ISSN 2277-3916 CVR Journal of Science and Technology, Volume 3, December 2012

Design Issues in Cloud-Hosted Applications

A. Seetha Ram Nagesh[1] and Suhail Afroz[2]
[1] CVR College of Engineering, Department of IT, Ibrahimpatan, R.R.District, A.P., India
Email: asnagesh@rediffmail.com
[2] CVR College of Engineering, Department of CSE, Ibrahimpatan, R.R.District, A.P., India Email:
suhailafroz@hotmail.com

Abstract – Cloud Computing is the hottest technology in the market these days, used to make storage of huge amounts of data and information easier for organizations. Maintaining servers to store all the information is quite expensive for individual and organizations. Cloud computing allows to store and maintain data on remote servers that are managed by Cloud Service Providers (CSP) .The concept of building or consuming services and applications that are hosted off-premises is becoming more attractive both to independent software vendors (ISVs) and to enterprises as a way to reduce costs, maximize efficiency, and extend capabilities. This paper describes the nature and use of cloud-hosted services and applications. It describes the benefits and the typical design issues, and the constraints and technology considerations often encountered when building and consuming these kinds of applications.

I. INTRODUCTION

From initial concept building to current actual deployment, cloud computing is growing more and more mature. Nowadays many organizations, especially Small and Medium Business (SMB) enterprises, are increasingly realizing the benefits by putting their applications and data into the cloud. The adoption of cloud computing may lead to gains in efficiency and effectiveness in developing and deployment and save the cost in purchasing and maintaining the infrastructure

Regarding definition of cloud computing model, the most widely used one is made by NIST as "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. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models."[1] The cloud computing model NIST defined has three service models and four deployment models. The three service models, also called SPI model, are: Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS) and Cloud Infrastructure as a Service (IaaS). The four deployment models are: Private cloud, Community cloud, Public cloud and Hybrid cloud.

Cloud computing represents the converging evolution of computing infrastructure and application models for building and consuming scalable distributed solutions. As techniques for building these kinds of applications have advanced, so too have the capabilities of the infrastructure on which they run. This synergistic evolution allows the infrastructure to be provisioned and maintained largely independently of the applications that it hosts. This in turn allows applications to take advantage of supporting infrastructure services and capabilities while they focus on their specific business functionality.

Many organizations have been able to realize the joint benefits of scalable application models and supporting infrastructure internally on-premises in their own data centres. However, it is the ability to leverage an off-premises out sourced application hosting infrastructure that is behind much of the excitement around cloud computing. The infrastructure provider focuses on hardware, networking, power, cooling, and the operating environment that supports application manageability, reliability, and scalability; leaving the organization free to focus on their application's business functionality. This provides many benefits in terms of reduced capital outlay and operating costs; and increased capacity, scalability and availability.

To leverage these benefits, cloud-hosted applications typically must be architected to follow a specific application model. This allows the cloud-hosting provider to generalize and optimize their operating environment support for application manageability, reliability, or scalability.

Different cloud-hosting providers have different application model requirements. Some adopt a virtual machine approach, where the application is developed and packaged along with its operating system image and the dependent runtime frameworks. Others utilize an application model that provides higher level abstractions for data access and storage , and for computation and communication. Still others provide higher level application models based on highly configurable applications that focus on specific vertical application functionality, such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM). Each of these approaches provides distinct advantages and disadvantages.

Furthermore, some off-premises hosted applications are self-contained and designed for users who interact with the application through a dedicated UI. Some of these applications are service-enabled, and provide both a UI and expose their functionality through an API (often exposed through standards such as REST or SOAP) so that they can be integrated into other applications, which themselves can be hosted either on-premises or off-premises. Some off-premises hosted services are specifically designed to provide functionality for integration into other applications, and provide no UI at all.

DOI: 10.32377/cvrjst0303
DOI: 10.32377/cvrjst0303

The paper is organized as follows. Section II discusses the services provided by cloud environment. Section III explains the benefits of the Cloud applications. Section IV exploits the design issues for both the ISVs and Enterprise customers in developing Cloud-hosted applications. Section V summarizes the relevant patterns for design issues discussed in Section IV.

II. Cloud Services

Cloud-based services generally fall into categories such as storage/compute, business services, and retail/wholesale services. Some common examples of these remote services are:

- Business services such as stocks and shares information, invoicing and payment systems, data interchange facilities, merchant services, and business information portals.
- Retail/wholesale services such as catalogues, stock query and ordering systems, weather and traffic information, mapping services, and shopping portals.
- Storage/compute services such as data storage and processing, data backup, source control systems, and technical or scientific processing services.

These remote services can be consumed by software that runs on-premises, in an organization's data center or on a user's machine, which may be a desktop computer or any other Internet-enabled device. This typically involves a mix of technologies and techniques that are referred to as Software plus Services (S+S) [2]. S+S refers to an approach to application development that combines hosted services with locally executed software. The combination of the remote services and the software running locally, with rich seamlessly integrated interfaces and user experience, can provide a more comprehensive and efficient solution than traditional on-premises silo applications. S+S is an evolution of several other technologies including Service Oriented Architecture (SOA), Software as a Service (SaaS), Platform as a Service (PaaS), and Web 2.0 community-oriented architectural approaches.

A. Cloud-hosted Services

- Building block service: A service designed to be consumed by or integrated with other applications or services. An example is a storage service or a hosted Security Token Service (STS) such as the Access Control Service in the Azure Services Platform.
- Cloud-hosting environment: An environment that provides a core runtime for hosting applications; and, optionally, building block services, business services, social network services, and hosting services such as metering, billing, and management.
- Home-built application: An application that you create in-house, usually specifically targeted at some task, scenario, or process you require; it will often address a need that cannot be sourced from a third party.
- Hosted application: An application (packaged or home-built) hosted as a service. It may be hosted internally on your own system, or hosted externally by a partner or hoster.
- Packaged application: An application created by a third party or vendor that may provide only limited customization capabilities based on configuration or plug-ins.
- Platform as a Service (PaaS): A core hosting operating system, and optional plug-in building block services, that allow you to run your own applications or third-party applications obtained from vendors, in a remote cloud hosting environment.
- Software as a Service (SaaS): Applications that perform comprehensive business tasks, or accomplish business services, and allow you to consume them as services with no internal application requirements other than composition and UI.

III. Benefits Of Cloud Applications

Cloud-hosted applications and services may be very beneficial to ISVs, and to service delivery or hosting companies that build, host and deliver services. They also offer benefits to large enterprises that generally consume hosted and cloud-based solutions.

A. Benefits for ISVs and Service Hosts

The key advantages for ISVs and service hosting companies building and offering cloud-based solutions are the following:

- Architectural Flexibility: Vendors can offer their customers a range of deployment options, including hosting for the services they require, and allow users to choose from a range of prebuilt features or choose which features of the application they will implement themselves. This can reduce the architectural liabilities for end users who are developing services
- Rich User Experience: ISVs and service providers can offer richer experiences to their customers by leveraging existing specialized services (such as Virtual Earth). Hosters can combine their offerings with other cloud services obtained elsewhere to offer additional value propositions, and make it easier for end users to integrate services.
- Ubiquitous Access: Services in the cloud persist user data and state, and resynchronize when the user reconnects from any location. This supports both offline and occasionally connected scenarios, which is especially useful for mobile devices where a constant connection or bandwidth cannot be guaranteed.

ISVs and service hosts may also consider entering the market for commercial reasons to take advantage of

DOI: 10.32377/cvrjst0303

monetization opportunities. The following are some examples:

- Vendors may wish to take advantage of an untapped market opportunity by offering a product that is not currently or easily available elsewhere, or use the cloud to offer lower end versions of their products to protect a high end franchise.
- Startup companies may use the cloud-hosted approach to minimize initial capital expenditure, and to take advantage of properties of the cloud such as elasticity (the capability to grow as required without high initial cost commitment).
- Vendors and users can create applications that generate income more quickly by taking advantage of ancillary services that are already available. For example, they can take advantage of payment and accounting systems in the cloud. Users can even build virtual stores without requiring large investments in IT equipment and networking capabilities.

B. Benefits for Enterprise Service Consumers

The key advantages for enterprises that consume cloud-based solutions are the following:

- Architectural Flexibility: In-house developers can create complete solutions that compose services in the cloud with local application code and their own services. IT departments can choose which features of the application they will implement themselves, and buy in other services that they require.
- Cost and Time Savings: IT departments can select the best cloud-based service for each task, and combine them to expose fully functional applications with shorter development times, and at a reduced cost. In addition, the reduction in the requirements for in-house IT infrastructure simplifies management, security, and maintenance costs.
- Economies of Scale: Companies can leverage economies of scale for industry average capabilities, and focus on their core activities. The economies of scale available from hosted applications arise from a range of factors, including reduced in-house infrastructure costs to better utilization of hardware that offers opportunities for reduced running costs. However, the gains in economies of scale must be balanced with the loss of control inherent with moving from on-premises to fully hosted applications.
- Offline Capability: The cloud can act as hub for roaming users. User data and state can be stored in the cloud and resynchronized when the user reconnects. Users can move between desktop and mobile clients seamlessly with fewer network configurations.

IV. DESIGN ISSUES

Several common issues are of concern to both ISVs and enterprise customers [3]. While they cover a range of different aspects of hosted and cloud-based scenarios, these issues can be categorized into specific areas.

- Data Isolation and Sharing
- Data Security
- Data Storage and Extensibility
- Multi-tenancy
- Performance
- Service Composition
- Service Integration

a) Data Isolation and Sharing

Hosters can implement isolation and sharing for databases and for database schemas. There are three basic models:

- Separate Databases: Each tenant has a separate database containing their own data schemas. This has the advantage of being easy to implement, but the number of tenants per database server might be relatively low, with subsequent loss of efficiency, and the infrastructure cost of providing services can rise rapidly. It is most useful when tenants have specific data isolation or security requirements for which you can charge a supplement.
- Shared Databases, Separate Schemas: All tenants use the same database, but have separate sets of predefined fields available. This approach is also easy to implement, maximizes the number of tenants per database server, and improves database efficiency. However, it usually results in sparsely populated tables in the database. It is most useful when storing data for different tenants in the same tables (commingling) is acceptable in terms of security and isolation, and when you can anticipate the predefined custom fields that will be required.
- Shared Databases, Shared Schema: All tenants use the same database and special techniques are used to store data extensions. This approach has the advantage that the number of custom fields you can offer is practically unlimited. However, indexing, searching, querying, and updating processes are more complex. It is most useful when storing data for different tenants in the same tables (commingling) is acceptable in terms of security and isolation but it is difficult to predict the range of predefined custom fields that will be required.

b) Data Security

Cloud-hosted applications must implement strong security, using multiple defense levels that complement one another to provide data protection in different ways, under different circumstances, and against both internal and external threats [7]. When planning a security strategy, consider the following guidelines:

DOI: 10.32377/cvrjst0303

- Filtering: Use an intermediate layer between a tenant and a data source that acts as a sieve so that it appears to the tenant that theirs is the only data in the database. This is especially important if you use a shared database instance for all of your tenants.
- Permissions: Use access control lists (ACLs) to determine who can access data in the application, and what they can do with it.
- Encryption: Obscure every tenant's critical data so that it will remain unreadable to unauthorized parties, even if they manage to access it.

c) Data Security Patterns

Depending on the multi-tenant model adpoted, consider the following security patterns:

- Trusted Database Connections (applies to all three multi-tenant models): The application always connects to the database using its own application process identity, independent of the identity of the user, and the server grants the application access to the database objects that it can read or manipulate. Additional security must be implemented within the application itself to prevent individual end users from accessing any database objects that should not be exposed to them. Each tenant (organization) that uses the application has multiple sets of credentials associated with their tenant account, and must grant their end users access to the application using these credentials. These end users access the application using their individual credentials associated with the tenant account, but the application accesses the database using the single set of credentials associated with that application. This means that a single database access account is required for each application (one for each tenant). Alternatively, you can use an STS to obtain encrypted login credentials for the tenant irrespective of the individual user, and use security code in the application to control which data individual users can access.
- Secure Database Tables (applies to the Separate Database model and the Shared Database, Separate Schema model): Grant a tenant user account access to a table or other database object. In the Separate Database model, restrict access on a database-wide level to the tenant associated with that database. In the Shared Database, Separate Schema model, restrict access on a per table basis to the tenant associated with specific tables.
- Tenant Data Encryption (applies to all three multi-tenant models): Secure the data using symmetric encryption to protect it, and secure the tenant's private key using asymmetric (public/private key pair) encryption. Use impersonation to access the database using the tenant's security context, and use the tenant's private key to decrypt the data in the database so that it can be used. The disadvantage is that you cannot index encrypted columns, which means that there is a tradeoff between data security and performance. Try to avoid using index fields that contain sensitive data.
- Tenant Data Filter (applies to the Shared Database\Shared Schema model): Use SQL views to select subsets of data from tables based on the tenant or user ID, or the tenant account's security identifier. Grant tenants access to only their views, and not to the underlying tables. This prevents users from seeing or accessing any rows belonging to other tenants or users in the shared tables.

d) Data Storage and Extensibility

Hosted data may be stored in variety of ways. Two different approaches are emerging for implementing data storage in hosted applications: hosted relational database management systems (RDBMS) and non-relational cloud-based storage. Relational database systems provide storage for structured data, and are more suited to transactional systems or applications that are I/O intensive; they also typically provide lower latency and advanced query capabilities. In contrast, cloud storage refers to any type of data storage that resides in the cloud; including services that provide database-like functionality, unstructured data services (for example, file storage for digital media), data synchronization services, and network-attached storage (NAS) services. Data services are often consumed in a pay as you go model, or in this case a pay per GB model (including both stored and transferred data).

Cloud storage offers a number of benefits, such as the ability to store and retrieve large amounts of data in any location at any time. Data storage services are fast, inexpensive, and almost infinitely scalable; however, reliability can be an issue as even the best services do sometimes fail [6]. Applications that are sensitive to high latency might also be affected as each interaction with the storage service requires network transversal. Finally, transaction support can be an issue with cloud-based storage systems. These systems generally focus heavily on partitioning and availability, and consistency cannot always be guaranteed.

e) Multi-tenancy

The idea of multi-tenancy, or many tenants sharing resources, is fundamental to cloud computing. Service providers are able to build network infrastructures and data architectures that are computationally very efficient, highly scalable, and easily incremented to serve the many customers that share them. Multi-tenancy spans the layers at which services are provided [9]. In IaaS, tenants share infrastructure resources like hardware, computer servers, and data storage devices. With SaaS, tenants are sourcing the same application (e.g., Salesforce.com), which means that data of multiple tenants is likely stored in the same database and may even share the same tables. When it comes to security, the risks with multi-tenancy must be addressed at all layers.

Individual tenants share the use of the hoster's hardware and infrastructure, as well as sharing databases and database systems. Service suppliers must provide a platform with appropriate capacity and performance for hosted services. They must also consider how to keep the cost structure under control, and how they will provide customization through configuration. There are four common stages in moving towards an efficient multi-tenancy architecture with user-enabled configuration. The following sections describe these stages.

- Custom: Each customer runs a separate copy of the software assigned only to that customer, and the only way to support multiple customers is to serve them with different copies of the software. Furthermore, because little is done to allow customization through configuration, each copy includes specific customer customizations in the form of custom extension code, custom processes, and/or custom data extensions. Although the software is, technically, delivered as a service (it does not run on the customer's premises), economy of scale cannot be achieved because each customer runs a different instance of the software. Although this could be a useful starting point to validate the business model, it must be avoided once the volume of customers increases. It is impractical to manage thousands of customers using this model.
- Configurable: The software can be tailored for each tenant through configuration and by avoiding the use of custom code. All the tenants run the same code; however, the architecture is still not multi-tenant and each customer runs their own copy of the code, even though the copies are identical. The separation can be either virtual (virtual machines on a same server) or physical (running on separate machines). Although this model is a considerable improvement over the custom model described above, the architecture still allows customization through configuration, and the computing power is not shared among the instances. Therefore, the provider cannot achieve economy of scale.
- Multi-tenant: The UI can be customizable per tenant, as can the business rules and the data model. The customization per tenant is entirely through configuration using a self service tool, which removes the requirement for the service provider to perform configuration. This level is almost the SaaS perfect case; the exception is any capacity to scale out. At this level, data partitioning means that growth can only be achieved by scaling up.
- Scalable: The architecture supports multi-tenancy and configuration, plus the capability to scale out the application. New instances of the software can be transparently added to the instance pool to dynamically support the increasing load. Appropriate data partitioning, stateless component design, and shared metadata access are part of the design. At this level, a Tenant Load Balancer (implemented using a round robin or a rule based mechanism) is introduced, maximizing the utilization of hosting resources such as CPU and storage.

This means that the total load is distributed across the entire available infrastructure. The data is also reorganized periodically in order to average the data load per instance. The architecture is scalable, multi-tenant, and customizable through configuration.

f) Performance

Cloud-hosted applications must be scalable to support increasing numbers of services, and increasing load for each service and tenant [11]. When designing services, consider the following guidelines for scaling applications:

- Design services and components to be stateless where possible. This minimizes memory usage for the service, and improves the opportunity to scale out and load balance servers.
- Use asynchronous input and output calls, which allow the applications to do useful work while waiting for I/O to complete.
- Investigate the capabilities of the hosting platform that can improve performance. For example, in Microsoft Azure, use queues to manage requests and worker processes to carry out background processing.
- Use resource pooling for threads, network, and database connections.
- Maximize concurrency by using locking only where absolutely necessary.

When scaling data storage and applications, consider the following guidelines:

- When scaling the data partition, divide subscriber data into smaller partitions to meet performance goals. Use schemes such as Hashing (to subdivide content) and Temporal (based on the time or date range in which the data is valid).
- Consider implementing dynamic repartitioning to repartition the data automatically when the database size reaches a specific maximum size.

When scaling data storage and applications investigate standard patterns, and the specific techniques and implementations provided by the hosting platform—some examples are data partitioning, load balancing, failover, and geographical distribution.

g) Service Composition

Users in enterprise-level organizations require access to many different document repositories, types of data, sources of information, and applications that perform specific functions. Traditionally, users interacted directly with each store or application, often using specific isolated applications. However, over time, enterprises have attempted to consolidate systems; often using

DOI: 10.32377/cvrjst0303

intranet Web portals or façade-style applications that connect to the appropriate downstream applications.

With the advent of services and SOA applications, IT departments can expose applications and data as services, either hosted in-house or bought in as SaaS. The service portfolios can still expose the combination of traditional local applications, internally hosted services, and remote services through portals, which hide the user from the implementations and allow IT departments to adapt the ranges of services quickly and easily. However, S+S and SaaS designs and technologies allow IT departments and enterprise customers to integrate services fully. Service integration can help to achieve the goal of a *many to one* model where all applications and services are available to the user through a composition architecture that effectively exposes them as a single application, as shown in Figure 1. A service integration mechanism combines the groups of applications in the portfolios and exposes them though a rich client that can interact with any service or application.

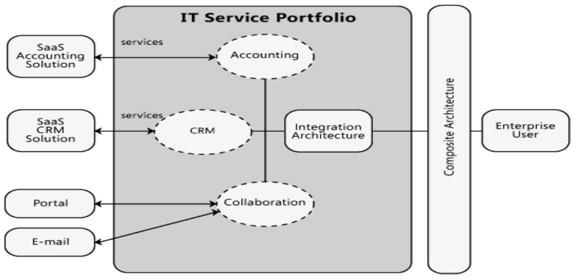

Figure1. Service Integration

h) Service Integration

Cloud-hosted solutions can help to mitigate some of the challenges encountered with traditional software, but add new and different challenges for the consumer of these services. Consider the following the challenges when moving to hosted cloud services and applications:

- Identity Management: Enterprise procedures for adding, updating, and removing users must be extended to the remote services. If the external service depends on user identity, which is very likely for SaaS and for S+S, the provisioning and deprovisioning processes must be extended. In addition, translation of in-house user identity into specific roles may be required, possibly through a federated service, to minimize the migration or duplication of individual user identities at the remote service host. Enterprise user account policies such as password complexity and account lockouts must also be compatible with those of the remote service supplier. If no SSO facility is available, there can be increased liabilities, maintenance costs, and operational inefficiencies.
- Data: Requirements of data operations, such as Extract, Transform, and Load and data integration, must be analyzed for compatibility with service capabilities. Hosted services may not support complex data storage patterns, which may affect the design of data entities and application architecture. In addition, data may need to be protected more securely to counterbalance the lack of physical security available when hosting in-house. However, applications can store sensitive or private data locally, and use the cloud services only for nonsensitive data.
- Operations: In-house integration services and client applications may not be compatible with services exposed by the service supplier, even when using industry standard protocols. You must also ensure that the service provider can generate appropriate reporting information, and determine how you will integrate this with your own management and reporting systems. In terms of service levels, Service Level Agreements (SLAs) may require revision to ensure that they can still be met when depending on the service provider for escalated support. Enterprises must also be prepared to implement help desk facilities that act as the first contact point for users, and define procedures for escalating issues with the service provider.
- Security: Enterprise privacy policies must be compatible with those of the service provider, and rules for actions that users can execute, such as limits on transaction size and other business rules, must be maintained—even if these are not part of the remote service capabilities. This may make the service integration infrastructure more complex. Procedures and policies for maintaining the security and integrity of data in the event of service or interconnectivity failure will also be required. Authentication, encryption, and the use of digital signatures will require the purchase of certificates from certified providers, and may require implementation of a Public Key Infrastructure (PKI). In addition, integration may require changes to firewall rules, and updates to firewall hardware and software may need to be required to provide filtering for application data and XML Schema validation.
- Connectivity: Some types of cloud-based applications rely on good quality broadband Internet connections to function well. Examples are online transaction processing and real time services such as voice over IP (VoIP) and Microsoft Office Communications Server. In some areas and some countries, this may not be available. In addition, services that require large data transfers such as backup services and file delivery services will generally run more slowly over an Internet connection compared to a local or in-house implementation, which may be an issue. However, messaging and other similar services may not be as dependent on connection

bandwidth or severely affected by occasional loss of connectivity.

- Service Level Agreements: Skills and expertise will be required to assess suppliers more comprehensively, and make choices regarding service acquisition and contracts. SLAs may also require revision to ensure that they can still be met when depending on the services hosted by a remote provider.
- Compliance and Legal Obligations: Compliance with legal and corporate directives may be affected by the performance of the service supplier, or these compliance directives and legal obligations may conflict if the service provider is located in another country or region. There may also be costs associated with obtaining compliance reports from the service supplier. Local laws and policies may prevent some types of applications, such as banking applications, from running in hosted scenarios.

V RELEVANT DESIGN PATTERNS

Key patterns are organized into categories such as Data Availability, Data Transfer, Data Transformation, Integration and Composition, Performance and Reliability, and User Experience as shown in the following table. Consider using these patterns when making design decisions for each category.

Category	Relevant patterns
Data Availability	Polling: One source queries the other for changes, typically at regular intervals. Push: A source with changed data communicates changes to the data sink every time data in a data source changes, or only at regular intervals. Publish/Subscribe: A hybrid approach that combines aspects of both polling and pushing. When a change is made to a data source, it publishes a change notification event, to which the data sink can subscribe.
Data Transfer	Asynchronous Data Transfer: A message-based method where the sender and receiver exchange data without waiting for a response. Synchronous Data Transfer. An interface-based method where the sender and receiver exchange data in real time.
Data Transformation	Shared Database: All applications that you are integrating read data directly from the same database. Maintain Data Copies: Maintain copies of the application's database so that other applications can read the data (and potentially update it). File Transfer: Make the data available by transporting a file that is an extract from the application's database so that other applications can load the data from the files.
Integration and Composition	Broker: Hide the implementation details of remote service invocation by encapsulating them into a layer other than the business component itself. Composition: Combine multiple services, applications, or documents into an integrated interface while performing security, validation, transformation, and related tasks on each data source. Portal Integration: Create a portal application that displays the information retrieved from multiple applications within a unified UI. The user can then perform the required tasks based on the information displayed in this portal.
Performance and Reliability	Server Clustering: Design your application infrastructure so that your servers appear to users and applications as virtual unified computing resources to enhance availability, scalability, or both. Load-Balanced Cluster. Install your service or application onto multiple servers that are configured to share the workload. The load-balanced hosts concurrently respond to different client requests, even multiple requests from the same client. Failover Cluster. Install your application or service on multiple servers that are configured to take over for one another when a failure occurs. Each server in the cluster has at least one other server in the cluster identified as its standby server.

Table I.
Patterns to be considered while Designing Cloud-Hosted Applications

DOI: 10.32377/cvrjst0303

CONCLUSION

Cloud Computing is the cost, time and performance effective. Some basic issues are the key concern in the Cloud Computing use and in the implementation for the Client as well as for Vendors.Cloud computing allows to store and maintain data on remote servers that are managed by Cloud Service Providers (CSP) .The concept of building or consuming services and applications that are hosted off-premises is becoming more attractive both to independent software vendors (ISVs) and to enterprises as a way to reduce costs, maximize efficiency, and extend capabilities.The current technology does not provide all the requirements needed by the cloud computing. There are many challenges to be addressed by the researchers for making cloud computing work well in reality. Some of the challenges like security issues and Data issues are very much required for the customers to use the services provided by the cloud. Similarly challenges like Security, performance issues and other issues like service comspostion etc are important for the service providers to improve the services. In this paper we have identified the challenges in terms of security issues, data challenges, performance challenges and other design challenges. We have provided an insight into the possible solutions to these problems even though lot of work is needed to be done in this regard

REFERENCES

[1] Peter Mell, and Tim Grance,"The NIST Definition of Cloud Computing," Version15, 10-7-09 http://www.wheresmyserver.co.nz/storage/media/faq-files/ cloud-def-v15.pdf.

[2] "Software + Services (S+S)" at http://msdn.microsoft.com/en-us/architecture/aa699384.aspx

[3] Traian Andrei, "Cloud Computing Challenges and Related Security Issues", Survey Paper

[4] Kresimir Popovic, et al., "Cloud Computing issues and challanges" MIPRO 2010 May 24-28 Opatija, Croatia, pages 344-349.

[5] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou, "Achieving Secure, Scalable, and Fine-grained Data Dccess Control in Cloud Computing", The 29th IEEE Conference on Computer Communications (INFOCOM'10), San Diego, CA, March 15-19, 2010.

[6] Bhavani Thuraisingham, Vaibhav Khadilkar, Anuj Gupta, Murat Kantarcioglu, Latifur Khan, "Secure Data Storage and Retrieval in the Cloud", The University of Texas at Dallas

[7] L. Kaufman. "Data security in the world of cloud computing". IEEE SECURITY & PRIVACY, 7(4), July- August 2009.

[8] Kevin Hamlen, Murat Kantarcioglu, Latifur Khan, Bhavani Thuraisingham, "Security Issues for Cloud Computing", International Journal of Information Security and Privacy, 4(2), April-June 2010

[9] "Multi-Tenant Data Architecture" at http://msdn.microsoft.com/en-us/architecture/aa479086.aspx

[10] Sadie Creese, Paul Hopkins, Siani Pearson, Yun Shen, "Data Protection-Aware Design for Cloud Computing",HPL-2009-192

[11] Michael Olson, K. Mani Chandy,"Performance issues in cloud computing for cyber-physical applications"