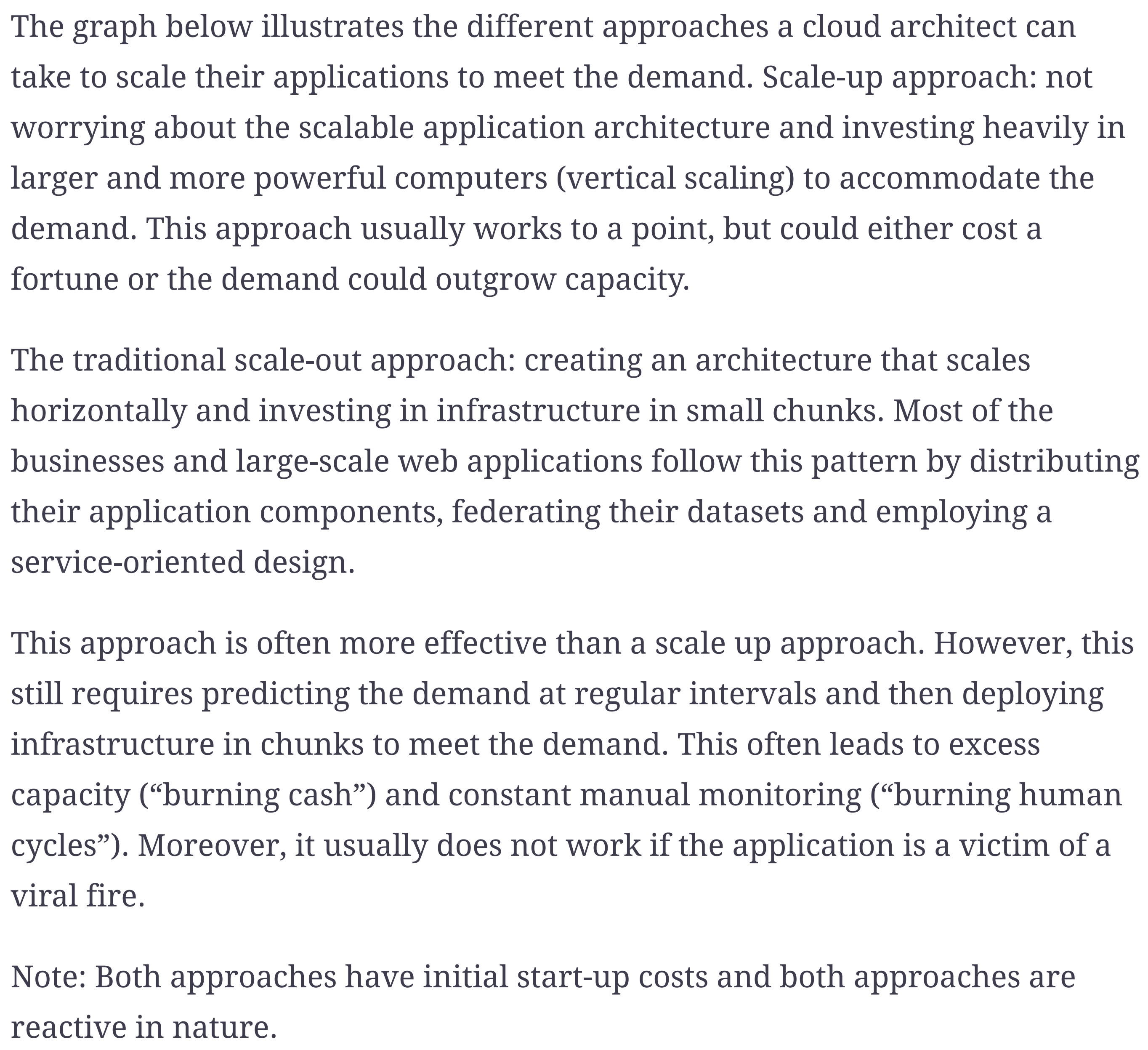
Understanding Elasticity on the Cloud

When you decide to move your applications to the cloud and try to map your system specifications to those available in the cloud, you will notice that cloud might not have the exact specification of the resource that you have on-premise.

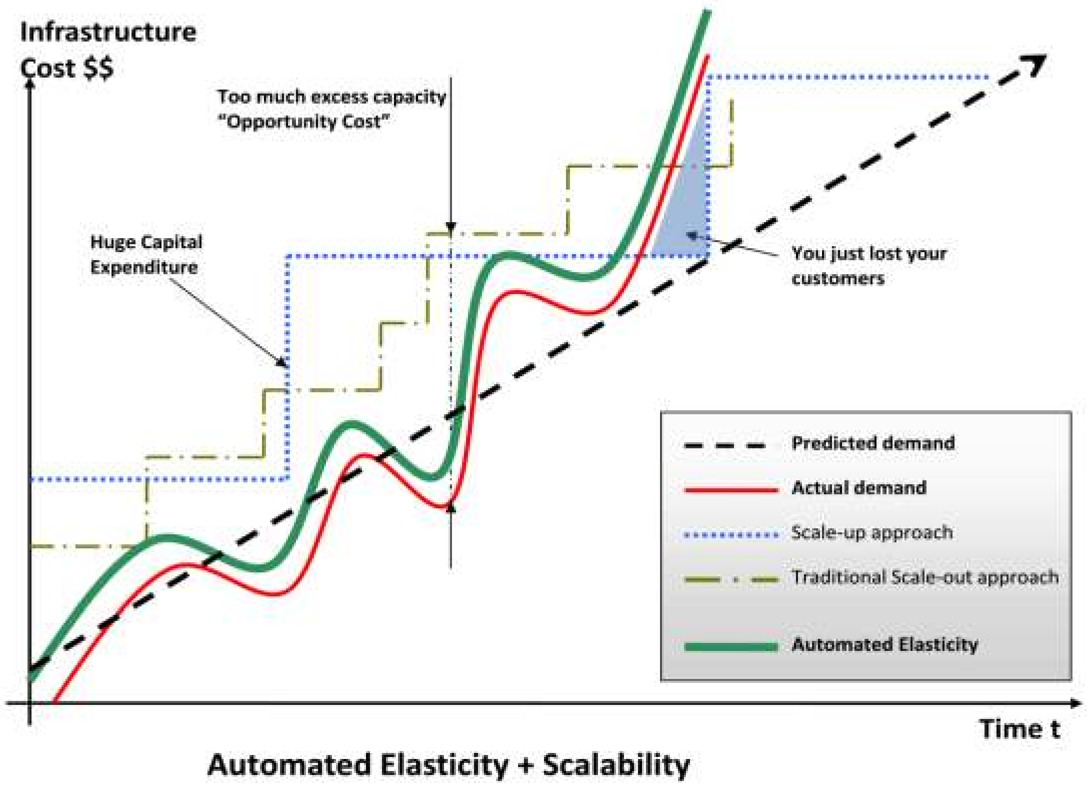


The graph below illustrates the different approaches a cloud architect can take to scale their applications to meet the demand. Scale-up approach: not worrying about the scalable application architecture and investing heavily in larger and more powerful computers (vertical scaling) to accommodate the demand. This approach usually works to a point, but could either cost a fortune or the demand could outgrow capacity.

The traditional scale-out approach: creating an architecture that scales horizontally and investing in infrastructure in small chunks. Most of the businesses and large-scale web applications follow this pattern by distributing their application components, federating their datasets and employing a service-oriented design.

This approach is often more effective than a scale up approach. However, this still requires predicting the demand at regular intervals and then deploying infrastructure in chunks to meet the demand. This often leads to excess capacity (“burning cash”) and constant manual monitoring (“burning human cycles”). Moreover, it usually does not work if the application is a victim of a viral fire.

Note: Both approaches have initial start-up costs and both approaches are reactive in nature.



Traditional infrastructure generally necessitates predicting the amount of computing resources your application will use over a period of several years. If you underestimate, your applications will not have the horsepower to handle unexpected traffic, potentially resulting in customer dissatisfaction. If you over-estimate, you're wasting money with superfluous resources. The on-demand and elastic nature of the cloud approach (Automated Elasticity), however, enables the infrastructure to be closely aligned (as it expands and contracts) with the actual demand, thereby increasing overall utilization and reducing cost. Elasticity is one of the fundamental properties of the cloud.

Elasticity is the power to scale computing resources up and down easily and with minimal friction. It is important to understand that elasticity will ultimately drive most of the benefits of the cloud. As a cloud architect, you need to internalize this concept and work it into your application architecture in order to take maximum benefit of the cloud.

Traditionally, applications have been built for fixed, rigid and pre-provisioned infrastructure. Companies never had the need to provision and install servers on daily basis. As a result, most software architectures do not address the rapid deployment or reduction of hardware. Since the provisioning time and upfront investment for acquiring new resources was too high, software architects never invested time and resources in optimizing for hardware utilization. It was acceptable if the hardware on which the application is running was under-utilized. The notion of “elasticity” within an architecture was overlooked because the idea of having new resources in minutes was not possible.

With the cloud, this mindset is changing. Cloud computing streamlines the process of acquiring the necessary resources; there is no longer any need to place orders ahead of time and to hold unused hardware captive. Instead, cloud architects can request what they need mere minutes before they need it or automate the procurement process, taking advantage of the vast scale and rapid response time of the cloud. The same is applicable to releasing the unneeded or under-utilized resources when you don’t need them. Or in with most modern cloud provider now have auto scaling.

If you cannot embrace the change and implement elasticity in your application architecture, you might not be able to take the full advantage of the cloud. As a cloud architect, you should think creatively and think about ways you can implement elasticity in your application. For example, infrastructure that used to run daily nightly builds and perform regression and unit tests every night at 2:00 AM for two hours (often termed as the “QA/Build box”) was sitting idle for rest of the day.

Now, with elastic infrastructure, one can run nightly builds on boxes that are “alive” and being paid for only for 2 hours in the night. Likewise, an internal trouble ticketing web application that always used to run on peak capacity (5 servers 24x7x365) to meet the demand during the day can now be provisioned to run on-demand (5 servers from 9AM to 5 PM and 2 servers for 5 PM to 9 AM) based on the traffic pattern. Designing intelligent elastic cloud architectures, so that infrastructure runs only when you need it, is an art in itself. Elasticity should be one of the architectural design requirements or a system property.

Question that you need to ask:

What components or layers in my application architecture can become elastic? What will it take to make that component elastic? What will be the impact of implementing elasticity to my overall system architecture?

Not Fearing Constraints

When you decide to move your applications to the cloud and try to map your system specifications to those available in the cloud, you will notice that cloud might not have the exact specification of the resource that you have on-premise. For example, “Cloud does not provide X amount of RAM in a server” or “My Database needs to have more IOPS than what I can get in a single instance”. You should understand that cloud provides abstract resources and they become powerful when you combine them with the on-demand provisioning model.

You should not be afraid and constrained when using cloud resources because it is important to understand that even if you might not get an exact replica of your hardware in the cloud environment, you have the ability to get more of those resources in the cloud to compensate that need. For example, if the cloud does not provide you with exact or greater amount of RAM in a server, try using a distributed cache like memcached or partitioning your data across multiple servers.

If your databases need more IOPS and it does not directly map to that of the cloud, there are several recommendations that you can choose from depending on your type of data and use case. If it is a read-heavy application, you can distribute the read load across a fleet of synchronized slaves.

Alternatively, you can use a sharding algorithm that routes the data where it needs to be or you can use various database clustering solutions. In retrospect, when you combine the on-demand provisioning capabilities with the flexibility, you will realize that apparent constraints can actually be broken in ways that will actually improve the scalability and overall performance of the system.