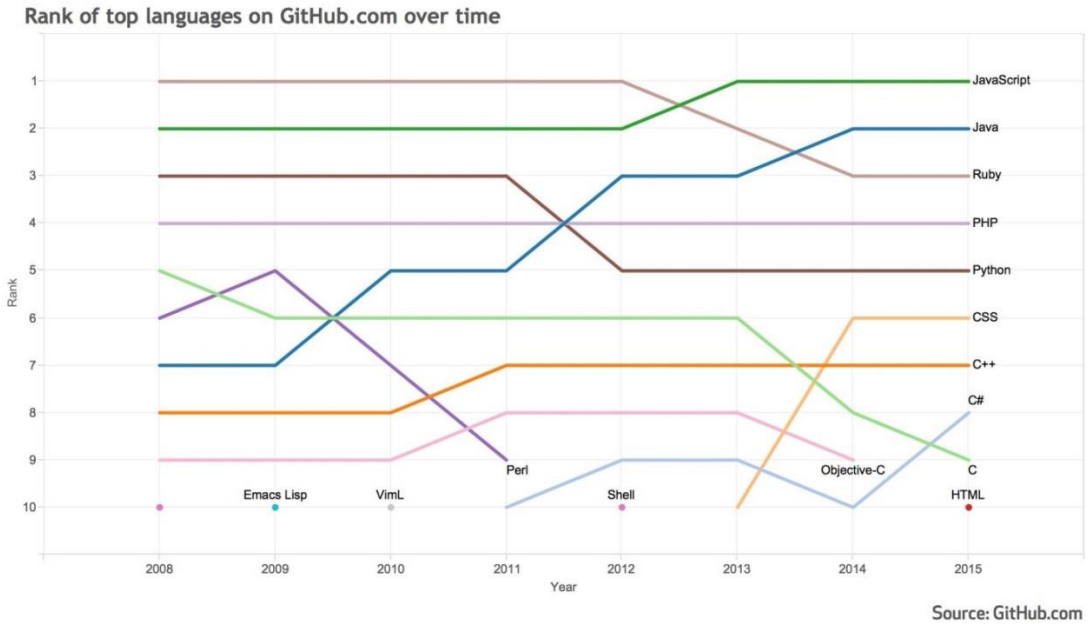


Fig. 4-1: Developer preference for language [25]

Each process focuses on certain factors of cloud application development as shown in Fig. 4-2 and allows to develop the application in easier, economic and resource efficient way. The pilot study has been carried out for selection of tools.

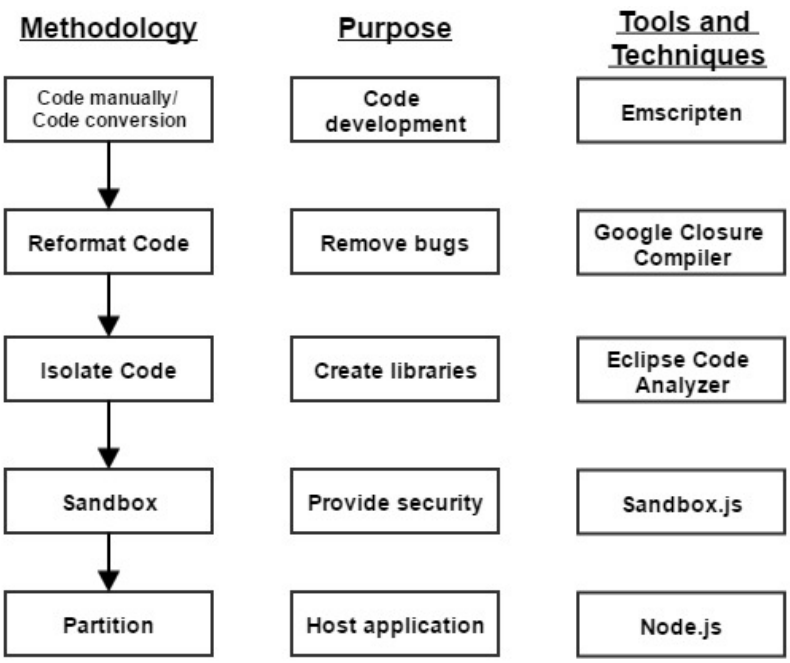


Fig. 4-2: Mapping of CRISP methodology to purpose and tools used

- **Scope of process specific tool chaining**

  The flowchart for whole code development for cloud is shown in Fig. 4-3. In proposed strategy there is provision to divide application into standalone cloud application or client-cloud application. Standalone cloud application doesn't contain modules to process data on client side. Client-cloud application allows to execute partial data locally on client side and rest of the data is processed on cloud side.

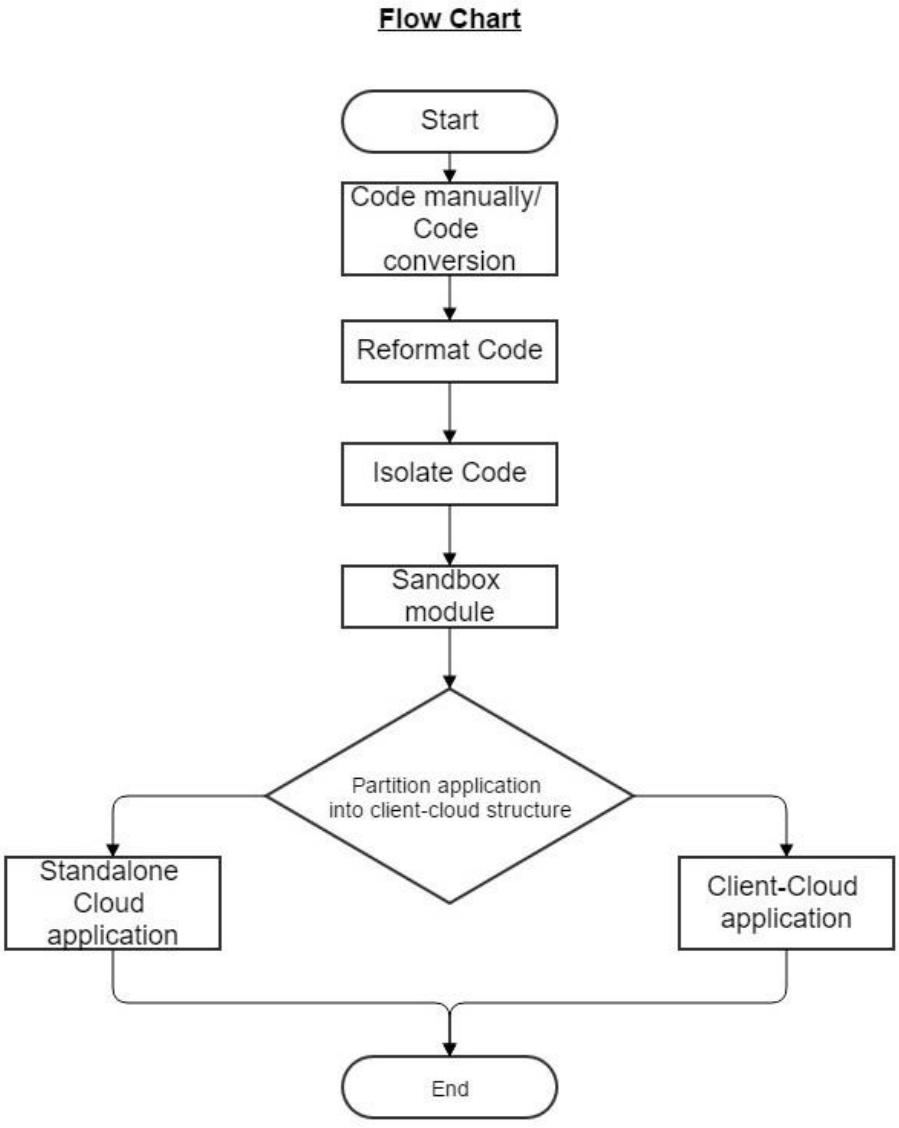


Fig. 4-3: CRISP strategy flow chart

Fig. 4-3 highlights the key processes of CRISP strategy applied for cloud application development. Various tools are available for each step. For pilot study of cloud application development we conducted experimental research on different application

development tools with various codebase and projects, ranging from 5 line codes to 400 lines of code. There are various tools available in market like Emscripten, Google closure compiler, jshint etc. for cloud application development. However each tool has its own advantage and disadvantage. Through the experiment conducted a process specific tool chain is found out that enables easy and faster application development. Table 4-1 summarizes available tools for a given process and also shows the preferred tool used in CRISP strategy that enable faster and easier cloud application development.

Table 4-1: Preferred tools for CRISP strategy from available tool set

S. No.	Process	Available tools	Preferred tools
1.	Code manually/ Code conversion	Emscripten, Cheerp, Firebug, Mandreel	Emscripten
2.	Reformat code	Google closure compiler, JS Lint, Eclipse code analyzer, JS beautifier	Eclipse code analyzer
3.	Isolate module	Eclipse code isolator, Intel XDK, YUI compressor	Eclipse code isolator
4.	Sandboxing	js.js, sandbox.js, jailed.js	Sandbox.js
5.	Partition	Google Cloud SDK, Firebug, Node.js	Node.js

4.1 Code manually/ Code conversion

To develop codebase for cloud application development, code can be either written manually or code of existing application could be converted into JavaScript. These two methods are explained below

4.1.1 Code manually

Code can be written in JavaScript for developing cloud application. JavaScript is high level language and provides rich API for development. The developers can merge existing code with HTML to provide graphical user interface.

Limitation - JavaScript code written manually may contain errors and bugs which needs to be removed for efficient cloud application deployment. The efficiency is improved by reformatting code in next step of strategy. Usually, an application is written in efficient C/C++ language and converted to less error prone JavaScript using transcompiler to get efficient JavaScript code.

4.1.2 Conversion using Emscripten compiler

Code conversion is used as a method to translate available application source code into JavaScript language to be used for further application development. In CRISP strategy, conversion is a process that uses Emscripten tool for converting C/C++ to JavaScript. This tool uses LLVM backend. The conversion process takes C/C++ codebase as input and converts it into JavaScript code and HTML files.

- Emscripten compiler

  The Emscripten compiler converts C/C++ GUI related content into HTML files while doing conversion. HTML file can be edited to add necessary changes. The conversion process of code from C/C++ to JavaScript is as shown in following Fig. 4-4.

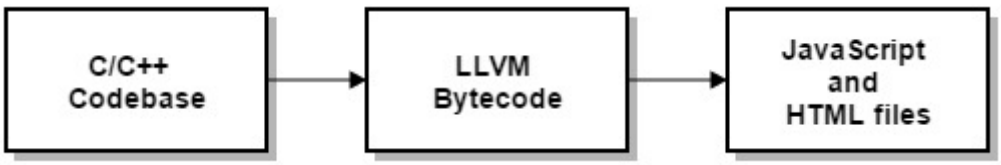


Fig. 4-4: Code conversion from C/C++ to JavaScript and HTML files

Emscripten is an effective tool that can be used to translate C/C++ code into JavaScript. It is the project developed under Alon Zakai working at Mozilla Foundation. It is a transcompiler that compiles C/C++ code into JavaScript. LLVM Clang is used as frontend in Emscripten. This allows source code in C/C++ to be executed in web browsers directly after code conversion. This facilitates use of web standards already present like HTML5, Web workers etc. and removes dependency on browser plugins. The process, functionalities and limitations of Emscripten tool are given in Table 4-2.

Table 4-2: Process, functionalities and limitation of Emscripten tool

Parameters	Characteristic
Tool Used	Emscripten
Process	i. Conversion of C/C++ to JavaScript starts by first compiling C/C++ codebase into LLVM assembly code by using LLVM Clang as frontend. Then LLVM assembly is translated into optimized JavaScript subset called as asm.js. The whole process is executed by Emscipten compiler. ii. The asm.js optimized subset acts as low level language and makes further optimization possible in browser. Arithmetic overflow , rounding errors and signing issues are checked by built-in Emscripten methods.
Functionalities	i. Applications compiled using Emscripten includes C and C++ runtime libraries implementation. These implementations allow converted code to independently carry out tasks like printing, accessing file system and memory allocation. ii. Emscripten provides facility of virtual file system using FS library, that is used in libc and libcxx. JavaScript has helper functions that can be used for adding and deleting files in virtual file system. iii. Direct C function calling is also supported in Emscripten. At compile time these C functions should be first named to avoid their optimization in the resultant compiled code. Then Emscripten uses its own cwrap function to wrap un-optimized C function and JavaScript function name is designated to it. Using cwrap function it becomes possible to use JavaScript variables as arguments in C functions.
Limitations	i. Using Emscripten, programs based on threads cannot be

	compiled because no threading is supported in JavaScript. ii. 64-bit integer implementation is not possible in JavaScript as numerical representation is done using 64-bit doubles. iii. Code generated using these code convertors, often contain bugs, error, dead code, un-optimized functions, redundant code and unreadable code structure.

This thesis proposes certain steps that optimizes and secures compiled JavaScript code. Each stage converts JavaScript code into a format suitable for being used in next stage. Finally optimized and refined library and module are generated suitable for application development in smartphones, clouds and servers.

4.2 Reformat code

This stage is crucial for formatting the code into more efficient and feature rich code. Certain code patterns result in more compilation time, inefficient use of resources, etc. There are also bugs introduced because of redundant code, inefficient translation, missing variables, dead code etc.

Reformat code is a method of modifying the code, optimizing it, improving security, removing bugs etc. It has been observed that refactored JavaScript code improves performance vastly when converted to asm.js format. We have proposed the method to refactor code by using directed graph as a technique to find dead code, unoptimized loops etc.

- **Eclipse code analyzer**

To reformat code, Eclipse code analyzer tool is used along with firefox code console to remove bugs, errors etc. This Eclipse tool uses code analyzer to analyze converted JavaScript code. The process, functionalities and limitation of Eclipse code analyzer are given in Table 4-3.

Table 4-3: Process, functionalities and limitation of Eclipse code analyzer

Parameters	Characteristic
Tool Used	Eclipse code analyzer

Process	i. Eclipse code analyzer is used to analyze the JavaScript code and find bugs, errors etc. Code analyzer contains code crawler, dependency generator and code console. Code crawler allows crawling whole codebase and finds static relationship present among code modules. ii. Dependency generator generates dependency matrix and dependency graph using static relationship found during code crawling. iii. Code analyzer after crawling whole codebase provides list of variables, functions classes and files. iv. The static relationship data along with the list of variables, functions, classes and files is used to create dependency matrix and dependency graph that helps to identify dead function, complex loops, unused variables etc. v. Code console is used to remove dead function and complex loop through manual coding.
Functionalities	i. Crawl whole JavaScript code base using code crawler. ii. Collect static relationship data present in JavaScript codebase. iii. Generate dependency matrix and dependency graph using static relationship data collected. iv. Provide code console to edit the JavaScript code and to remove bugs, errors, dead function etc.
Limitations	i. The depth of code crawling needs to be decided to get required data. Selecting crawling depth too low gives less information while crawling too deep gives excess data. Hence, crawling depth should be selected carefully. ii. Dead code, bugs and errors could only be identified. They need to be removed manually. No automatic correction is provided.

- **Dependency matrix and dependency graph generation process**

  A two dimensional dependency matrix is generated by listing variables, function and classes along both rows and columns. The relationship between each variable, function and class is shown by this matrix. A row or column element can be a variable, function class or file. An element in a row is related to other element, if the value of intersection cell of the given row and column is 1 or -1 otherwise it is 0 to represent null relationship. The relationship flow from row element to column element is marked as 1. The relationship flow from column element to row element is marked as -1. When there is no relationship then it is marked as 0. An excel fie output showing two dimensional dependency matrix is shown in Table 4-1.

- **Dependency matrix and dependency graph generation example for calculator**

  A calculator program in C language is taken to generate dependency matrix and dependency graph. The calculator program is fetched to Eclipse code analyzer. The static relationship data is created and dependency matrix is generated as shown in Table 4-4.

Table 4-4. Dependency matrix for calculator program

	Functions	main()	input()	select()	add()	subtract()	multiply()	division()	output()
1	main()	0	1	0	0	0	0	0	0
2	input()	-1	0	1	0	0	0	0	0
3	select()	0	0	0	1	1	1	1	0
4	add()	0	0	-1	0	0	0	0	1
5	subtract()	0	0	-1	0	0	0	0	1
6	multiply()	0	0	-1	0	0	0	0	1
7	division()	0	0	-1	0	0	0	0	1
8	output()	0	0	0	-1	-1	-1	-1	0

The generated dependency matrix is used to draw dependency graph as shown in Fig. 4-5. A node in a graph can represent variable, function, node or file. The dependency graph contains start and end node. A forward relation between nodes is represented by 1. A reverse relation between nodes is shown by -1 and 0 represents no relation between nodes.

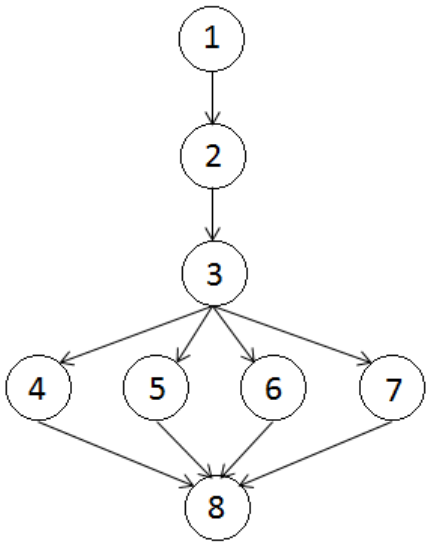


Fig. 4-5: Dependency graph for relation matrix of calculator

Machine generated code is often cluttered and generated in unreadable format. Code reformat tool includes JS beautifier library to unscramble generated code. This library takes JavaScript code as input and generates structured JavaScript code having variables and functions formatted in readable form.

All this modification allows generating optimized JavaScript code in this stage. This code can be easily modified further to divide it into reusable modules in next stage.

4.3 Isolate module

Due to increasing market demand there is need for economic and fast deployment strategy. Code reusability is one of the major advantage of JavaScript. Machine generated code modules increases reusability and performance. This facilitates faster deployment on multiple clouds and reusing same modules for different projects.

Reformatted code obtained in previous stage is often large and contains translated JavaScript code along with asm.js library in one file. There is significant overhead when JavaScript compiles such large JavaScript files. The large JavaScript code can be divided into libraries as explained through Table 4-5. These divided libraries enable code reuse and reduces compilation time.

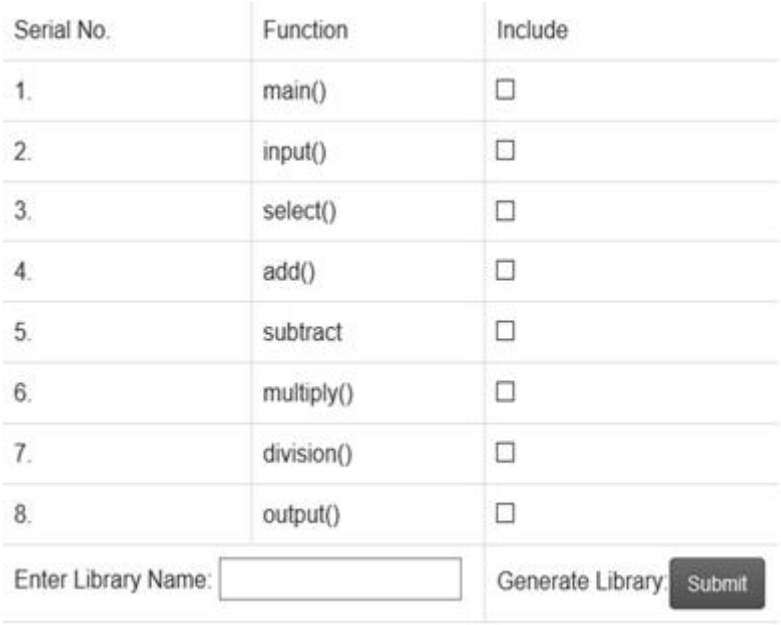
Manually dividing code into libraries would be a laborious task and time consuming process. Separation of code into libraries often introduces bugs and error. An effective

method should be used to divide large codebase to libraries. Before dividing codebase, the original file should be backed up in repository to make it available for future use. The division process of code should follow following condition to provide executable libraries:

- The common class and function should be implemented in all divided libraries.
- Division process should not break dependency present between function and classes.
- No modification should be done in original file.
- If needed HTML interface should be generated for different libraries

Table 4-5: Process, functionalities and limitation of Eclipse code isolator

Parameters	**Characteristic**
Tool Used	Eclipse code isolator
Process	i. In this step for dividing code into libraries, code isolator module present in Eclipse is used. ii. The dependency matrix also contains dependency of JavaScript function and files. From dependency graph we can identify which function depends on which other function. In this tool, after reformatting code we get a list of function available, along with checkbox for each function as shown in Fig. 4-6. iii. According to requirement we can select necessary function by checking the checkbox to include that function in separate library. After checking all the function we need to click submit button. The tool creates a library containing all the checked function along with functions on which given checked function depends. iv. This process can be repeated to create more sub libraries. Thus by this tool we effectively divide given library into sub-libraries. These sub-libraries allow better reuse of code and

	less compilation time. This improves overall performance of applications in which it is used. The tool uses YUI compressor library to reduce the size of the code. This allows faster downloading of libraries when used in websites or cloud applications.  **Fig. 4-6**: The code isolator module v. Finally these libraries are packaged with respective html files obtained during conversion process to create a module. This module can be used for different cloud application development.
Functionalities	i. Code console in code isolator allows us to modify JavaScript code. It is a text area that takes JavaScript code from previous stage and provides necessary editing functionality. Now dependency graph created is used for identifying dead function, complex loops, incomplete functions, unused variables etc. as follows: - A dead function or unused variable can be identified when a function node or unused variable is separate from graph. The dead function can be modified to make it executable. The unused variable can be eliminated.

	- A complex loop can be identified by cluster of loop cycles present in graph. It can be optimized by removing the unnecessary loop iteration. ii. Code reformat tool includes JS beautifier library to unscramble generated code. This library takes JavaScript code as input and generates structured JavaScript code having variables and functions formatted in readable form.
Limitations	i. The given module may still contain some bugs and errors that can cause fault and security issue in platform being used. Next stage implements methods to solve this problem. ii. Managing sub-libraries could become difficult as codebase increases.

4.4 Sandboxing

A cloud based application may need to be integrated with other existing applications. Converted code usually have dead code and complex structure. Even after reformating code and isolating them into libraries certain bugs may still exist. This may cause issues which may lead to security holes, hacking and project failure. This security issue is resolved by sandboxing the converted modules. This technique restricts the the faults and bugs within sandbox.

Sandboxing is a method that encapsulates codes inside a closed environment and provides restricted access to resources. The security is provided by making it necessary for any process to have execution privilege. A process is executed only if it has a particular execution privilege. The sandboxing process is explained through Table 4-6.

- **Building sandbox.js library**
    - The sandbox.js library is developed by converting V7 browser engine to JavaScript library using code conversion. This JavaScript library acts as sandbox.

- When executed the sandbox library encapsulates other scripts and executes the code inside it. The security attack done on module will only effect code inside sandbox. The attack doesn't affect other programs and thus security is achieved through sandboxing.

Table 4-6: Process, functionalities and limitation of sandbox.js

Parameters	**Characteristic**
Tool Used	Sandbox.js
Process	i. To provide better security and control over the converted JavaScript code we sandbox the module libraries created in previous stage. ii. The sandboxing method used in this stage follows principle of least privilege. This allows us to contain the JavaScript libraries and other files in a constricted environment. By sandboxing our module we restrict the modules access to resources of underlying platform. This ensures that when any converted JavaScript code is executed in JavaScript engine it won't cause bugs or illegal operation. iii. To provide fine grained security and sandboxing we use sandboxing libraries like sandbox.js or jailed.js. These libraries execute scripts inside sandbox. The data flow is done by using secure API between modules. iv. The attack effects only the sandbox and doesn't affect underlying platform. Thus, sandbox secures the platform.
Functionalities	i. Encapsulate the application code inside restricted environment. ii. Allows to control execution privileges of each module individually.
Limitations	i. It could be used to secure modules against certain attacks only. It

	cannot provide security against many attacks. ii. The sandboxing makes program execution slow.

At this stage we have secured the JavaScript module by using sandboxing. Now sandboxed modules can be efficiently used for cloud application development. In next step we discuss deploying application to cloud.

4.5 Partition

With increase in size of cloud application, a platform may not provide enough resources to execute the application. In resource constraint scenario it is more effective to divide application into client-cloud module.

Partitioning is the final step in our methodology to deploy our JavaScript application to cloud. Here partitioning is the method of moving resource intensive module or whole application to cloud server. The whole application is divided into client module and remote application module. The client module runs on client side like laptop, smartphone, microcontrollers etc. The remote application module runs on cloud. However cloud can be used to host client module also. In this case client module is loaded in browser as client accesses the cloud service.

To deploy an application to cloud we need a cloud platform that can host the application module and provide resources for execution. We can either opt to subscribe for available cloud services like Google cloud platform, Amazon EC2 etc. or we can set up our own cloud using available software stacks.

The JavaScript modules can be hosted through cloud platform and makes it possible to provide various services through cloud. Different cloud services provide different methods to host application. We will be using Google cloud service to host application modules using node.js application runtime as explained through Table 4-6.

Table 4-7: Process, functionalities and limitation of Google cloud SDK

Parameters	Characteristic
Tool Used	Google cloud SDK with node.js runtime
Process	**A. Create Google cloud storage bucket** We create Google cloud storage bucket as shown in Fig. 4-7 using following steps: i. Using Google cloud platform console access Google cloud storage browser. Google cloud storage browser is the utility to maintain various file services. ii. Create bucket by clicking its option. iii. Give name of the bucket iv. Specify storage class for bucket. v. Decide location of the bucket. vi. Finally click create vii. On successful completion of steps following prompt will be generated: "There are no objects in this bucket" **B. Upload cloud application files** We have created a host area from where we can serve files. Now we need to upload application files for hosting by using following steps: i. Access the bucket created in previous step ii. Upload the application modules iii. Using file dialog browse and select desired file. Specify this file as index file Files name, type, size and last modified date are shown in bucket after successful upload. The index file loads the rest of the cloud application when accessed through browser or client-side module. Thus cloud application module is hosted and is used to provide service. **C. Provide share privilege to files** Finally we need to provide share privilege to allow file hosting

<table>
<tr><td></td><td>through cloud by following steps:

i. Using cloud storage browser, browse each of the files uploaded and created.
ii. Click checkbox to share publically

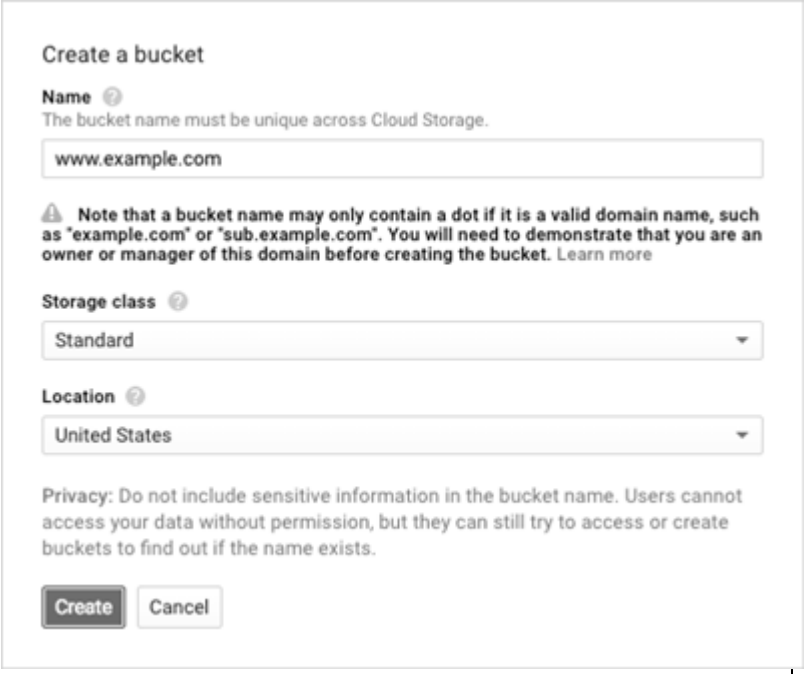
Thus application is now serviced through cloud. With this cloud service can be provided to different customers.
 

Fig. 4-7: Bucket creation</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Functionalities</td><td>i. Create cloud hosting space for hosting files.
ii. Control access privileges of each resource. This step provides security by restricting file access.
iii. Add, delete and modify files in storage bucket.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Limitations</td><td>i. To access extra services, subscription is needed.
ii. Limited bandwidth for free cloud hosting</td></tr>
</table>

With this final stage of methodology, easy and secure cloud application development using JavaScript is made possible. The strategy is evaluated using a case study of portable language recognizer and translator in next chapter.

Chapter 5

Evaluation and Validation

To evaluate CRISP strategy, we have built a portable recognizer and translator in JavaScript using C++ codebase and migrated it to cloud. The C++ codebase has functionality to captures image and recognize the English text using OCR library. A text editor is present to edit text and then fetch it to translator. Translator module translates from english language to another language.

The testbed for Cloud application is given in Fig. 5-1.

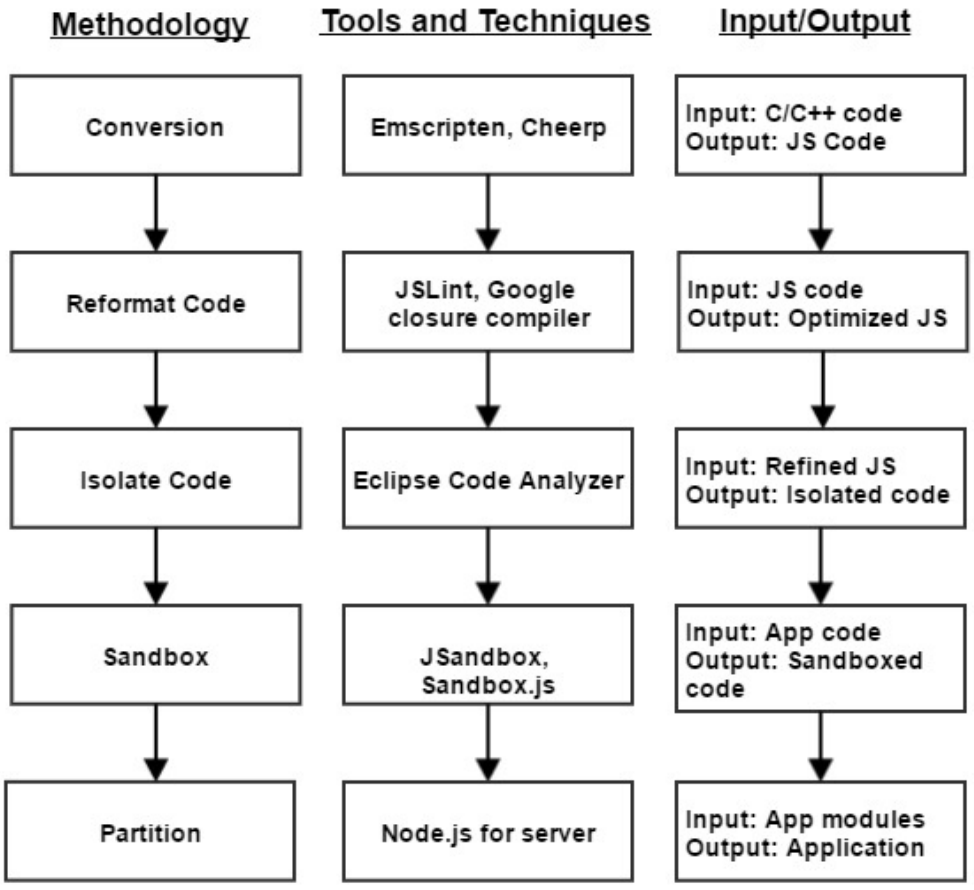


Fig. 5-1: CRISP strategy testbed

5.1Cloud application development

We have portable language recognizer and translator as C++ application and needs to be deployed on cloud. To convert C++ application into JavaScript based cloud modules, we implement strategy by following steps:-

i. Code/Conversion

The C++ code of application is fetched to Emscripten compiler and we get converted JavaScript codebase as shown in Fig. 5-2.

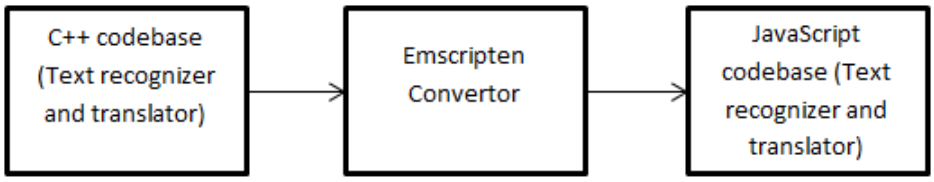


Fig. 5-2: Code conversion from C++ code to JavaScript code

The Emscripten compiler is accessed through command console as shown in Fig. 5-3. The converted JavaScript OCR library is included with HTML files as shown in Fig. 5-4.

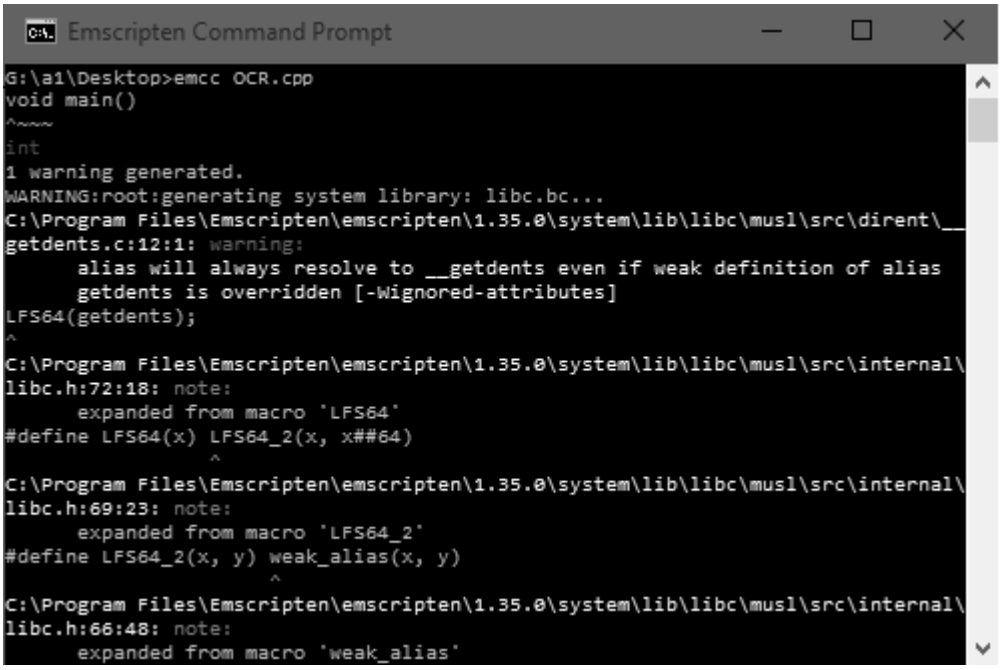


Fig. 5-3: Emscripten is used to convert OCR.cpp to JavaScript code

ii. Reformat code

In this step, we use Eclipse code reformat tool to identify and remove dead code, bugs and errors as shown in Fig. 5-4, Fig. 5-6 and Fig. 5-7. We also add certain function to control brightness and contrast of image captured. Certain functions are written in each module to add needed functionality. This improves efficiency of translated code and utilizes cloud resources effectively.

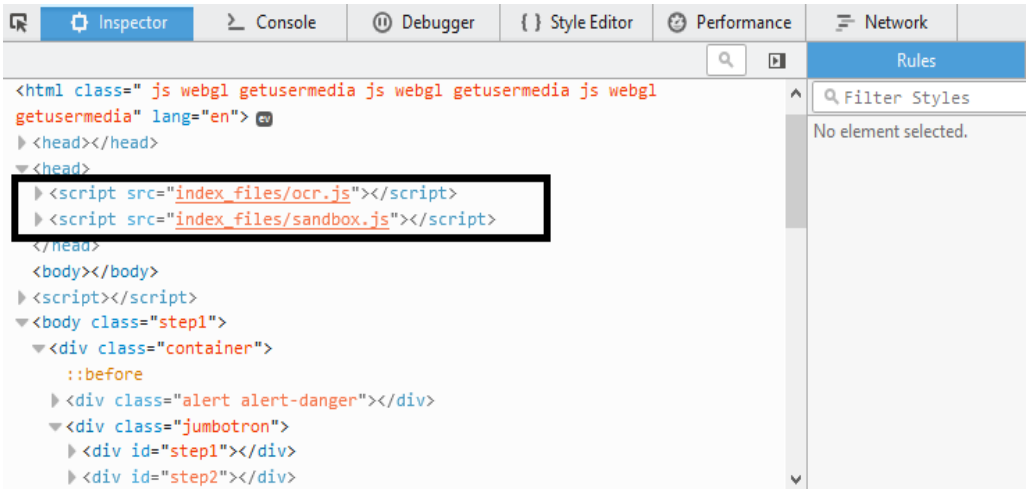


Fig. 5-4: Camera module reformatted to remove bugs and sandbox.js library is integrated

iii. Isolate module

The reformatted code is taken and edited in Eclipse code isolator tool. Dependency graph is generated and is used to identify dependency of function.

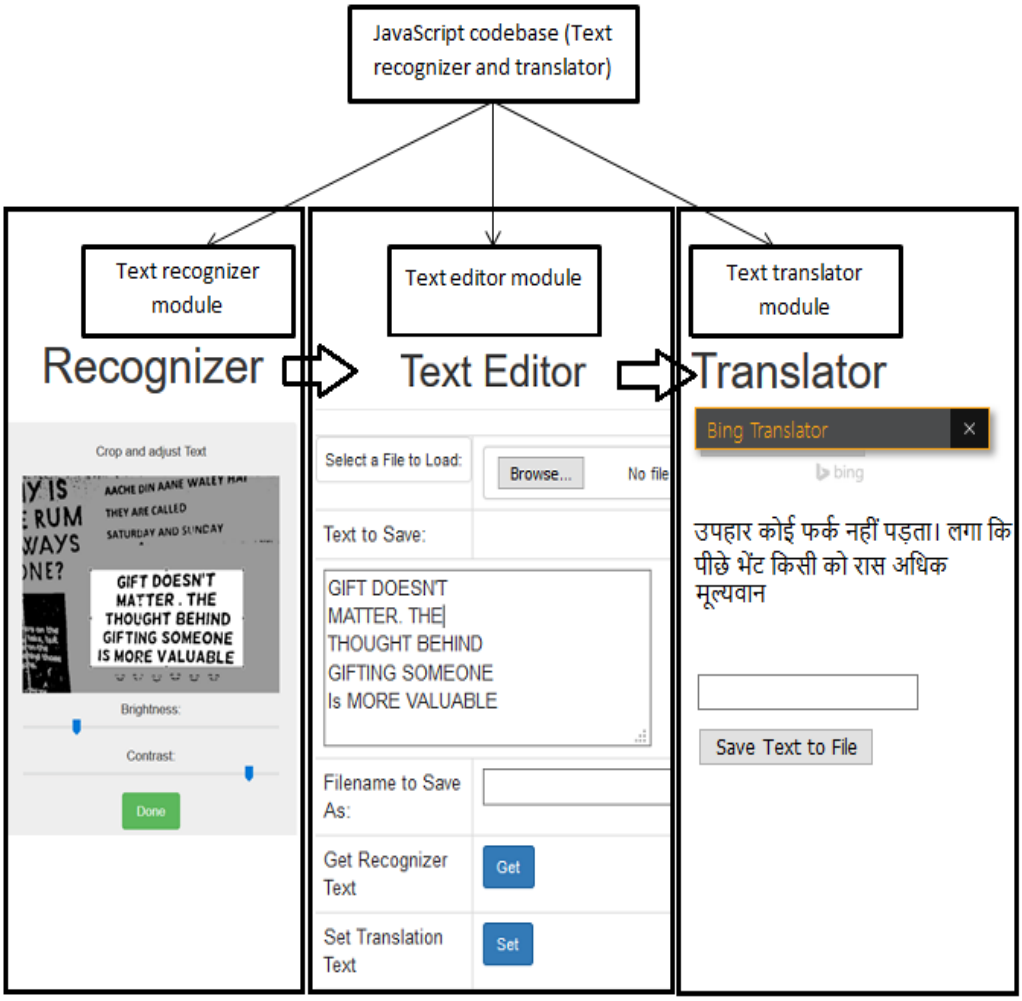


Fig. 5-5: The single module is divided into three independent modules

Then we separate whole JavaScript codebase into ocr.js library, textbox.js library and translator.js library as shown in Fig. 5-5. Respective libraries are packaged with html files to generate recognizer module, text editor module and translator module. This enables us to reuse a given module thus reduces budget for future project by using same module for development.

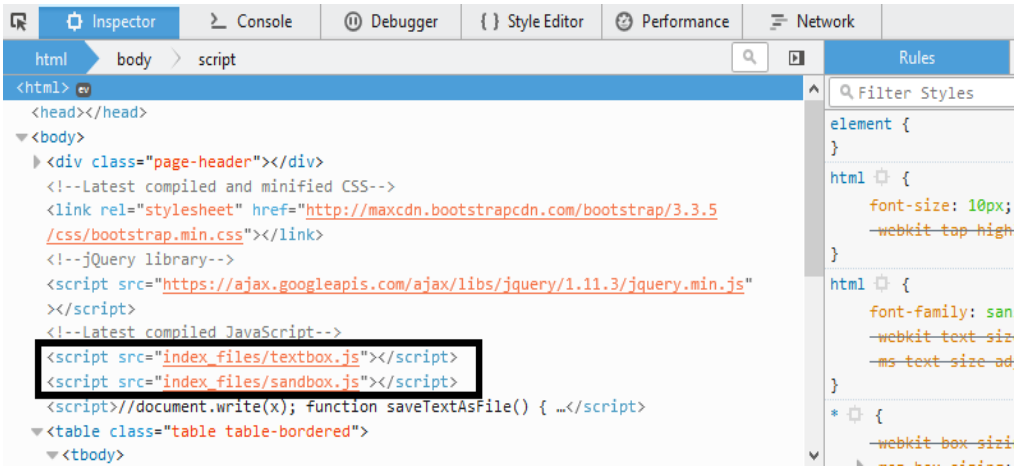


Fig. 5-6: Text editor module reformatted and sandbox.js library is integrated

Care should be taken while dividing library to include all dependency code in each library. This ensures bug free library.

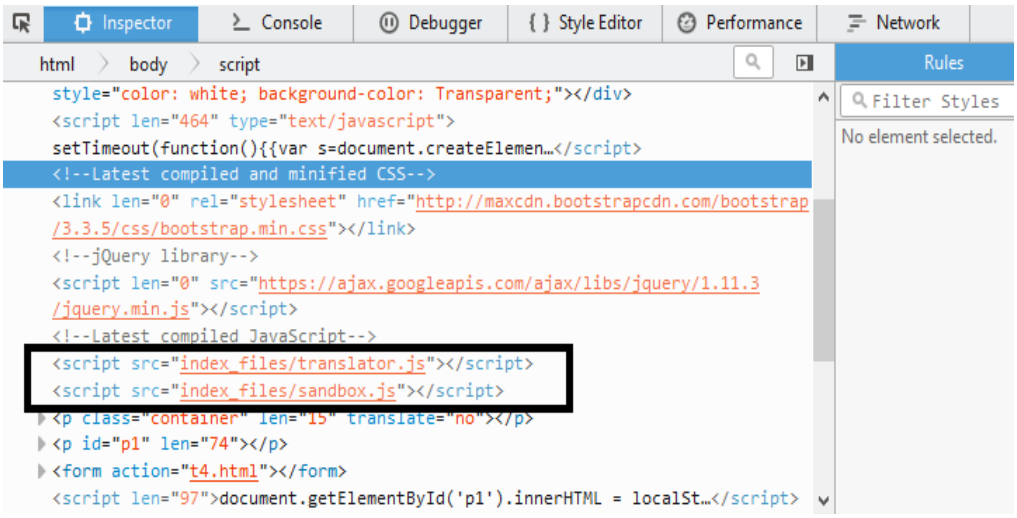


Fig. 5-7: Translator module reformatted and sandbox.js library is integrated

iv. Sandbox module

Modules from previous stage are sandboxed using iframe attribute of HTML and using sandbox.js library. Each sandbox is configured individually to

restrict certain privileges for each module. Sandbox.js library is integrated in each module as shown in Fig. 5-4, Fig. 5-6 and Fig. 5-7. For example, to provide security we will be allowing same origin policy for text recognizer module to allow camera access as shown below:

<iframe src="recognizer.htm" sandbox="allow-same-origin"></iframe>

This allows to secure modules from certain security threats. Thus we provide security to modules

v. Partition

In this stage we move all resource modules to cloud server. We use node.js as backend server over windows 10 platform with specification 8GB RAM, 2 TB hard drive, Intel i7 processor, bandwidth 1mbps to provide cloud services. As modules are sandboxed, they provide security to execute them over cloud and client side. In this step we create a bucket to upload all modules and give privileges to execute over cloud environment.

The application is built on client-cloud model. When the application is accessed from client side then client side files are downloaded. The local processing for text recognition and text processing is done on client side. The resource intensive processing like translation is done on cloud side. Thus our strategy balances the processing overhead by partitioning the application and offloading resource intensive processing.

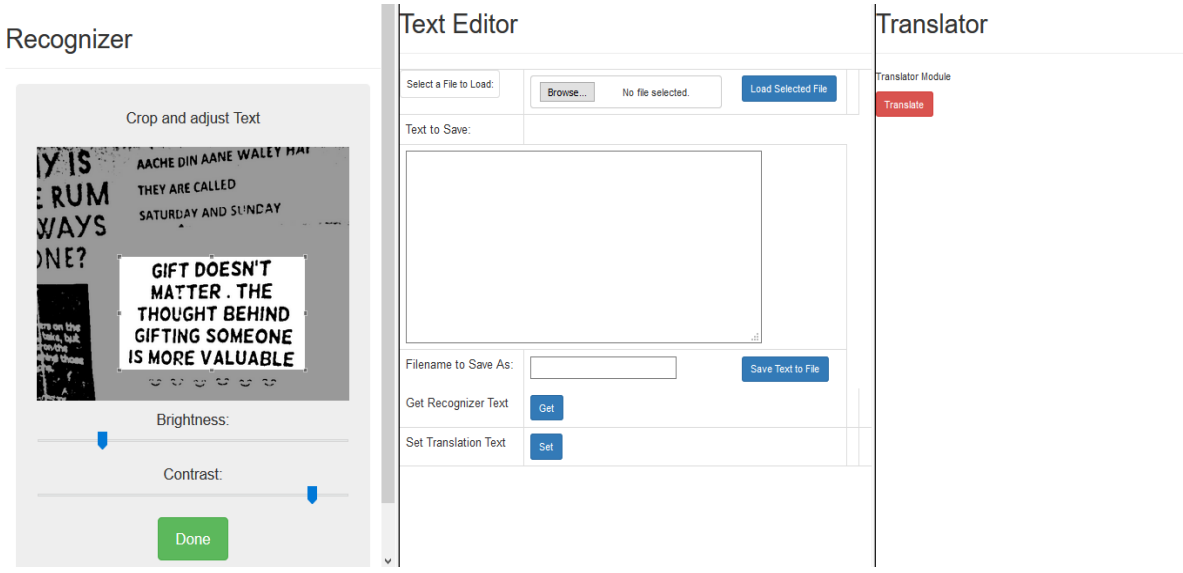


Fig. 5-8: Text recognizer and translator cloud application

The application developed is shown in Fig 5-8. Here each module text recognizer, text editor and text translator are separate modules and sandboxed separately. Each module is independent of other module and can execute independently. This allows us to use a given module for different application thus we achieve reusability.

5.2 Working of portable language recognizer and translator

The portable language recognizer and translator capture an image and recognize text using converted JavaScript library. The recognized text can be edited in text editor module. Finally, the recognized text is translated into Hindi text using cloud service. The whole process happens in step by step process as shown in Fig. 5-9.

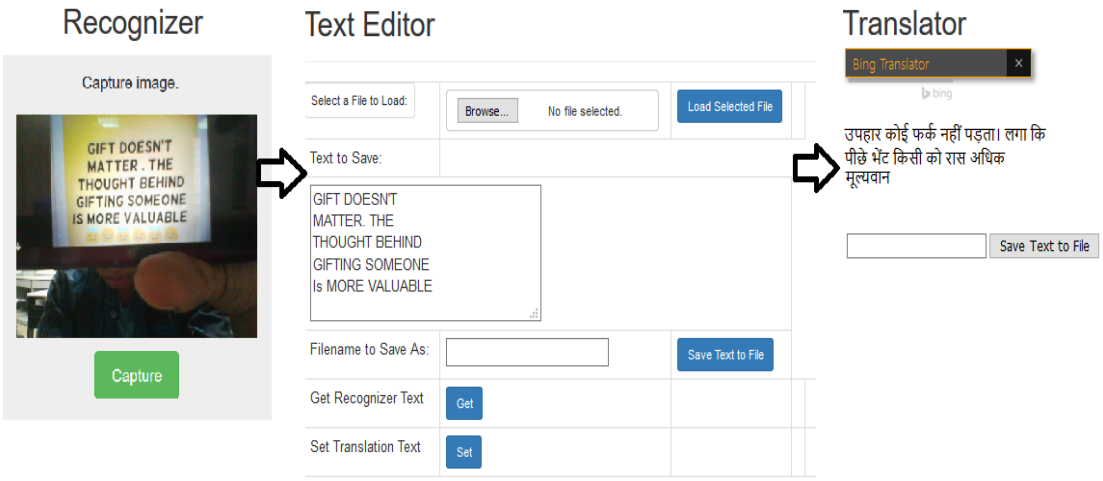


Fig. 5-9: Working of portable language recognizer and translator

5.3 Resource consumption comparison

We have executed the developed application over local cloud and we measure the processing power used for each module. The C++ application and Un-optimized JavaScript application are also executed and processing power used is measured. All the performance data is tabulated in Table 5-1. The relative CPU RAM usage comparison chart for 1 GB RAM is shown in Fig. 5-10. Thus we get overall perspective of CPU RAM consumption.

Table 5-1: Relative CPU RAM usage of Native C++, Un-optimized JavaScript and cloud application

Module\Platform	Native C++	JavaScript (Un-optimized)	Cloud JavaScript (Optimized)
Recognizer module	8%	38%	25%
Text editor module	3%	18%	12%
Text translator module	7%	24%	16%

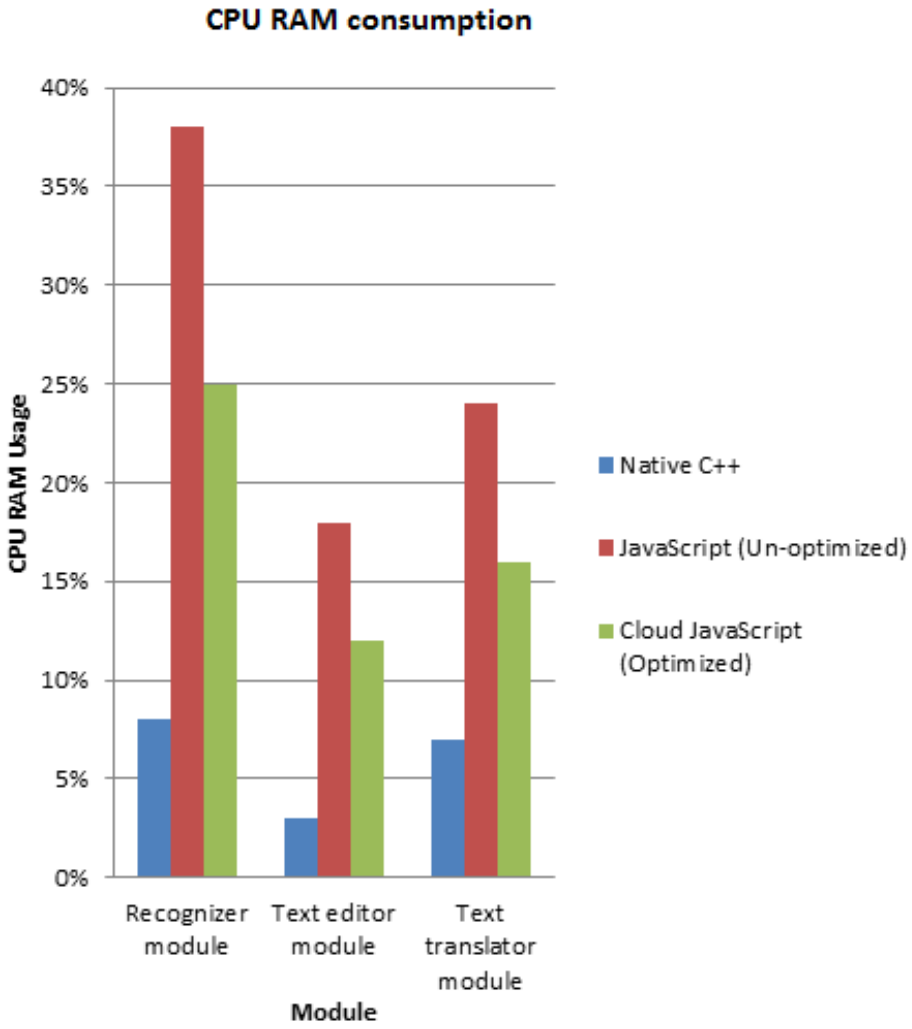


Fig. 5-10: CPU RAM usage for different modules

The CPU processing power utilized by cloud application developed through CRISP strategy is less compared to un-optimized JavaScript code. The processing power used

is within acceptable limit to provide effective service. The proposed strategy is useful to create cloud application.

Thus evaluation of CRISP strategy is successful and proved that it can be used for cloud application development.

Chapter 6

Conclusion and Future work

6.1 Conclusion

The demand for fast cloud application development is increasing in IT field. There are many organizations who want to develop cloud applications or want to migrate their application to cloud. The proposed CRISP application development and migration strategy allows such organizations to develop cloud application and to migrate their services to cloud. This benefits them by decreasing the deployment time to cloud, by reducing bugs, improving security, increasing efficiency and by reducing hosting charges. The methodology can be easily applied on different applications and enables to provide service through cloud. When using this strategy the resource consumption decreases greatly. Also, the performance and efficiency of application developed increases. Finally, it is concluded that the proposed strategy gives better efficiency and performance than un-optimized code and enables effective cloud application development that can be used to provide cloud service.

6.2 Unique Contribution

This thesis made a commendable contribution by providing CRISP strategy for immature cloud application developers and small size organizations. CRISP strategy is a process specific tool chaining technique which not only made cloud application development easier but also faster. The strategy also enabled easy deployment and migration on cloud applications. The selection and considerate sequencing of appropriate processes, tools and techniques have made CRISP strategy a successful endeavor. JavaScript language is one of the mostly used language and can be used easily by developers. It can be used for both client and server side programming. This language provides multi-platform execution facility through different runtime engines and utilizes less CPU RAM. The JavaScript code is easily reusable and reduces development time. Thus JavaScript has been used effectively for cloud application development.

6.3 Limitation

Limitations of CRISP strategy are as follows:

- It is useful only for lightweight application. Applications requiring high processing power could not be efficiently developed using proposed strategy.
- It is not suitable for hard real time applications. Application that are time critical require high efficiency and performance. CRISP doesn't provide required performance time for hard real time applications.

6.4 Future work

CRISP strategy can be further improved to add new cloud application development technique while keeping it simple to adopt. The application developed can be made more efficient by performing more rigorous reformatting of translated JavaScript code. Customized tools can be developed for partitioning application into client and cloud. Code reusability can be increased by adopting better modularization of translated code.

References

[1] N. Ansari, S. Patil, A. Navada, A. Peshave, and V. Borole, “Online C/C++ compiler using cloud computing”, *Multimedia Technology (ICMT),2011 International Conference on IEEE*, pp. 3591-3594, 2011.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski and M. Zaharia, “A view of cloud computing”, *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, 2010.

[3] R. Barnes and M. Thomson, "Browser-to-Browser Security Assurances for WebRTC", *IEEE Internet Computing*, vol. 18, no. 6, pp. 11-17, 2014.

[4] P. V. Beserra A. Camara R. Ximenes A. B. Albuquerque and N. C. Mendonca, “Cloudstep: A step-by-step decision process to support legacy application migration to the cloud”, *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the IEEE*, pp. 7-16, 2012.

[5] I. Bochon, V. Ivens and R. Nagel, “Challenges of cloud business process management”, *Cloud Computing for Logistics*, pp. 119-139, 2015, Springer International Publishing.

[6] J. Brodkin, “Gartner: Seven cloud-computing security risks”, *Infoworld*, pp. 1-3, 2008.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, “Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services”, *Algorithms and architectures for parallel processing*, pp. 13-31, 2010, Springer Berlin Heidelberg.

[8] D. Catteddu, “Cloud Computing: benefits, risks and recommendations for information security”, *Web Application Security*, pp. 17, 2010, Springer Berlin Heidelberg.

[9] I. Chaniotis, K. Kyriakou and N. Tselikas, "Is Node.js a viable option for building modern web applications? A performance evaluation study", *Computing*, vol. 97, no. 10, pp. 1023-1044, 2014.

[10] M. Cho, Y. Han, M. Kim and S. Kim, "O2WebCL: an automatic OpenCL-to-WebCL translator for high performance web computing", *J Supercomput*, vol. 71, no. 6, pp. 2050-2065, 2014.

[11] G. Chun, S. Ihm, P. Maniatis, A. M. Naik, and A.Patti, “Clonecloud: elastic execution between mobile device and cloud”, *Proceedings of the sixth conference on Computer systems* ACM, pp. 301-314, 2011.

[12] "Action against Cybercrime", *Cybercrime*, 2016. Available: http://www.coe.int/t/dghl/cooperation/economiccrime/cybercrime/cy-activity-interface-2010/.

[13] M. de Assunção, A. di Costanzo and R. Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters", *Cluster Computing*, vol. 13, no. 3, pp. 335-347, 2010.

[14] R. Di Cosmo, M. Lienhardt, J. Mauro, S. Zacchiroli, G. Zavattaro and J. Zwolakowski, “Automatic Application Deployment in the Cloud: from Practice to Theory and Back”, *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 42, 2015, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[15] M. Ficco, F. Palmieri, and A. Castiglione, “Modeling security requirements for cloud-based system development”, *Concurrency and Computation: Practice and Experience*, vol. *27*, no. 8, pp. 2107-2124, 2015.

[16] C. Freeman, plan. js: A Motion Planning Library for Javascript, 2016. Available: freemancw.github.io.

[17] S. Frey and Hasselbring, “An Extensible Architecture for Detecting Violations of a Cloud Environment's Constraints during Legacy Software System Migration”, *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on IEEE*, pp. 269-278, 2011.

[18] B. Furht and A. Escalante, *Handbook of cloud computing*, vol. 3, 2010, New York: Springer.

[19] J. Galenson, C. Rubio-González, S. Chasins and L. Gong, “Research. js: Evaluating Your Research on the Web”, 2012.

[20] A. Greenberg *et al.* "The cost of a cloud: research problems in data center networks", *ACM SIGCOMM computer communication review*, vol. 39, pp. 68-73, 2008.

[21] “A comprehensive look at the path to cloud migrations”, 2014, Available: http://searchcloudcomputing.techtarget.com/essentialguide/A-comprehensive-look-at-the-path-to-cloud-migrations

[22] “Visual studio”, 2016, Available: http://visualstudio.com/

[23] "Gartner cloud survey",2014,Available: http://www.gartner.com /newsroom /id/1684114

[24] "Jet Brains", 2016, Available: http://www.jetbrains.com/

[25] "Language trends on github", 2016, Available: https://github.com/blog/2047-language-trends-on-github

[26] "Eclipse", 2016, Available: https://www.eclipse.org/

[27] W. Iqbal, M.N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud", *Future Generation Computer Systems,* vol. 27, no. 6, pp. 871-879, 2011.

[28] C. Jones, "Software Defect Origins and Removal Methods", *Namcook Analytics,* 2013.

[29] J. Kleimola, "DAW Plugins for Web Browsers", *Proc. 1st Web Audio Conference (WAC-15), Paris,* 2015.

[30] M. Klems, M. Nimis and S. Tai, "Do clouds compute? a framework for estimating the value of cloud computing", *Designing E-Business Systems. Markets, Services, and Networks,* pp. 110-123, Springer Berlin Heidelberg.

[31] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading" *INFOCOM, 2012, Proceedings IEEE*, pp. 945-953, 2012.

[32] Ronald L. Krutz and Russell Dean Vines. "*Cloud security: A comprehensive guide to secure cloud computing*." Wiley Publishing, 2010.

[33] Leopoldseder, L. Stadler, C. Wimmer, and H. Mössenböck, "Java-to-JavaScript translation via structured control flow reconstruction of compiler IR", *Proceedings of the 11th Symposium on Dynamic Languages*, pp. 91-103, ACM, 2015.

[34] S. Letz, S. Denoux, Y. Orlarey, & D. Fober, "Faust audio DSP language in the Web", *Linux Audio Conference,* 2014.

[35] J.J. Levandoski, D.B. Lomet, M.F. Mokbel, and K. Zhao, "Deuteronomy: Transaction Support for Cloud Data", *CIDR,* vol. 11, pp. 123-133, 2011.

[36] B. Li, J. Li, J. Huai, T. Wo, Q. Li and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments.", *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on IEEE,* pp. 17-24, 2009.

[37] Y.F. Li, P.K. Das, and D.L. Dowe, “Two decades of Web application testing—A survey of recent advances.”, *Information Systems*, vol. 43, pp. 20-54, 2014.

[38] C. Low, Y. Chen and M. Wu, “Understanding the determinants of cloud computing adoption”, *Industrial management & data systems*, vol. 111, no. 7, pp. 1006-1023, 2011.

[39] J.L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I.M. Llorente, “Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios”, *Concurrency and Computation: Practice and Experience*, vol. 27, no.9, pp. 2260-2277, 2015.

[40] L. A. Meyerovich, & B. Livshits, “ConScript: Specifying and enforcing fine-grained security policies for Javascript in the browser”, *Security and Privacy (SP), 2010 IEEE Symposium on IEEE*, pp. 481-496, 2010.

[41] P. Mohagheghi, & T. Sæther, “Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the remics project”, *Services (SERVICES), 2011 IEEE World Congress on IEEE*, pp. 507-514, 2011.

[42] I.J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A.E. Hassan, “A large-scale empirical study on software reuse in mobile apps”, *Software, IEEE*, vol. 31, no. 2, pp. 78-86, 2014.

[43] M. Peter, and T. Grance. "The NIST definition of cloud computing”, pp. 20-23, 2011.

[44] A. Ojamaa and K. Duuna, “Assessing the security of Node. js platform”, *Internet Technology And Secured Transactions, 2012 International Conference for IEEE*, pp. 348-355, 2012.

[45] Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A.D. Keromytis, “The Spy in the Sandbox--Practical Cache Attacks in Javascript”, *arXiv preprint arXiv:1502.07373,* 2015.

[46] P. H. Phung, M. Monshizadeh, M. Sridhar, K.W. Hamlen and V. N. Venkatakrishnan, (2015). “Between Worlds: Securing Mixed JavaScript/ActionScript Multi-party Web Content”, *Dependable and Secure Computing, IEEE Transactions on IEEE*, vol. 12, no.4, pp. 443-457, 2015.

[47] J.G. Politz, S. Eliopoulos, A. Guha, and S. Krishnamurthi, “ADsafety: Type-based verification of JavaScript sandboxing”, *arXiv preprint arXiv:1506.07813,* 2015.

[48] C. Rohlf, and Y. Ivnitskiy, “The security challenges of client-side just-in-time engines”, *IEEE Security & Privacy*, vol. 2, pp. 84-86, 2012.

[49] D.G. Rosado, R. Gómez, D. Mellado, and E. Fernández-Medina, “Security analysis in the migration to cloud environments”, *Future Internet*, vol. *4*, pp. 2, pp. 469-487, 2012.

[50] S. Subashini, and V. Kavitha, “A survey on security issues in service delivery models of cloud computing”, *Journal of network and computer applications*, vol. *34*, no. 1, pp. 1-11, 2011.

[51] H. Takabi, J. Joshi, and G.J. Ahn, “Security and privacy challenges in cloud computing environments”, *IEEE Security & Privacy*, vol. 6, pp. 24-31, 2010.

[52] A. Taly, U. Erlingsson, J.C. Mitchell, M.S. Miller and J. Nagra, “Automated analysis of security-critical javascript apis”, *Security and Privacy (SP), 2011 IEEE Symposium on IEEE*, pp. 363-378, 2011.

[53] J. Terrace, S.R. Beard, and N.P.K. Katta, “JavaScript in JavaScript (js. js): sandboxing third-party scripts.”, *Presented as part of the 3rd USENIX Conference on Web Application Development (WebApps 12)*, pp. 95-100, 2012.

[54] S. Tilkov and S. Vinoski, “Node. js: Using javascript to build high-performance network programs”, *IEEE Internet Computing*, vol. 14, no. 6, pp. 80, 2010.

[55] V. Tran, J. Keung, A. Liu, and A. Fekete, “Application migration to cloud: a taxonomy of critical factors”, *Proceedings of the 2nd international workshop on software engineering for cloud computing*, pp. 22-28, 2014.

[56] J. Vilk and E. D. Berger, “Doppio: breaking the browser language barrier”, In *ACM SIGPLAN Notices*, vol. 49, no. 6, pp. 508-518, 2014.

[57] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao and C. Fu, “Cloud computing: a perspective study”, *New Generation Computing*, vol. 28, no. 2, pp. 137-146, 2010.

[58] Q. Wang, C. Wang, J. Li, K. Ren and W. Lou, “Enabling public verifiability and data dynamics for storage security in cloud computing”, *Computer Security–ESORICS 2009*, pp. 355-370, 2009, Springer Berlin Heidelberg.

[59] D.G. Feng, M. Zhang, Y. Zhang, and Z. Xu, “Study on cloud computing security”, *Journal of software*, vol. 22, no. 1, pp. 71-83, 2011.

[60] T. Wood, K. K. Ramakrishnan, P. Shenoy and J. Van der Merwe, “CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines”, *ACM Sigplan Notices*, vol. 46, no. 7, pp. 121-132, 2011.

[61] A. Zakai, “Emscripten: an LLVM-to-JavaScript compiler”, In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pp. 301-312, 2011.

[62] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges”, *Journal of internet services and applications*, vol. 1, no. 1, pp. 7-18, 2010.

List of Publication

Ayush Sahu and Ashima Singh, “Securing IoT devices using JavaScript based Sandbox”, *in International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT-2016), IEEE*. [Accepted]

Video Link

https://youtu.be/gqH-5f4C1ho

Plagiarism Certificate

ORIGINALITY REPORT

5%	4%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dione.lib.unipi.gr Internet Source	1%
2	www.gepeq.dep.ufscar.br Internet Source	<1%
3	Bangui, Hind, and Said Rakrak. "Mobile Cloud Middleware: Smart Behaviour for Adapting Cloud Services", 2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems, 2014. Publication	<1%
4	Submitted to University of Salford Student Paper	<1%
5	www.inderscience.com Internet Source	<1%
6	aisel.aisnet.org Internet Source	<1%
7	publications.polymtl.ca Internet Source	<1%
8	"Practical Memory Deduplication Attacks in Sandboxed Javascript", Lecture Notes in	<1%