IT@Intel White Paper

Intel IT
IT Best Practices
Cloud Computing and IT Business Agility
October 2012

Developing a Highly Available, Dynamic Hybrid Cloud Environment

Although open source solutions for the hybrid cloud are still evolving, we believe there is enough maturity in the marketplace to enable Intel to begin using hybrid cloud hosting to bolster our consumer-facing web services' agility and reliability.

Executive Overview

Intel IT is exploring hybrid clouds—a mix of public and private clouds—for consumer-facing web services. The hosting environment for such services requires a different approach than what is necessary for our internal Office and Enterprise environment.

The issues we need to address include the following:

- Consumers expect web services to be always available.
- Demand for services can be unpredictable, spiking and dropping without warning.

We are working to accelerate Intel's adoption of a hybrid cloud by establishing key design strategies.

- Hosting applications across three availability zones (two at Intel and one at an external provider) provides a distributed hosting presence that supports high availability.
- Designing applications and the hosting environment for automated self-healing enables speedy detection and correction of application and infrastructure failures.
- Designing our hybrid cloud so that it can automatically meet unpredictable demand, using and relinquishing resources as demand increases and decreases.

We are also actively encouraging application development teams to design cloud-aware applications that gracefully accommodate infrastructure outages and that can be concurrently active at multiple locations.

We have established three primary teams—cloud engineering, cloud system administration, and cloud integration and brokering—to further facilitate the implementation of our hybrid cloud.

Although open source solutions for the hybrid cloud are still evolving and interoperability issues need to be abstracted when operating across public and private cloud instances, we believe there is enough maturity in the marketplace to enable Intel to begin using hybrid cloud hosting to bolster our consumer-facing web services' agility and reliability.

Munir Ghamrawi
Cloud Database Engineer, Intel IT

Das Kamhout
Principal Engineer/Intel IT Cloud Lead, Intel IT

Sridhar Mahankali
Cloud Network Engineer, Intel IT

Contents

IT@INTEL

The IT@Intel program connects IT professionals around the world with their peers inside our organization – sharing lessons learned, methods and strategies. Our goal is simple: Share Intel IT best practices that create business value and make IT a competitive advantage. Visit us today at www.intel.com/IT or contact your local Intel representative if you'd like to learn more.

BACKGROUND

As part of our cloud computing initiatives, Intel IT has created a roadmap to a public-private, or hybrid, cloud hosting model using primarily open source solutions. This hybrid cloud model will enable us to achieve even higher levels of agility, scalability, and efficiency. We are working to accelerate the adoption of a hybrid cloud by establishing strategies such as working with business groups to design cloud-aware applications and designing Intel's infrastructure to be more cloud-centric.

Over the last few years, Intel has been exploring and enabling several consumer-focused web services such as the Intel AppUp® center, which is already available to consumers. Providing a hosting environment for such consumer-focused services involves an approach that significantly differs from hosting traditional IT applications, especially in regard to two key areas: availability and demand.

- **High availability.** In contrast to our internal Office and Enterprise environment, which Intel has been able to support with about a 99.7-percent availability level, consumers expect web services to be always available. Our goal is to attain 99.99-percent availability for consumer-facing web services, which translates to a little more than 52 minutes of annual downtime.
- **Unpredictable demand.** Computing demand in our Office and Enterprise environment is generally predictable. We can observe demand growth patterns over months and years, and plan accordingly. However, with consumer-facing web services, the demand is less predictable, spiking and dropping without warning, depending on the popularity of the application or service.

We explored different approaches to address the requirements for 99.99-percent availability and plan for unpredictable demand.

- Provision enough resources internally to handle any amount of demand
- Exclusively use public cloud providers
- Use the hybrid cloud model

SOLUTION

We made a strategic decision to adopt the hybrid cloud model, because its flexibility allows us to dynamically adjust the amount of capacity we are using in the public or private hosting environment. Our financial analysis has shown that hosting internally is more cost effective compared to hosting on the public portion of the hybrid cloud. Consequently we are using the private portion of the hybrid cloud to absorb the majority of the demand, while diverting unanticipated or a sudden increase in demand to the public portion of the hybrid cloud.

Although open source solutions in the hybrid cloud space are still emerging and evolving, we believe there is enough maturity in the marketplace to enable Intel to begin to use hybrid cloud hosting to bolster our consumer-facing web services' agility and reliability.

Developing a highly available, dynamic hybrid cloud environment requires a significantly different hosting design than we have used for our internal Office and Enterprise environment. However, we plan to implement concepts, modules, and capabilities from the hybrid cloud model into our enterprise private cloud as well.

In implementing a hybrid cloud environment, we needed to consider changes in application design and modify our internal organizational structure.

Design Strategies

To accommodate the high availability needs and unpredictable demand associated with hosting consumer-facing web services, we have developed three key design strategies.

ACTIVE-ACTIVE-ACTIVE HOSTING MODEL

We determined that hosting applications across three availability zones (two of them hosted at Intel and one at a public cloud provider) provides a distributed hosting presence that supports 99.99-percent availability even if an individual data center is available at only 99 percent.

Enhanced service availability is achieved by having the service actively handle consumer transactions concurrently at all three locations—called *an active-active-active hosting model.* Because applications are hosted at and synchronized between three locations, if one data center fails, global load balancers help redistribute the load across the remaining locations. This model assumes that the hosted applications are designed to function in an active-active-active fashion and can work around any infrastructure failures. (For more information on application design, see the Cloud-aware Application Design Tips sidebar.)

Figure 1 shows a high-level view of the active-active-active model.

The three hosting sites are interconnected using fully meshed virtual private network (VPN) tunnels to enable cross-site data synchronization. By using another set of VPN tunnels, the architecture integrates with certain key centralized management capabilities, such as service management and patch management, thereby avoiding duplication of the management infrastructure.

By using a public cloud provider, we can quickly add a hosting presence for externally facing applications on the East Coast of the United States, where we have fewer data centers. As we expand our hybrid cloud structure into other geographic regions, we can apply the same method to quickly develop an external hosting presence in an area where Intel may not have physical data centers.

Another benefit of using a public cloud provider is that it provides a sustained baseline of activity, which helps ensure that the infrastructure is already in place and only scaling is necessary. If demand suddenly grows, the public cloud provider's deployed infrastructure capacity enables us to quickly add application virtual machines (VMs) to meet consumer demand. As the demand subsides, we can release the additional resources. If demand sustains, we can deploy additional infrastructure resources at Intel data centers to rebalance the application transaction load. This model helps us optimize cost while effectively meeting our business demands.

Database Synchronization and Data Sharding

Database synchronization is an important aspect of implementing an active-active-active application hosting model. For the data to be highly available, it must be replicated between sites. However, we must balance this requirement against cost and performance.

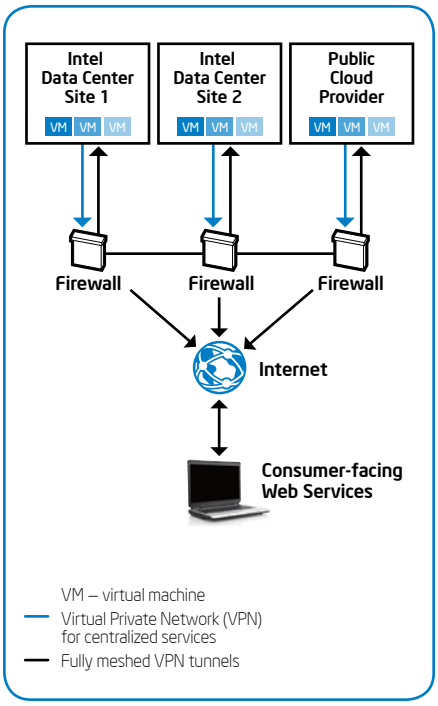

Figure 1. To help provide high availability, our active-active-active hosting model uses two internal hosting locations and one external location.

As shown in Figure 2, we use data sharding to manage our scaling needs locally within the site and remotely to other sites. A shard is a portion of a database, consisting of one or more rows. In addition, we use replication to mirror each shard to another site. Using what we call *GEO shards*, we write data for a specific geographic region to a specific designated site. We started with three sites within the United States and can add more sites as needed. Within each GEO shard, we can create local shards to enable scaling within each site.

To minimally affect application availability and to help prevent data loss, any write to a shard is replicated to an alternate site so there is always a local copy and a remote copy of that shard. For example, if a user initially accesses the Intel data center in region 1 and updates data, that data is stored in region 1's data center and synchronized with a remote site such as the Intel data center in region 2. For that particular data, writes always happen in region 1. A copy is kept in region 2. If we lose a local shard, the application accesses the shard's local replica automatically, and if we lose an entire data center, the application accesses its remote replica automatically.

We use another database synchronization mechanism, called *circular replication of data*, for targeted use cases. With circular replication, all data updates in a particular data center are replicated in a circular fashion to all other data centers hosting that application in an active-active-active manner. For example, updates in the region 1 data center are replicated to the region 2 data center and then replicated from region 2 to the region 3 data center in an asynchronous manner but with minimal latency. In this way, all the data is available at all data centers. However, this model entails higher network and storage costs.

AUTOMATED SERVICE RECOVERY

Our second key design strategy is to design applications and the hosting environment that are self-healing. To achieve 99.99 percent availability, we are limited to a little more than 52 minutes of annual downtime, whether scheduled or unscheduled. Within this constraint, human intervention to fix any problems that arise will almost certainly use all these available minutes; manual recovery means we will miss our service level agreement in the first incident that occurs in a year.

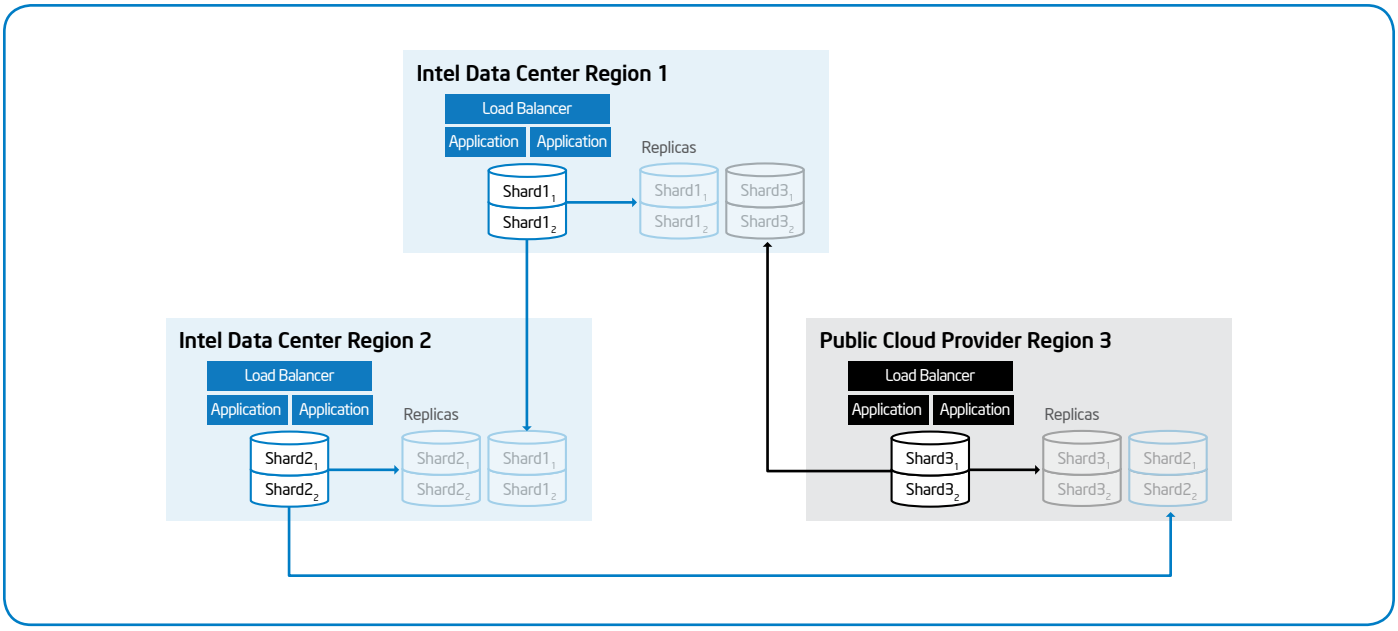

Figure 2. We found that replicating data shards between two of the three hosting sites is an efficient way to achieve high availability.

Given the critical need to use automation to address the majority of problems, we developed a self-remediation framework that consists of three primary functions, as shown in Figure 3.

- **Watcher.** We use an open source solution that implements integrated monitoring of the whole infrastructure-as-a-service (IaaS) solution stack—including compute, network, and storage capabilities—as well as monitoring of the application itself.
- **Decider.** We developed an in-house function that examines the received alerts and conducts a correlation analysis to identify the problem and initiate a remedial action.
- **Actor.** We use an open source solution that executes a sequence of pre-defined steps that help resolve the problem. This tool also uses a domain model described in the configuration management database to ensure that the infrastructure configuration does not drift from the desired settings for proper application functioning.

Currently in our first phase, our remedial actions consist of three levels. Depending on the nature of the problem, we can take out of service a single node, a set of nodes constituting an application scaling unit, or a data center. While this action is taking place, a notification is sent to a support person, who can diagnose the problem and take any additional actions.

As our processes evolve, we expect the Decider to perform increasingly complex analyses and the Actor to take increasingly complex remedial actions, thereby minimizing human intervention. For example, if the Watcher detects a faulty node, the Actor could take that VM out of service and automatically provision a new one.

AUTOMATED SCALING

Similar to automatically detecting and responding to failures, we also are designing our hybrid cloud so that it can automatically meet unpredictable demand.

Working with one of our lead customers, we have developed the concept of a scaling unit, called a *pod*, which consists of gateway servers, web servers, and a database. The number of servers in a pod is determined by the service and expected demand, with our goal being to maximize consumption and throughput across infrastructure components.

By programmatically monitoring a pre-defined metric or set of metrics, such as transactions per second, we can determine when certain thresholds are reached and initiate the provisioning of another pod to add capacity. After demand subsides, the extra pods can be decommissioned. This approach helps us make efficient use of our infrastructure resources and contain our public cloud provider costs while also enabling us to scale as needed.

Currently, we are working on developing automatic provisioning of a new pod, where human intervention is required only to initiate the pod creation process. We expect the development of automatically triggered pod provisioning and decommissioning to follow.

Architectural Details

Our private cloud environment, shown in Figure 4, takes advantage of the open source software capabilities that OpenStack* provides. These capabilities enable our cloud operating environment to expose compute, network, and storage resources using APIs, to which we can add additional customized automation.

The common compute pool hosts three types of services.

- **Cloud controller services.** These OpenStack services provide APIs to the private cloud hosting environment and perform several cloud-orchestration functions. Examples of services include provisioning VMs on a particular host, managing dynamic IP addressing, and managing storage volumes.
- **Tenant applications.** These are the consumer-facing web applications that the hosting environment is being designed for. The environment supports multiple tenants, with each tenant segmented and isolated from the other tenants using network segmentation.

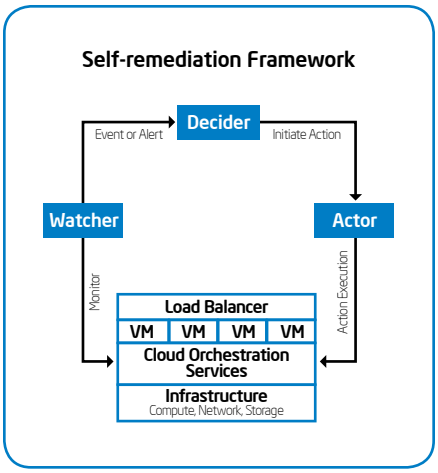

Figure 3. Our self-remediation framework consists of three functions: a Watcher, a Decider, and an Actor.

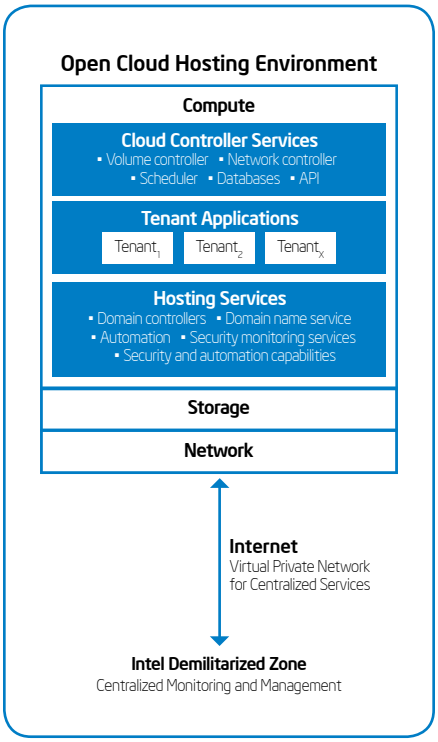

Figure 4. The capabilities of open source software make common compute, network, storage, and centralized management resources available to tenant applications.

- **Hosting services.** These include core infrastructure and application services required to operate an application hosting environment. Key services include domain controllers for administrator authentication and authorization, domain name service (DNS) for name-to-IP address mapping, security monitoring solutions, and monitoring and automation capabilities.

STATEFUL AND STATELESS ZONES

From a compute and storage perspective, we have segmented our private cloud hosting environment into stateful and stateless zones.

Services that are foundational to the hosting environment, such as DNS and domain controllers that provide core administrator authentication and authorization, are hosted in the stateful zone. These services are backed by enhanced redundancy and more robust storage. This framework ensures greater resistance to failure of these VMs and thereby greater availability of these services to the overall hosting environment.

Cloud-aware and automatically provisioned VMs are deployed in the stateless zone. For VMs deployed in this zone, the tenant application is expected to be designed to be resilient to failure of individual VMs. For example, an application can transparently retry a transaction to take advantage of other available VMs performing the same function.

Cloud-aware Application Design Tips[1]

The following design guidelines can help application development teams design cloud-aware applications that behave reliably, even when outages occur, and that provide an optimal user experience.

- **Treat everything as a service.** In the cloud, compute, network, and storage, as well as cloud applications, are software services. Increasingly, capabilities from suppliers are being delivered as software as a service (SaaS), rather than as on-premise solutions. Because cloud applications both provide and consume other web services, application capabilities should be partitioned into granular components that can be implemented, tested, and scaled separately. In essence, cloud-aware applications are a form of distributed system.
- **Use APIs that conform to REpresentational State Transfer (REST) constraints.** RESTful APIs are the building blocks of cloud applications. They enable easy re-use of an application's capabilities, support appropriate scaling of individual capabilities, and shield applications from underlying technology implementations.
- **Separate compute and persistence.** Complete separation of compute and persistence provides deployment and scaling flexibility. For example, use web services for all data storage, including log files and debugging streams, which shields the application from underlying changes in storage technology and decreases deployment overhead. Because nothing is stored locally on the compute instance that is running the cloud application, the application can run in an internal cloud, public cloud, or in both locations.
- **Design for failure.** Although the goal is zero failure, in reality components fail, services become unavailable, and latencies increase. However, cloud applications must function even when these difficulties occur. Designing applications to survive failures in a distributed environment affects decisions throughout the architecture but is especially important to consider in relation to the user experience. Applications should gracefully degrade and recover when services are unavailable.
- **Architect for resilience.** Instead of focusing on the mean time between failures, which tries to reduce the frequency of errors, cloud architects should focus on the mean time to recovery (MTTR). An architecture designed with a focus on MTTR accepts imperfection and enables rapid identification and resolution of problems when they occur.
- **Operationalize everything.** All services should be easy to maintain and easy to troubleshoot and should require the least number of people to operate. Application development teams should gather feedback about applications and services that lead to improvements in the quality and operational supportability of applications. Additionally, information gathered by instrumenting, logging, and analyzing application behavior will lead to operational improvements.
- **Implement security at every layer.** In a public cloud, a perimeter security approach is not sufficient, and a more comprehensive approach is needed. Examples include encrypted transport into the cloud, secure coding and access control inside applications, and encryption at rest. Application development teams should test and analyze the security of every API and data source.

[1] Adapted from Open Data Center Alliance "Cloud Aware Applications" white paper: www.opendatacenteralliance.org/docs/Best_Practices_whitepaper.pdf

SECURITY GROUP SEGMENTATION

As shown in Figure 5, to provide intra-tenant security segmentation, we use functional or application tier-based logical groupings of access control rules. These rules are applied to an OpenStack construct called *security groups*. Based on the VM's function, one or more security groups are associated with that VM when it is provisioned.

For example, all the Internet-facing web servers for a particular application are assigned to the Presentation Layer and Admin security groups. The Presentation Layer security group's rules allow only web application access from the Internet. The Admin security group's rules allow access for remote administration of the VMs from a set of designated hosts. Similarly, database VMs are assigned to the Database Layer and Admin security groups; the Database security group's rules allow database access only from VMs belonging to the Application Layer security group. With this type of capability we can implement flexible and VM context-aware logical segmentation within our OpenStack-based private cloud environment.

To incorporate the public cloud provider into our hybrid cloud environment, we created a logically isolated cloud environment on the public cloud. We also utilize capabilities available at the public cloud provider in a way that resembles the structure of the OpenStack-based private cloud hosting environment. These capabilities include network and security group segmentation, load balancers, monitoring solutions, and VM sizing. In addition, we developed an automation utility that helps abstract OpenStack APIs and the public cloud APIs away from the hybrid cloud administrators and tenant customers.

Application Design Changes

We are actively promoting the concept of cloud-aware applications. Traditional applications assume that infrastructure is always available, which is not necessarily true in the cloud. Application development teams need to design their applications to gracefully accommodate infrastructure outages and to be cloud-aware in order to be concurrently active at multiple locations.

Because building cloud-aware applications involves several new strategies, we are working to help promote key criteria and solutions in order to design simplified, fault-tolerant, modular services that run in a virtualized, elastic, multi-tenant environment (see the sidebar). Some aspects of application development that developers need to consider include the following:

- Avoid maintaining application states in memory; instead, write to a shared database so user context persists across application hosts and can survive individual host failures.
- Use process threads that resume on reboot.
- Resynchronize the state by reloading messages from queues.

Our intent is to have the web services layer be completely stateless. That is, if a user's transaction comes to data center A, subsequent connections might go through a different data center, but the process is transparent to the user and the results are the same as if all connections were with data center A.

Organizational Changes

To more effectively implement a hybrid cloud, we formed three new teams to help design and manage our cloud.

- **Cloud engineering.** Develops the designs and solutions for the cloud ecosystem, including infrastructure designs, manageability, and the automation solutions surrounding self-remediation, self-service, and automated scaling.
- **Cloud system administration.** This new operating model breaks down traditional organizational boundaries. Individuals on this geographically dispersed team are expected to expand their skills beyond their core domain of technical expertise and conduct operational activities across the overall cloud environment—including network, compute, and storage. Having a broad level of expertise is also critical to ensure agility whenever human intervention is required to address an event.
- **Cloud integration and brokering.** Working closely with end customers, this team integrates customers' applications into the cloud environment and provides design consultations to ensure that applications can take optimal advantage of the underlying cloud ecosystem.

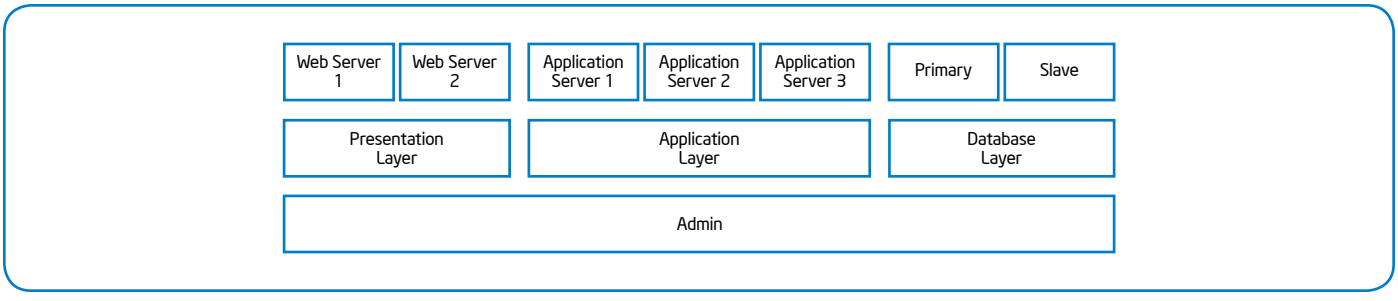

Figure 5. We use security groups for tenant segmentation.

CONCLUSION

Hosting consumer-facing web services in a hybrid cloud has required us to develop cloud design strategies that differ significantly from the ones we use for our internal Office and Enterprise environment. In particular, consumer-facing web services require a higher level of availability than typical enterprise applications. Also, demand for such services is more unpredictable.

To help accelerate Intel's adoption of a hybrid cloud, we have established three key design strategies that support high availability and dynamic demand.

- Applications are hosted across three availability zones, two at Intel and one at an external provider.
- Automated self-healing enables speedy detection and correction of application and infrastructure failures.
- Automated provisioning enables us to use and relinquish resources based on demand.

In addition to making Intel's infrastructure more cloud-centric, we are encouraging the development of cloud-aware applications that can gracefully degrade and recover from infrastructure outages and that can be concurrently active at multiple locations. The organizational changes we have made, including the formation of new teams, is also helping to facilitate implementation of the hybrid cloud.

Although open source solutions for the hybrid cloud are still evolving, we believe the marketplace is mature enough to allow us to use hybrid cloud hosting to bolster the reliability and agility of our consumer-facing web services.

CONTRIBUTORS

Jim Chamings
Winson Chan
Joel Cooklin
Mike Pedeupe

ACRONYMS

DNS	domain name service
MTTR	mean time to recovery
REST	REpresentational State Transfer
VM	virtual machine
VPN	virtual private network

For more information on Intel IT best practices, visit www.intel.com/it.