**Published: 15 September 2022**

**Shane Korbel**

**Improving datacenter utilization through containerized service-based architecture**

The modern datacenter's computing capabilities have far outstripped the applications running within and have become a hidden cost of doing business due to how software is architected and deployed. Resources are over-allocated to monolithic applications that sit idle for large parts of the day. If applications were architected and deployed differently, shared services could be used for multiple applications as needed. When combined with powerful orchestration software, containerized microservices can both deploy and dynamically scale applications from very small to very large within moments—scaling the application not only across a single datacenter but across all datacenters where the application(s) are deployed.

In this paper, we analyze data from an application(s) deployed both as a single monolithic codebase and as a containerized application using microservice-based architecture to calculate the performance and computing resource waste are both architected and deployed. A modern approach is offered as a solution as a path from how to go from a monolithic codebase to a more efficient, reliable, scalable, and less costly deployment model.

Software advances of the 1980s through the early 2000s were primarily limited by the hardware that it ran on. The finite number of resources available on hosts meant that software was constrained or rate-limited in processing abilities. After the millennium turn, hardware advances started to surpass software's ability to consume all the available resources. This shift in dynamics has led to extra resources allocated to other software or where a single physical host could run many virtual hosts. Even with this deployment model, resources go unused and represent a significant opportunity for cost savings by modifying how we build and deploy software.

Datacenters and computer hardware have undergone several significant transformations over the last four decades. In the early '80 s, mainframes took up entire rooms and performed specific tasks over and over very quickly. Mainframes at the time, and for the most part, remain specialized computing devices that are not widespread. Size and cost limited mainframes to large organizations to do specialized processing quickly. When hardware became smaller and less expensive during the '90 s, a shift was seen to more generalized computing machines and the rise of distributed architecture, where datacenters were now comprised of hundreds or thousands of servers running parts of applications.

These smaller distributed computing systems grew more powerful at the turn of the millennium. The distributed servers became able to run more than a single application and eventually more than a single guest Operating System (OS), leading to a rise in virtualization of servers where many server instances may run on a single host. Virtualization led tech giants like Amazon, Microsoft, and Google to create large networks of datacenters across all geographies, starting the rise of infrastructure as a service as early as a decade ago. Infrastructure as a service has companies shifting away from using on-premises datacenters to cloud-based solutions for hosting applications or hybrid on-premises/cloud deployments.

Software architecture over the decades has undergone an extensive series of transformations. In this paper, and to keep things focused, only changes that came about because of datacenter hardware advances will be explored. Early software was purposely built for processing bulk data over and over on mainframes. When distributed systems started to take over, and hardware was more generalized, the software could be designed to run at scale by adding more systems running the same software and then balancing each system's load.

In the last decade, two factors have altered the way software is architected: containerization and service-orientated architecture. Building software to run in containers is not a new concept. Containerization was created and has been around for decades to keep the kernel safe by isolating the running software kernel to take all the host's resources and compromise its stability. Keeping this critical interface operating makes containerization so appealing to prevent software that is not part of the operating system from affecting system stability. Containerization saw very little use until the Service-Oriented Architecture (SOA) model was developed. SOA redefines a way to make software components reusable by defining service interfaces that can be used by other software. These interfaces utilize standardized communication so that they can be readily incorporated into new applications without the need to perform reintegration each time.