TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 8, August 2014, pp. 6369 ~ 6379
DOI: 10.11591/telkomnika.v12i8.5912

Dynamic Virtual Programming Optimizing the Risk on Operating System

Prashant Kumar Patra*[1], Padma Lochan Pradhan[2]
[1]Dept. of CSE, College of Engineering & Technology, BPUT, Bhubaneswar-751003, Orissa, India
[2]Dept. of CSE, Central Institute of Technology, Raipur, CG, India
*Corresponding author, e-mail: citrprcs@rediffmail.com.

Abstract
This virtual programming language is a primary method of improving the performance and protecting system resources in around the clock on anywhere and anytime platform. This proposed article is contributes to the development of an optimal method that aims to determine the minimal cost and time to be invested into secure and robust computing in right way, right time. everywhere and everytime of the globe. Furthermore, the virtual programming method optimizing the (CPU, MEMORY) space & time of the operating system. This dynamic code fully support to the current virtualization to co-op with large scale complex infrastructure and applications. We have to define, design, develop & deployment the virtual programming (C++, JAVA & RAIN) on real time operating system to participates the millions of users able to access on line web and mobile application simultaneously. We have to integrate this VPL with real time hardware, software and middleware to facilitate to customers requirement. This dynamic code is communicating, interfacing, message passing, replicating among the several subjects and objects over a distributed super scalar environment (MIMD). This proposed method optimized the system attacks and down time by implementing VPL on complex heterogonous infrastructure.

Keywords: *virtual programming language, java enterprise edition, multiple instruction on multiple data, risk mitigation, ROI: return on investment*

1. Introduction

The high level programming language is a set of instructions, commands, scripts and other symbols & syntaxes use to write the application software packages for specific scientific and commercial purpose. The programming languages that the developer use to write source code to solve for specific problem is called high-level languages (Ada, BASIC, COBOL, REXX, C, C++, JAVA). These codes can be translated, interpreted and compiled into a low-level language, which is recognized directly by the computer hardware as zero and one (i.e; low level languages). The high-level languages are designed to be easy to writeable, readable, reliable (WRR), available, robust, scalable and understandable by human being. The programmers to write pesudcode, source code using syntax, logical symbol and just like English words, grammar as per flow chart and algorithm (specification). For example, the control statements and reserved English words like go to, for loop, do while, if then else, continue and break are used in most major programming languages to construct the programmed to solve our purpose. The logical operators and symbols (&&, | |, ++, <, >, == and !=) are common syntaxes are available in all most all high level programming languages as available on today on concurrent, parallel and distributed environment. There are many high-level languages are similar enough that programmers can easily understand source code written in multiple languages like C, REXX, PASCAL, Ada, COBOL, C++, Java & C# for multiple purpose [1-2].

1.1. Existing High Level Languages [1]

Now a day, there are many programming languages are available and massively used in both commercial & scientific applications are using in parallel, distributed and concurrent operating system. The traditional programming languages is an early stage of conventional programming languages are (C, PASCAL, COBOL, FORTRAN, Ada & REXX) commonly known as procedure oriented programming (POP). In the procedure oriented approach, the problem is

Received November 21, 2013; Revised February 18, 2014; Accepted March 6, 2014

viewed as a sequence of things to be done such as reading, calculating and printing. The POP is primary focus on a function. A typical program structure of procedural programming is show in Figure 1.

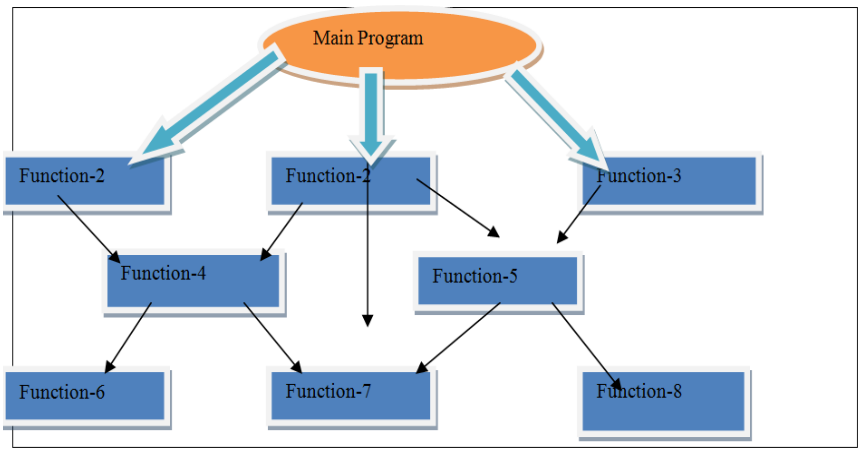

Figure 1. Architecture of POP

The technique of hierarchical decomposition has been use to specific the task to the completed for solving program. The procedure oriented programming basically consist of writing a list of instruction (or actions) for the computer to follow and organizing these instructions into groups know has function. We normally use a flow chart to organize thus action and represent the follow of control from one action to another. While we concentrate on the development of function very little attention is given to the data that are being used by various functions. What happens to the data? How are they affected by the function that works on them? In a multi function program, many important data items placed as global so that they may be accessed by all the functions. Each function may have it's own local data. Figure 2 show the relationships of data and functions in a procedure oriented program.

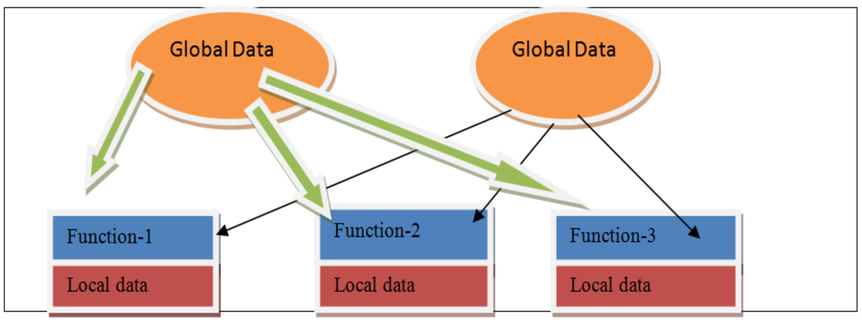

Figure 2. Relation between Data & Function

The global data are more vulnerable to a inadvertent change by a function in a large program it is very difficult to identify what data is used by which function. In this case, we need to revise an external data structure we also need to revise all function access the data. This provides an opportunity for bugs to creep in to the codes. Another serious drawback with the procedural approach is that it does not model real world problems very well. This is because function is action-oriented do not really corresponding to the elements of the problem [1- 2].

1.2. Characteristics of POP [1]

a) Emphasis is on function and algorithms, but neither data nor object.
b) Large programs are divided in two smaller programs known as functions.
c) Most of the function share global data.
d) Data move only around the system from function to function.
e) Function transforms data from one from to another.
f) Employs top-down approach in program design.

1.3. Advatntage of Function in OOP Languages (Message oriented Language) [8]

a) Function makes the lengthy and complex program easy and in short forms. It means large program can be sub-divided into self-contained and convenient small modules having unique name.
b) The length of source program can be reduced by using function by using it at different place in the program according to the user's requirement.
c) By using function memory space can be properly utilized. Also less memory is required to run program if function is used.
d) A function can be used by many programs.
e) Function increases the execution speed of the program and makes the programming simple.
f) By using the function portability of the program is very easy.
g) It removes the redundancy (occurrence of duplication of programs) i.e. avoids the repetition and saves the time and space.
h) Debugging (removing error) becomes very easier and fast using the function sub programming.
i) Functions are more flexible then library function.
j) Testing (verification and validation) is very easy by using function.
k) User can build customized library of deferent function use in daily routine having specific goal and link with the many program similar to the library function.[1], [8].

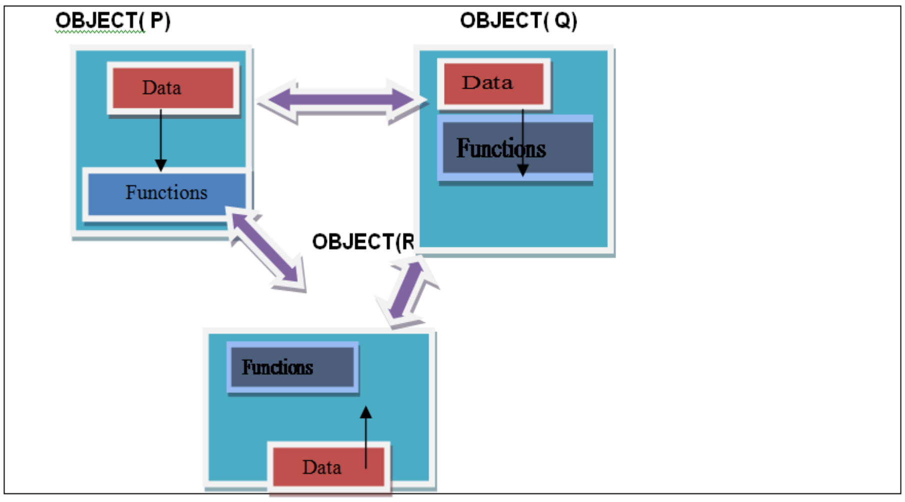

Figure 3. Communication among Data & Function in OOP

2. Existing Operating System

As per Flynn's classification of computer architecture in 1972 (layering technology), The layering approach of hardware & software is defined as follows: This is the traditional hardware, software and layering architecture available as on today, but we are moving to virtualization world to more optimize our hardware, software, resource cost & time [5].

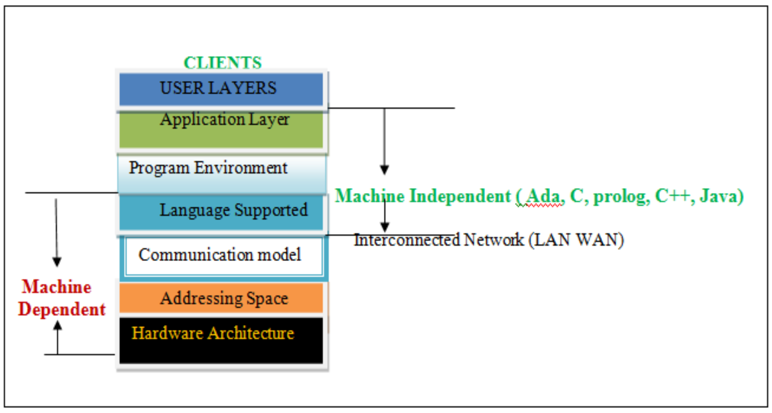

Figure 4. Architecture of OS

The concurrent programming involves the notations for expressing potential parallelism, so that operations may be executed in parallel and the techniques for solving the resulting synchronization and communication problems (producers & consumers). The concurrent, distributed and parallel operating system have been used at various times to describe various types of concurrent programming (C++,JAVA & RAIN). There are many more multiple processors, shared memory, share caches store to implementation concepts and are not important from the programming language points of view at all the time [5, 7].

2.1. Existing Programming Languages & Data Collection

The high level programming language is a computer language, the programmers use to develop applications, commands, scripts, and other set of instructions for a computer to execute for specific purpose. We have to find out the several different programming and scripting languages currently listed in our database collection and survey for the specific requirements as per customer's requirement in multi-computer environment for multiple-purpose is defined on Table 2. There are several programming languages list out in category wise as on today.

Table 1. Data Collection [3-5], [7], [10]

SN	Heterogeneous Operating System (MIMD)	DESRIPTIONS CATEGORY	TYPES OF LANGUAGES (POP, FPL, PPL, CCP, OOP)	PURPOSE
01	UNIX, Win, NT, Linux	APPLICATIONS AND PROGRAMS DEVELOPMENT	C, C++, JAVA, C#, PERL	Commercial & Scientific
02	UNIX, Win, NT, Linux	ARTIFICIAL INTELLIGENCE DEVELOPMENT	LOGIC (Prolog), LISP, Haskell, ML	Mathematical & Scientific
03	UNIX, Win, NT, Linux	DATABASE DEVELOPMENT	Dbase, Fox pro, SQL, My SQL, Sybase, Oracle, Ingress.	Business & Commercial, ERP
04	UNIX, Win, NT, Linux	GAME DEVELOPMENT	C, C++, JAVA, RUBY, Ant	Entertainment
05	UNIX, Win, NT, Linux	DEVICE DRIVER (HARDWARE)	C, C++, JAVA, C#, PERL	Mathematical, Scientific & Benchmarking,
06	UNIX, Win, NT, Linux	INTERFACE DEVELOPMENT	C, C++, JAVA, C#, PERL	Mathematical, Scientific & Benchmarking,
07	UNIX, Win, NT, Linux	INTENET, INTRANET & WEB DEVELOPMENT	DHTML, HTML, JAVA SCRIPT, PERL, AWK, SED, Ant, PHP, VBScript. Python.	Business & Commercial, ERP &Web Engineering
08	UNIX, Win, NT, Linux	SCRIPTING DEVELOPMENT	HTML, XML, VBScript, Perl, PHP, Python, Ruby, Ant, Perl	Business, Telecom. Commercial, ERP, Web & Mobile Mathematical & Scientific

3. Problem Statements

a) Programming cost is more.
b) Security, Integrity, Scalability problem.
c) Updation & Change management problem (Change Management).
d) The codification, updation, modification change when requirement changes as per algorithm changes times to time, the codification is fully depending on algorithm.
e) Hungering of CPU & Memory on POP & FPL.
f) Optimize the change & incident management. (Fault Tolerance)

3.1. Reseacrch Questions: Why we need VPL?

a) To optimize Space & Time
b) To design and develop internet, intranet & extranet programming (GUI) for complex N-tier architecture on heterogeneous platform.
c) To maximize the performance of CPU & Memory (bench marking).
d) Automatic memory management (Automatic garbage collection: AGC)
e) Increasing the business (large volume of data, Information, data warehousing & data mining).
f) Increasing the millions of users (Web & Mobile computing).
g) Increasing the hackers as well as experiences hackers.
h) Increasing the hardware & software capabilities (N-th bits processor & number of CPU, Memory).

3.2. Research Objective

Our proposed Virtual Programming will be great helpful for E-commerce, E-governess, science to science, product to product, business to business & society to society:

a) Business continuity planning & disaster recovery planning (BCP/DRP)
b) Support to internal & external system audit. (CMDB & ITIL)
c) Keep the system balance among OS, Network, Application & Various types of devices, sub-systems, resources & users need. (Fault tolerance)
d) Improve the security.
e) Improve the interoperability.
f) Improve the performance, benchmarking, fault tolerance, reliability & high availability.

3.3. Objective of Virtual Programming

The virtual programming is the most recent programming and different task for different environment. It is therefore important to have modularizing programs by concern partitioned memory area for both data and function. The object is considered to be a partitioned area of computer memory that stores data and set of operation that can be access data. The virtual programming emphasis is on data rather than function & procedure. The programs are divided into what is called objects. The function, operate on data of an object. The data are hidden, that cannot be access by external functions. The objects may communicate with each other through functions. The data & function can be added, when business rule changes. The virtual programming solves the following problems like: real time system, simulation model, AI, internet, extranet, multimedia, gamming & parallel programming [1-2].

The high level programming languages like C++, C# and Java has become largely popular in the web based application in around the world. The Java's became highly popular and widely acceptances programming language for intranet, internet and extranets, especially that can be write once and run it anywhere (byte code) any time on heterogeneous platform. The object oriented Java language green paper initially developed by Sun Micro-System in USA in 1991, The Java is a simple, object oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, scalable, resource dependencies, high available, high-performance & throughput, multithreaded and dynamic. The object oriented programming JAVA interfacing, interacting and synchronizing, communicating, interpretable, scalable with frontend (API- VB, Java, java script), middleware (Apache, Tomcat, Iplanet, Web logic, Jbose) & backend Database (Oracle, My SQL, Sybase) and tightly coupled with Operating System. It is fully supported to service oriented architecture & programming (SOA & SOP) [2], [4], [6], [9].

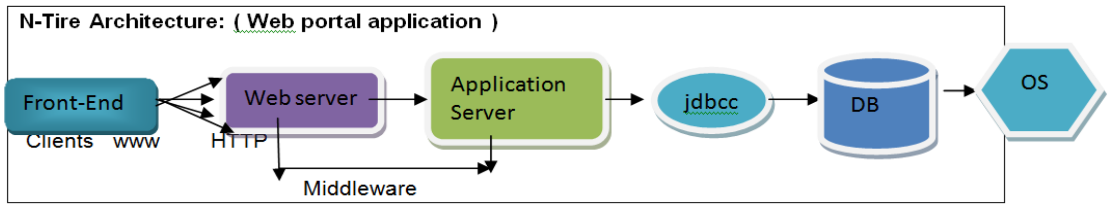

Figure 5. Architecture of Web Portal

5. Proposed Method: Dynamic Virtualization Program (Polymorphism)

We have to define, design, develop and deployment (D^4) this propose virtual programming (Polymorphism) for web based services and fix up the majors automated system configuration to maintain residual risk. Meanwhile, we have to maintain the system control by applying automated method, model, mechanism (M^3) & tools on operating system level to optimize the risk and maximize the decision management as per business requirement and availability of resource & technology.

Our VPL should be design in such way, that the file system, shell and kernel automatically protected, detected & corrected in around the clock for millions of users.

We have to implement dynamic VPL to optimized the system attacks and down time by implementing VPL mechanism & automated memory mgmt, mean while improving the throughput of the File, Memory and Processor & Kernel system. Finally, we have to maximize the performance & minimize the cost of the operating system. Our objective is that fix up the risk at highest (H, M & L) level with minimal cost and time.

6. Research Method (Action & Reaction Applied to Newton's Third Law)

The polymorphism can be defined as one interface to control access to a general classes of actions. There are two types of polymorphisms one is compile time polymorphism and the other one is run time polymorphism. The compile time polymorphism is functions and operators overloading. The runtime time polymorphism is done using inheritance and virtual functions. The polymorphism (different behaviors) means that functions assume different shapes & forms at different times. In case of compile time it is called function overloading. The program is divided into two functions, the two or more functions can have same name but their parameters list should be different either in terms of parameters or their data types as per business requirement. The functions which differ only in their return types cannot be overloaded. The compiler will select the right functions depending on the type of parameters passed. In this cases of class constructors could be overloaded as there can be both initialized and uninitialized of the objects. This following test level program focusing the working of compile time functions overloading and constructor overloading.

Example of polymorphism as follows:

```
class Operatingsystem {
        void which() {
        System.out.println(" HPC.");
        }
}
class Solaris extends Operatingsystem {
        void which() {
        System.out.println("Solaris");
        }
}
class AIX extends Operatingsystem {
        void which() {
        System.out.println("AIX.");
        }
}
```

```
class Operatingsystem {
	public static void main(String[] args) {
		Operatingsystem ref1 = new Operatingsystem();
		Operatingsystem ref2 = new Solaris();
		Operatingsystem ref3 = new AIX();
		ref1.which();
		ref2.which();
		ref3.which();
	}
```

One of the powerful tool for using polymorphism behavior is to use the same method name but in the same class, over and over again to get (achieve) the desired effects as we needed. How can we use polymorphism in java program to solve our purpose that one mentioned in bench marking sections.

Overloaded Methods as follows(Compile Time):

```
Public class Operatingsystem extend Solaris {
    Public void makePerform() {
        System.out.println("Good!");
}
Public void makePerform( Boolean failoured) {
       System.out.println(" failoured");
}
Public void makePerform( Boolean failoured) {
If ( failoured ) {
       System.out.println(" failoured");
   }
}
```

Overridden Methods (Run Time):

In polymorphism, we can create a method in a super class (parent class), then in a sub class also define that method.

```
public abstract class Operatingsystem {
  public class Solaris extends Operatingsystem {
     public void makePerformed() {
        System.out.println( "Good");
}
 Public void makePerformed(Boolean failover) {
    If failover) {
       System.out.println( "Failover");
        }
   }
}
```

Public class AIX extends Operatingsystem. Solaris & AIX are subclasses of Operatingsystem because they extend Operatingsystem.

Dynamic Method Binding (Late Binding):

Dynamic method binding is how Java decides what method to call when it has to decide between the super class and the sub class.

```
Public static void main(String{} args ) {
        Operatingsystem solaris = New Solaris( );
```

We have to just made a Solaris but declared it as an Operatingsystem, Normally we can do in this way:

```
Public static void main(String{} args ) {
   Solaris solaris = New Solaris( );
```

However, we can also declare a variable by its super type or abstract type to make it more generic way.

```
Public static void main(String{} args ) {
 Operatingsystem operatingsystem;
}
Public static void main(String{} args ) {
     Operatingsystem solaris = new Solaris( );
     Operatingsystem aix = new AIX ( );
     ArrayList<Operatingsystem> operatingsystem = new ArrayList(Operatingsystem>( )
     operatingsystems.add(solaris);
     operatingsystems.add(aix);
}
```

# Save the program ovcmethod.java (source code file)
 # java ovcmethod.java (created .class file internally)
# Run time .class file (ovcmethod. class) byte codes any where & any time on a heterogeneous operating system.

We have to test & experiment on these java & class file on heterogeneous hardware & software for benchmarking purpose to evaluate the operating system performance (Processor, memory & time). Accordingly the test result, we can able to validate the operating system components as described in below (Table 2).

6.1. Work Flow Diagram of the Recent Virtual Concept

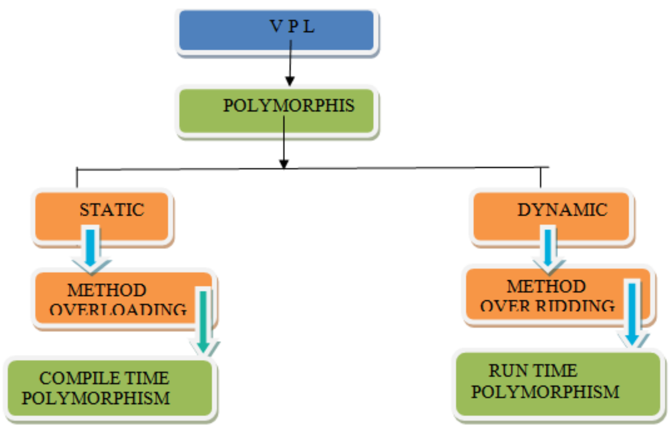

Figure 6

6.2. Proposed Bench Marking of Operating System

This research paper contributes to the development of scalable programming metrics on operating system that aims and objective to determine the performance factors at optimal cost & time to be invested into optimal model & mechanisms deciding on the measure component of operating system resources (i.e. Processor, Memory, Encryption key & KERNEL). That's why we are calling as it "dynamic programming model" for risk analysis. Furthermore, the model & mechanism optimize the cost, time & resources is supposed to reduce the operating system risks. We have to optimize the technology & resource cost and maximizes the performance factor (throughput) of OS and business.

We have to implement our idea based on the assumption data on BCP. How the operating system performance maximize as per our business requirement. Our objective is that maximizes our Business (throughput) & minimizes our Technology & Resources.

6.3. Performance Analysis

We can test the various virtual programming sample like (.JAVA & .CLASS) compile time polymorphism & run time polymorphism on different type of operating system. We have to analysis how the Processor, Memory, Cache and other sub components are working on the heterogeneous platform. How is the behaving the hardware & software, when millions of users working same time for same data & information.

The operating system validation and verification for the high performance computing to manage E-Commerce, E-Payment and product like B2B, B2C, P2P & G2G for superscalar computing. These system validation, verification & benchmarking can be define in the dynamic programming. We have to maintain the risk free environments on the hardware, software & application (algorithm) on basis of the following data. The verification and validation of hardware & software on super scalar environment as follow to satisfying to the B2B, B2C, Web & Mobile computing. We have fix up our others control tools like AES, MD5, SSH & SSL as per business requirements.

6.3.1. Theoretical Benchmarking

We have to make new idea to compare various types of UNIX operating systems parameters like processor, memory, file system, kernel on identical Virtual programming of Java or C++ performing on standard product like B2B, B2C, M2M, & P2P. The relative performance of the systems on identical tasks is more important to us than the absolute best performance that could be achieved for any individual system through system specific fine tuning and performance analysis. For comparison purpose ,we have only source code available for testing of Processor, Memory & Kernel parameters, our benchmarking methodology is the black box approach available in sequential & as well as randomize way to determine our objective. We usually attempt to explain curious & interesting results through continuous testing and benchmarking rather than investigations of Memory, CPU & Kernel code etc.

Why we need Benchmarking? To make the system high fault tolerance, which is satisfying to BCP & DRP.

6.4. Proposed Operating System Scalable Metrics

The following dynamic data is helping to our purpose for benchmarking.
We have to update these data dynamically as per implementation of J2EE as follows:

Table 2

B	X	X	X	X	X	X	X	Business	J2EE	HA	VLIW	MIMD
K	M	M	M	M	M	M	M	Kernel	M	HA	VLIW	MIMD
V	Java	Java	Java	Java	Java		A=2^n	VPL	J2EE	HA	VLIW	MIMD
E	16	32	64	128	256	512	E=2^n	Encryption	J2EE	HA	VLIW	MIMD
P	32	64	128	256	512	1024	P=2^n	Processor	J2EE	HA	VLIW	MIMD
M	16	32	64	128	256	512	M=2^n	Memory	J2EE	HA	VLIW	MIMD
C	L	L	M	H	H	H	C=2^n	Control	J2EE	HA	VLIW	MIMD

Where X : Volume of the business (unknown), M: Kernel Value (unknown), VLIW: Very large instructions word, MIMD: Multiple Instruction on Multiple Data, HA: High Availability, C: Control, RM: Risk Mitigation. When control is high, then risk is law as per Fuzz's law applied in to benchmarking.

6.5. Practical Approach for Risk Assessment

We have to verify & validated the real time operating system integrity, high availability, reliability, scalability of virtual programming languages (polymorphism) on heterogeneous operating system platform and we have to study the behaviors of the sub system of the operating system like: Shell, File, Kernel, Processor, Memory & Encryption key as per business requirement & availability of technology. We can apply some review method on internal UNIX operating system for our benchmarking purpose to mitigate the risk factor. This benchmarking method can be applied on traditional as well as Web based application which is going to be facilitated to the performanance analysis, benchmarking, fault tolerance and risk assessment over a complex real time operating system.

6.6. Experimental practice on benchmarking: Under SUN SOLARS UNIX

Table 3. Verification & Analysis [3], [5], [7], [10]

SN	ACTION PLAN (SUBJECT-INPUT)	DESCRIPTIONS	RISK ANALYSIS (OBJECT-OUTPUT)
01	iostat	Input /output statistics	CPU & Device Utilization, HA availability, Reliability & integrity of Processor. PRIMARY RISK ASSESSMENT
02	pmstat	Processors statistics	Global Statistics among all the processors & users : PRIMARY RISK ASSESSMENT
03	vmstat	virtual memory statistics [MEMORY ACTIVITIES]	Statistics of all the processor runable, block, swap, free buffer, input/output block devices, CPU detail, system, user, idle, waiting stage. HA availability, Reliability & integrity of Memory. PRIMARY RISK ASSESSMENT
04	sar	system activities	Activities report on: paging & swapping of OS detail. PRIMARY RISK ASSESSMENT
05	ps –ef | grep	ACTIVITIES OF PROCESSOR	The suspious processor or orphan/dead one. [space & time complexity issue] SECONDARY RISK ASSESSMENT
06	lsof l more	FILE SYSTEM ACTIVITIES	list of open files system which is very high risk. SECONDARY RISK ASSESSMENT
07	/etc/system	KERNEL SYSTEM ACTIVITIES	Can be update the kernel PRIMARY RISK ASSESSMENT
08	/etc/ssh/sshd_config CKM file system Automated Control	Cryptography enable through ssh implementation AES: 256 bits chipper ssh-key gen -b 1024 -f /etc/ssh_host_key -n '' chmod - - - /etc/ssh/ssh_config	Preventative control, n=1024, 2048, 4096 chimed r w x (i. e. 4 2 1) – blank is nothing [H, M, L]
9	/var/adm/message	Date & time stamp of events	Hardware & Software problem analysis

7. Results and Analysis

The subject and object can able to mapping, integrate, communicate, synchronize and optimize through through real time operating system. This virtual programming utilities and application will be more measurable and accountable for performance, fault tolerance, throughput, bench marking and risk assessment on any application over a complex IT infrastructure. How is the system behaving, when millions of users accessing the same piece of data & information in around the clock (high availability, scalability and reliability). We can only review practically based on theoretical idea. But, we have to review and justify the system behavior of space & time complexity based on machine size and problem sizes [5].

On behalf of the /etc/system script, we can update and improve the kernel capability as per business requirement and that can be help to our machine size and problem sizes analysis (Table 1-3).

How is behaving the UNIX server along with its sub components, when we are running on the different processor, memory & encryption key on the same programming & application or reverse way? The /var/adm/message scripts will be give the output statistics of hardware and software problem of real time event management system including date and time stamp on unix machine.

In this ways, we can improve the performance, benchmarking, fault tolerance and risk assessment at a time to utilizing the virtual programming application, which is help to our business, technology and society in around the globe.

8. Benefits and Advantage

The multiple operating system specifications will be involve and develop in response to market needs to protecting, detecting and correcting developer & user’s investment in existing systems and applications. The availability of the NT, Win, UNIX system from multiple vendors & suppliers gives users freedom of choice some of alternate solutions. There are many more wide range of applications - built on the NT, Win, UNIX system's strengths of scalability, availability and reliability that mission critical business needs can be meet the customer’s choice.

a) Optimizing the programming codification & complexity.
b) Optimizing the hardware resources as per business requirement.
c) Optimizing the CPU, Memory (SPACE), Cost & Time .
d) Supporting heterogeneous hardware, software, application, business, client & relevant resources
e) Highly secure, readable, writable, reliable, scalable and high available
f) Codification is faster, easier & scalable.
g) Best practice of ROI, TCO, ROA, TQM for BCP/DRP.

9. Future Work

We have to work on automatic memory management on various operating system as well as current virtualization system.

Our testing methodology and practice should be improve the benchmarking standard on hardware, operating system (Processor, Memory & Kernel), software, application & dynamic programming languages like C++, JAVA & Rain.

10. Conclusion

The optimization is great things on hardware, software, operating system and programming languages, when applications have been written and are working correctly. The good design decisions and best practices in application design and implementation provide a sort of pre-optimization by eliminating many of the source coding issues that might have required optimization. We maximize the memory & minimize the process to achieve the minimum cost and time to run our business in right way and right time in around the globe.

The multi-tier web and mobile applications provide a more flexible, scalable and reliable environment for business-critical applications and their 24 x 7 availability requirements. Now a days, increasingly sophisticated deployments introduce additional complexity tools in system, on complex infrastructure and application interaction. The operating system and hardware selection is the best practices for application design, coding, the performance and application monitoring can help maximize application performance today and scalability, extenbility for tomorrow.

The OS like UNIX systems have always pioneered the high-performance networking, distributed applications and distributed storage managements that are the under lying of robust, high-performance N-tier web and mobile applications. However, the optimizations within N-tier applications themselves can make substantial contributions to the performance of those (J2EE) applications in addition to making them easier to understand, more maintainable and more scalable & reliable on any where any time platform.

References

[1] Balagurusamy EB. Object Oriented Programming C++. New Delhi, India: Tata McGraw Hill. 2010.
[2] Balagurusamy EB. Programming in Java. New Delhi, India: Tata McGraw Hill. 2007.
[3] Das, Sumitabh. UNIX System V UNIX Concept & Application. Delhi, India: Tata McGraw Hill. 2009.
[4] Herbert, Scheldt. The Complete Java Reference. New Delhi, India: Tata McGraw Hill. 2010.
[5] Hwang, Kai. Advance Computer Architecture. New Delhi, India: Tata McGraw Hill. 2008.
[6] Khalid A. Programming Guide to Java. Delhi, India: Pearson India. 2008.
[7] O’ Reilly. Essential of System Administration. O’ Reilly Media: USA. 1995.
[8] Rachhal, Singh. Programming in C++. Delhi, India: Kalyani Publisher. 2003.
[9] Steven, Halznar. Java Black Book. Delhi, India: Dreamtech. 2008.
[10] Sun-Microsystems. UNIX Sun Solaris System Administration. USA. 2002.