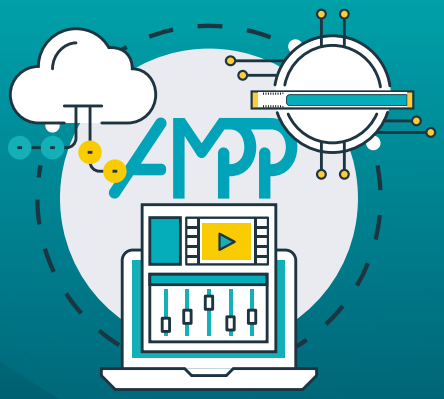
Grass Valley

TECHNICAL BRIEF



Rigidity, Redundancy and Resiliency

In a modern cloud environment, is redundancy a waste of resources?

The traditional broadcast model for the oft-cited five nines of uninterrupted performance requires duplication of all essential components in the production chain. Because the components must be available in case of a fault, these resources of necessity remain largely idle. While this was never a good optimization of resources, in today's cloud environment where workflow runtime is metered and all resources, whether used or on reserve, have an associated cost, keeping a fully redundant system on hot standby is wasteful.

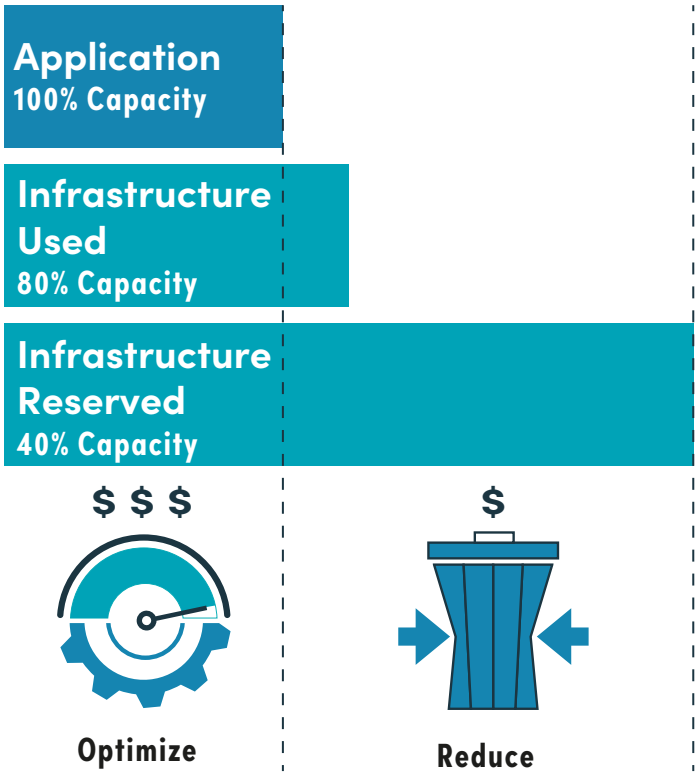
Many early cloud-hosted components offered in the broadcast market are lift-and-shift, monolithic developments that require full duplication in the cloud to preserve the standard of uninterrupted performance, thus locking in the additional cost of redundancy.

For these systems to work, they rely on rigid networks with a very fixed operating environment to maintain connectivity with each other and contributing components. Any fault in the system has the potential to result in system failure with no graceful way to recover.

The best cloud-based systems today have thrown out that older way of thinking. They focus instead on achieving the ultimate goal, which is a high level of resilience. Resilience is measured by the system's mean time to failure (MTTF) and mean time to recover (MTTR). Other measures of resilience can include the system's ability to:

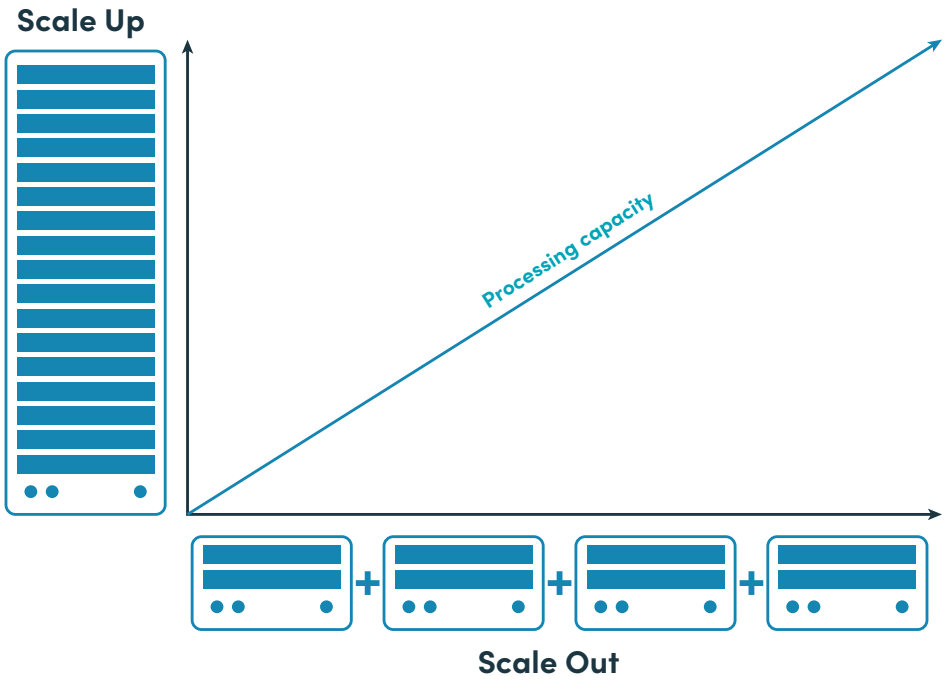
- Forecast potential faults
- Isolate impacted components
- Protect against potential faults
- Remove faults and recover from a fault state
- Restore optimal system performance (Muhammad, 2019)

Cloud Resource Consumption



Rigidity

Rigid network resources, physically assigned in metro- and wide-area networking (WAN) networks, aren't easily manipulated into making dynamic changes. The complexity of the network makes it difficult to maintain their existing connections, while at the same time, build new agile platforms with on-demand, multi-tenant services. (Raab, 2016)



Redundancy

The intentional duplication of system components to increase a system's stability. An absolute measure of the additional components supporting system resilience. (Muhammad, 2019)

Resiliency

A system's ability to recover from a fault and maintain persistency of service dependability in the face of faults. A relative and continuous measure of the impact of fault on the system operation. (Muhammad, 2019)

By focusing on the objective of resilience rather than redundancy, it becomes much easier to make decisions that measure and improve reliability while optimizing for the lowest cost and operator complexity.

It is the nature of cloud-based processing to occasionally have faults. A data center may suffer a power loss. A server or app may fail. To maintain high standards of performance, resiliency in cloud systems must be a consideration for every aspect of the system design: the network, the platform, and the solutions that run on those platforms.

Compensating for potential faults begins with software-defined networking (SDN). SDN enables at-scale network virtualization and spin-up of a virtual network infrastructure to more flexibly, and cost-efficiently, meet customer's demands.

In a well-designed, modular system with elastic capacity, a fault does not have to automatically cascade to an error and then failure. When a cloud solution is modular all the way to the microservices level, it can enable resiliency strategies that surgically target replication, rerouting and restarting specific resources on an as-needed basis. Each service externalizes its state — indicates to the rest of the system its health status — so that an effective error detection mechanism can provide early warning of potential faults and isolate, stop and restores potential problem areas as a preventive action. The fault remains invisible to the system operator — all without human intervention during the recovery event.

Resiliency in the cloud begins at the network level with the concept of scaling out.

In a traditional broadcast workflow, the common model of addressing large scale production was to purchase the largest hardware processing frames available. Many companies, including Grass Valley®, have established market leadership around providing equipment with the largest possible processing capacity. Regardless of the size, these processing capabilities have a limit both in terms of number of possible computations and geographic access to the processors.

In well-designed cloud systems, instead of maximizing the capacity of a single machine, additional processing capacity is added across multiple servers that share the work. Servers are deployed and orchestrated as needed with something like Amazon ECS or a technology such as Kubernetes. Unlike scaling up, which has a single location, a single health status, and a maximum size, scaling out provides essentially unlimited ability to scale, geo-agnostic access, and much more elasticity in working around faults.

Models of scaling up vs scaling out impact recovery response times. There is not a one-size-fits-all solution. The desired amount of response time and standby resource must be designed using the Service Level Agreements of the individual cloud services in the design.

A similar modular architecture for the SaaS solution running on the network is a critical part of preemptive fault responses. The control system monitors the health of each module used in the solution.

A general rule of thumb is: the larger the functional unit that must fail before the recovery process is initiated, the longer the system takes to recover. This is why the concept of microservices is so important. A solution based on microservices, as opposed to a monolithic deployment, makes it much easier and faster to forecast and isolate potential faults, while bringing up new resources to compensate. A monolithic system only has a single state for the entire system. When the entire system is down, it takes much longer to recover.

There are a variety of methodologies in cloud technology to provide a resilient system. While it isn't critical to understand all the different technology options, it is important to ensure that the total system meets your performance requirements.

Consider the following as you review options for developing a cloud-based media production system.

- ✔ Is the SaaS application platform-agnostic? Being constrained to a single cloud provider may limit your ability to acquire new resources where you need them.
- ✔ How important is it to my project to be geographically dispersed? Consider geography both in terms of local access to data centers and in terms of distribution of backup resources.
- ✔ What forecast mechanisms are in place to pre-emptively recruit resources prior to a potential failure?
- ✔ If a system were to fail, what failover response time do I require?
- ✔ Is the proposed system capable of meeting that failover response time?
- ✔ Does the recovery provide an instance that is identical to the initial instance in terms of capability and performance?
- ✔ Do I require the recovery system to be identical, or can it be "good enough?"