A Guide to Architecting the Active/Active Data Center
Introduction

With the average cost of downtime per hour estimated at a whopping $163,674\(^1\), ensuring continuous availability for applications has become a top business priority. Aware of the business ramifications, organizations have gone to great lengths to keep their databases up and running, building disaster recovery (DR) systems to minimize the damage an outage can create.

Because of cost pressures, most organizations today are relegated to running active/standby systems. These architectures can take hours to restore operations and recover essential data following a failure. Within a single data center, line of business application recovery can take up to 30 minutes even when adequately set up for DR. For application processes that must run across data centers, recovery time can multiply, adding up to hours of downtime, with lost revenues and unmet service level agreements (SLAs) driving costs higher.

By their very design, active/standby scenarios depend on a secondary site that sits idle until a disaster hits. For most organizations, the operational cost of moving to active/active architectures has been too high to justify – such a design requires sophisticated application, database, and networking configuration. As a result, few organizations have implemented this approach. New technologies, however, are making it easier – and more affordable – to architect for active/active operations.

Active/Active Data Centers – The Gold Standard

Running two or more data centers in active/active mode means all data centers can serve application traffic at the same time. In addition to enabling continuous app availability, running the additional data center(s) in active mode also delivers increased capacity and better performance. This additional capacity improves the economics of active/active operations, since the systems are delivering throughput vs. sitting idle.

Implementing an active/active architecture requires more than just replicating the database from one data center to another; applications themselves must understand the database infrastructure and know details such as which servers are closer when clusters span different data centers.

Without this insight into database topology and locality, applications can’t send load optimally. With random load balancing, the application will send as much traffic to the remote data center as the local one, increasing WAN traffic and slowing application performance.

Furthermore, understanding locality for writes is essential for supporting cross-data center failover. Applications need to understand both read and write locality – reads should be sent to the closest available local server within the same data center, and writes should be sent to the current primary server in the appropriate data center.

---

\(^1\) Aberdeen Group research report, "Downtime and Data Loss: How Much Can You Afford?" 8/16/2013
Existing Approaches to Supporting Active/Active Operations

The three most common approaches to architecting active/active operations are building mechanisms within the application itself, relying on the database, and creating an abstraction layer.

1. Building Mechanisms for Active/Active Ops into the Application

Building support for active/active operations into the application takes a huge amount of engineering resources and continual maintenance. With systems being accessed not only synchronously (via user devices) but also asynchronously (via API calls from third-party services), supporting user and device locality within applications is highly complex and labor intensive. These architectures get exponentially more complex when you want to span more than two regions. Overall, this approach has suffered from such a poor ROI that few organizations have embraced it.

2. Relying on the Database for Active/Active Ops

Relying on the database to achieve active/active operations usually requires a third-party cluster management system (such as Galera) for master/master capabilities. This approach creates a slew of issues, including application consistency restraints that can negatively impact the user experience. In addition, this approach replicates all writes in all data centers, which decreases its value from a capacity standpoint.

Building active/active ops within SQL Server has also proven very challenging. AlwaysOn Availability Groups in SQL Server 2012/2014 don’t support location or replication awareness. The application is offered a random server for its connection – the first replica in the list would frequently not be the closest available server, and without replication monitoring, its data could be stale. Both scenarios can cause performance and data consistency issues that are difficult to diagnose and resolve.

In the same manner, because the AlwaysOn Availability Group Listener can’t perform read load balancing, all read loads go to only the first available read replica, leaving other read replicas in the group idle.

3. Creating an Abstraction Layer to Support Active/Active

According to the database consultancy Pythian, the abstraction layer approach provides a compelling alternative. “This architecture gives you a way to abstract away all the database and application variations,” says Aaron Lee, managing principal for DevOps at Pythian.

Lee contends that for 80% of organizations, using database load balancing software offers the best solution for supporting active/active operations. The software – which provides a range of traffic management capabilities including load balancing – understands the database architecture, knows the SQL state, and manages the SQL queries based on this deep knowledge. The software supports replication and performance awareness, ensuring quality of service and consistency rules – without having to build the capabilities into applications or dealing with the limitations of what a database natively supports.
The database load balancing software creates an abstraction layer, separating the apps from the databases. The software delivers load in the most optimal way, with no application or database changes.

**How ScaleArc Enables Active/Active Operations**

The ScaleArc software provides a range of traffic management capabilities, enabling an agile data tier. The software supports four key features that make it possible for organizations to easily adopt and benefit from active/active operations:

- Automatic support for read/write split
- Geo-aware load balancing
- Replication-aware serving of data
- Auto failover within or between data centers

1. **Automatic support for read/write split**
   ScaleArc has the ability to understand which queries are reads vs. writes and then appropriately distribute the traffic accordingly, sending all write transactions to the primary server and load balancing all reads across the available secondary servers. With ScaleArc, you don’t need to modify your applications with configurations such as read-intent strings to get the performance and reliability benefits of read/write split.

2. **Geo-aware load balancing**
   ScaleArc uses Time To First Byte to understand server response times, so the software will send queries to the servers that are closer and will send more load to those better-performing servers automatically.

3. **Replication-aware serving of data**
   ScaleArc monitors replication lag through a replication monitoring engine. The software will automatically avoid sending traffic to a node that has fallen behind the user-set threshold for replication delay.

4. **Auto failover within or between data centers**
   ScaleArc leverages the failover architected in the database to provide automatic failover at the application layer. The software recognizes a failover is underway, queues inbound queries to avoid application errors, and then forward the queries once the secondary is promoted to primary. The software can also route read queries to other available secondaries throughout the failover, ensuring continuous application availability. Because the application is directed to communicate with the ScaleArc abstraction layer, changes in the database structure behind that layer are transparent to the application, enabling cross-data center failover with no application changes (see diagram).

“**This [abstraction layer] architecture gives you a way to abstract away all the database and application variations.**”

-- Aaron Lee

**Managing Principal for DevOps, Pythian**
Figure 1. Running in HA mode in two data centers, ScaleArc sits transparently between the database and application servers. Since ScaleArc operates at the SQL protocol layer, it’s also completely app agnostic.

Apps connect to the ScaleArc instances in each data center, via a common DNS name. As a result, the application thinks it is always accessing the same database server – regardless of the server location where ScaleArc forwards the queries.

Impact of the Cloud

With capacity elasticity and vertical scale, the cloud can be less expensive than on-premise deployments for implementing active/active from a capacity allocation standpoint. The cloud also makes it very simple to migrate operations to a new region to support moves or changing demand.

With ScaleArc, you don’t need to modify your applications with configurations such as read-intent strings to get the performance and reliability benefits of read/write split.
However, without the ability to migrate IP addresses between machines, most existing HA methodologies don’t work in the cloud, as automated failover must also be supported across data centers in different geographies. This shortcoming, along with smaller instances, can make achieving active/active operations more problematic in the cloud.

ScaleArc solves this problem in the same manner it enables cross-data center failover for on-premise deployments. Because ScaleArc routes database traffic at Layer 7 or the Application layer, it can also automatically redirect traffic flows across data centers during failure.

### Reducing Risk and Adding Value

ScaleArc’s database load balancing software gives organizations a seamless and cost-effective way to implement active/active operations. With the ability to send load to multiple data centers, businesses can better utilize all database capacity. Rather than maintaining a cold DR site, organizations maximize performance and increase uptime with a second site that’s always servicing traffic.

A live secondary makes it possible to identify potential performance problems in the data stack—in advance of a failure. ScaleArc’s solution for active/active ops gives enterprises the confidence of knowing the secondary data center is ready to take load. In addition, because all data centers are now active, each site can be smaller. With support for distributed load, organizations don’t require as many servers in the primary and secondary data centers—about 30% fewer. If a failure happens, the site may slow slightly but will still remain operational.

Active/standby operations cost 2.5x to 3x the cost of a single data center, accounting for full redundancy and the operational costs of connecting and maintaining the two sites. Active/active operations, on the other hand, cost about 1.4x to 1.8x, since each site operates less equipment. Organizations can leverage resources more efficiently, while ensuring system reliability and lowering costs. Most importantly, the active/active architecture ensures application continuity, delivering the most value to today’s businesses, whether on-premise or in the cloud.

To learn more about these best practices, see ScaleArc’s “Active/Active Operations” webinar.