Cloud Computing: Privacy, Mobility and Resources Utilization

Tariq Alwada'n[1], Omar Al-Zitawi[2], Jalal Omer Atoum [3]

[1]*Faculty of Technology, The World Islamic Science and Education University, Jordan*
[2]*King Hussein Faculty of Computing Sciences, Princess Symaya University for Technology, Jordan*
[3]*College of Science and Engineering, Southern Arkansas University, USA*

Abstract: *The fascinating world of Cloud computing has definitely changed the way of using computers and the Internet. The impact it has left so far on how IT and business services are delivered and managed is undeniable. However, the evolutionary change that Cloud computing has left on the IT landscape has given rise to a range of concerns by Cloud providers and customers. The current study introduces and examines two major problems in Cloud computing system. The first one is the Cloud internal and external data security and client's privacy, and the tasks mobility and resources utilization. The study also provides practical solutions to the two aforementioned problems and a prototype, called SPI, which has been successfully tested. The present study suggests using SSL with proxy server and secured access to members control panel for Cloud external data security. As for internal security and client's privacy, multiple approaches have been applied.*

Keywords — *Cloud computing, privacy, mobility, IaaS, Saas, PaaS, SPI*

I. Introduction

A. *Cloud Computing*

Cloud computing, Grid computing, and Cluster computing are intended to permit access to huge quantity of computing power by combining resources and providing a solo system interface. A significant goal of these computing technologies is to provide utility computing. Utility computing is described as an IT service model in which service provider creates computing resources and delivers them on demand [1]. Clients pay service providers on pay-per-use billing method. It is basically similar to public utility service provision in which the users are charged on monthly basis as per the usage of the given utility (gas, water, power, telephone) [2].

Cloud computing has been claimed as an umbrella term to explain a class of complicated on demand services provided by the business providers [3]. Organizations and individuals can access and use applications from all around the globe on-demand through Cloud computing as it provides a model through which computing structure is seen as a "cloud" [4]. The major concept of this model is providing "storage, computing and software as a service".

Several professionals in the field have tried to give a definition of Cloud computing. One of these definitions is introduced by [4]: "Cloud is a parallel and distributed computing system consisting of a collection of inter-connected computers that are dynamically provisioned and presented as one or more unified computing resources based on SLA (service-level agreements) established through negotiation between the service provider and consumers."

B. *Mobility*

The capacity to move or rearrange virtual or physical computing resources including tasks, application software, and data between Cloud environments traversing a wide or a local network is termed as mobility [5].

Mobility enhances the management between resources and services due to the heightened usage of resources [6]. With the assist of mobility, the services can move all over the Cloud to obtain data from cloud nodes, implement on those nodes, and carry the results back to their original nodes. This can enhance the utilization and Cloud services flexibility [7]. Mobility may be divided into computational mobility and personal mobility. Personal mobility means that the user of Cloud is capable of performing the task at sites far from the definite physical hardware which eliminates the need to move the jobs along with the hardware. Users have the ability to initialize a job, for instance webmail, at a particular site and then migrate it to another location irrespective of the machine type. Moving an actual part of computer hardware such as PCs notebook and Personal Digital Assistants (PDAs) from one site to another is called as computer (or computational) mobility [5].

This paper is interested in computational mobility which deals with the movement of software. Migration of data, control, object, and link can also be called as computational mobility. Mobility can be classified into two categories; Weak mobility and Strong Mobility. Weak mobility allows code to move through the networks. In some cases, the codes have initial data assigned but without execution states (for example, the state of the computation is lost at the first node). An example of a weak mobility system is Code-on Demand (CoD) and Remote Execution Evaluation (REE) [8, 9].

On the other hand strong mobility, which our research builds on, is the ability of a computational environment to mobile the code and execution state (the context of execution) to start again at a new resource. The execution state comprises of running code, saved processor registers, program counter, local variables, and return addresses. A set of organizing execution controls operates in a process on the user machine and then accesses the remote resource by the in vocation of a remote process. One of the most important features of strong mobility is that jobs can choose to migrate between sites while it is processing information. The reliability challenge of partial failure is decreased because the job state is only in one site at a time [5].

This paper is organized as follows. Section II is the architecture structure and components. In Section III we give an overview of the Deployment Model. Section IV takes about Development Tools, System Design and Workflow. Section V and VI discuss the Implementation and Testing phase. Section VII gives an overview about Detailed Testing Sample Showing Policy Appliance, followed by conclusion and references.

II. Architecture Structure and Components

The paper suggests the following requirements; user's mathematical problem input, user's specification of the problem calculation requirements (hardware, privacy level, priority, processing location, and encryption mechanism) and users' need to access their space and track their tasks progress anywhere and at any time. The paper attempts to manage and control the privacy for mobility in Cloud system to reduce time, effort and eventually save money.

The diagram for the proposed prototype is shown in Fig. 1. The output of the suggested phase represents a temporary design. This diagram presents an overview of the intended system by unfolding the objects and classes within the system. It presents a wide-range of use cases from representing a domain-specific design to a comprehensive structure of the system.

In Fig 1 we introduced our proposed combined Cloud modules which represents the systems components and their connection together and how they are related. It consists of two main parts: Client side and the Virtual Cloud control system; we used the main modules which are IaaS , SaaS and PaaS in the back end to construct our virtual cloud.

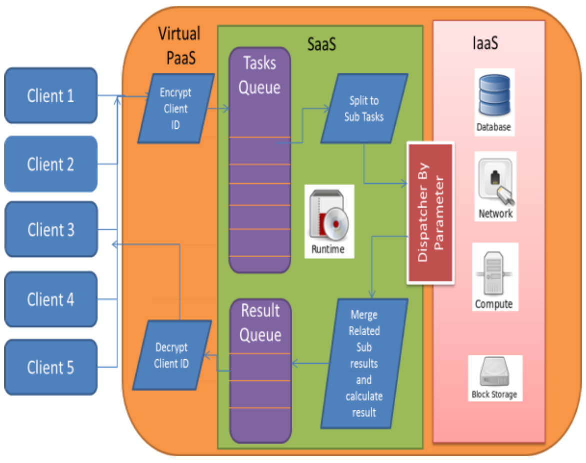

Fig 1 Design for the study proposed prototype

The proposed system consists of components such as (client, dispatcher and IaaS resources) and process such as (encryption and decryption system), queuing system, splitting, merging system and dispatching resources. The Dispatching system is divided into four parts:

1) **Clients Side Application:** Used to encrypt and submit the needed tasks for cloud processing.

2) ***L1 SaaS Queuing System:*** *Contains two queues (Tasks Queue) and (Results Queue).*

- Tasks Queue adds internal system sequence (ID number) for each task and forwards the Encrypted Client ID and the Internal Assigned Sequence Number to the Results Queue controller to be able to dispatch the task's result back to the client.
- Results Queue matches each received result with the previously received client ID and dispatches it to the decryption Agent to process it.

3) ***L2 SaaS Tasks:*** Splitting Agent and Results' Merging Agent.

4) ***IaaS Dispatching Agent:*** controls Task's Resource Allocation according to the included parameters and returns results back from processing resources to the result queue.

From Fig 1 the Queuing concept is designed to handle tasks processing with adding a priority handler to manage ordering tasks in the queues and on the other hand to take results back and get them ready for dispatching to the client.

III. Development

The prototype proposed in the paper is called SPI. The name is derived from the initials of the three Cloud computing modules, namely SaaS, PaaS and IaaS, all integrated in the proposed system. The prototype is based on a Cloud computing environment merged with the grid computing through the SaaS module with the clients interface handling mathematical linear and polynomial equations. The three parts coming next shed light on the prototype incorporation with Saas, PaaS, and IaaS providers.

A. Incorporation with SaaS provider

Incorporation of PaaS provider with Primary SaaS (Software as a Service) model provides clients with the ability to use value chain facilities on pay-per-use approach. PaaS provider incorporation with SaaS service provides consumer with the facility of range of languages to deploy their application. In addition, in terms of advertising they offer magazines or forums the opportunity to publish these services.

Organizations should mull over whether they want to acquire a public PaaS platform or to get a private PaaS platform. For a comparatively small or an average business, a public PaaS is more feasible and practicable. However, for a fairly large business, organizations should consider private PaaS platform.

B. Incorporation with PaaS provider

PaaS is the model that shows running applications over the Internet by renting hardware and software infrastructures. (Fig 2) shows the key attributes that characterize PaaS. This model provides the clients with an opportunity to get virtualized servers and software services on rent for maintaining and executing current applications and developing new ones.

The control and administration of the prime Cloud infrastructure is not in the hands of the client. The system control, server control, OS control, and storage control are under the supervision of the provider, although the consumer has access to posted tasks control and the configuration settings specified for the task handler in the hosting environment.

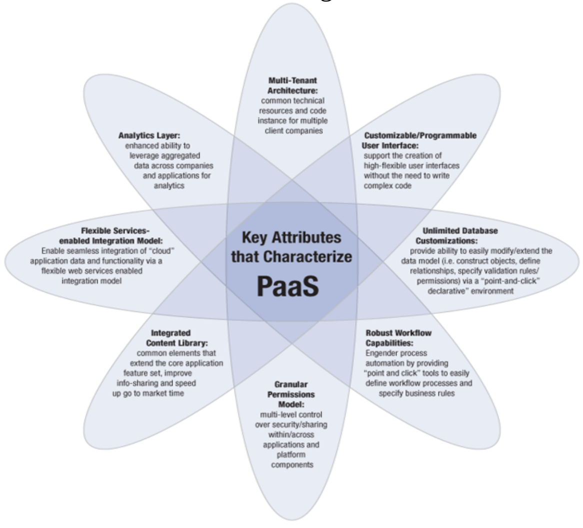
Multi-Tenant Architecture: common technical resources and code instance for multiple client companies
Analytics Layer: enhanced ability to leverage aggregated data across companies and applications for analytics
Customizable/Programmable User Interface: support the creation of high-flexible user interfaces without the need to write complex code
Flexible Services-enabled Integration Model: Enable seamless integration of "cloud" application data and functionality via a flexible web services enabled integration model
Key Attributes that Characterize PaaS
Unlimited Database Customizations: provide ability to easily modify/extend the data model (i.e. construct objects, define relationships, specify validation rules/ permissions) via a "point-and-click" declarative" environment
Integrated Content Library: common elements that extend the core application feature set, improve info-sharing and speed up go to market time
Robust Workflow Capabilities: Engender process automation by providing "point and click" tools to easily define workflow processes and specify business rules
Granular Permissions Model: multi-level control over security/sharing within/across applications and platform components

Fig 2 PaaS characteristics

C. Incorporation with IaaS provider

The incorporation between PaaS provider and fundamental IaaS (Infrastructure as a Service) provider offers to developers the chance to determine the availability of hardware resources directly without getting in touch with IaaS providers themselves. Developers need not to worry about hardware resources and their data. They just have to make contact with PaaS provider and leave the rest to it. Generally, these incorporations are carried out through APIs (Application Programming Interfaces) that help the IaaS model makes accessible to PaaS provider.

D. 4. Development Tools, System Design and Workflow

To develop such a solution, our system proposes the usage of Microsoft based Clouds with .net runtime, ASP.net, C#.net, and MS SQL DB Clusters. Fig 3 shows the proposed infrastructure design of SPI, containing the following elements:

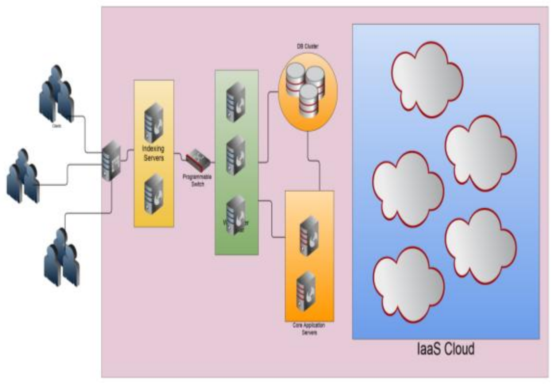

Fig 3 SPI infrastructure design

1) ***The Proxy server:*** implements a layer of privacy for the Cloud server. Proxy server [1] is contacted by the client for various tasks, such as requesting a file or connection request to different servers. These days most proxies provide anonymity and access to the content on World Wide Web.

2) ***Indexing servers:*** Indexing servers are used to map requests coming through the proxy server to a free web server on the back for load balancing.

3) ***Web server cluster:*** Web server cluster is a number of clustered servers configured for hosting the web portal that serves the clients requests and

[1] A proxy server is a system that operates as a mediator between client device (which is demanding the service) and another server. A user when demands some service (Any resource) from another sever after joining the proxy server, the proxy server assess the demand as a method to abridge and manage its complexity. A user when requests an Internet service to the proxy server, it searches it in the cache of earlier downloaded pages. If the page exists in the history, it provides it to the requester without requiring passing on the request to the network. In other case i.e. if the server does not find the web page in the history, it requests the demanded page from the Internet using one of its randomly generated IP address and passes on that web page to the user. At present, proxy server system is mainly used to provide access to the Internet with secrecy.

passes it back to the backend core Application servers.

4) ***DB cluster:*** DB cluster is used to store all cloud configurations and clients encrypted data.

5) ***IaaS Cloud:*** IaaS Cloud is a set of remote machines connected and controlled by the application servers used to serve and run clients forwarded tasks.

IV. IMPLEMENTATION

In the prototype system proposed in the current paper, a system that uses PaaS for Cloud computing is created to execute specific client task such as linear mathematical equations, defining the processing parameters for each task submitted and controlling its security level, priority, processing location, and resources needed. The prototype is developed using asp.net, c#.net, JSON and SQL Server programming language, and VMware environment.

Linear and polynomial equations are implemented as they are used in many different applications. In this prototype, tasks management, control of privacy and tasks mobility in Cloud system prototype were implemented. The prototype was used as method.

Since the development of this application is specific for the mobility, a Cloud hosted web application is used as a SaaS to be accessible from any location and at any time. In addition, multiple processing units in different locations are used as IaaS. When SaaS and IaaS are combined, a virtual PaaS emerges.

A. *Solution Processing Mechanism*

For each client's task before getting in the SaaS, tasks processing queue gets encrypted along with the client's ID. The following parameters have to be sent along with each Task:

- Encryption Type (if Null a random encryption type will be assigned)
- Client ID
- Task's Data (linear Equation parameters)
- Task's Type (has to be processed in sub parts or has to be processed as one task)
- Resources Needed and physical location of processing resources
- Tasks priority (from client input)

Clients Application side gets the client's ID and the Tasks Data encrypted, using the selected Algorithm. Then it forwards the tasks to the Queue to start processing it in SaaS Control System after assigning internal Task ID to match them with results later.

IF Task type is set to (1), each Task gets dispatch sequentially from the queue and splice into sub tasks , otherwise it gets forwarded directly to IaaS dispatching checker to determine which resource will be used to process the submitted task or sub task according to resource parameters sent with each task. Once processing is done, the result gets back into the IaaS dispatching agent to send it back to the result queue and the client. IaaS dispatcher controls IaaS resources utilization to make sure resources are wasted.

B. *Prototype Mobility*

The main idea of Cloud system is the mobility and the accessibility of information from any device anywhere in the world at any time. The mobility is to the same extent as the web-based system as it can be accessed using any Internet connected device.

1) *Tasks and Resource Migration*

The prototype proposed in the current paper is based on a task and data mobility using priority and resource parameters which direct mobility to move the tasks from one IaaS resource to another according to three conditions: (1) task priority, (2) tasks completion percentage, and (3) task resources specifications.

According to the IaaS Dispatcher's policy, once a task arrives to the IaaS dispatcher, it gets into a number of filters, namely task resources parameters filter and task priority filter. Once the task gets through the resource filter and gets assigned to an IaaS resource, which may have other tasks in process, the incoming task gets into the second filter which compares the task's priority for the incoming task with the priority of the task under process. If the priority of the incoming is less than the one of the current running task, the task gets into the waiting queue until the current task finishes processing and gets dispatched out of the resource. However, if it is higher, the current processing task gets into a third filter which checks the current processing percentage. If it is below 50%, then the current process gets terminated, the new task with the higher priority gets into its place, and the IaaS dispatcher checks if any other resource has similar resource requirement to send the old process to it for processing. Otherwise it gets into the waiting queue for the original IaaS resource. See Fig 4 and Fig 5.

2) *Data/Application Software Migration*

The prototype is controlled by a SaaS application which is a web portal that can be moved and hosted on any server anytime and anywhere, as long as it maintains the same domain name. All PaaS resources connect to each other, using a configuration files hosted in the PaaS core server, which contains all information such as: IP addresses, IaaS details and aspects and clients DB.

This information is stored in a secured SQL server DB, so basically for moving the SaaS web application, one has to host in a new server, publish it and update the SaaS records and the DNS information to point to the new SaaS server. This update should be within interval rate of 5 second on the IaaS clients Agent and the dispatchers' queues.

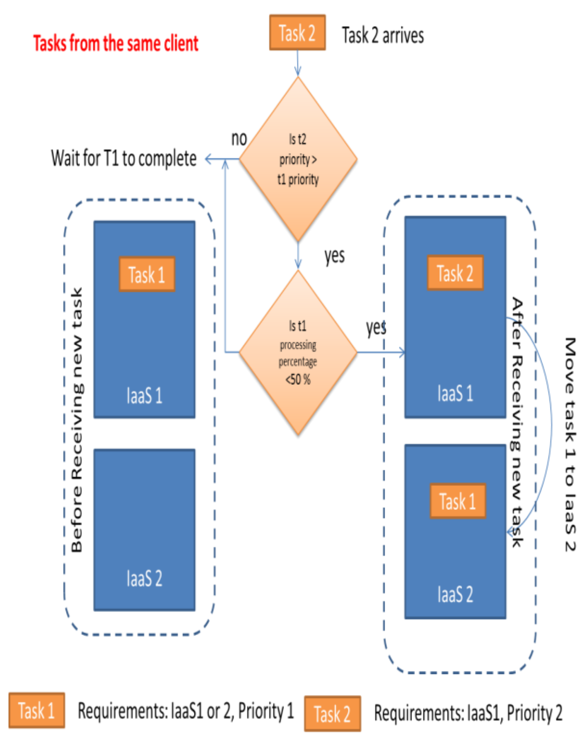
Tasks from the same client
Task 2
Task 2 arrives
Is t2 priority > t1 priority
no
Wait for T1 to complete
yes
Is t1 processing percentage <50 %
yes
Before Receiving new task
Task 1
IaaS 1
IaaS 2
After Receiving new task
Task 2
IaaS 1
Task 1
IaaS 2
Move task 1 to IaaS 2
Task 1
Requirements: IaaS1 or 2, Priority 1
Task 2
Requirements: IaaS1, Priority 2

Fig 4 Tasks mobility for two tasks from the same client

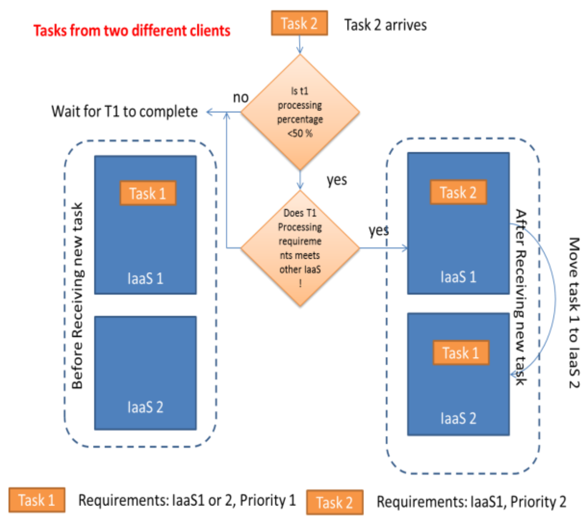
Tasks from two different clients
Task 2
Task 2 arrives
Is t1 processing percentage <50 %
no
Wait for T1 to complete
yes
Does T1 Processing requirements meets other IaaS !
yes
Before Receiving new task
Task 1
IaaS 1
IaaS 2
After Receiving new task
Task 2
IaaS 1
Task 1
IaaS 2
Move task 1 to IaaS 2
Task 1
Requirements: IaaS1 or 2, Priority 1
Task 2
Requirements: IaaS1, Priority 2

Fig 5 Tasks mobility for two tasks from the two different clients

C. *Prototype User Privacy and Security*

The system applies an automatic random encryption (or a specific encryption algorithm if specified by the client while submitting the task) on the tasks client ID. This encryption key is applied before the tasks get posted to IaaS tasks queue, which prevents the clients' identity from being revealed within the PaaS system. Moreover, the application is secured with SSL certificate with 256 bit RSA, to secure packets transmitting through the network. In the prototype, Data Protection application programming interface (DPAPI) and key relationship management is used.

D. *Prototype Screen Shots and Description*

This section shows the screen shots of the SPI prototype and its features and services given to SPI admins and clients.

The client will be introduced to the Dashboard that contain regional state, feeds, client task process, and IaaS subtask process as shown in Fig 6. While the admin has a full informative and statistical dashboard which can be used to monitor the overall system status, IaaS utilization and resources, load and control IaaS Processing Agent as in Fig 7.

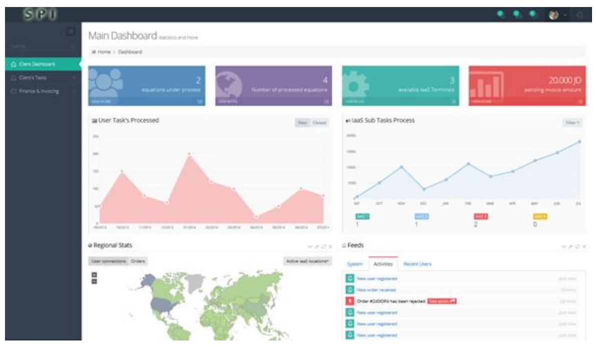

Fig 6 Client dashboard

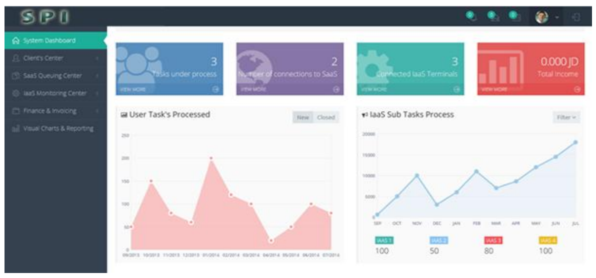

Fig 7 Administration main dashboard

V. TESTING PHASE

To test and evaluate the proposed system prototype, the researcher has formed 30 different test cases. Table I shows the first tenth testing cases. Also Fig 8 shows the comparing results between numbers of clients to the overall processing time in associated to the number of equations. These testing cases include various numbers of clients, various numbers of equations (linear, polynomial, and linear polynomial), various complications of equations, and various client policies.

The testing environment consists of five VMware virtual machines installed on a testing laptop with the following aspects:

Table I General Testing Results

	No. of clients	No. equ	Equ type	Equations complexity	IaaS 1 CPU utilization	IaaS 1 Processing percentage	IaaS 2 CPU utilization	IaaS 2 Processing percentage	IaaS 3 CPU utilization	IaaS 3 Processing percentage	Processing time
1	2	6	L	15	9%	38.89%	3.75%	16.67%	10%	44.44%	0.54
2	3	12	P	13	19%	32.05%	11.88%	20.30%	28%	47.65%	1.404
3	4	18	P	25	98%	43.33%	77.50%	34.44%	50%	22.22%	5.4
4	1	4	L	12	1%	10.42%	1.63%	27.08%	4%	62.50%	0.144
5	5	30	L	21	13%	31.75%	11.13%	28.25%	16%	40.00%	0.945
6	6	5	P	30	31%	27.78%	40.00%	35.56%	41%	36.67%	2.7
7	3	10	L	50	75%	40.00%	47.50%	25.33%	65%	34.67%	4.5
8	1	25	P	40	54%	43.00%	21.25%	17.00%	50%	40.00%	3
9	7	15	P & L	30	100%	57.14%	62.50%	15.87%	100%	26.98%	9.45
10	4	20	L	25	81%	32.50%	93.75%	37.50%	75%	30.00%	6

- 2 TB Sata Hard desks
- 16 GB RAM
- Intel Core i7 2.60 GHz
- 64 bit windows 8.1
- Each VM Machine has 100 GB HD, 2 GB RAM, shared host processor, and virtual IP Address visible for the current LAN.

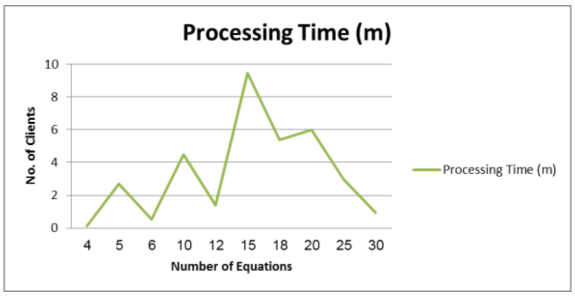

Fig 8 The Processing Time Resulted from comparing the numbers of clients associated to the number of equations (measured in minutes).

In addition to the five VMs which are used as IaaS resources, the testing environment includes:

- 1 VM which is used to host the SaaS Web Application,
- 2 VM which is used to test the client side connection to the PaaS, and
- 2 IPhone and 3 Android devices which are used as clients.

The following parameters are used to evaluate the proposed SPI prototype as shown in Table I:

- Number of clients: specifies the number of clients connected to the SPI prototype.
- Number of equations: how many equations submitted by each client.
- Equation type: what type of equation submitted by clients (Linear, polynomial and linear polynomial).
- Equations complexity: for linear equation the complexity specifies the number of unknowns in each equation while for polynomial equations; complexity specifies the maximum power of the unknown.

Formally, the following measures are used to evaluate these cases:

- CPU utilization: the maximum utilization of the IaaS CPU throughout the test.
- Processing percentage: the percentage of tasks processed by each IaaS from the total number submitted by clients calculated using this formula – Percentage = (number of processed tasks by the IaaS/the total number of tasks submitted by all clients during the test) x 100%.
- Processing time: the duration of processing all tasks during the each test measured in minutes.

Based on the results of evaluating each test case, the following is concluded:

- When IaaS resources CPU utilization reaches 100% and gets fully utilized; "tasks queue" starts filling up especially when clients start specifying a dedicated resource specification. This will prevent the task from being processed by any other available IaaS and also that means the SPI didn't drop any job and keep it in "Task Queue".
- Giving the client the ability to choose the IaaS specifications decrease the overall cloud system utilization. Also it increases the load on specific IaaS resources while leaving the others unutilized or partially utilized and in the same time helps the client to decide which place can run his job.
- The best scenario for SPI is working with full efficiency on linear equations up to 20 equations and complexity up to 25 while it starts to reach high utilization levels when it exceeds those limits.
- The best scenario for SPI is working with full efficiency on polynomial equations up to 18 equations and complexity up to 30 while it

starts to reach high utilization levels when it exceeds those limits.
- Security levels in our prototype all data and client information are very high; secured from being breached by internal or external users.

VI. Detailed Testing Sample Showing Policy Appliance

Two clients connected to the same LAN (using Samsung Note device), One from VM1 and the second connecting from a mobile device. Each of the clients logs in and submits 3 equations 2 polynomial and 1 linear with complexity level 15 (15 unknown parameters for the linear equations and a maximum power equal to 15 for the unknown parameter of the polynomial equation shows in Table II). The complexity level 15 is chosen for its ability to record the execution time and to record the mobility action. The processing steps include**:**

Client one submits his/her first task with the following parameters: Virtual machine 2; priority is not specified. Since the client has not specified a priority level, the SaaS adds default priority (0) to the task once it reaches the Tasks Queue. When it is its turn, the SaaS controller dispatches the first task from client 1 to the IaaS dispatcher to specify its processing IaaS. The dispatcher moves the task to VM no. 2 as it is free, and VM no. 2 is specified with the task's parameters. IaaS 2 starts processing the task.

In the meanwhile, client no. 2 submits his/her first task with the following parameters: VM no. 1; priority level 1. The task then is forwarded by the Task Queue to IaaS dispatcher which moves it to VM number 1 (IaaS 1). IaaS 1 starts processing the first task submitted by the second client.

Client no. 1 submits his/her second task with the following parameters: VM no. 2; priority level 2. Once the task reaches IaaS dispatcher, it gets into the task resources filter. As IaaS no. 2 is specified and is not free, the dispatcher compares client 1's first task (which is under process) with client 1's second incoming task. The dispatcher finds that the priority of task 2 is higher than the priority of task 1. Accordingly, another check is made to examine the processing percentage of task 1. If the processing percentage is less than 50%, the dispatcher takes task 1 and return it back to task queue (processes it in another IaaS according to task 1's requirements). The incoming task, which is task 2 that has higher priority, is the one that is being processed. If the processing percentage is more than 50%, which was not the case, task 1 will continue to be processed while task 2 is queuing.

IaaS 1 finishes processing client 2's task 1 and sends it through the IaaS dispatcher to the result queue, and then sends it back to client 2.

IaaS 2 finishes processing client 1's task 2 and returns the result back to the client through the result queue of IaaS dispatcher. Once task 2 is processed, IaaS dispatcher calls for the removed task (client 1's task 1) and gets it into IaaS 2 to be processed.

Client 2 submits a new task (task 2) to the task queue with the following parameters: VM no.2; priority 1. IaaS dispatcher gets notified but keeps the task waiting in the queue as the requested IaaS is busy with another task. Once IaaS 2 finishes processing client 1's task 1, it gets client's 2 task 2 into IaaS 2 to be processed.

Client 2 submits a new task (task 3) with the following parameters: VM 3; default priority. The task gets forwarded from the tasks queue to the IaaS dispatcher which moves it to IaaS 3 for processing as per the task's parameters set by client 2. IaaS 3 finishes the processing of task 3 and sends it back to client 2.

Client 1 submits his/her third task without specifying the VM machine requirements and sets its priority to 1. The task gets into the task queue and gets forwarded to IaaS dispatcher. Since VM requirements are not specified, and all IaaS resources are free, the task randomly gets assigned to one of the available IaaS resources it run in VM 3. The task is then processed, and the result is sent back to client 1. Table II shows the scenario of the processing steps explained above.

Table II Result sample

	Equation 1	**Equation2**	**Equation3**
User 1	15 * VM: no 2 Priority: 0	15 * VM: no 2 Priority: 2	15 * VM: undefined Priority: 1
User 2	15 * VM: no 1 Priority: 1	15 * VM: no 2 Priority: 1	15 * VM: no 3 Priority: 0

The equation was sorted out in the following order:
- On VM1: Task (Client 2 , task 1)
- On VM2: Task (Client 1, task 1), task (Client 1, task 2), task (Client 2, task 2)
- On VM3: Task (Client 2, task 3), task (Client 1, task 3)

VII. Conclusions and Future Work

This paper proposes a mixed usage of the three service models of cloud computing (IaaS, SaaS, and PaaS) to get the most of each model and create a new fully automated cloud computing environment that maintains the highest level of security possible and at the same time fulfils the mobility concept for the system as a whole and the client tasks. By merging IaaS and SaaS, a new kind of virtual PaaS, called SPI, has emerged that has the best features of each cloud type, decreases the disadvantages inherited in the system, and minimizes any security risks. Numbers of future works that can be implemented in the field of cloud systems such as: User Responsive (Encoding); clouds are usually very complex and difficult to encode. This is very difficult to modify the given cloud application according to the particular requirements.

REFERENCES

[1] M. Rouse. What is public cloud. Retrieved from Whatis.com. (2014, October 12).

[2] Cisco Systems. Managing the Real Cost of On-Demand Enterprise Cloud Services with Chargeback Models. White Paper.(pp. 10), 2010

[3] S.Ahmed and S. Maria. Cloud Computing: Paradigms and Technologies. F. Xhafa and N. Bessis (eds.), Inter-cooperative Collective Intelligence: Techniques and Applications, Studies in Computational Intelligence 495, DOI: 10.1007/978-3-642-35016-0_2, Springer-Verlag Berlin Heidelberg, 2014.

[4] C. Y. R Buyya. Vision, hype and reality for delivering computing as 5th utility . In Cloud computing and emerging IT platforms, 2009.

[5] T.Alwada'n. Security for Mobile Grid Systems. PhD Thesis. School of Computing, De Montfort. (pp. 26-28), 2012.

[6] T.Athanaileas. An agentbased framework for integrating mobility into grid services. Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE Operating Systems, and Applications (pp. 31:1-31:6). Brussels: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2007.

[7] S.Wong and K.Ng. Performance evaluation of mobile grid services. Proceedings of the 2nd KES International conference on Agent and multiagent systems: technologies and applications (pp. 557-566). Berlin, Heidelberg: Springer-Verlag, 2008.

[8] J.Stamos, and D. Gifford. Remote evaluation. ACM Transactions on Programming Language and Systems, (pp 537-565), 1990.

[9] C.B.Westphall, and C. Westphall. Management and security for grid, cloud and cognitive networks. Revista de Sistemas de Informao da FSMA 8, (pp. 8-21), 2011.