



Herbie Pearthree

# Always On: Assess, Design, Implement, and Manage Continuous Availability

## Introduction

Continuous availability's principles encompass the ability to transparently withstand component failures, the ability to introduce changes nondisruptively, and the ability to withstand a catastrophe transparently or nearly transparently. In this IBM Redpaper™ publication, we describe methods to assess, design, implement, and manage continuous operations and continuous availability so that the reader can understand the principles and apply them to their availability evolution roadmap.

Continuous availability is built upon the underlying principles behind high availability (HA) to bypass component failures and it requires the pattern of service parallelism: the business service is fully functional and running on more than one cloud or data center. Service parallelism is the key enabler behind nondisruptive changes and Disaster Transparency, the ability to withstand a catastrophic outage without needing recovery. As practitioners often focus only on the technology, evolving toward continuous availability is typically a disruptive change encompassing people (a change in culture), applications (developed for resiliency), process (continuous operations), and technology.

Achieving continuous availability (CA) or even near continuous availability (nCA) of business services across geographically distributed systems is an attainable goal based on proven technologies available to the practitioners over the past few years with modern application platforms and patterns. The difference between continuous availability and near continuous availability can be thought of as zero outage changes and "disaster transparency" – meaning no human being is involved to bypass a failure versus near continuous availability requiring short outage planned changes and "service restoration" - where human beings must be involved to bypass a failure whether it be to change the direction of replication, wait for replication synchronicity, or other control tasks. Regardless of whether a business application can be provisioned on a platform of continuous availability or near continuous availability, it is key that the perception of the user is that the service is always on.

The basic technical concept that enables continuous availability and near continuous availability is the ability to run a service from multiple “clouds” in parallel (also known as *service parallelism*). Each “cloud” is capable of running the business service independently of its peers, yet replicates state and persistent data to its peer “clouds”. Enabling the IT platform is as straightforward as implementing the three enabling technologies:

- ▶ Global traffic management, which intelligently routes users to one of the service “clouds”
- ▶ Non-persistent application data grid where sessions and non-persistent data can be replicated across “clouds”
- ▶ Guaranteed application level data replication, which enables data to persist in all clouds whether it fits the requirements of Atomicity, Consistency, Isolation, and Durability (ACID) – typically a requirement in the financial sector or eventual data consistency, which fits most other sectors (similar to the Google, Bing, Facebook, eBay, and so on)

Eventual data consistency is not a new concept, because the concept of memo-post or store and forward has been practiced in the financial sector since the advent of automated teller machines. Yet unless the business services application architecture, development, and operations processes are designed to take advantage of the flexibility and constraints of the continuous availability IT platform and availability patterns, the business availability goals will not be met.

Brewers CAP Theorem on distributed systems limits the technology solution to providing only two of the three guarantees:

- ▶ Consistency: All distributed nodes have a single up-to-date copy of all data at all times.
- ▶ Availability: Every request receives a success/failure response.
- ▶ Partition tolerance: System continues to run despite arbitrary message loss or failure of part of the system. For example, the network, stops delivering messages between server sets.

Given Brewers’ limits, it is extremely important to accurately assess the business requirements of every business service to determine which architectural pattern to apply to each specific service based on business requirements. In addition to the business requirements, the application architecture and platform must be assessed to best determine how to mitigate whichever of Brewers’ three guarantees cannot be met. It is also important to consider these limitations as they apply to the application presentation and business logic layers (also known as the *systems of engagement*) independently of the data layer (also known as the *system of record*). A common pattern might have the application pattern accepting the consistency risk although the data layer might not be able to risk consistency and therefore must choose to accept the availability risk. After the design is correctly assessed, the design phase begins and establishes the application, data, and the IT platform patterns to implement.

In parallel with the design phase, the practice of continuous operations must be established to maintain continuous availability. Typically, this is a disruptive change because the historical model of non-integrated business, development, and reactive delivery silos must be replaced with a close partnership among business, development, and business service (also known as *Biz/DevOps*)-aligned proactive delivery teams.

Managing continuous availability and near continuous availability requires zero or near-zero outage changes and can significantly change the existing incident, problem, and change processes. Take, for example, the IBM® website [www.ibm.com](http://www.ibm.com). Because the delivery team can manage zero outage changes even with an agile application development and deployment model, change management calls and planned changes are performed during normal business hours daily. Incident and problem processes must be changed to empower immediate bypass of failed components – you cannot wait for a ticket to get routed correctly and approvals requested before you bypass actions. To ensure that changes are successful and incidents are immediately handled, delivery operations must understand the business service end-to-end, be aligned to the service, and be empowered to take mitigation actions.

## **Assess**

There are multiple assessments that must be made to establish the roadmap toward continuous availability for our client's critical business services. We have found many examples of organizations that vary in their maturity and readiness to evolve to extreme levels of availability. Before designing a continuous availability or near continuous availability solution for a business service, it is imperative to assess the business service requirements, applications, and processes, and the organization's current state of maturity. The basic premise is that they must start with HA and disaster recovery (DR) – this seems somewhat obvious but it is clear that many organizations in the financial sector are bound to back-office systems based on technologies common in the 1980s, which inhibit their journey beyond HA/DR. It is imperative that assessments dive deeply into each technology domain and involve experts in each. These assessments must span people, process, applications, and technology – with people and process being the most time-consuming and difficult to change.

### **Assess the business service requirements for availability**

Many organizations are already aware of their most critical business services, especially if regulatory requirements or business requirements have mandated DR solutions for those. However, often the business is driven from the Chief Marketing Officer (CMO) and CxO levels where they believe everything must be always on. This non-functional requirement is driven by the knowledge that from a user perspective, the service must always be there, regardless of time of day. This perspective is valid because consumers access services, anytime, anywhere. However, it is not feasible, both from a technology perspective and a cost perspective to enable every business service to be always on. For a business service application to ride on a continuously available platform, it must be modernized – often a disruptive leap that requires significant development funding, extensive testing (“know how it works, know how it fails”), and an integrated operations model. Therefore, we must help organizations identify and focus on their most critical business services first so they know how to direct their funding.

Let us look at an example based on an international airline availability assessment. On initial investigation, discussions with the CMO indicate that the ability to sell tickets and upsell services is the most important business service. The Chief Operations Officer's critical business driver is “Keep the planes on schedule and in the air”. The CIO's perspective is to keep all the systems available as much as possible.

Given the broad perspectives represented at the CxO-level interviews, the assessment approach recommended development of “use cases” that are representative of the core business functions that support the airline's daily operations so that each of the scenarios traverses across multiple members of the top critical IT systems groups.

The following IT systems groups were defined:

- ▶ Flight operations
- ▶ Departure control system
- ▶ Airport operations
- ▶ Aircraft maintenance
- ▶ Crew operations
- ▶ Electronic communications
- ▶ Reservation system

The following use cases were identified:

- ▶ Print boarding pass
- ▶ Scan boarding pass
- ▶ Release a flight
- ▶ Reroute aircraft
- ▶ Purchase seat
- ▶ Check in passenger
- ▶ Electronic communications

With these use cases, interviews and data collection activities identified twenty-nine supporting applications, environments, and infrastructures that compose the foundational support of the airline operational IT architecture. These became the focus of the availability assessment.

See Figure 1 for the business criteria categorizations – the color nomenclature is used as clients have various terms for identifying business criteria categories. See Figure 1 for an overview of the categories.

Business Criteria	Platinum <5% Continuous Availability “Always On”	Gold <5% Near Continuous “Almost Always On”	Silver 20-40% High Availability “Usually On”	Bronze 50-70% Moderate Availability
<b>Business Function</b>	✓ Targeted to applications & business functions that, if unavailable, will result in either financial or legal penalties based on regulatory restrictions	✓ Targeted to business applications and functions that present a potentially broad impact across the internal organization	✓ Targeted to applications that support analysis of business functions	✓ Targeted to non-critical, back-end, offline business functions
<b>Business Impact</b>	✓ Typically assigned to the 5-10% of applications that drive revenue & profits, resulting in severe negative impact to brand reputation	✓ During critical processing windows, cannot afford to be without this function	✓ Typically backend processes with minimal impact to higher class services	✓ Typically less desirable methods are available to achieve same business function to support tolerance for extended outages
<b>Tolerance for Downtime</b>	✓ Ability to provide continuous availability potentially achieving 24x7x365 availability	✓ Ability to provide constant availability within a defined processing window with availability requirements reduced outside the window	✓ Ability to provide consistent availability within a defined processing window	✓ Availability desired but not mandated with extended outages tolerated by business
<b>Component Failure Impact</b>	✓ Component failures will not cause disruption in service	✓ Component failures should not present a disruption in service	✓ Redundancy at the subcomponent level limits outages based on a single subcomponent failure	✓ Potential outages due to single points of failure inherent within technology & application design
<b>Maintenance and Change Impact</b>	✓ Maintenance & changes required to be concurrent and/or staggered, with no interruption to service	✓ Maintenance & changes required to be concurrent or predefined outage window for change introduction	✓ Maintenance & changes require predefined outage window where changes can be introduced	✓ Maintenance & changes require a liberal outage window where changes can be introduced
Continuous Operations Required				

Figure 1 Business criteria categorizations

## Assess the business service application architecture

Can the application take advantage of the technologies that enable continuous availability and mitigate the missing Brewer's Theorem guarantee? Although this might sound like a simple question, the answer can be complex. Many existing applications have evolved over time through business requirements, acquisitions, consolidation, and so on. Many have numerous dependencies and numerous interfaces that all must be evaluated. Often, the business might not even be aware of all the dependencies and interfaces even though they are tightly coupled and must be included due to that tight coupling. The application assessment must include discovery of these dependencies and interfaces, and identify the application availability patterns that fit.

Just like assessing business services applications' readiness for cloud adoption, assessing and evolving applications for higher levels of availability might be difficult and might cause a disruptive evolution. Both cloud and continuous availability patterns fundamentally change the relationship between an application and its computing resources. Both are platforms designed for the non-functional requirements underneath a business application.

There are two adoption strategies for cloud and continuous availability patterns that enterprises must adopt:

- ▶ Cloud/continuous availability-ready applications can be adopted to run in a private cloud or continuous availability platform.
- ▶ Cloud/continuous availability-centric applications are natively built for the cloud or continuous availability platform with a different set of assumptions.

Cloud/continuous availability-ready applications are often built without cloud assumptions in mind. They are deployed using existing standard technologies, such as Java/JVM Platform, Enterprise Edition and relational databases, for example. In the case of cloud/continuous availability-ready applications, applications only need minimal knowledge of topology – follow Java Platform, Enterprise Edition best design practices or SQL best practices, for example. However, they require different enabling technologies to take advantage of capabilities, such as synchronization and auto-scaling, and must be updated to support the non-functional requirement of continuous availability. Granted, not all traditional or historical technologies can fit continuous availability patterns and therefore will remain, at best, HA with fast failover.

Cloud/continuous availability-centric applications are built for the platform and assume that individual components are untrustworthy, temporary, and will fail. They assume that all work must happen in distributed clusters (not stretch clusters) and are built by composition. The application must be fully aware of the platform topology to take advantage of the technologies that enable automatic fault transparency, and state and data replication methods. The applications must also handle resiliency – what used to be a non-functional requirement, failing gracefully without impact, and in some cases making up for the deficiencies in what the platform can deliver. This awareness usually requires a rewrite of an application because the existing applications do not meet the “shared nothing” and synchronization requirements of cloud and continuous availability platforms.

Several practices are provided that assisted with this assessment in previous client engagements.

We learned that it useful when decomposing an application's readiness and abilities for moving to a more available platform, to start simply and dive deeper. Often, evolving the application's presentation or business logic layers (systems of engagement) is not difficult because the layers are not so tightly coupled to the data. The challenge comes down to the data layer. Here, we have to focus again on the data requirements: absolute or eventual data consistency and the RPO.

First, we must talk about application architectures because not all applications are designed to take advantage of the availability patterns provided by technology. This is a critical assessment step because many clients purchased applications from vendors where they have little control over development requirements. Therefore, we must fit the solution to the limitations of the application. Where the business controls the application development, they can fund the development to support either the Active/Query solution to provide near continuous availability or the Active/Active solution to provide continuous availability.

Figure 2 is helpful in explaining the application resiliency patterns to organizations planning their evolutions. It is important to inspect these application patterns with the key understanding that the requirements for the systems of engagement (the presentation and business logic layers) might be evaluated independently of the systems of record (the data layer). In most cases, systems of engagement have no data consistency requirements and change most often – a perfect fit for multi-active clusters. These four patterns are presented in a simple way to help assess the application’s existing capabilities and its future best availability pattern. Under the description column, note the differences in patterns using recovery time objective (RTO) – the amount of time to restore the service and RPO – the amount of possible data loss in a catastrophic situation.

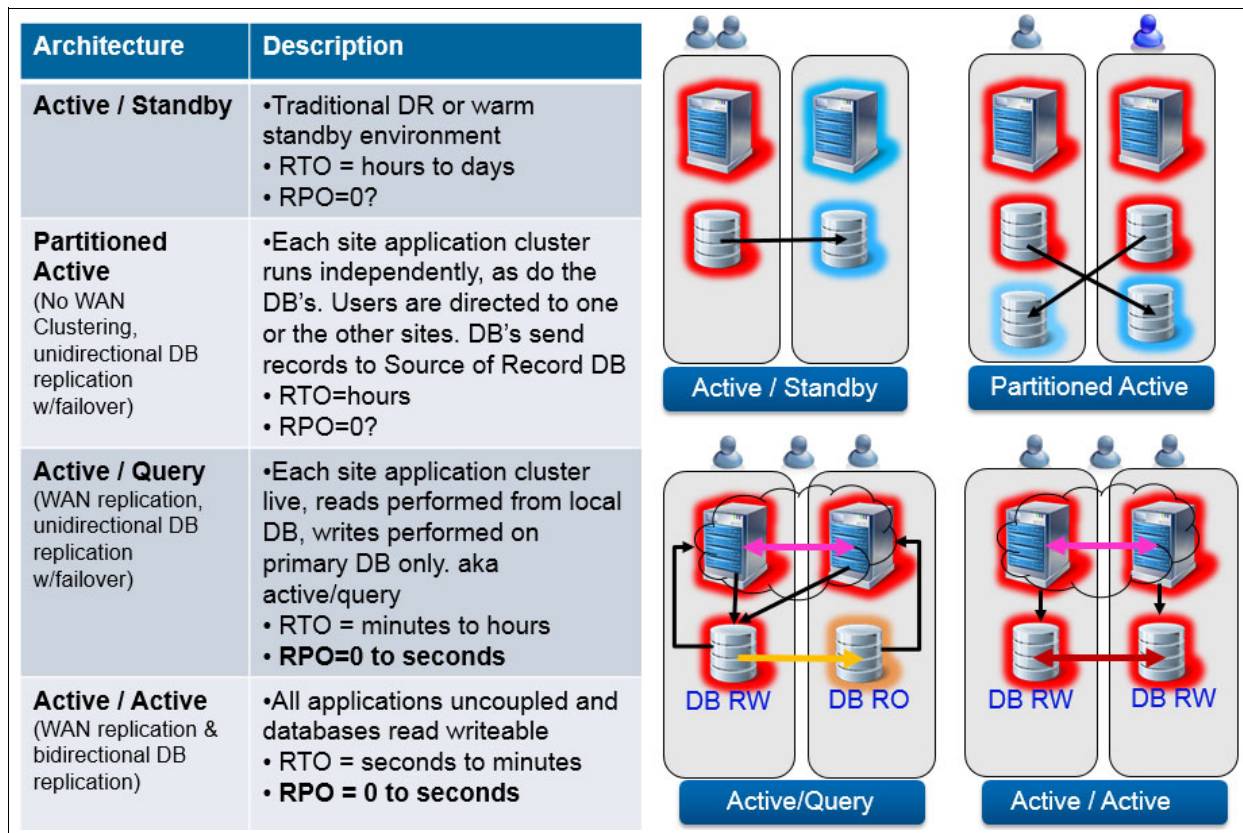


Figure 2 Application resiliency patterns

The following architectures are described in Figure 2 on page 6:

- ▶ Active/Standby is the traditional architecture since the first IT failure – old, proven, and costly for the minimal risk avoidance it provides. Often in practice, keeping the standby “cloud” identical to the active “cloud” is difficult because many organizations choose to use the standby “cloud” for development rather than having it unused – this practice significantly increases their RTO. Advanced organizations use a variant of this pattern for concurrent application releases, where the active “cloud” will have application version X in production, the standby will have version X+1 and they will switch from active to standby for production traffic during the change. This operational practice might also be referred to as “*concurrent versioning*”. This mature operational practice of evolving Active/Standby to active/warm and integrating into application change and release practices can significantly reduce planned downtime because the interruption of service is shortened from perhaps many hours to minutes when cutting over from the active “cloud” to the warm “cloud”.
- ▶ Partitioned Active is one step beyond Active/Standby in that both “clouds” can be used with users directed to one or the other “cloud” and there are no application changes required. This redirection can be based on geographic location, account number, and so on – the point is that some users do business from one “cloud”, others do it from the other and they never mix unless there is a catastrophic failure. Data is replicated to each “cloud” in case of a catastrophe where all work can be cut over to the surviving “cloud”. This partitioning of users enables the partitioning of data for uni-directional replication, avoiding data conflicts and ensuring atomic consistency. Like mature organizations using Active/Standby cutover for concurrent application releases, this same concept can be employed in the Partitioned Active pattern increasing the potential for zero-outage application deployments.
- ▶ Asymmetric Active or Active/Query means that only one read/write database (also known as *systems of record*) exists with the replicas being used for read-only workloads. Users can be served from either “cloud” with all their data reads coming from the local database. Applications must change because they need to share the session state and be aware of the routing mechanism to direct writes to the systems of record database, and reads to the local read-only replica. There are network appliances available that can intercept the SQL statements and route them, reducing the impact on application development. Another method is to expose the data layer through a web service and use Layer 7 URI routing on Server Load Balancers or similar appliances to route the reads and writes separately. Like the previous solutions, mature organizations can also apply zero-outage application releases to this environment by following the same X and X+1 concurrent versioning practice mentioned previously.
- ▶ Active/Active means that all “clouds” provide the same service, with data reads and writes at any “cloud” synchronized. This method provides transparent fault tolerance, even at the “cloud” level. The service is available in all locations except during planned and unplanned outages when only one “cloud” provides the service. This level of availability typically requires application changes to ensure that applications generate unique indexes, keys, and so on for data consistency. Therefore, this architecture is perhaps the most expensive and difficult to implement due to software development costs.

## Designing for continuous availability

IT architects are well practiced in the art of designing infrastructure platforms based on common technologies and patterns, including HA. However, in the past they have been given business requirements that only address functional requirements, for example, Windows .NET environment, Oracle WebLogic Java Platform, Enterprise Edition environment, IBM WebSphere® Java Platform, Enterprise Edition environment, Mainframe CICS/IMS/IBM DB2®, and so on. In the past, the architectural decisions made did not consider any of the methods that enable higher levels of availability because the Active/Standby and DR solutions are independent of those methods. When considering a design for continuous availability or near continuous availability, the architect must first address the non-functional requirement of continuous availability of the platform and map that back to the business functional environmental requirements. This is a disruptive thought in the practice of the IT architecture because it requires architects to think differently, and the business application to comply with the requirements mandated by the availability platform.

Although availability requirements might be considered non-functional, they are absolute requirements when designing for continuous availability. The platform solution options are tightly coupled to the application availability patterns discovered during the previous assess phase of requirements gathering. The solution will also be based on the capability of the middleware and the database platforms to replicate the state and data, ensuring service parallelism from the “clouds”.

Rather than focus on the hundreds of product and pattern designs available, we must instead establish some guiding principles to the design phase and leave it to the practitioners to integrate the platform-specific requirements. Platform-specific requirements are well documented for every platform and must be followed, assuming they are updated to support continuous availability patterns.

### Continuous availability design guideline principles

The guiding principles behind HA design are a prerequisite to the principles listed:

- ▶ Continuous availability’s core principles encompass the ability to transparently withstand component failures, the ability to introduce changes non-disruptively, and the ability to withstand a catastrophe transparently or nearly transparently. Always keep that in mind.
- ▶ Think differently: The principles listed might sound counterintuitive but they have been proven successful in many cases. The IBM <http://www.ibm.com> website that has been active continuously since June 2001 is the case study for these principles. You will find similar principles followed in all other mature organizations where continuous availability is the guiding principle in the organization (for example, Google, Facebook, eBay, Amazon, USAA, and so on).
- ▶ Keep it simple, stupid (KISS): It is easy to add complexity into any solution, even solutions whose goal is to achieve outstanding levels of availability. Complexity adds obfuscation and difficulty during problem resolution and adds more points of failure.



- Include concurrent versioning in the design: The ability to run concurrent versions of applications or platforms might be a design principle from HA design. However, in many cases, we have found organizations unable to perform concurrent versioning due to architectural design limitations.

The concurrent versioning solution can be two clusters within a “cloud” allowing staggered deployments, or the solution can even be enabled by staggering different versions per “cloud” in a multi-cloud configuration (such as Google, Facebook, and many others). Business requirements will drive the longevity of versioning. The duration that application release version N and version N-1 may (or must) coexist will be driven by business requirements and therefore will drive the architectural decisions.

- Include the capability to perform continuous operations in the design: This principle is similar to the “Include concurrent versioning in the design” principle because it enables operations to change anything at any time without affecting the business service. Versioning addresses application releases, middleware updates, OS updates, and other changes. You must also design to facilitate concurrent infrastructure maintenance, network maintenance, and component maintenance. All systems must be continuously updated for security and resiliency purposes, so the design must enable that concurrently. The simplest example is at the network layer where devices are typically deployed in pairs, upgrades are done one device at a time without interrupting the service. Middleware cluster technology is another example where clusters are made up of N+1 nodes so that any node in the cluster can be taken offline for maintenance without affecting the rest of the cluster.
- Design the solution as though only one “HA cloud” exists: This might sound like an oversimplification, but it is necessary to reinforce this fact because we have to remember that we must design for the business application environment identified during a Business Impact Assessment. All the difficulty that comes into designing a typical data center or “cloud” still exists – no single points of failure can exist (although architectural decisions might conclude that you do not need HA pairs – more on that later). You will tie the “clouds” together with technologies described later and either dedicated circuits or virtual private networks (VPNs) to allow the clouds to stay in sync. After the design is finalized for the functional requirements, we can then clone that design for the other “clouds” for the parallelized business service. Remember, we want each “cloud” to be able to completely function on its own and to also be able to replicate its state and data with its peers.
- Fail small: Everything breaks, so we need to ensure that the failure affects as few users as possible. Horizontal scaling addresses this principle well, but many organizations have workloads that need to scale vertically. Often, this is a dilemma to be solved by the architect and operations teams, but we need to address this in the design phase to enable this principle. We have seen a case study where an organization had deployed nearly 500 virtual machines (VMs) on the same physical frame – although plausible in practice, the risk of a frame failure now affects 500 VMs. Additionally, by embracing virtualization for mobility (the next principle), you will be required to design sufficient spare compute and memory so that all workloads can be moved off a frame for frame maintenance. The larger the frame, the more unused space you will pay for in reserves.
- Virtualize nearly everything: Virtualization provides flexibility and mobility, both enablers of continuous availability and near continuous availability. Virtualization flexibility enables you to scale up and scale out as needed. Mobility enables you to evacuate a frame for frame maintenance. Although the promise of concurrent hardware and software maintenance is coming to fruition, mature organizations will proactively move all services off a frame before they perform even “concurrent maintenance” to avoid the risk of it not working. Virtualization of course goes beyond the compute domain. Many networking, security, application accelerator appliances, and storage solutions support virtualization (for example, Software Defined Environments (SDEs)).

- ▶ Automate nearly everything: The leading cause of unplanned outages is the combination of people (40%) and processes (40%). Hardware and operating systems are only 20% of the problem. This 80% encompasses the application realm, as well as the people and processes that create the applications. Automation ensures that precise, repeatable tasks are performed in the correct order, avoiding human error. Automation also allows the automatic bypassing of failed components and services far faster than any human being can respond. Remember that if a business has the expectation of a 99.99% service level agreement (SLA) on a business service, then only 4 minutes of downtime are allowed a month. Have you ever seen a person respond to an incident in 4 minutes? There might be policies where you require a human being to act rather than use automation. Take, for example, bypassing an intermittent or partial failure. Although automation can automatically bypass an intermittent issue, it is possible that it might bypass everything for a badly misbehaving application and therefore bring down the business service in error. Design for automation, and grant exceptions to align with business policies.
- ▶ Design for failure, which is also known as “Know how it works, know how it breaks”: Everything breaks, so we need to ensure that our design considers this principle. The most mature organizations fully embrace this principle. Netflix, for example, has the “Chaos Monkey” that breaks random things to ensure that the service continues uninterrupted. If it interrupts service, the platform design or application must be changed to mitigate that situation the next time. At the most primitive level, this principle can be accomplished using the “cleaning person” method: walk on the data center floor, unplug something at random, and see what happens. When integrating services across “clouds”, it is extremely important to know how the technologies that allow integration fail – we must avoid using a technology that spreads the failure domain beyond individual “clouds” to achieve extreme levels of availability.
- ▶ Applications must also be designed for failure: *Everything breaks*; repeat this fact three times. Complex applications might have dependencies that are external to the continuous availability platform that are at best HA. If an external service fails, the application must handle this transparently. Rather than the entire application failing due to the weakest link, it must gracefully mitigate the failure or inform the user that the service is temporarily out of order.
- ▶ Avoid HA takeover if at all possible: It is common in a single “HA cloud” design to implement HA pairs of infrastructure services (for example, authentication, mail, network time protocol, domain name service, and so on) and HA takeover pairs at the application and data layers. In many cases, we have found HA takeover technologies to induce complexity and in fact only provide the perception of HA because it is easy to misconfigure the HA takeover and to only discover that it is not functional at the worst possible moment – when it is needed. HA takeover also has the risk of the “split brain” issue where if for any reason the pair cannot communicate, each will act to become the master and therefore the service will also fail. HA takeover is still valid for network and security components because it is significantly more mature in those domains and a fault at either of those domains will affect all services within an individual “cloud”.
- ▶ Availability is provided by the peer “clouds”: In the design, the fault domain is isolated to each individual “cloud”. We cannot allow a technology to be introduced that will cause a failure to span into more than one “cloud”. Therefore, if a business service fails in one “cloud”, bypass that service in that “cloud” and rely on the peer “clouds” to provide the service. Let us take a simple example of an authentication service. Instead of designing it to have HA takeover per “cloud”, instead rely on Server Load Balancing to configure the service to be consumed from the local cluster in all cases except when it is down for planned or unplanned maintenance. While it is down, the Server Load Balancer will then provide the service from any one of the peer “clouds” where the same exact service cluster exists. This concept minimizes complexity and costs while always providing the service.

- ▶ **Share nothing:** This principle holds true in the cloud, too. Share nothing between “clouds” because you must isolate the fault domain within a “cloud”. Any data that needs to persist must be placed in a data store that is replicated within a cluster and within the “clouds”. The output of a Business Impact Assessment will leave you with a list of business services with availability requirements within the four defined tiers. Within the Platinum and Gold tiers, you have applications and platforms that are designed for continuous availability and near continuous availability. Therefore, operations teams can perform concurrent maintenance on any system or component. These zero-outage change environments will have the ability to change things at any time. The HA environments will still require a planned outage for changes. If you break this principle, even at the storage level, you will be constrained to the least available environment’s change windows.
- ▶ **Design availability zones:** A Business Impact Analysis will define the list of business services that fit into all four of the availability tiers. Each availability tier will have its own change windows, either zero-outage changes or downtime for planned maintenance. Building on the share nothing principle, we must apply this to availability zones within a “cloud” as well so that nondisruptive maintenance can occur in the zones that support it, without affecting the HA zones and vice versa. Do not share network, security, compute, storage, or facilities between these availability zones. Depending on the business requirements, you might end up only having two availability zones per “cloud” – the continuous availability zone and the HA zone. In other situations, you might have three availability zones where you have the continuous availability zone, the near continuous availability zone (remember, this one requires short planned outages), and the HA zone.
- ▶ **Add global traffic management:** Global traffic management is a service typically provided by an appliance and is similar to the intelligent Domain Name Service (DNS). Its goal is to reply to a DNS query with an ordered list (or the best list only depending on business requirements) directing users to the best “cloud” for their business service. Typically, it is integrated with the client’s Server Load Balancing appliances so ensure that you stay with the same vendor. Because it is heavily reliant on DNS, it is imperative that the DNS is managed by the same team that manages the Global Traffic Managers and Server Load Balancers because they are essential enablers of zero-outage changes at the “cloud” layer and at the individual “cluster” layer. Mature organizations have moved these responsibilities from network support teams into the middleware support teams because they are integral to the zero-outage change process.
- ▶ **Add an in-memory application grid for stateful applications to loosely couple the “clouds” together:** Stateless applications are feasible in solutions where the applications themselves have been written with a focus on avoiding the state. This is a relatively new concept and is not always plausible because a business might require the state to be maintained. If the business requires the state to be maintained, the state must be maintained across a cluster within a “cloud”, and across the “clouds” transparently – it must persist. This requires awareness of the size of the state metadata – keep it small to minimize latency. This enables the movement of a user from one cluster to another, and even from one “cloud” to another without the interruption of their existing workflow. Additionally, grids support non-persistent data caching, which helps with scalability. Although this adds a layer of complexity to the solution, it enables transparent movement of the user when bypassing failures. IBM WebSphere eXtreme Scale allows the grid to span clusters and “clouds”. Oracle’s Coherence allows the grid to span clusters and “clouds”, too.

- ▶ Add application-level data replication to loosely couple the “clouds” together: Data keeps all businesses running and therefore it is important to ensure that data persists in all “clouds”. Storage-level replication allows data to be replicated at the block level to peer “clouds”, yet it typically does not support the ability to write data on both sides. Bidirectional data replication allows writes to occur in all “clouds” and keeps them in sync at half the speed of light. Of course, the business’ requirements will mandate what can be done based on their RPO and data consistency requirements, either ACID or eventual data consistency. ACID will require synchronous replication within metro distances (< 40 km (24.8 miles)). Eventual data consistency is what we get with asynchronous replication. The IBM InfoSphere® Replication Server based on Q-replication enables this across distances, as does Oracle’s GoldenGate replication server.
- ▶ Never stretch a cluster across clouds: Clustering technologies are outstanding in providing platform flexibility and availability, but they were never intended to span “clouds”. Many cluster technologies communicate with protocols designed for the local area network (LAN) that are network intensive and susceptible to latency if “stretched” across the wide area network (WAN) to another cloud. Additionally, if you stretch the cluster (technically feasible within a metro distance of < 40 km (24.8 miles)), you have also stretched the failure domain across “clouds”. If the stretched cluster fails, you bring down the service in all “clouds” that participate in the cluster. Instead, couple the clouds together with state grids and data replication. IBM PureScale and Oracle RAC enable this concept of stretch data clusters spanning metro distant “clouds”. However, if your goal is continuous availability, you must isolate the fault domain to each cloud and must instead build a PureScale or RAC cluster in each “cloud” and add logical data replication. Additionally, there will come a time when you must bring down the cluster for upgrades and maintenance – if you stretch it, you have to bring down the service in both “clouds”.
- ▶ Include an OoR “cloud”: This is mandatory to protect the business service from the “3-Fs”: fires, floods, and fools (malicious hackers, miscreants, enemies, under-trained operators, and so on), and is a requirement in many industries. The requirements vary. Some require an OoR data center based on distance, others are more aware of flood zones, earthquake zones, and power grids. Often, this requirement is designed as a DR solution. Although DR might be the best we can do for unchanged historical systems, we want to evolve OoR to be able to be used in a “fast failover” mode where it is kept in sync with everything in the production “clouds” and can even be integrated into changes enabling zero-outage changes in the production “clouds”. Additionally, an OoR solution that is kept in sync with a remote metro pair can be used for analytics, reporting, batch operations, and beta application releases through partitioning. There is no reason for idle expensive systems – integrate them and use them.
- ▶ Do not forget security: Critical to continuous availability is the ability to protect the platform and application from miscreants (the fools in the 3-Fs). As you architect the security posture in a single “cloud”, you must architect it in every “cloud”. The only difference between a single “cloud” and multiple “clouds” is that security incidents and logs must flow into at least two of the “clouds” so that a catastrophic failure does not take away the visibility into your security incident dashboards.
- ▶ Do not forget performance engineering: The best physical IT pattern for high levels of availability is moot if there is not sufficient compute, memory, storage, or network capacity available to the business service applications. Ensure that the development teams have performance engineering integrated into their development process. Ensure that the business makes development and operations aware of all media events that might drive spikes in demand well in advance. Often, simple application changes can be made to reduce resource consumption. Where applications cannot be changed, reserve IT capacity must be available if significant increased demand is expected.

## Continuous availability design patterns

Following the continuous availability design guiding principles and our case studies, it is apparent that a 3-site solution with at least one site OoR provides the best availability options at the lowest cost and lowest risk. Whether it is for continuous availability or near continuous availability, operationally, it enables the continuous availability guiding principles. A brief example about the capacity required and risk benefit for a 2-site solution compared to a 3-site solution follows.

Let us take a simple example to illustrate this 3-site concept where we only discuss the systems of engagement layer and ignore the systems of record. Let us say a fictional widget service requires a minimum of four web servers and four application servers to handle the total business transactional load (that is,  $(4+4) = 8$ ). If I want to run the widget service in two locations for availability purposes, I need four of each in both locations (that is,  $(4+4) + (4+4) = 16$ ). If we remove one location from service (either planned or unplanned removal), we still have the capacity that we need (that is,  $(4+4) + (0) = 8$ ). However, if during a planned change where we have one location disabled, the other location fails, we now have no capacity available (that is,  $(0) + (0) = 0$ ). If I instead consider three locations, I now only need two web servers and two application servers in each location (that is,  $(2+2) + (2+2) + (2+2) = 12$ ). If any of the three locations become unavailable either due to a planned or unplanned outage, I still have four web servers and four application servers (that is,  $(2+2) + (2+2) + (0) = 8$ ) available. In this scenario, our worst case scenario again becomes when one location is down for planned or unplanned maintenance and we lose another location, but in the 3-site pattern, we still have 50% capacity available (that is,  $(0)+(0)+(2+2) = 4$ ). Although not ideal, we can survive by reducing the unneeded services until we can recover one of the other two locations.

The same reasoning why three sites are more flexible at a better cost than two sites also applies to a four or more “cloud” solution if your design optimizes the capacity plan so that each cloud has adequate capacity if a cloud fails. (The capacity requirement for each cloud becomes less as more clouds are added.) When using public clouds where the cost of a data center does not come into the equation, more is better, you can fail small without impact.

This section focuses on the four high-level 3-site patterns of multi-site design for CA and nCA:

- ▶ Active Standby within metro with OoR DR
- ▶ Active Active within metro with OoR DR
- ▶ Active Active (2-Active) within metro with OoR query and fast failover
- ▶ Three Active (3-Active) geographically distributed (can be N-Active as well)

### High-level perspective of the four near and continuous availability patterns

Figure 3 on page 14 shows a high-level perspective of the four near and continuous availability patterns.

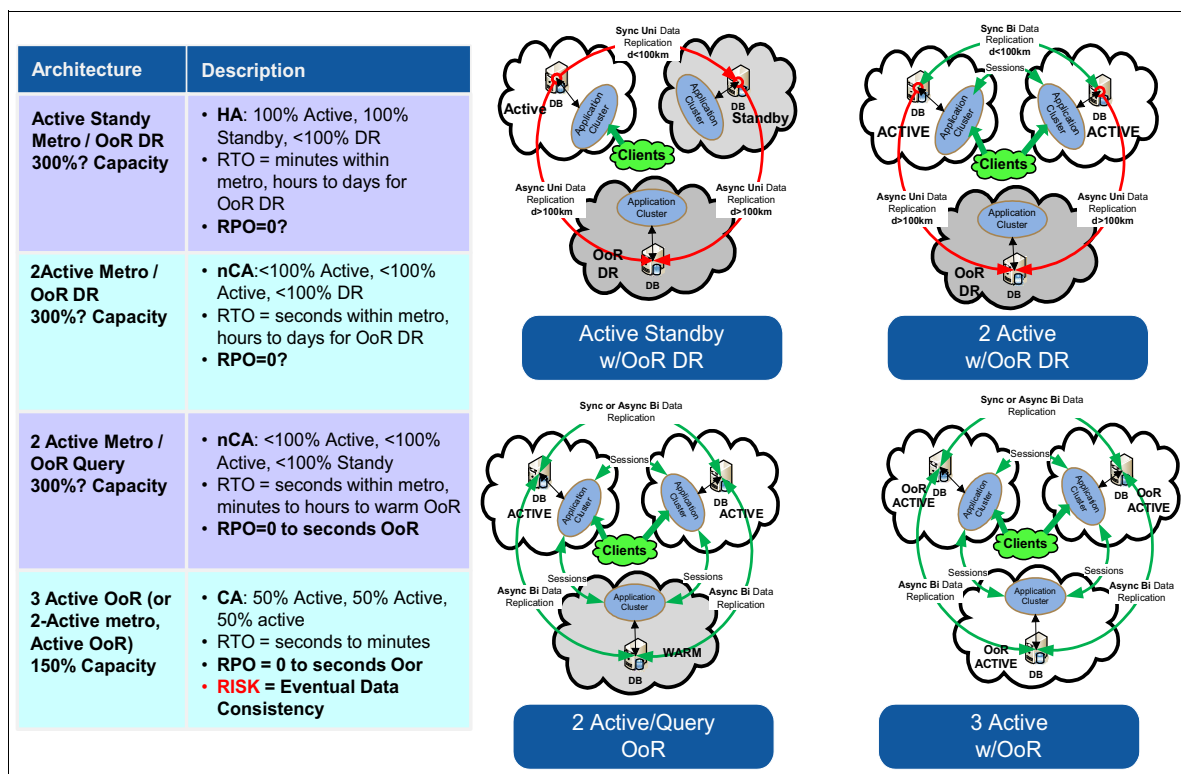


Figure 3 High-level perspective of the four near and continuous availability patterns

In Figure 3, the business requirements for RPO, RTO, and data consistency (whether ACID or eventual) are key to the architectural decisions made here. These business requirements will drive which of the four patterns will be the future state and chart the journey toward continuous availability. Although these patterns are driven by the data requirements, variations and efficiencies can be made easily at the presentation and business logic layers (for example, the systems of engagement can be 3-Active and the data layer can be 2-Active within a metropolitan area ensuring ACID requirements).

### Active/Standby metro pair with OoR DR

This is the standard and traditional Active/Standby with OoR DR model practiced for many decades. It provides only HA. It is used here to indicate a possible starting point of an organization's journey toward continuous availability (Figure 4 on page 15).

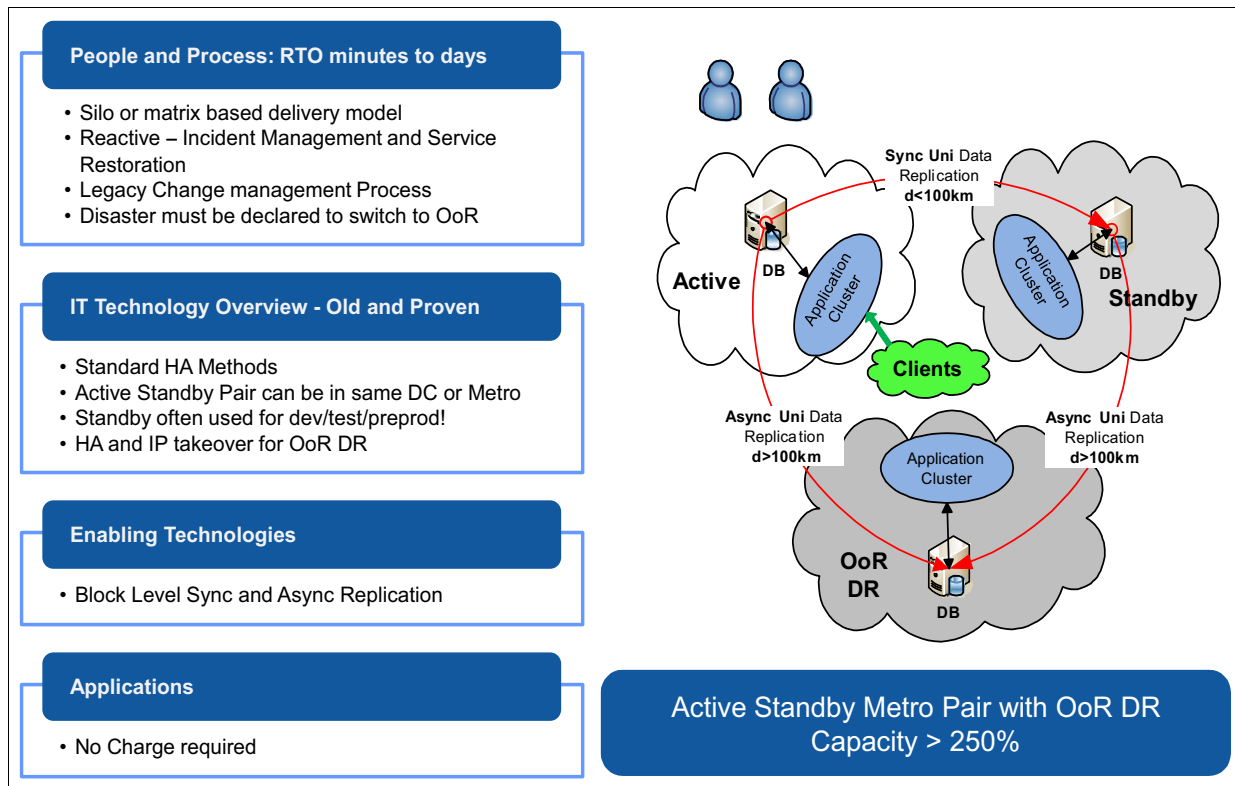


Figure 4 Active Standby metro pair with OoR DR

Often, the Active/Standby pair is within the same data center and therefore provides no protection from a data center catastrophe (FFF: Fires, Floods, or Fools). Variants include the standby data center being in a different data center within a metro distance or within the same data center. Organizations often use the standby data center for dev/test/pre-production, increasing the recovery time if the primary data center or any component failure brings down the active service. Much human effort is required to support this model. Incident management involves critical situations and service restoration, often not finding the true root cause due to the pressure of recovering the service.

Mature organizations have achieved high levels of availability by using this pattern through stringent operational practices. If the Active and Standby pair is kept in sync operationally (which means that a service request performed in the Active “cloud” is immediately also performed in the Standby “cloud”), this architecture can enable concurrent versioning. Version X can be in production in the Active “cloud” while version X+1 can be deployed and tested in the Standby “cloud”. During an extremely short planned outage change window, the Standby can be made Active, and the Active can be made Standby. The duration of the planned outage depends on many factors, including the duration for the uni-directional replication to be synchronized with the Standby “cloud”, the time that it takes to change the direction of the replication from Standby to Active, and the time that is required to redirect users to the X+1 instance. This concept can also be applied to catastrophic failures of the Active “cloud”. For a mature organization, this can reduce the RTO to minutes within the metro pair yet the OoR DR cutover will still take hours at best to days.

## Active active metro pair with OoR DR

This is the common pattern (Figure 5) that is typically seen in the mature financial sector where continuous availability is required during business hours and the RPO=0 or data consistency requirements are ACID. Planned changes can be performed in off hours, because mature organizations can shorten the planned outage duration using staggered deployments and upgrades. The active pair is within synchronous distance (typically < 40 km (24.8 miles)) allowing writes to occur on both sides of the Active/Active pair. And, the DR site is OoR meeting DR requirements. This pattern enables the core principles, allowing faults to be isolated to one of the “clouds”. Concurrent versioning as application releases can be staggered to each active “cloud” and therefore non-destructive changes. Typically, each cloud is designed in an HA manner with the ability to handle 100% of the transactional workload each, for a total of 200% transactional capacity. If concurrent versioning is required within each cloud, more than 100% capacity will be required in each “active” cloud if a dual-cluster approach is taken within each cloud.

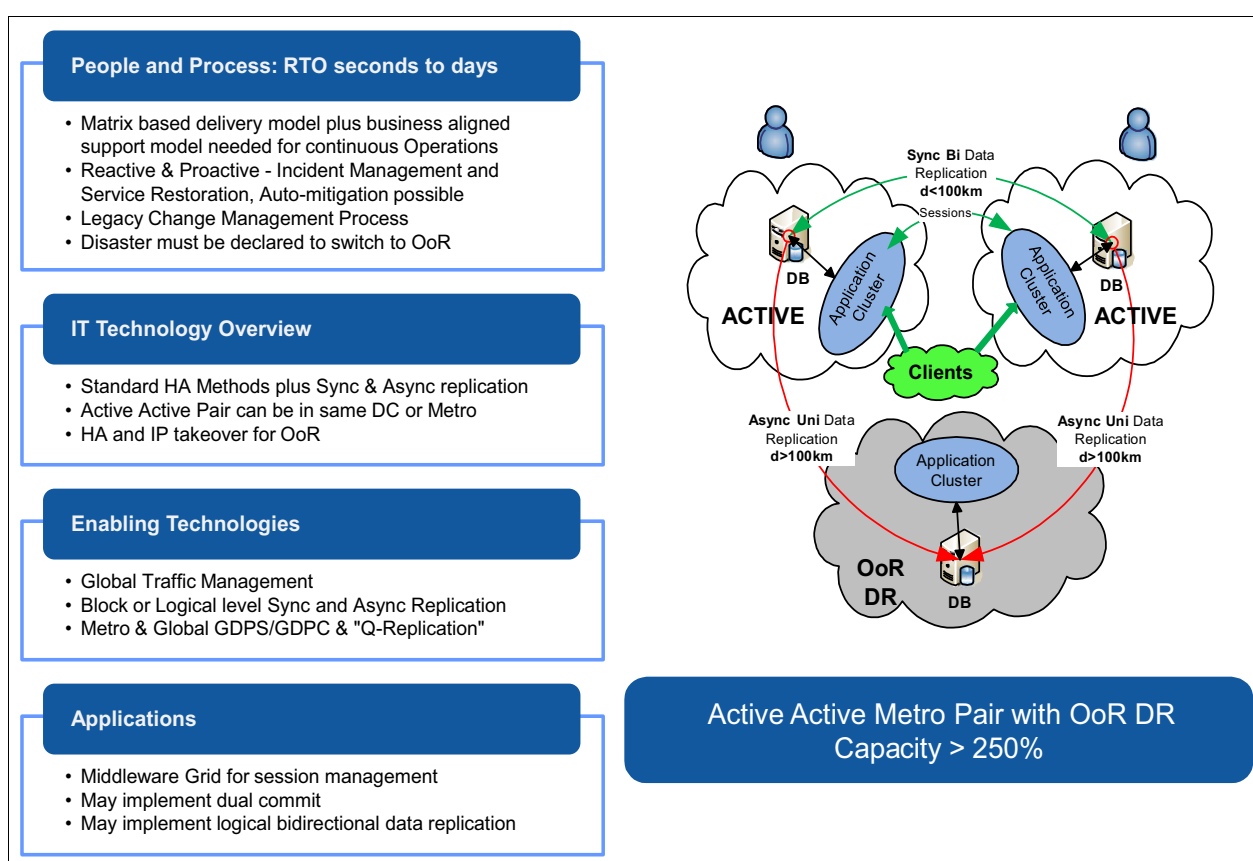


Figure 5 Active active metro pair with OoR DR

The state and data replication are enabled synchronously within the active pair abiding by the ACID requirement. The OoR is connected with uni-directional block or application-level replication providing eventual data consistency. Because this is a DR solution, DR planning and testing must still be conducted.



## Active Active metro pair with OoR query and fast failover

This pattern (Figure 6) is a more mature version of the previous Active/Active with OoR DR pattern. This is most likely as far as we can take an organization whose data policies require RP0=0 and ACID consistency requirements and DR requirements. The novel concept here is that rather than using the OoR “cloud” in a DR scenario, it is instead integrated into day-to-day operations and can be used for analytics, reporting, batch processing, read only queries, and in fact might be used as an Active component when maintenance is required on a component affecting both the Active pairs within the Metro. Additionally, the OoR data center might run the systems of engagement actively along with the metro pair’s systems of engagement, allowing reduced capacity and cost because each system of engagement “cluster” will only require 50% of the compute, memory, and network capacity if two clouds will always be available.

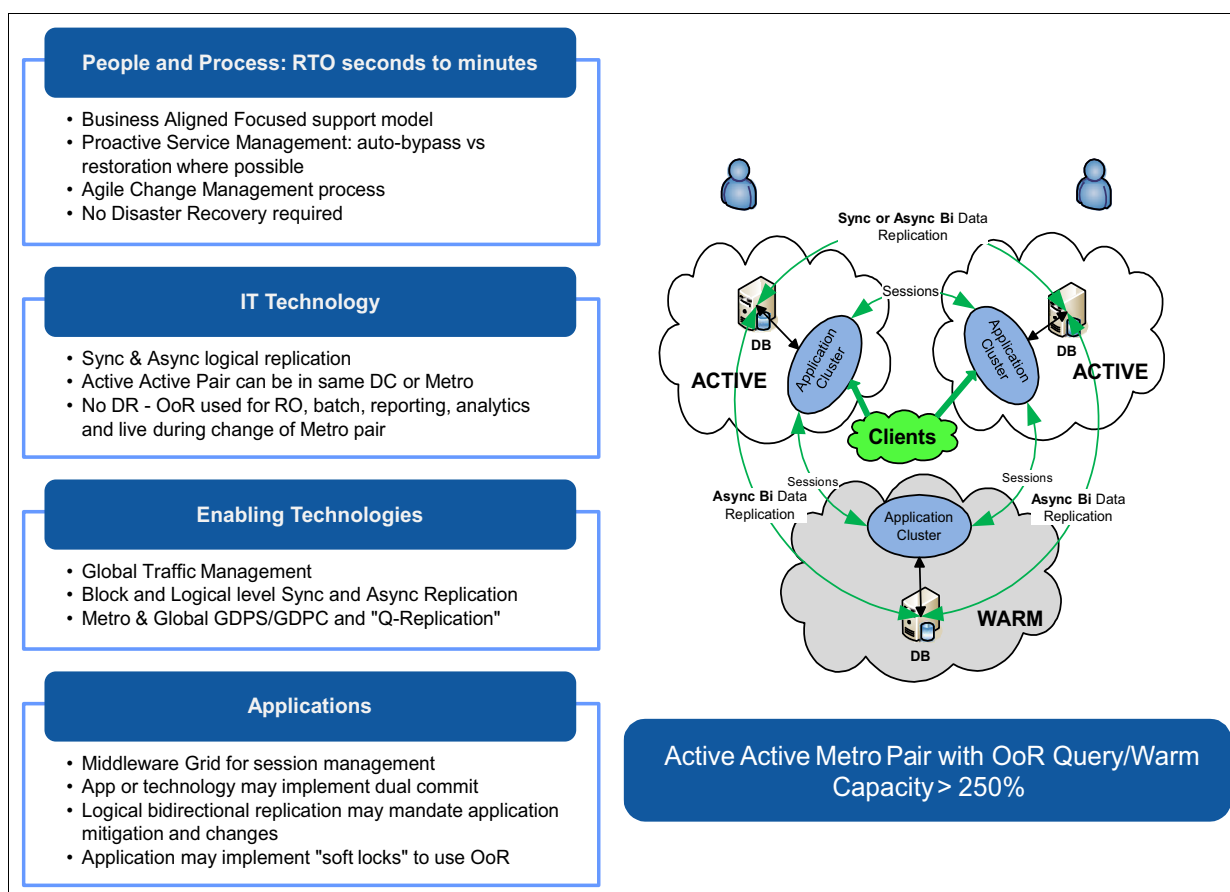


Figure 6 Active active metro pair with OoR query and fast failover

Like the previous Active/Active with OoR DR pattern, the state and data replication are enabled synchronously within the Active pair, complying to the ACID requirement. The OoR is connected with uni-directional or bidirectional application-level replication providing eventual data consistency to the OoR “cloud”. The advantage of this pattern is that it meets OoR DR requirements and also allows the warm “cloud” to be used actively for the systems of engagement and passively for the systems of record. Mature organizations will manage change across all three clouds, ensuring application consistency at all times and enabling fast failover to the OoR when needed.

## Three-Active geographically distributed

This is the 3-Active model (Figure 7) that has been used to keep IBM.com always on since June 2001 and fully uses business service parallelism, which is also referred to as “*N+2*” *resiliency*. The key business decision enabling this pattern is that of eventual data consistency. Data can be written to any of the three “clouds”; it is captured at the source; and it is applied to its two peers with a replication delay based on the distance between the data centers. For IBM.com, the greatest distance between the clouds is approximately 2,800 km (1,739.8 miles) and causes a 4 - 7 second replication delay based on the size and volume of the data objects. This consistency risk is accepted by the businesses whose services enjoy this level of continuous availability.

The biggest advantage of this pattern is that it enables continuous availability at less cost than Active-Active patterns. In a 3-Active pattern, each “cloud” only needs 50% of the overall transactional capacity (compute, memory, and network) because there will always be two of the “clouds” available during all planned changes, providing 100% total transactional capacity. Storage capacity remains at 100% per “cloud”. Of course, a business might decide that each “cloud” must have 100% transactional capacity or even more for concurrent versioning within each “cloud”.

Changes are deployed one “cloud” at a time, allowing the service to be available at all times. For IBM.com, we have actually moved the location of our “clouds” eleven times in the past fourteen years without business service interruption. We have also transparently survived the “3-Fs”: fires (Shamburg, IL), floods (Bethesda, MD), and fools (anonymous attack, backhoe fiber cut, our own operational errors, and so on).

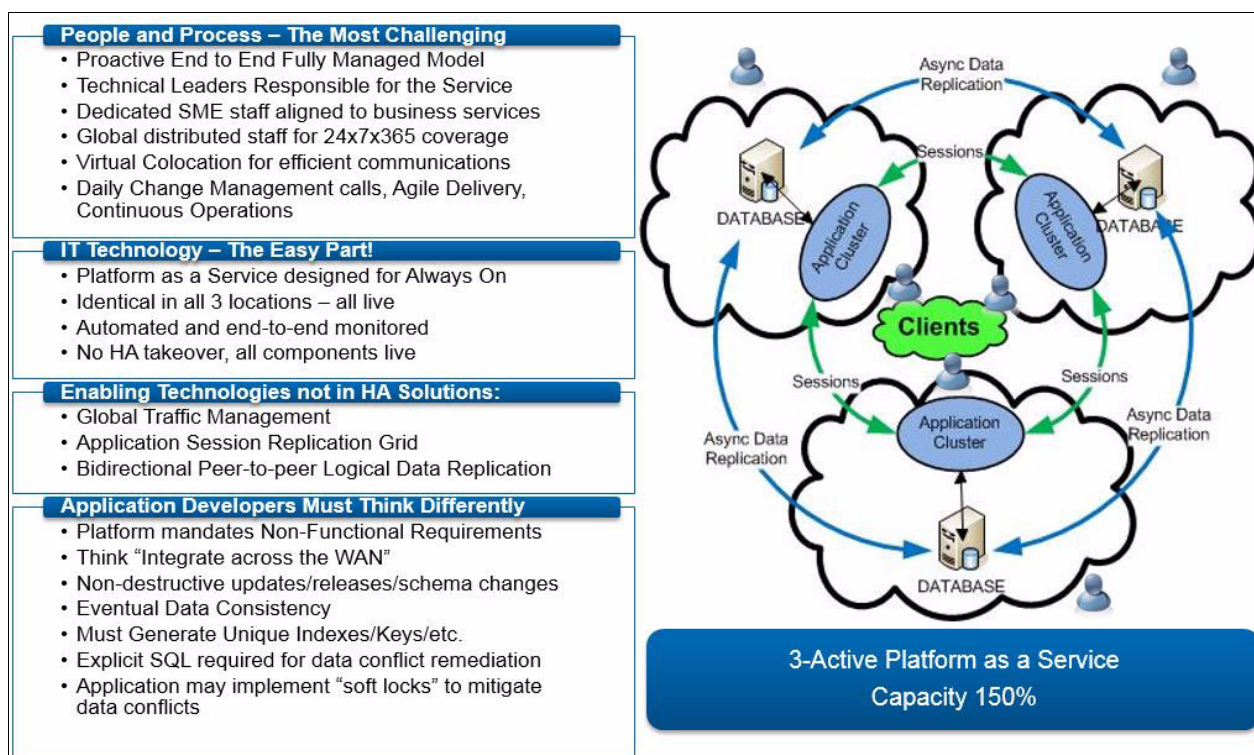


Figure 7 Three-Active geographically distributed

Enabling technologies are the same as the Active-Active with an OoR warm site pattern: global traffic management, clustered applications synchronized across the WAN using session grid technologies, and logical data bidirectional peer-to-peer (also known as *multi-master*) replication. All three clouds are active at all times, enabling service parallelism, concurrent changes, and disaster transparency.

The technology is fairly straightforward, and the platform is designed from the ground up to enable continuous business services. No HA Active/Passive server pairs exist; all clouds are identical in all components; and nearly everything is automated to ensure consistency and automated service bypass in a failure.

The risks in enabling the OoR cloud to be integrated into the read/write data layer are probable data conflicts due to eventual data consistency due to the distance between the “clouds”. The logical data replication, for example, the IBM InfoSphere Q-Replication or Oracle GoldenGate, can apply basic business rules to conflict remediation, although conflict potential increases with the latency between sites and the volume of distributed writes. What the system itself cannot remediate, it will store and then advise the operators of the conflicts that must be manually remediated. This manual remediation is driven by the business consistency requirements.

Applications must adapt and change to this pattern; they must be uncoupled from the data layer as they must also be with the 2-Active pattern. Because that replication latency across distances is a physics problem that cannot be overcome, the mitigation of data conflicts must be handled at the application layer. Similar to synchronous methods of locking databases, applications might establish a “soft lock” mechanism to mitigate latency and conflicts if this is required by the business. Therefore, the application checks for a soft lock before a write. If no soft lock exists, the application sets a soft lock with a time to live on it before any data write and then waits the worst case replication delay before writing and releasing the soft lock.

This pattern will mandate specific application development guidelines to enable the sharing of the state and to use bidirectional application-level data replication. The technology that enables these methods will identify those development requirements and must be shared with all application development teams that write applications for this platform.

Often, the cost and risk advantage of this pattern is not obvious because the first instinct is to perceive it as extremely expensive. Significant cost savings can be gained because the infrastructure requirements for the systems of engagement, or the presentation and application business logic layers, requires only 150% (50% in each of the three clouds) of compute/memory/network capacity in the 3-Active pattern, versus a 2-Active pattern that requires 200% (100% in each of the two). Figure 8 on page 19 attempts to clarify the benefits.

	Active / Standby	2-Active	3-Active			
Production cpu/ mem/ network capacity	>200%	>200%	>150%	Pre-Prod 25%-50%	250%	
Production Capacity with out of Region DC	>200%	300%-600%	>150%		225%	
Platform Availability	99.5%	99.999%	99.99999%	Standby 100%	200%	Pre-Prod 50%
Availability during Planned Changes	99.5%	99.5%	99.999%		175%	
Failure Impact	100%	50%	33%		150%	Active Site 3 50%
Disaster Recovery Time	Hours to Days	0 to seconds in region, hours to days OoR	0 in region to seconds OoR		125%	
Incident Response	Manual Failover	Automatic Bypass in region else manual	Automatic Bypass	Active 100%	100%	Active Site 2 50%
Maintenance Windows	YES	Sometimes	No		75%	
					50%	Active Site 1 50%
					25%	

Figure 8 Pattern benefits



The worst case capacity scenario in this 3-Active model is when you have one “cloud” down for planned maintenance and you have an unplanned incident affecting the business service in another “cloud”. If this happens, you only have 50% overall transactional capacity during the time that it takes to restore service from either the planned outage or the unplanned outage. This condition can be mitigated through contingency planning: business pre-approved actions that can be taken by the operations team to disable non-critical services during this period until service is restored in one of the other “clouds”. Business pre-approved is a critical point here; there is no time to escalate to the business for approval when an incident happens. The operations team must be empowered to take mitigation actions immediately.

Let us look at a case study of this pattern (Figure 9 on page 20). It applies to IBM.com’s electronic support services where a hybrid approach is used. The presentation and business logic layers are deployed in a 3-Active pattern, and the data layer is deployed in a 2-Active pattern with OoR warm fast failover to comply with ACID data requirements. The key advantage here is that the presentation (web) and business logic (app) tiers, which users experience and changes the most often, support all the core CA principles using the 3-Active pattern. Changes can be performed at these tiers at any time without affecting the business service.

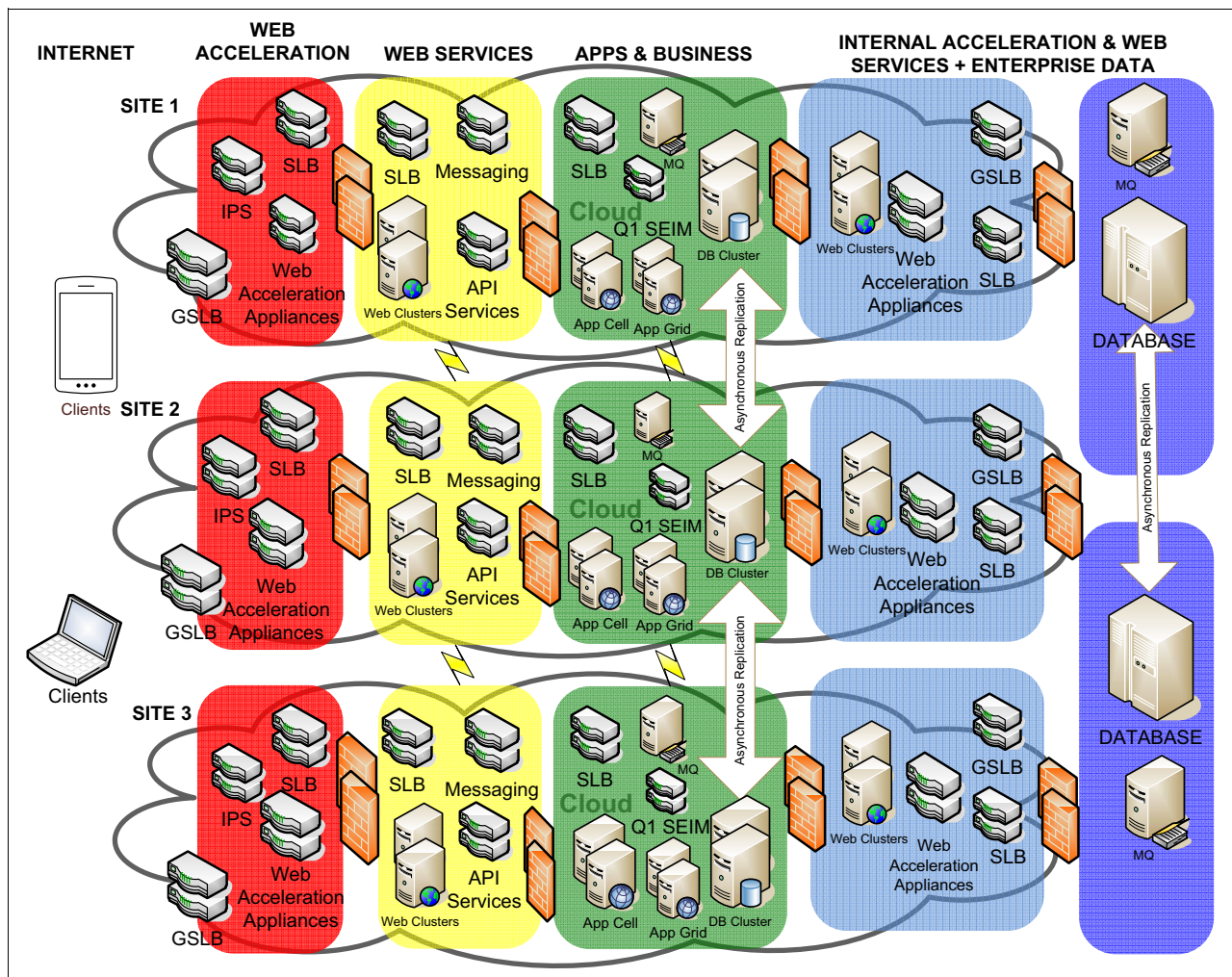


Figure 9 Case study

Each cloud is identical in all components in the red, yellow, green, and light-blue security zones, the 3-Active systems. Databases that exist in the green zone comply with the eventual data consistency requirement. The databases in the dark blue security zone (enterprise data systems of record) comply with the ACID requirement because they are deployed in a 2-Active pattern in one location and the OoR data is kept in sync with asynchronous replication. During normal operations, all 3-Active clouds use data only from the active data pair in the blue zone in the upper-right corner of the diagram. When maintenance is required to the 2-Active data pair in blue, a page is deployed to the 3-Active presentation layer. Users are advised that the service will be temporarily disabled (up to 5 minutes) while the service is halted while replication from the 2-Active data pair is synchronized with the OoR warm data source. After this data is in sync, connections are then made from the 3-Active presentation and business logic tiers to the now “hot” OoR data source. The big advantage is that application changes can be made concurrently without interruption of the business service, which changes the most. The maintenance window for the data tier in blue is reduced from the typical 6 - 8 hour window down to a 5-minute planned outage window.

Figure 10 on page 21 is another example that might clarify this concept with a 3-Active distributed layer and a 2-Active and OoR warm data layer with ACID requirements.

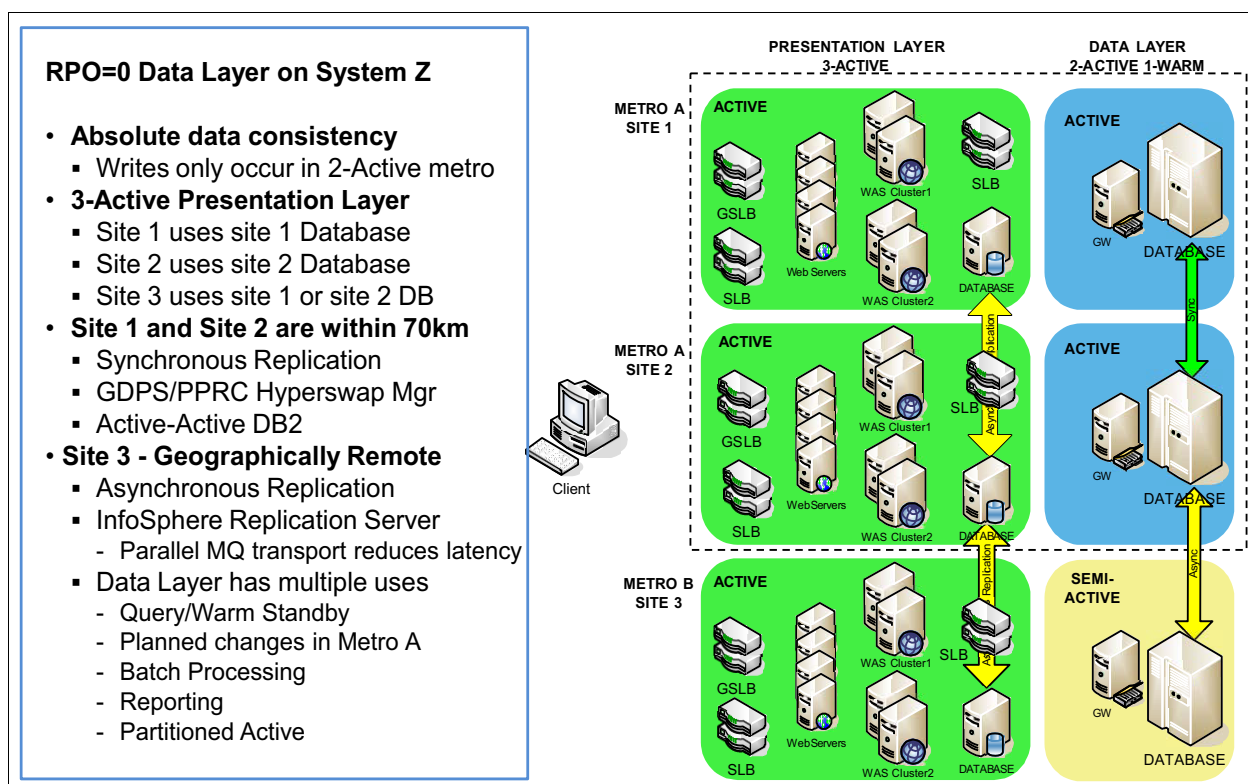


Figure 10 Three-Active distributed layer and a 2-Active and OoR warm data layer

Another variant of this pattern (Figure 11 on page 22) deploys two of the active “clouds” in one data center (where a client only has two data centers) and the third “cloud” is OoR. If the “Share nothing” principle is strictly followed within the data center with two “clouds” and the data center is designed as a tier 3+ or tier 4 data center, this pattern can significantly increase the resiliency of the business service and still provide the flexibility and availability inherent to the 3-Active pattern.

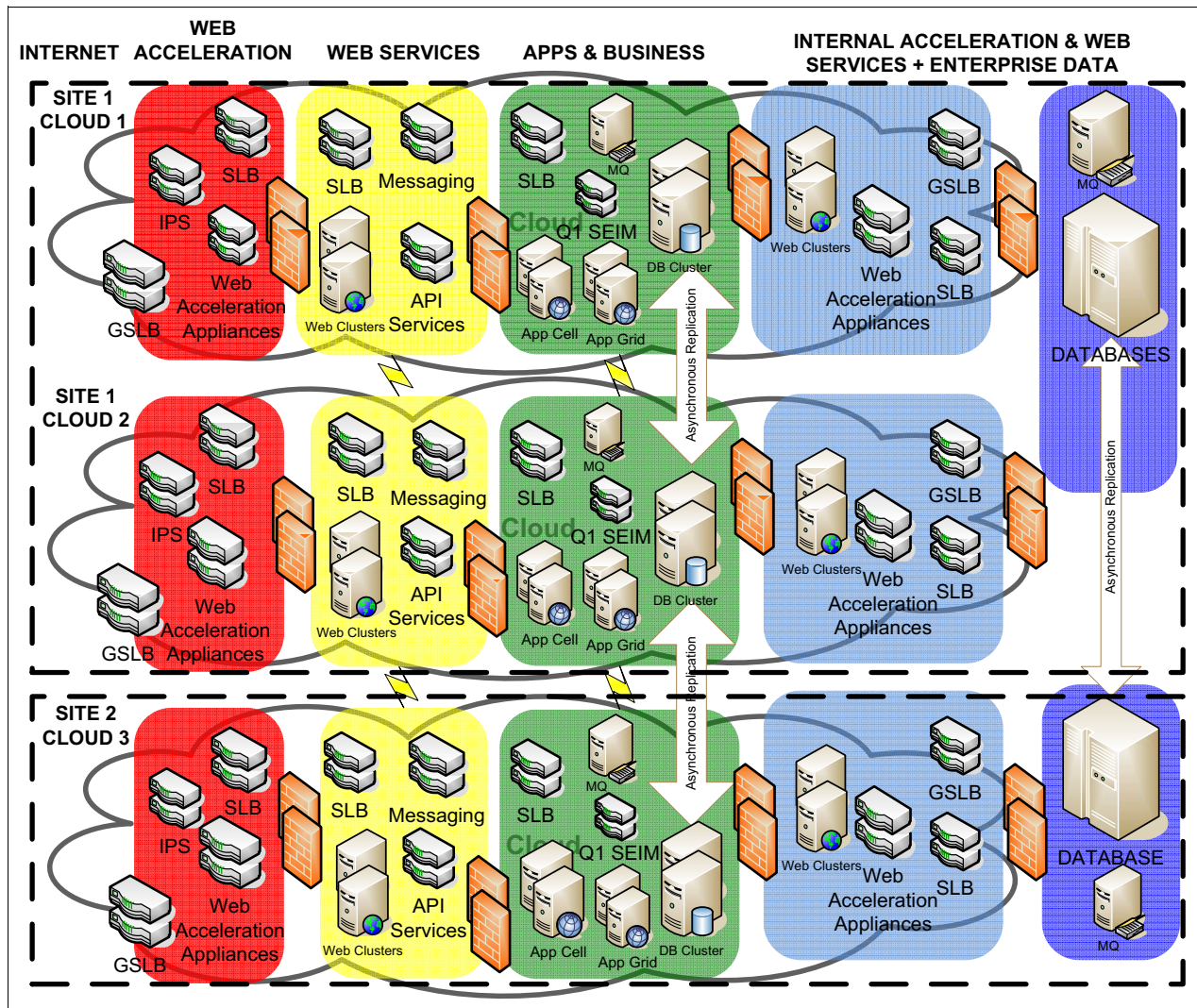


Figure 11 Case study pattern variant

## Continuous availability design summary

IT solution design can be complex, especially when designing in higher levels of resiliency and considering which application resiliency pattern fits which continuous availability design pattern. This document describes the high-level patterns to assist the practitioner in identifying the best possible solution for the business requirements. Not every application can take advantage of the continuous availability patterns due to application restrictions, but it is imperative that the best pattern is applied to the most mission-critical business services. Of course, each technology solution has its own requirements, which also must be considered, but those requirements are far beyond the scope of this document. Consider the 3-Active pattern the best fit (and best cost) for all presentation and business logic layers (also known as *systems of engagement*) and design with that in mind. Then, consider which continuous availability pattern best fits the business service data requirements (*systems of record*) and design specifically for the data layer with the continuous availability guiding principles as the reference.



## Implementing continuous availability

Implementation practices are well documented and specific for all technologies, so we only describe the variations that enable success when designing for continuous availability. These statements might be counter to the way that IT is implemented today and is disruptive, but they are based on the experience that we have seen in mature organizations with extreme levels of availability.

### Lifecycle operations are involved in design, transition, and readiness testing

Enabling continuous operations is one of the key factors to enabling higher levels of resiliency. For continuous operations to be successful, the operations teams need to completely understand how things work and how things break.

**Important:** For this knowledge to be correctly attained, the team that will be responsible for lifecycle operations needs to perform this transition. We suggest that the operations teams are directly involved in the design and implementation phases as well. This approach also provides immediate feedback and remediation if the design does not fulfill all the continuous availability design guiding principles.

This approach also ensures that the detailed knowledge and documentation are created for the solution by and for the people responsible for lifecycle maintenance and incident response.

**Important:** During implementation, it is critical to break all things during readiness testing to understand and document incident response documents and automation gaps.

Typically, the operations team members involved in the design phase are the team leaders with extensive breadth of knowledge. They are responsible for documenting the end-to-end business service solution, and enabling their greater team with that knowledge through distributed learning sessions. During the implementation phase, the lifecycle teams implement the solution instead of a temporary transition team so that the knowledge stays within the lifecycle team. The lifecycle operations teams also perform the readiness tests and fault scenarios to build the knowledge and expertise to quickly bypass incidents and keep the business service running.

### Build one “cloud” at a time and then interconnect them

Continuous availability requires multisite solutions that are identical in each site for business service parallelism. Each “cloud” must be built out as a stand-alone unit with the standard methods and readiness tests that are currently suggested as the preferred practice. After two “clouds” are built and independently tested, it is important to tie them together across the network so that the state and data replication can occur. The state and data replication are typically specific to the business service workload requirements, and vary by technology. Some might require state replication; some might not. Some might require unidirectional data replication; some might require bidirectional replication. The business service application must be deployed on each cloud independently, and then the state and data replication policies must be enabled based on the business requirements.

**Note:** Test the performance of the business service application before you interconnect the clouds and after you interconnect the clouds. Replication methods often add compute and memory overhead, so compute and memory overhead must be considered during the design and verified during the implementation.

## Organizational alignment for managing continuous availability

Managing continuous availability with continuous operations is perhaps the hardest and longest part of the journey toward continuous availability because it requires a disruptive cultural and operational change to business service-aligned continuous operations. The Business, Development, and Operations groups must all be aligned and communicate bidirectionally to proactively focus on the business service and embrace continuous availability as the mission. The non-functional requirements for availability and performance must be considered as stringently as the functional requirements. In fact, you can argue that availability needs to be reclassified as a functional requirement in the “always on” era of pervasive computing that is core to the current user population.

## IT operations changes and the Biz/DevOps model

Just as the journey toward cloud enablement with the adoption of application patterns will change the operational model, so does the adoption of Continuous Operations because Continuous Operations in itself is a pattern. Organizations must evolve from reactive to proactive. They will become less compartmentalized and more integrated. In the past, large and complex heterogeneous IT environments have encouraged specialization in IT operations organizations – commonly referred to as “matrix” or “silo” organizations where specialists have one skill and handle service requests and incidents within their specialty without awareness of the impact of their actions on the business service. Continuous availability and cloud-aligned operations teams manage pre-integrated and optimized patterns with complete awareness of the impact of their actions on the business service.

With pre-integrated and optimized patterns, fewer things can go wrong and therefore you need smaller specialized teams working more closely on areas, such as performance management and incident management. You will need more generalists and fewer specialists – there will always be the need for both, but the ratio will change in the favor of well-skilled generalists. Generalists will focus on the business service with the end-to-end understanding of the patterns. Specialists will focus on the architecture and lifecycle management of the patterns, and manage the framework that enables patterns to be consistently deployed and managed.



## The matrix delivery model versus business service-aligned delivery model comparison

To illustrate these disparate operational models, we use two of the IBM delivery models to show the differences: the matrix delivery model, and the IBM CA Services delivery model. The IBM matrix delivery model is based on years of experience managing complex heterogeneous infrastructures where each business service application mandated the IT architecture required to run the business service. Given this complexity, often different for each style and vendor solution of each business service, it was necessary to align skills based on underlying technologies. The IT staff was unable to develop end-to-end understanding because there were no common patterns. This alignment on the technologies that were used also narrowed the number of skills needed and enabled skilled practitioners to become subject matter experts (SMEs). Skills are grouped according to the various hardware platforms, operating systems, hypervisors, and the many middleware components.

This matrix is needed so that skills can be shared and grow among the differing technologies, yet it leaves a gap because no generalist perspective is aligned to the business service. The gap is temporarily filled during design and transition with integration architects whose skill ties the technologies together to fit the business application service requirements. These integration architects are often reassigned after successful integration and readiness testing of the solution, and the integrated end-to-end knowledge that will be helpful during incidents leaves with them. Granted, they typically leave the lifecycle team with a well-documented “initial state”. However, the business forces change to the initial state, so the documentation often loses credibility over the years through neglect (a process problem – remember that 40% of unplanned outages are attributed to process). Performance is often reactively managed by another group that analyzes performance reports at certain intervals and suggests capacity increases to the operations teams. Often, that advice is too late.

When service outage incidents occur in a business service that is served by the matrix model, another role comes into play as the incident manager needs to tie all the skills together. This incident manager runs the critical situation calls, orchestrating and triaging the issue, and trying to get the various skills involved in delivering the business service to work together for service restoration. Everybody in operations has experienced these situations and will agree that they do not provide the ability to immediately restore service given the pressure and chaos inherent in this process. This matrix triage process does not enable rapid service recovery that facilitates extreme levels of availability. Remember that a 99.99% SLA (near continuous) allows only 4 minutes per month of downtime. A 99.999% SLA (the perception of continuous) allows only 27 seconds per month.

Figure 12 on page 26 shows a matrix operations model. The skills groups identified on the perimeter of the large center circle are responsible for lifecycle support. The human beings in each skill group are designed to be interchangeable; if one person is ill or on vacation, another takes their role and handles their incidents, problems, and changes. The skills in the center of the circle, the various architects, are typically only present during the design and implementation phases and are focused within their own domain.

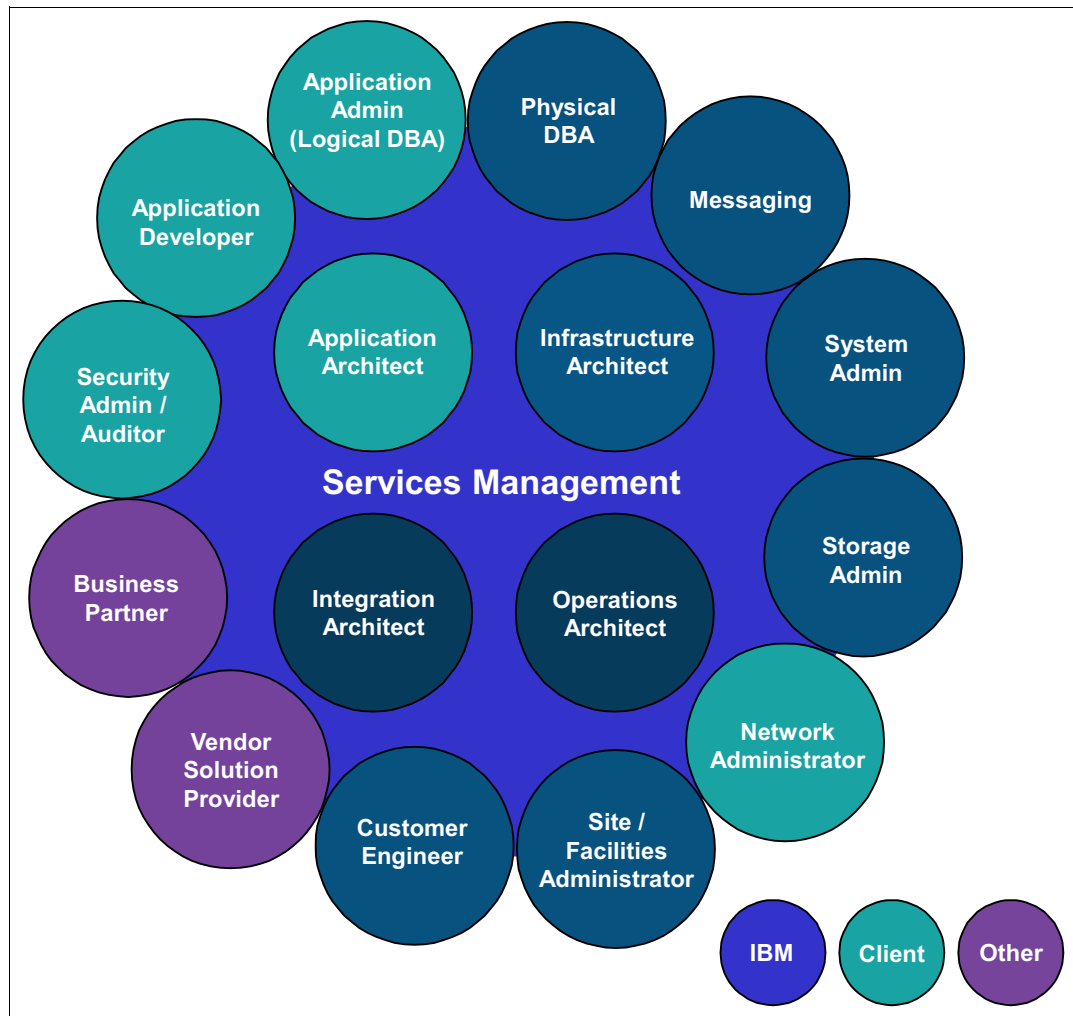


Figure 12 Matrix operations model

Let us consider an operations model that is aligned to the business service, the IBM GTS Continuous Availability Services organization. In 1998, this organization was chartered from their initial founding with the sole guiding principle of continuous availability. The mission was clear and simple, failure was not an option after some visible failures occurred when trying to rely on HA patterns and technologies and the reactive matrix delivery model. This new team was given the continuous availability mission and left to design, implement, and manage the Continuous Availability solution and the operational model. They were advised to question everything and empowered to change pattern solutions and operational processes to make them more agile (within the constraints of audit and governance). Initially, they were all collocated in one physical location while establishing the operational model. Within a couple years, the staff moved back to their original locations and are now distributed around the US, Europe, and Asia, providing business continuity and continuous operations. They are now “virtually collocated” by using persistent chat rooms aligned to skills, business services, teams, and transition projects.

They were formed with only one business service pattern: Continuous Availability for IBM World Wide Sponsorship Marketing Events (CMO driven: The Olympics, Wimbledon, The Masters, and so on). That one continuous availability pattern included the infrastructure layer, and the presentation and business logic layers. Two teams of generalists evolved: one team aligned with all the infrastructure components, and one team aligned with the presentation and business logic components. Because the mission was also aligned and funded to achieve the marketing goals, a strong relationship was established between the business, the application development team, and the operations team because they all shared ownership in the success. The development team and operations team were informed of business plans well in advance and updated on a regular basis regardless of whether plans changed. The development team worked in tandem with the operations team because they had to design their business service application to fit the continuous availability platform. They formed an intra-team relationship similar to the DevOps model that evolved from the entrepreneurial post .COM companies (for example, Google, Netflix, eBay, and so on). They monitored everything and wrote custom code that kept them up-to-date with minute-by-minute statistics and trends that showed the health of the overall service and all the underlying components.

As technology evolved, it enabled more patterns to be deployed onto the continuous availability platform, and in mid-1999, www.ibm.com boarded onto the 3-Active platform. This growth in business service application patterns meant that the team responsible for the presentation and business logic layers grew in capabilities with the growth in patterns. It soon became apparent that we needed to establish another team focused on the end-to-end business service, so the SMEs can focus on the patterns. With the growth in patterns, more middleware SMEs joined the team, and they have evolved to the operational organization that is shown in Figure 13.

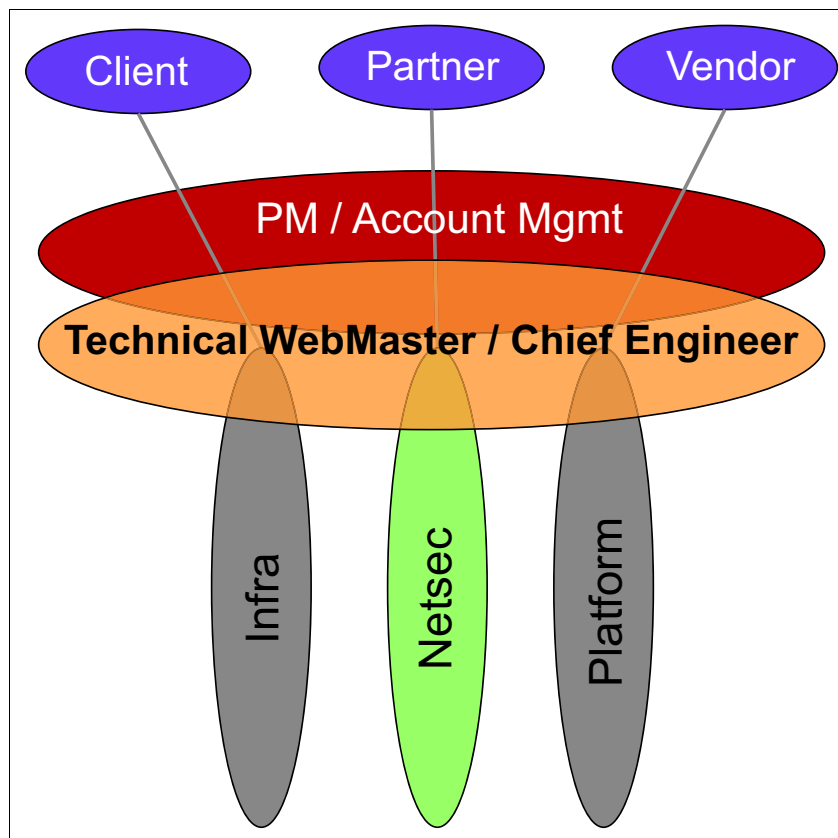


Figure 13 Operational organization

The vertical ovals in Figure 13 on page 27 are the generalist teams of SMEs. The “Infra” team is responsible for Infrastructure as a Service, “NetSec” is the network and security team, and “Platform” is the team responsible for the Platform as a Service (also known as *middleware*). Each team consists of specialists in multiple skill groups similar to the matrix model skills, and they are also the architects of the solutions and patterns within their knowledge domains. The horizontal ovals in the center represent the project/account management team responsible for the business relationship with our clients, vendors, and partners. The Technical Webmaster is the technical generalist that takes the business service requirements and applies them to the associated continuous availability pattern. The Technical Webmaster is responsible for the end-to-end service and is measured on two areas: continuous availability and client satisfaction. They are held accountable for delivery of the end-to-end service. The Technical Webmasters are involved with the business client and development team from concept all the way through lifecycle. They perform many of the roles mentioned in the matrix model: Performance Analyst, Availability Manager, Integration Architect, Incident Manager, and orchestrator of the day-to-day changes applied by the technical teams, ensuring that no changes negatively affect the business. The organization thinks that the Technical Webmaster role is the most innovative operational enhancement that enables successful continuous operations and extreme availability.

### **Agile incident, problem, and change processes**

The obvious operational change to continuous availability is the ability to process incidents immediately and bypass problems so that the business service is not affected. Continuous availability SLAs do not allow time to route tickets and dispatch them to the appropriate operations team before acting. Mature organizations have automated incident response activities: the automatic opening of incidents and immediate notification of incidents are pushed into a persistent chat room that all team members join during their working hours. If automation cannot remedy the incident, immediate action is taken by 24x7 first responders who follow the incident playbook document number posted with the incident. These playbooks are generated during readiness testing and before any business service goes live. If a playbook needs updating, a problem ticket is opened against it. If the 24x7 responders cannot remedy the problem after bypassing it, they have the choice of paging in the appropriate SME or technical webmaster on-call staff, or leaving the problem bypassed until normal business hours when the SMEs are at work. The correct continuous availability design enables the capability to bypass a problem and leave it in a broken state without a negative impact to the business. If needed, level 3 support experts can be called in to examine the system in its broken state, allowing true root cause analysis to occur. After the root cause is correctly identified, corrective action is taken, and the failed component is put back into the flow of business transactions.

Not so obvious is the necessity to align change processes to agile development processes. Abiding by the continuous availability design guiding principles, non-destructive changes can be performed any time of day or night rather than during a planned maintenance window during non-business hours. This causes a disruptive change to the change management processes. Rather than have change review boards that meet, review, and approve changes on a weekly basis, planned changes are reviewed and approved daily by the project manager and technical webmaster aligned to the business service and peer reviewed by SMEs within the same infrastructure, network and security, or middleware platform team. Daily calls for the entire staff describe the previous 24 hours of major incidents and review for recurring problems, and also notify all members of the changes that are planned for the following 24 hours that affect more than one business service.

One of the key lessons learned in mature continuous operations organizations is the necessity to control the schedule of business driven changes. Many mature continuous operations organizations have the ability to make changes at any time. For example, without having a method to manage business driven changes, the businesses can all decide they want to deploy their agile-developed applications every Monday. When you support tens of lines of business and their various business services, all changes cannot happen on the same day. They need to be staggered throughout the week because an operations organization does not have enough staff to orchestrate every business service change on a single day. For that reason, the concept of velocity and scheduling is applied. *Velocity* is a tool that the business can use to indicate the frequency of planned business service changes. Agile development teams typically have a high velocity because changes are propagated into production every two weeks for each business service application. Other development teams might have a velocity of releases that is much slower, monthly or even quarterly. Their velocities are based on a point system that reflects the frequency and difficulty of change. This enables the operations team to ensure that they have adequate resources available, and enables the business to adequately schedule the changes based on their velocity points available each week. If the business exceeds their planned velocity for whatever reason, they must request an exception, which must be approved by the management team.

### **Governance in the continuous availability world**

Governance in and around a continuous availability platform is slightly unique. With the success attained by building a continuous availability platform for mission-critical workloads and exceeding expectations comes the increase in demand from other lines of business that do not yet have continuous availability. Mature CIO organizations have formed Business Classification Review Boards whose mission is to correctly identify which tier of a service a business service requires. The Business Classification Review Boards review their business plan to ensure that they can afford the increased development and operational expenses required to attain continuous availability. This Business Classification Review Board uses a technique similar to the Business Impact Analysis mentioned in “Assess” on page 3 and approves or disapproves their request for a higher tier of availability.

After the Business Classification Review Board has approved the request to engage the Continuous Availability operations team, a project manager and technical webmaster are assigned to the business and work with the business and development teams to perform a “fit analysis”. A “*fit analysis*” is an assessment similar to the Business Application Maturity assessment where the business service functional and non-functional requirements and the application architecture are reviewed to ensure that the application architecture fits one of the existing continuous availability patterns without breaking any of the guiding principles. If the application architecture fits, it proceeds into the typical onboarding process.

If the business service application architecture requires a new pattern or deviation in the existing continuous availability patterns, it then goes in front of the operations teams’ Architecture Review Board. The mission of the Architecture Review Board (ARB) is to ensure that any new pattern fully complies with the continuous availability guiding principles and reference architecture. The ARB is made up of two team members from each of the delivery teams (Infrastructure, Network and Security, Platform/Middleware, and Technical Webmasters) and each team has one vote, yes or no. For the business service to board, the yes vote must be unanimous. Any dissent must be based on the guiding principles and reported to the management team. The management team has veto rights and might temporarily grant an exception based on needs of the business, with a date defined for mandatory compliance. Failure to meet that date can result in multiple actions: freeze all application releases until the application architecture fully complies, temporarily increase costs based on the additional support costs of a non-compliant application, or eject them from the continuous availability platform back to the HA platform.

## Proactive service management

Proactive Service Management Guiding Principle – It pays to be paranoid. You must monitor and trend everything to detect abnormalities before they create incidents and problems. This action is a disruptive leap from reactive to proactive service management. Immature operations organizations monitor only the infrastructure services and react when services fail. Mature organizations monitor every aspect of the business service from inside the environment, and externally. They have failed if a user or the business client reports a problem of which they are not aware. They gather real-time statistics and display them in simple to view trend graphs. These graphs are specific to the business services and display statistics within each cloud and across all clouds. This graphical perspective is extremely important in enabling non-destructive changes – the change orchestrator must be able to ensure that business service consumption has been quiesced in a “cloud” before making the change.

Figure 14 shows an example from the IBM Continuous Availability Services team’s traffic trend graph for [www.ibm.com](http://www.ibm.com). In the graph, you can clearly see the traffic quiesce as changes are rolled out to each “cloud”. This graph is a 4-day view. Notice how the blue graph represents the first cloud on which the change is performed. Notice after live traffic was brought back to the blue site, an issue that was not identified in pre-production testing was identified and traffic was again drained from the site and the issue was resolved. After the issue is resolved, the change was then again controllably performed on the second cloud shown in red, and finally, the third site shown in green.

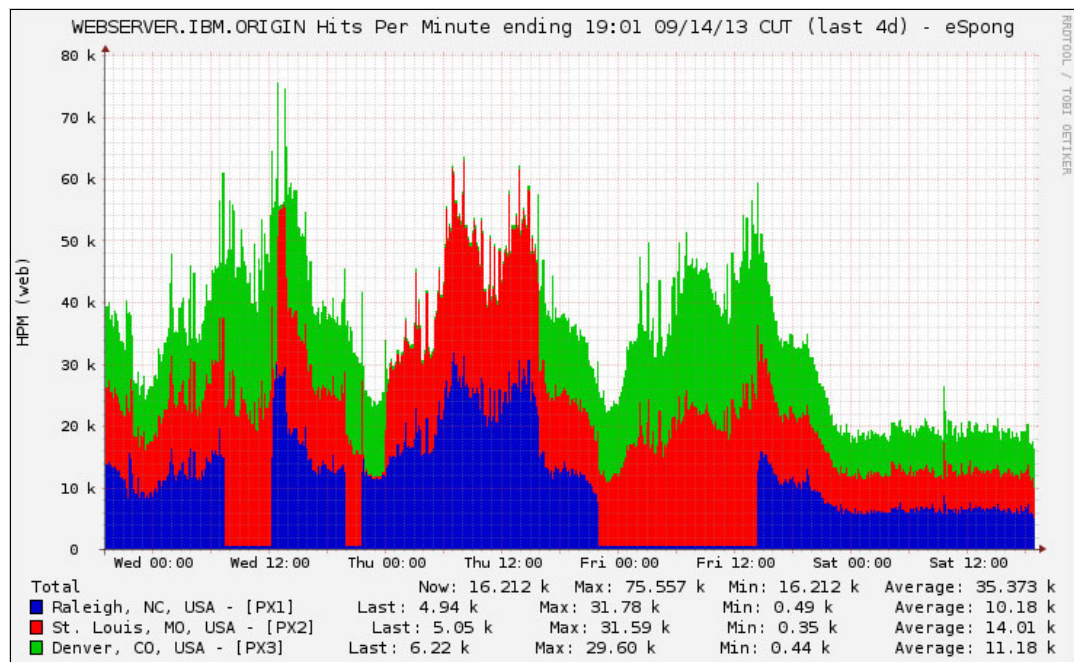


Figure 14 Traffic trend graph

An additional requirement for services management is similar to security management. Each “cloud” must be able to monitor itself independently, and all agents must report back to a minimum of two of the “clouds” to ensure visibility even in a catastrophic failure of any one cloud. Without an “always on” view, it is difficult to confirm that you are providing an “always on” service.

## Conclusion

Achieving extreme levels of availability, whether continuous availability or near continuous availability, is an achievable goal with what can be a long journey. These levels of availability require an entire organization to fully embrace the non-functional requirement of “always on” and accept the changes that this will mandate in their business, development, and operational procedures. These procedural changes encompass people, process, and technology and cannot be achieved changing only one of the three.

This document provides guidance about how to assess, design, implement, and manage extreme levels of availability. Because every organization and IT solution is different, practitioners will need the knowledge and tools to enable them to identify the maturity of the organization and its IT solutions, and assist their clients in understanding where they are currently and their journey forward. Our suggestion is to start with the business services that are most critical to the business, ensure that they are highly available and support concurrent versioning, and chart the roadmap toward Business/DevOps organizational change, fault tolerant business applications, and continuous operations delivery on a continuous availability platform.

## Author

This paper was produced by the following author:

**Herbie Pearthree** is a Senior Technical Staff Member and Acting CTO of the IBM Continuous Availability Services team in GTS America’s Delivery Technology and Engineering department. He is based in RTP North Carolina, US. He has over 25 years of experience in Internet-facing Enterprise Architectures specializing in resilient systems that are “always on”. He attended the University of Florida and Indiana University, focused on electrical engineering, business, and philanthropy. His areas of expertise include the entire enterprise platform, from the network, security, and infrastructure layer all the way up through the software stack. He has written extensively about continuous availability methods and speaks with IBM clients around the globe to help them understand methods of continuous availability and how to align their operational model to enable better availability and to reduce downtime.

Thanks to the following people for their contributions to this project:

- ▶ Bertrand Portier
- ▶ Scott Simmons
- ▶ Gunnar Karlsson
- ▶ Tom Parette
- ▶ Serge Bourbonnais
- ▶ Dale McInnis
- ▶ Jeffrey Calusinski
- ▶ Carol A. Miller
- ▶ Anantha (Srin) Rao
- ▶ James F. Walton

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new IBM Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "because IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright International Business Machines Corporation 2014. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document REDP-5109-00 was created or updated on July 14, 2014.



Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:  
[ibm.com/redbooks](http://ibm.com/redbooks)
- ▶ Send your comments in an email to:  
[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)
- ▶ Mail your comments to:  
IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400 U.S.A.



## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>


The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2®

IBM®

InfoSphere®

Redbooks®

Redbooks (logo) ®

Redpaper™

WebSphere®

The following terms are trademarks of other companies:

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.