FORGET DOWNTIMES, ENSURE FLAWLESS SERVICES AT ALL TIMES!

HIGH AVAILABILITY

ZEXTRAS®

TABLE OF CONTENTS

Introduction

In recent years, IT's progressive increase has made High Availability systems popular in industries whose **income relies directly on their users' satisfaction and system availability**. Hospitals and data centers that require high availability of their systems to perform daily activities also hugely benefit from redundant infrastructures.

One of the most important features of **Zextras Carbonio** is its replication capabilty. This means with Zextras Carbonio, even if your servers experience some downtime, **your services will never go down** user side.

There are different ways to achieve this, such as having a complete replica of your infrastructure; this doesn't work for you simply because it's costly. Zextras, on the other hand, has overcome every drawback of traditional approaches by introducing new methods and concepts to provide users with an **extremely efficient replication system.**

What Is High Availability

High availability (HA) is an essential characteristic of a system that guarantees a **specific availability level** for a certain period.

For example, an enterprise might need a system availability or uptime of at least 99.99% in a year (which means 53 minutes of downtime per year). HA systems are the only way to guarantee a desirable level of availability.

Availability refers to the ability of the user to obtain service by accessing the system. Regardless of the reason, if a user cannot access the system, it is considered **unavailable** from their point of view. This period of unavailability is referred to as **downtime**.

SNIA Online Technical Dictionary
by Storage Network Industry Association defines **High Availability** as:

> *[Computer System] The ability of a system to perform its function continuously (without interruption) for a significantly longer period of time than the reliabilities of its individual components would suggest.*

As you see, high availability is somehow ambiguous, and even SNIA acknowledge that as follows:

> *High availability is not an easily quantifiable term. Both the bounds of a system that is called highly available and the degree to which its availability is extraordinary must be clearly understood on a case-by-case basis.*

Exploiting this ambiguity is tempting for vendors. Wise customers should distinguish between important aspects of availability and the jargons used only to convince customers. The best bet for customers is to educate and familiarize themselves with these concepts. This is what exactly Zextras has been trying to do by providing you with as much technical information as possible with maximum clarity.

Why Replica and Availability are so Important

In August 2013, Amazon ended up losing over $990,000 in a 15 minutes downtime. These figures are enormous even for a large company like Amazon.

In recent years, IT's progressive increase has made HA systems popular in industries whose income relies directly on their **users' satisfaction** and **system availability**. Hospitals and data centers that require high availability of their systems to perform daily activities also hugely benefit from HA systems.

Avoiding Huge Costs

Although the cost of downtime varies by the company size and industry, it is widely accepted as very expensive. These are some of the **costs you can easily avoid** by High Availability:

Immediate - Such as loss in employees' productivity and idle time. It can be different in industries. For example, *Network Computing* on its contingency planning research reports **the cost of a single hour of downtime on average for different industries** as below:

Brokerage services	$ 6,480,000
Energy	$ 2,800,000
Financial	$ 1,500,000
Health care	$ 636,000
Media	$ 90,000

Collateral - Such as customer loss, share price drop, reputation loss, employee discouragement, etc.

Who Needs It

The higher performance we gain from High Availability is always beneficial. However, it costs too. You must ask yourself:

- Is the availability of your services to everyone essential in running your business?
- Are these potential downtimes damaging for your company?

If your answer to either of the above questions is positive, the decision is much easier.

Measuring Availability

This section will talk about some common measurements for availability and introduce a framework to understand these measurements. The first step to measuring high availability is to characterize downtime.

How We Characterize Downtime

Downtime definitions can be different for different systems and enterprises. Failed servers, disks, operating systems, and applications can be rudimentary candidates for characterizing downtime. Alternatively, a slow server, performance, and data unavailability are more specific and sophisticated standards for this purpose. However, downtime should be defined based on the single-user point of view.

The system is down even if a single user cannot fulfill their task on time.

The goal is to provide end-users with the best possible solution to finish their tasks conveniently. When a user cannot achieve their goal, we consider the system to be down regardless of the reason. Alternatively, if a component fails but the user can complete their job, we don't consider the system down.

How We Measure Availability

Simply put, to measure high availability, all the system's failure modes should be known and defined. It includes networks and applications. In other words, all the failures have a recovery time with an upper bound. It means that we know how long a failure keeps the system down. Even if we don't know how to prevent specific failures, we should know them and prepare to recover if they happen.

A high available system can tolerate a failure in a critical component and recover from it in an acceptable known and bounded time.

It's pretty common when someone tells you they need 100 percent availability because their project is so important that they can't tolerate any downtime. No one is not against 100 percent availability, but considering how much it may cost, we are bound to find a sweet spot between the cost and availability.

One hundred percent uptime might be achievable if we need it only for a few hours a day. However, suppose 100 percent uptime is required 24/7 the whole year. In that case, the costs will skyrocket so much that only the most profitable enterprises can consider it, and even in that case, 100 percent availability is almost impossible to achieve for the long term.

Common Metrics

High availability is costly, and the closer you get to 100%, the cost increases quicker.

Let's say a server with no particular safeguard measures renders 99 percent availability. Now let's duplicate this server with an identical one as a substitute when the first server fails. The second server also has 99 percent availability. Therefore, theoretically speaking, we can expect a combined availability of 99.99 percent. It's achieved by multiplying the downtime of the first server, which is 1%, by the uptime of the second one, which is 99%, as the second server will be in use only over the 1% downtime of the first server. Therefore, we get 0.99% more availability. By adding the new 0.99% to the original 99%, we get a **99.99% availability**. This is the **theoretical uptime** for the combined servers.

The two servers' availability doesn't simply add up to 99.99% **in practice**. Because **the failover**, which is the substitution of the first server after it fails with the second one, **takes time**, and in the meanwhile, both the servers are down. Other fail factors impact both servers, such as power outages. All these factors reduce the 99.99% combined availability.
We cannot simply consider the increase in availability as linear. That's why the IT industry tends to use the **number of nines** for stating availability.

Percentages of a particular order of magnitude are sometimes described as the number of nines or class of nines in the digits.

Availability %	Downtime per year	Number of 9s
99%	3 days 15 hours 36 minutes	two
99.9%	8 hours 45 minutes 36 seconds	three
99.99%	52 minutes 34 seconds	four
99.999%	5 minutes 15 seconds	five
99.9999%	32 seconds	six
99.99999%	3 seconds	seven

Availability is a measure of the time in which the server operates according to our downtime definition. A simple equation used for this purpose is:

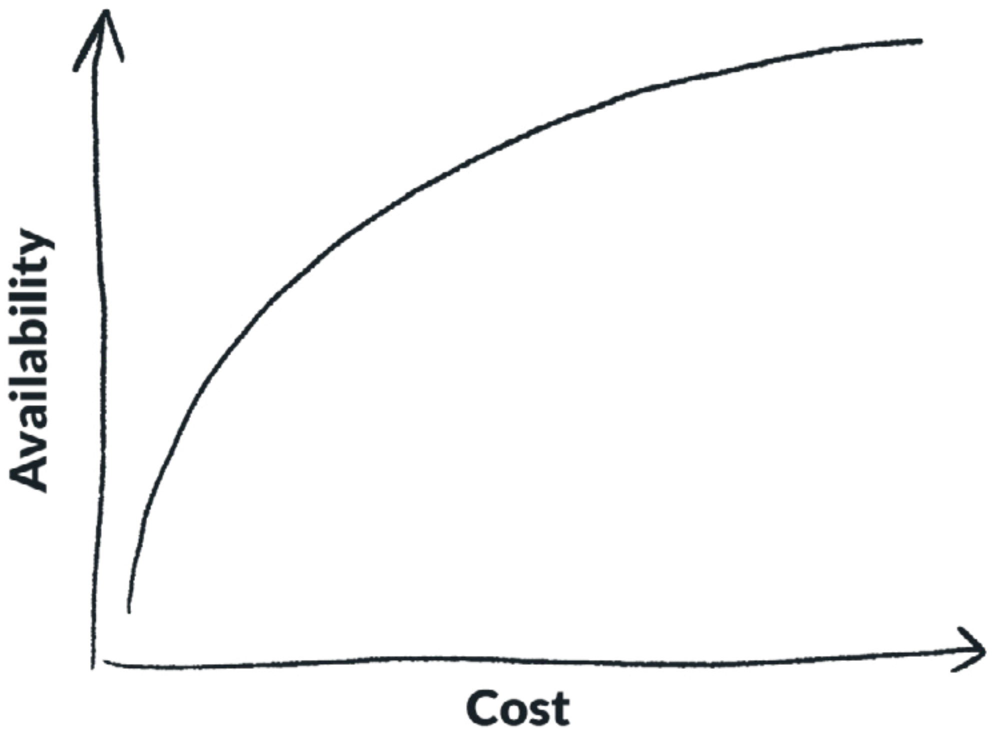
$$Availability = F / (F + R)$$

Where F is the average expected time between two failures for a given system and R is the maximum recovery time of the problem causing the failure.

For instance, the availability of a system with an F of 1000 hours and an R of 1 hour is 1000/1001 which is 99.9%. Decreasing R to 1/10, which is 6 minutes, adds another 9 to its availability which is 99.99%.

Increasing the availability level is extremely difficult, and the higher the availability level, the harder it gets.

This can be observed from the formula as well. By decreasing R to zero, availability increases to 100%, while the bigger the F, the less impact R has on availability. Based on the same reasoning, the following diagram shows the relation between investment on availability and the achieved availability.



Some Caveats

Although this scale is easy to understand, it doesn't take everything into account. The impacts of different downtime sessions can be different. Let's consider an outage scenario in two enterprises with the same availability. The same outage may cause a company to **lose customers** while it only causes **negligible disruptions** for another company's users. Another example would be the cost of downtime in an online shop. A 50-minute outage in a year means 99.99% uptime; however, the outage on a Friday night in December costs much more than Sunday midnight. As you see, **the number of these nines doesn't reflect the difference between the two**.

Let's consider another case in which different components are available in a network with their own level of availability. For simplicity, let's say all have the same availability of 99.99%. If one component fails, the whole system goes down as the availability of this system for the end-user translates to the availability of **all the components in the chain**. If there are seven components, since they have all the same level of availability, it might intuitively seem evident that the whole system has the same. However, if you do the math, 99.99% availability of seven components gives us 99.997% or 99.93%. This might not seem to be a big difference; however, if you calculate the downtime of the system for these two figures, you'll get:

Availability %	Downtime per year
99.99%	52 minutes 34 seconds
99.93%	6 hours 5 minutes 8 seconds

As you see in the table above, this system's **actual downtime per year is much bigger** than what one might expect based on their intuition. The correct way to look at it is to consider availability in terms of the downtime itself, not the number of nines.

Let's see it in action. For simplicity, consider that components won't be down simultaneously. Since every component is prone to **52m34s** of downtime each year, seven components will yield **7×52m34s** per year, or around **6 hours per year or 99.93%**. As you see, availability in terms of the number of nines is not intuitive enough to give a customer a clear idea of the actual availability of a system.

Zextras Carbonio High Availability

High availability (HA) is an essential characteristic of a system. Traditionally high availability systems consist of a **set of loosely coupled servers** with failover capabilities. This is far from being perfect.

Zextras has tried to overcome every drawback of simple traditional approaches by introducing **new methods and concepts** to provide users with an extremely efficient HA system.
Replication and Heartbeat technologies are two examples that you can learn more about in the following.

Zextras Carbonio offers more than a traditional HA system.

How Zextras Carbonio Achieved It

Although HA is exceptionally **advantageous** in many industries, implementing a well-designed HA system can be complicated given the vast range of software, hardware, and deployment options. However, a successful effort always starts with determining business requirements and defining our needs. This is precisely what Zextras has done to pursue its HA approach.

Zextras achieved this goal using the **application approach**, as opposed to **hardware** and **operating system** approaches.

Let’s Compare Three Approaches

We can achieve HA through different approaches based on fail tolerance:

1 **Hardware (VSAN, clustered hardware)**

When several machines write on the same disk with the support of a hypervisor.

2 **Operating system (DRDB + VIP)**

Like in the Hardware approach, OS writes on different physical disks.

3 **Application**

This approach is far more intelligent and considers what is happening to the objects.

Advantage of Application Approach

The main advantage of the Application approach over the other two methods is that **it considers what happens to the files**. The other two methods don't care about the essence of the operation affecting the file and continue replicating it over all the nodes. For example, an admin makes a mistake and physically deletes a file on a mailstore. In both cases, the same operation is replicated over all the other nodes, even if the operation was originally a mistake.

1. In the **Hardware approach**, when you delete the file from the disk, the same will be done on other instances using the same disk no matter what.

2. In the **Operating system approach**, when you delete the file from one physical disk, the OS doesn't care and does the same operation on the second one.

3. In the **Application approach**, you know what is supposed to happen to which data and decide which operations should be replicated, which files we can alter, and which accounts should have HA. For example, you know if it's a log or temp file, or maybe it's something that the application shouldn't alter, such as removing one file or encryption at a low level.

Using the **Application approach** is a revolutionary change. This smart nature extensively improves the availability of Zextras Carbonio servers, which is also considered **more economical** compared to other methods.

Operation awareness, being more economical, replication mesh, Heartbeat technology are achievable only with this approach. You will learn more about these in the next sections.

How Carbonio Replica and High Availability Work

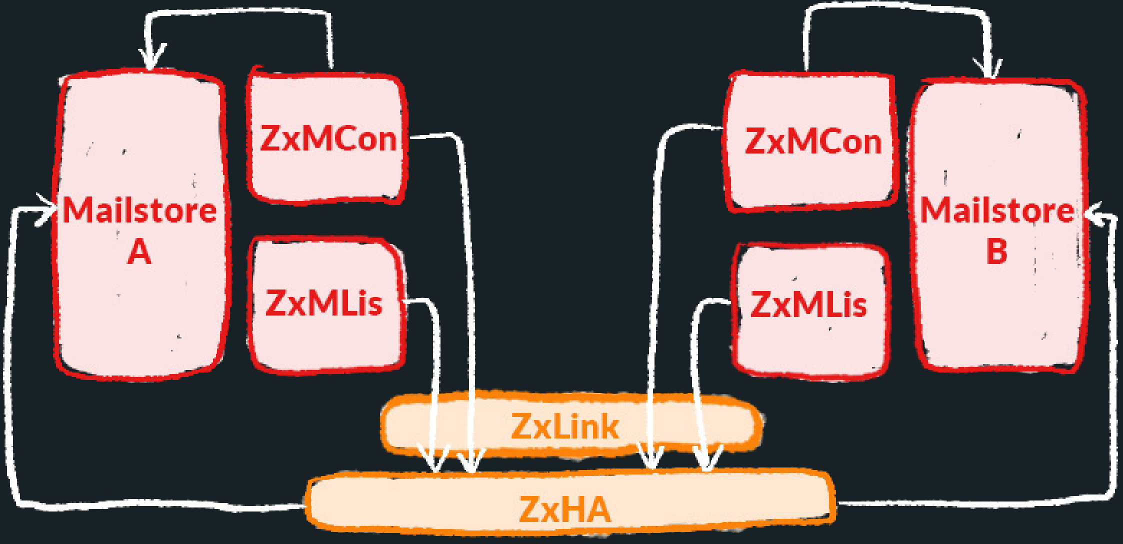
To better understand how Zextras achieves high availability, we need to know how high availability **basic concepts** such as *eliminating the single points of failure* are applied.

High Availability Basics

Let's see how these concepts exert in components such as; LDAP, Proxy, MTA, and Memcached.

- **Elimination of single points of failure:** Having more than one server to do the same service. We should have the possibility of installing each component of the infrastructure more than once so that failure of a component does not mean failure of the entire system. e.g., Having multiple MTA so if one MTA goes down, we won't lose the main queue. Mailboxes are a single point of failure.

- **Failover**: This is a backup operational mode in which a secondary system continues the functions of a system component if the primary one goes offline. Users may never see a failure. This is achievable by an internal load balancer. Your system can have interruptions, like when users need to refresh the page, however, you **won't lose anything**. The system does that **automatically**, and no admin action is required.



Zextras Carbonio Active Replica System Features

In this section, we will talk about the unique features of the Zextras Carbonio HA system. A server equipped with these features benefits the end-user experience and eliminates system administrators' unnecessary burden of maintaining, recovering, and failover. Let's take a look at how they work.

Service Granularity

All the components are in **multiple instances** and **redundancy**. All the apps in our mailstores use soap calls to do operations such as uploading, downloading, sending messages, changing status, etc. There are two essential components:

1. A **listener** (ZxMLis) - Each node has its listener that copies the same operation performed in the mailstore into a bus (ZxHA) shared between them.

2. A **consumer** (ZxMCon) - Each node has a consumer that reads the operations queue stored in ZxHA and performs them on its mailbox. It simulates the user interactions.

Zextras Carbonio uses this design to let you **choose which accounts** can use High Availability. For instance, some computationally intensive operations such as indexing can be excluded from HA, so each server performs them using its own resources. This intensely reduces the number of bytes sent to the bus.

Independency

The whole process is **autonomous (independent from the store)**. It means that mailstore A sends information to the bus regardless of the mailstore B status. Even if the mailstore B was offline when the operation is stored, it could read the bus whenever available. Alternatively, B can read the operations stored in the bus even if A is not available. The system can recreate every operation because all the necessary information is available on the bus.

Independency: The bus used in the Zextras Carbonio HA system is **independent of the source or destination stores**. It means that even if the stores are unavailable, the system can retrieve the necessary information to recreate the operations.

This is possible thanks to storing information on an autonomous bus instead of the stores.

Speed

Since this bus is based on a memory map, it's fast. Technically speaking, there are two different concepts when it comes to storing data.

- Storing the metadata,
- Storing the physical object.

The same concept is used in designing our central storage. The file uploaded in A can be available in B simply by reporting its ID.

Zextras Carbonio HA bus is based on a **memory map**.

Avoid Replicating Errors

This approach also lets you **avoid replicating application errors**. For example, the system will not replicate the operations thanks to a kernel panic when a mailstore gets corrupted.

Another scenario is when a massive account update can cause network workload; in this case, the administrator can easily pause the replication of those accounts to avoid network workload.

More Economic

This approach also helps you to balance wisely. Let's say you're **comparing different HA solutions** for your infrastructure. Replicating all the infrastructure can be costly, **depending on the size** of your infrastructure. With Zextras Carbonio HA, each store can be a replica of another one which is a lot more economical **regardless of the size** of your infrastructure. To understand better, consider the following scenarios:

1. One way would be **replicating all the infrastructure** to prevent any slowdown or downtime. Replicating all the infrastructure is not a big deal in situations with limited accounts, say only 1000. However, replicating the infrastructure is not viable if you have 20 stores, each with 3,000 accounts. It costs you a lot of money while most probably all the 20 stores will not go down simultaneously.

2. Another solution would be Zextras Carbonio Replication Technology. In our approach, **each store could potentially be a replica** of another one. Let's say you have 3,000 accounts on store A replicated on two other stores (1000 accounts on one store and 2000 on another, and so on). Therefore, each store acts as a peer creating a mesh.

Replication Technology

The great thing about Zextras Carbonio HA is that depending on how big your infrastructure is and how capable your stores are, you might tolerate losing several stores before the whole infrastructure is considered at fault. Having this account-based replication with a mesh structure is another advantage of the application approach.

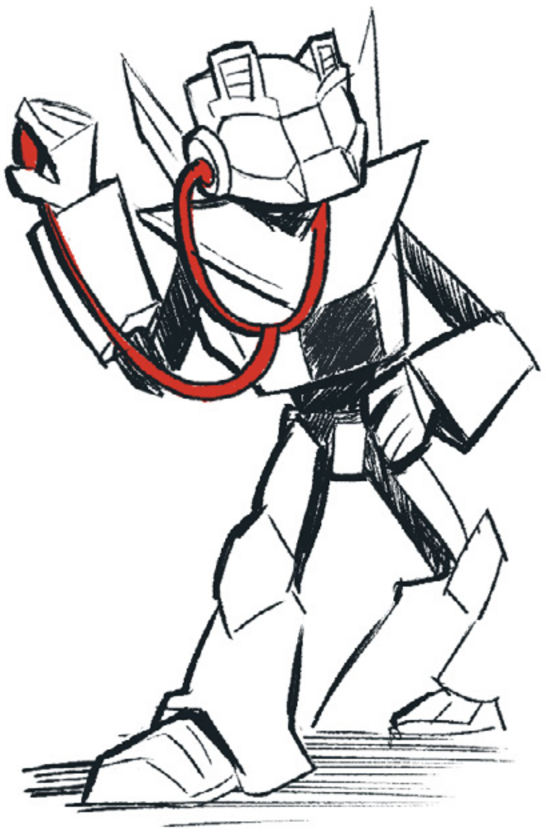
For instance, let's say you have four servers and four accounts using the application approach. Having an account on a server is specified as **Active**. **Active Replica** means that the account is replicated on that server.

	Account 1	Account 2	Account 3	Account 4
Server 1	Active	Active	Active Replica	Active
Server 2	Active Replica	Active Replica	Active	
Server 3	Active Replica	Active Replica		
Server 4	Active Replica			

You can also choose which accounts to be replicated based on their importance. (Note that it's different from having a backup, this technology aims to have multiple copies on other servers that are all available.)

Heartbeat Technology

Heartbeat plays a crucial role in ensuring High Availability, acting as **the pulse of the infrastructure**, monitoring the health of nodes, and facilitating seamless fail-over mechanisms.



Heartbeat is responsible for detecting the availability and **health status of a service** within the infrastructure.

It operates by sending **periodic messages**, known as heartbeats, between service nodes to ascertain their operational state. These messages serve as indicators of the service functionality and connectivity and can be used to route service requests to a healthy instance within the cluster.

Carbonio implements most of those heartbeats messages through Consul and the Sidecar Health checks. Those messages are evaluated by the HealtCheck system according to service implementation:

- **published service**, like MTA or Proxy, must be managed by an external/edge balancer;
- **internal stateless services**, such as Files or Auth, are managed by the internal Service Discover routing;
- **internal state full services**, such as LDAP, Postgres, or the Mailstore, must be managed by specific scripts according to the infrastructure need.

This is because **not all the services** use the same heartbeat parameters such as interval timing, network availability, and failure detection thresholds.

One of the most challenging scenarios is the "**Split-Brain**". This occurs when network partitions or communication failures cause the cluster to separate into multiple disjointed subgroups, each perceiving itself as the primary active cluster. Split-brain scenarios lead to conflicting state information and can result in data inconsistency, service disruptions, and degraded system performance.

Mitigating the split-brain issue requires proactive measures and strategies, but the most common is implementing a **quorum-based approach**, which ensures that the majority of nodes must agree on the cluster's operational state before taking action. It requires an odd number of Health Check controllers, to ensure the quorum is always agreed.

ZEXTRAS®

Want to know more?

Visit zextras.com

Zextras ©2024