the Availability Digest

www.availabilitydigest.com
@availabilitydig

Adding High Availability to the Cloud

Paul J. Holenstein
Executive Vice President
Gravic, Inc.
August 2014

Companies are moving more and more IT services and utility applications to public clouds to take advantage of the economy and flexibility of cloud computing. There are no initial expenditures for equipment, data center space, or staff. A company must only cover the costs of the computation, storage, and communication resources that it uses.

However, there remains a reluctance to move critical applications to the public cloud and for good reason. Notable instances of public cloud failures and data loss have been frequently reported, and security is a concern because the organization no longer has direct control over its environment.

There is motivation, though, to find a way for critical applications to take advantage of the benefits of cloud computing. A hybrid approach that assigns critical processing to highly available private systems such as HP NonStop servers and noncritical processing to the public cloud is a concept that is gaining momentum. Data replication plays an important role in this approach by providing the 'glue that binds' virtual machines running in the public cloud to highly available private servers handling the critical roles.

In this article, we discuss the role that data replication solutions can play in public, private, and hybrid-cloud computing for critical applications and how it can lower your IT costs, improve fault tolerance of your applications, and increase flexibility.

Cloud Computing

A cloud computing environment comprises a network of compute resources, including CPU, memory, persistent storage, and communication capacity. These resources are assigned to users of the cloud on an as-needed basis. If a particular application running in the cloud is suddenly presented with an increased transaction rate, the cloud infrastructure will assign more resources to it. As the application's load decreases, excess resources are returned to the cloud pool. The cloud business model is pay-as-you-go, meaning a user pays only for the resources actually utilized.

The architectural foundation for clouds is *virtualized machines* (VMs), as shown in Figure 1. A VM is a virtualized server that runs with other VMs on a single physical host server. A VM comprises a guest operating system running one or more applications. Supported guest operating systems typically include Linux, Windows, and some UNIX operating systems.

The VMs share the compute, memory, communications, and storage resources of the host server. Access to and allocation of these resources is adjudicated by a *hypervisor*. Typically, multiple interconnected host servers exist within the cloud.

When a user decides to run an application, the cloud will create a VM running on some physical server in the cloud environment. The guest operating system to be used for that VM will be booted onto that server, and the user application will be started. Thereafter, the user's application runs just as if it had sole possession of its own physical server – it is unaware that it is sharing resources with other VMs. Except for the overall load sharing of the physical resources, the applications and data running within the VM are isolated from all other VMs running on the same physical equipment.

A key point is that the physical representation of a VM is nothing more than a server image stored on SAN or NAS storage accessible by all physical host servers in the cloud. The server images contain the executables for the guest operating system used by the VM and the executables for all of the applications that are to be started. This accessibility provides a great deal of flexibility in a cloud. A VM can be moved from one physical server to another: simply terminate it on the old server and reboot it onto the new server via its server image on the common storage. This flexibility allows the cloud to move VMs around for load balancing and to relocate them in the event of a physical server failure.

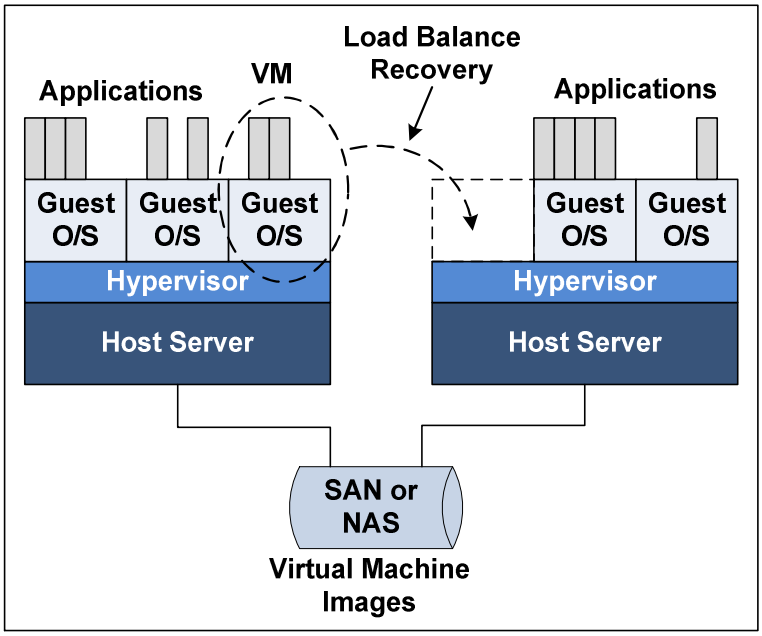

Figure 1: Virtual Machines in the Cloud

Equally important, if a VM begins to handle a larger load and requires more resources than it can get from its current physical server, it can be moved to another server that has the necessary resources. As that need dwindles, the VM can be relocated to a server whose resources more closely match its needs. The resources used by a VM are monitored by the cloud and are used to bill the user for his actual usage.

There are *public* clouds, *private* clouds, and *hybrid* clouds. A public cloud serves any user who wishes to subscribe to its services. A private cloud is owned and managed by an organization for its own internal use. An application running in a hybrid cloud spreads its resource requirements across public and private clouds or other internal IT infrastructures (for example, an HP NonStop server which is not running as a part of any cloud).

Cloud Lock-In and OpenStack

Today's large public clouds include Amazon Web Services (AWS) and Microsoft Azure. The problem with these clouds is that they are proprietary. Once a user selects a cloud for their applications, they are locked in. It is very difficult to move to another cloud provider.

A move is underway to correct this situation. OpenStack[1] is an open source cloud suitable for private and public clouds. As the OpenStack website explains, OpenStack is a "large scale open source cloud project and community established to drive industry standards, end cloud lock-in, and speed adoption of cloud technologies by service providers, enterprises, and government agencies to build massively scalable public and private clouds using freely available Apache-licensed software." OpenStack is patterned after Amazon's AWS cloud. It consists of three major facilities, as shown in Figure 2.

OpenStack Compute includes VMs to provide compute capacity. Most major hypervisors are supported, as are Windows, Linux, and some UNIX guest operating systems. An OpenStack VM can have two IP addresses – one for internal communications and one for external communications.

OpenStack Storage is a massive block store that provides persistent storage. It is modeled after Amazon's Elastic Block Store (EBS).[2] Storage assigned to an OpenStack VM does not survive when the VM is terminated. If the storage must be persistent, it must be directed to OpenStack Storage.

OpenStack Image uses OpenStack Storage to catalog and manage server images.

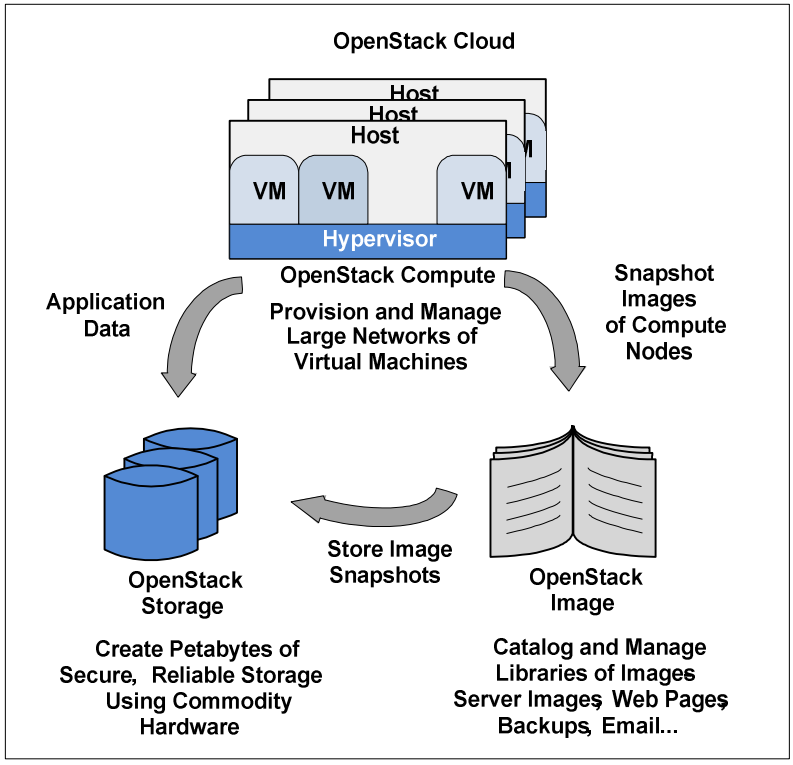

Figure 2: The OpenStack Architecture

OpenStack is currently used by the Rackspace cloud and by HP's CloudSystem Matrix, which allows companies to set up their own private, public, or hybrid clouds with HP's guidance. OpenStack is enjoying explosive growth with companies that are implementing their own clouds. If OpenStack becomes the de-facto standard for new clouds, the problem of cloud lock-in will largely disappear.

[1] OpenStack – The Open Cloud, *Availability Digest*; April 2012.
http://www.availabilitydigest.com/public_articles/0704/open_stack.pdf
[2] Amazon's Availability Zones, *Availability Digest*; November 2011.
http://www.availabilitydigest.com/public_articles/0611/amazon_availability_zones.pdf

Integrating a Data Replication Engine with the Cloud

Figure 3 shows a simplified view of a typical data replication engine. The role of the replication engine is to replicate data generated by a source environment to a target environment in real-time. The source and target environments can be on the same system, or on completely different heterogeneous systems.

The replication engine comprises an Extractor and an Applier. The Extractor resides in the source environment. It is fed data updates from applications (the *push* model), or it reads updates from a change queue (the *pull* model). The change queue may be generated by the applications, or it may be a transactional data log such as those maintained by transaction processing managers, e.g., the Audit Trail in HP NonStop systems or the Redo Log in Oracle databases. Various transaction processing managers are supported by different cloud implementations. For instance, OpenStack supports MySQL, and Amazon AWS supports MySQL, Oracle, and SQL Server.

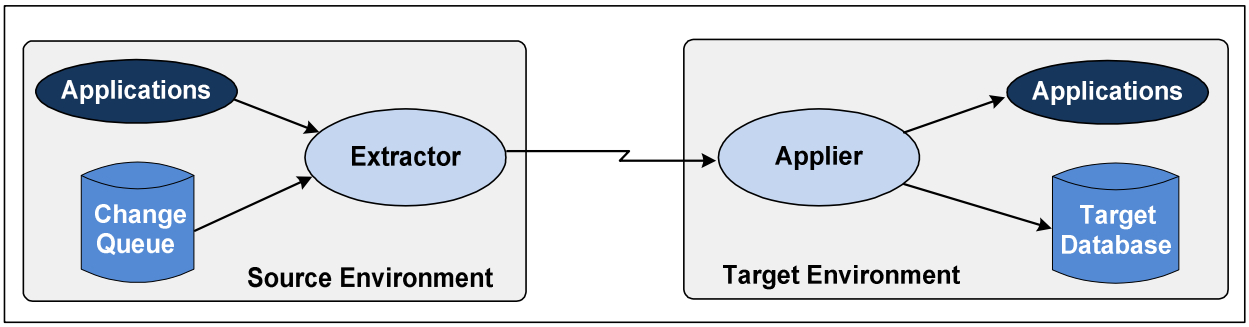

Figure 3: A Data Replication Engine

As the Extractor receives updates, it sends them over a communication channel to the Applier resident in the target environment. There, the Applier applies the data updates to a target database and/or sends them to target applications. In this way, the target environment is kept synchronized with the source environment.

In order to be effective in cloud environments, the replication engine must be heterogeneous. The source and target platforms may be different, as may be the source and target databases. The replication engine must guarantee the transactional integrity of the data it is applying to the target environment. All target databases and applications must be available for application use (both read and write) during replication.

Replication may be uni-directional or bi-directional (carried out in both directions). With bi-directional replication, a change to either database is replicated to the other. Thus, an application may be active in both environments, and the two environments are kept synchronized. A transaction can be sent to either environment for processing and will achieve the same result. This architecture is known as an *active/active* system.

Data Replication in the Cloud

As shown in Figure 4, the data replication engine can be integrated into the cloud so that it can transfer data internally between cloud applications and databases as well as between cloud application/databases and external systems. This capability depends upon the fact that for many cloud systems (OpenStack and Amazon AWS included), a VM may have two IP addresses – one for internal communications and one for external communications.

For data transfer from the cloud to an external system, the Extractor is configured to run inside a VM in the cloud. The cloud-resident Extractor uses the local change queue to collect data from local cloud applications or databases. In the event-driven push model, the Extractor receives the updates from the

change queue as they are generated and sends them to the Applier on the external system. In the pull model, the Applier is configured to connect to the Extractor and pull the changes periodically. Regardless of the model, the Extractor uses the VM's external IP address to send updates to the Applier located in the external system. The Applier applies the updates to a target database or sends them to target applications, as appropriate.

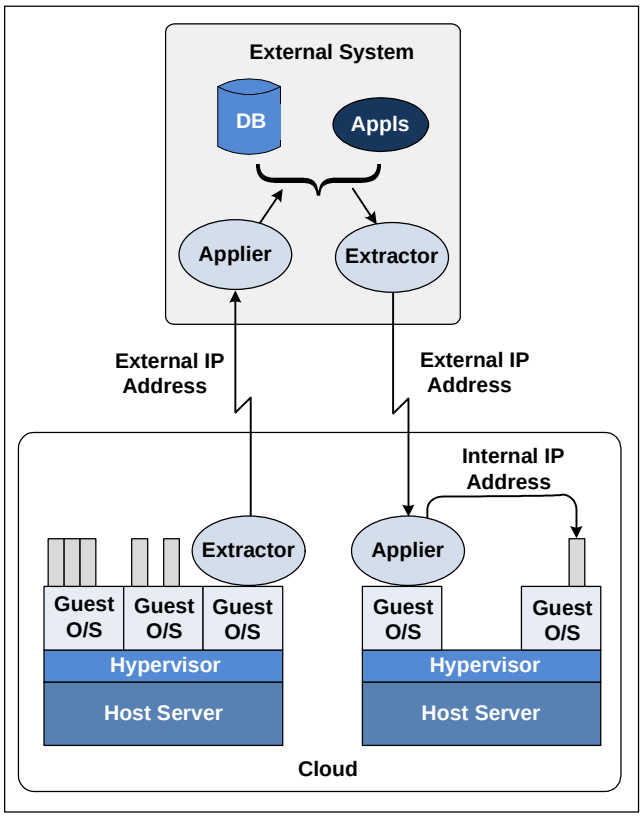

Figure 4: A Data Replication Engine in the Cloud

For data transfer from an external system into the cloud, the Applier is configured in a VM. It receives updates via its external IP address from the Extractor resident in the external system and sends these updates to applications via their internal IP addresses or applies these updates directly into the databases stored in the NAS or SAN.

Multiple instances of the data replication engine can be configured to support multiple external systems. Thus, by configuring replication engine components inside VMs running in the cloud, replication engines can transfer data in either direction between the cloud and external systems.

Data Replication In-The-Cloud Use Cases

There are many ways that a data replication engine can be used to enable businesses to take advantage of the benefits of the cloud while avoiding some of the issues discussed earlier. Most of these techniques exploit the replication engine's ability to keep cloud applications and data synchronized with external systems, thereby enabling businesses to shift noncritical functions to the cloud while maintaining critical or sensitive functions in the company's internal systems (i.e., in a hybrid cloud architecture).

In the following examples, the critical systems could be HP NonStop servers; they are the pinnacle of massively parallel, continuously available systems and are typically used for the most demanding IT

business services. Alternatively, they could be any of the many other platform and database types used in data processing applications. Potential uses of data replication engines to add resiliency to cloud applications are displayed in Figure 5.

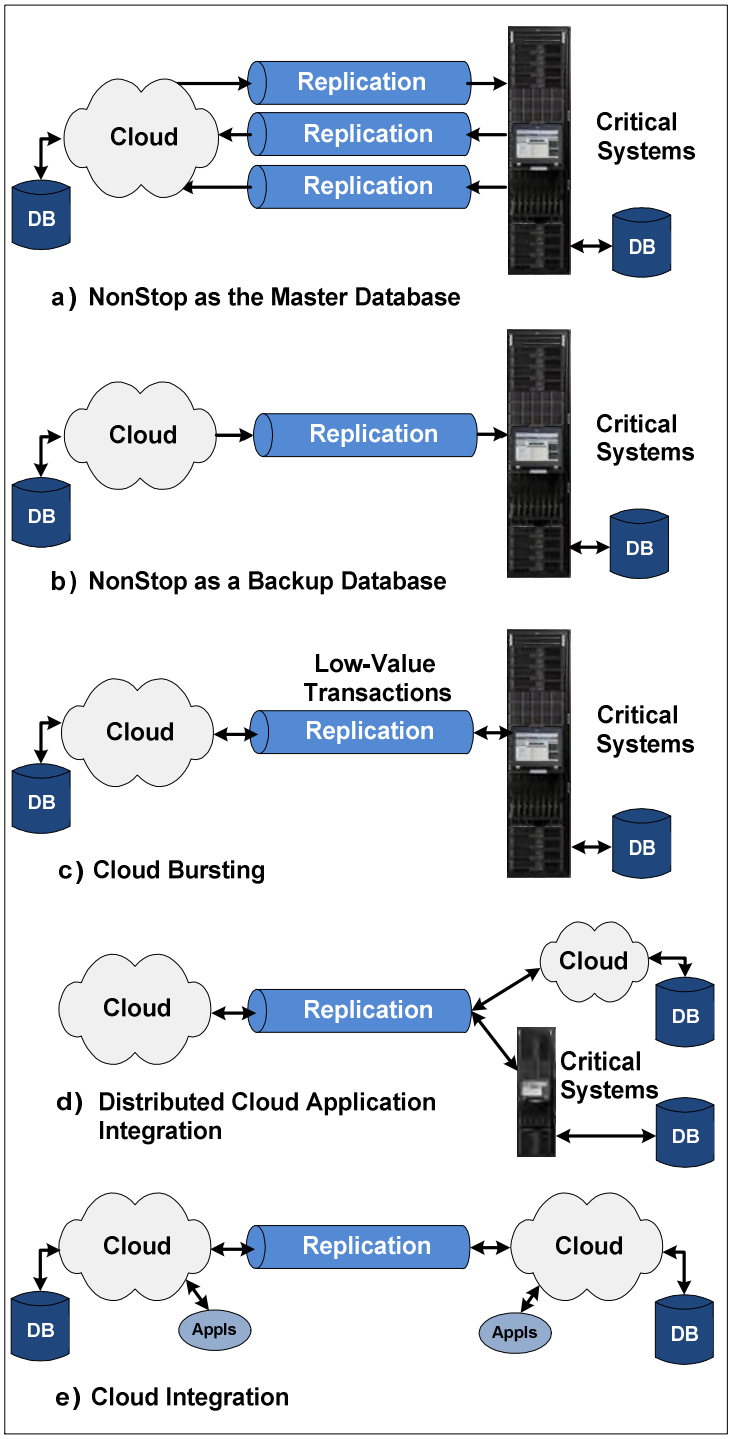

Figure 5: Replication Use Cases

Critical System as the Master Database

In this example the critical system is used as the master database with a replicated copy kept locally by all cloud applications (Figure 5a). This configuration can operate in two modes depending on the application architecture.

In the first mode, all updates are made to the critical system master database. The replication engine replicates these updates to local copies of the database in the cloud, so the cloud applications have access to the latest data and can perform fast, local, read operations on that data. This configuration is particularly useful for applications that have a high look-to-book ratio. For instance, a travel agency system spends a lot of time searching for deals and answering queries about hotel, plane, and car reservations. These searches are read operations and can all be executed quickly against the cloud local replicated copies of the master database. Only when a reservation is to be made is it necessary to update the master database.

In the second mode, update operations are made by applications to the cloud local database(s).The replication engine replicates these updates to the critical system master database, applies them, and can re-replicate these changes back to any other cloud-based copies of the data. In this way, the master database is kept current and comprises all updates made by applications anywhere in the cloud. It is the single, complete, database of record.

Critical System as a Backup Database

Another solution to protect critical data in the cloud is to provide backup facilities outside of the cloud. The replication engine can be used to make periodic (or continuous, streaming) backups of data that is being applied to a cloud-resident database (Figure 5b). The backup database can then be used to recover and restore data in the event that the cloud databases are lost or become corrupted.

Cloud Bursting

Cloud bursting entails sending some of the transaction load to the cloud if the critical system becomes heavily loaded. If the critical system begins to grow heavily loaded, then it can send low-value transactions to the cloud applications for processing. High-value transactions can continue to be processed by the critical server. Replication technology supports cloud bursting architectures (Figure 5c) by replicating database changes between the cloud and critical system databases, thereby ensuring that applications running in either environment are operating on the same, current, and consistent data.

Distributed Cloud Application Integration

Some applications may be distributed between both the cloud and critical systems (or between clouds). When an event is generated by one application, it needs to be propagated to another application for further processing. The travel agency application is a good example. When a reservation is to be made, the cloud based “look” application may raise an event to cause the critical system “book” application to actually make the reservation. Some replication architectures (Figure 5d) provide mechanisms to detect such application events when they are raised and send them to target applications (or databases) to be acted upon. In this way, replication technology provides the event detection and distribution fabric between distributed applications, enabling applications running in the cloud to be integrated with other applications running either in the cloud or on critical systems.

Cloud Integration for High/Continuous Availability as well as Application Interoperability

Figure 5e shows the replication engine integrating one cloud environment with another for providing high or continuous application availability. In some cases, the cloud operators may provide sufficient capability to provide these business continuity services, but their offerings may not be sufficient, complete, or able to work heterogeneously. They simply may be too expensive. For these cases, the data replication engine can replicate the data from one cloud environment uni-directionally into another (for example to

implement a highly available disaster recovery cloud-based architecture), as well as bi-directionally between two cloud environments (e.g., to implement a continuously available active/active business continuity cloud-based architecture).

In another similar example, the major clouds today are proprietary and cannot work together. However, a replication engine can be used to connect VMs and their applications in one cloud with those of another heterogeneous cloud. In this case (Figure 5e), the VMs in the different clouds are communicating with each other.

This architecture allows applications to be spread among multiple clouds and still interoperate. In fact, the same application can be running in multiple clouds in an active/active mode. If one cloud fails, then all transactions are simply routed to the surviving cloud, and the application users are unaffected. Moreover, if the shift in transaction workload is significant, the surviving cloud will automatically assign additional resources to the application to maintain the application's performance levels. The replication software is able to handle the necessary data reformatting between heterogeneous cloud-resident databases and applications.

Shadowbase Data Replication Software – Integrates Applications with the Cloud

The Shadowbase replication engine is designed to integrate diverse applications and databases. Its application to the cloud is simply an extension of the multiple source and target environments that it already supports.

A large, cloud-based application may comprise many different systems with different missions. Each system is implemented on a "best-fit" platform. There may be a myriad of heterogeneous platforms, applications, and databases that make up the application. Add to this mix the application components that reside in one or more clouds. A powerful, flexible, fast, and reliable data distribution fabric is required to interconnect these systems. The Shadowbase data replication engine fulfills this role.

Data must be passed from external information sources to the cloud applications and between the systems and cloud applications themselves. Data transfer must be secure and Shadowbase software supports encrypting the communication traffic, for example via SSL connections. Data transfer must be very fast, with high capacity and minimum latency. The Shadowbase engine's process-to-process architecture eliminates disk-queuing points that can slow down replication and otherwise consume extensive disk resources. Sub-second replication latency is achieved. The replication engine can be multithreaded, including the communication channels, so that any desired data transfer capacity can be attained.

In some cases, the data is being generated by an application. In other cases, it is being taken from a database. The data must be delivered to a target application or to a target database. The Shadowbase engine supports heterogeneity. It can receive data as it is generated from any supported application or database and can deliver it to any supported application or database, inside or outside the cloud.

Replication between diverse applications and databases often requires complex reformatting and restructuring of the data. Shadowbase software includes provisions for many types of reformatting. If it does not inherently support a specific type of reformatting, it provides user exits so the user can define extended reformatting algorithms.

The Shadowbase engine can replicate data synchronously or asynchronously. Asynchronous replication takes place after the fact and has no impact on the source application. Synchronous replication guarantees that an update is either made to both the source and target environments or that it is made to neither. Both synchronous and asynchronous replication guarantee the transactional consistency of the target database and that the target database is available for application processing during replication.

Shadowbase software is architected for continuous availability. If one of its components fails, it is automatically restarted and replication continues uninterrupted. If the target environment fails, then the

Shadowbase replication engine queues all events until the target environment is restored to service. It will then drain its queue of saved events to bring the target environment back into synchronization with the data source and resume replication.

The data distribution fabric within a complex cloud environment and between the cloud and external systems must be low latency and provide high capacity. It must be fundamentally heterogeneous and be able to deal with any application or database as a source or as a target. It must be able to reformat and restructure data on the fly as it moves data from one source to another totally different target. It must be highly reliable.

All of these are attributes of Shadowbase replication: low latency, high capacity, heterogeneous, powerful message processing, flexible end points, and high availability. Integrating heterogeneous data resources is a formidable challenge, and one that has been solved by Shadowbase software, which is positioned to form the ideal foundation for the data distribution fabric for cloud computing, public, private, or hybrid.

Summary

Tremendous incentives exist for organizations to move to cloud computing, since there are few initial expenditures and hiring new staff is not necessary. However, the track record for clouds is still spotty. Major cloud failures routinely make the headlines, and security is an issue. Thus, executives are reluctant to trust their critical applications and data to the cloud.

The Shadowbase data replication engine is an example of a tool which can alleviate many of these concerns. It serves as the glue that binds a company's critical applications and data to trusted corporate internal systems while allowing the cloud to perform less critical functions, for which the cloud is well-suited. With Shadowbase software solutions, an optimum compromise of application safety versus cost can be achieved, enabling businesses to take advantage of the benefits of the cloud while avoiding the pitfalls.