DEVELOPER STORY

ZEISS Demonstrates the Power of Scalable Workflows with Ampere® Altra® and SpinKube

By Scott Fulton III

SNAPSHOT

Challenge - The cost of maintaining a system capable of processing tens of thousands of near-simultaneous requests, but which spends greater than 90 percent of its time in an idle state, cannot be justified. Containerization promised the ability to scale workloads on demand, which includes scaling down when demand is low. Maintaining many pods among a plurality of clusters just so the system doesn't waste time in the upscaling process contradicts the point of workload containerization.

Solution - Fermyon produces a platform called SpinKube that leverages WebAssembly (WASM), originally created to execute small elements of bytecode in untrusted web browser environments, as a means of executing small workloads in large quantities in Kubernetes server environments. Because WASM workloads are smaller and easier to maintain, pods can be spun up just-in-time as network demand rises without consuming extensive time in the process. And because WASM consists of pre-compiled bytecode. It can be executed on server platforms powered by Ampere® Altra® without all the multithreading and microcode overhead that other CPUs typically bring to their environments — overhead that would, in less compute-intensive circumstances such as these, be unnecessary anyway.

Implementation - As a demonstration of SpinKube's effectiveness, ZEISS Group's IT engineers partnered with Ampere, Fermyon, and Microsoft to produce a system that spins up new WASM pods as demand rises in a just-in-time scenario. The demonstration proves that, in practice, a customer order processing system running on SpinKube, compared to a counterpart running with conventional Kubernetes pods, yields dramatic benefits. According to Kai Walter, Distinguished Architect at ZEISS Group,

"When we looked at a runtime-heavy workload with Node.js, we could process the same number of orders in the same time with an Ampere processor VM environment for 60% cheaper than an alternative x86 VM instance"

Kai Walter, Distinguished Architect, ZEISS Group

Source: [How ZEISS uses SpinKube and Ampere on Azure to Reduce Cost bt 60%]

Background: The Overprovisioning Conundrum

It's still one of the most common practices in infrastructure resource management today: overprovisioning. Before the advent of Linux containers and workload orchestration, IT managers were told that overprovisioning their virtual machines was the proper way to ensure resources are available at times of peak demand. Indeed, resource oversubscription used to be taught as a "best practice" for VM administrators. The intent at the time was to help admins maintain KPIs for performance and availability while limiting the risks involved with overconsumption of compute, memory, and storage.

Because of their extensive experience with object cache at AWS, the Momento team settled on caching for their initial product. They have since expanded their product suite to include services like pub-sub message buses. The Momento serverless cache, based on the Apache Pelikan open-source project, enables its customers to automate away the resource management and optimization work that comes with running a key-value cache yourself.

At first, Kubernetes promised to eliminate the need for overprovisioning entirely by making workloads more granular, more nimble, and easier to scale. But right away, platform engineers discovered that using Kubernetes' autoscaler add-on to conjure new pods into existence at the very moment they're required consumed minutes of precious time. From the end user's point of view, minutes might as well be hours.

Today, there's a common provisioning practice for Kubernetes called paused pods. Simply put, it's faster to wake up sleeping pods than create new ones on the fly. The practice involves instructing cluster autoscalers to spin up worker pods well in advance of when they're needed. Initially, these pods are delegated to worker nodes where other pods are active. Although they're maintained alongside active pods, they're given low priority. When demand increases and the workload needs scaling up, the status of a paused pod is changed to pending. This triggers the autoscaler to relocate it to a new worker node where its priority is elevated to that of other active pods. Although it takes just as much time to spin up a paused pod as a standard one, that time is spent well in advance. Thus, the latency involved with spinning up a pod gets moved to a place in time where it doesn't get noticed.

Pod pausing is a clever way to make active workloads seem faster to launch. But when peak demand levels become orders of magnitude greater than nominal demand levels, the sheer volume of overprovisioned, paused pods becomes cost prohibitive.

ZEISS Stages a Breakthrough

This is where ZEISS found itself. Founded in 1846, ZEISS Group is the world leader in scientific optics and optoelectronics, with operations in over 50 countries. In addition to serving consumer markets, ZEISS' divisions serve the industrial quality and research, medical technology, and semiconductor manufacturing industries. The behavior of customers in the consumer markets can be very correlated, resulting in occasional large waves of orders with a lull in activity in between. Because of this, ZEISS' worldwide order processing system can receive as few as zero customer orders at any given minute, and over 10,000 near-simultaneous orders the next minute.

Overprovisioning isn't practical for ZEISS. The logic for an order processing system is far more mundane than, say, a generative AI-based research project. What's more, it's needed only sporadically. In such cases, overprovisioning involves allocating massive clusters of pods, all of which consume valuable resources, while spending more than 90 percent of their existence essentially idle. What ZEISS requires of its digital infrastructure instead are:

- **Worker clusters with much lower profiles**, consuming far less energy while slashing operational costs
- **Behavior management capabilities** that allow for automatic and manual alterations to cloud environments in response to rapidly changing network conditions
- **Planned migration in iterative stages**, enabling the earlier order processing system to be retired on a pre-determined itinerary over time, rather than all at once

"The whole industry is talking about mental load at the moment. One part of my job... is to take care that we do not overload our teams. We do not make huge jumps in implementing stuff. We want our teams to reap the benefits, but without the need to train them again. We want to adapt, to iterate — to improve slightly."

Kai Walter

Distinguished Architect, ZEISS Group

The solution to ZEISS' predicament may come from a source that, just three years ago, would have been deemed unlikely, if not impossible: WebAssembly (WASM). It's designed to run binary, untrusted bytecode on client-side web browsers — originally, pre-compiled JavaScript. In early 2024, open source developers created a framework for Kubernetes called Spin. This framework enables event-driven, serverless microservices to be written in Rust, TypeScript, Python, or TinyGo, and deployed in low-overhead server environments with cold start times measurable only in milliseconds.

Fermyon and Microsoft are principal maintainers of the SpinKube platform. This platform incorporates the Spin framework, along with the containerd-shim-spin component that enables Fermyon and Microsoft are principal maintainers of the SpinKube platform.

This platform incorporates the Spin framework, along with the containerd-shim-spin component that enables WASM workloads to be orchestrated in Kubernetes by way of the runwasi library. Using these components, a WASM bytecode application may be distributed as an artifact rather than a conventional Kubernetes container image. Unlike a container, this artifact is not a self-contained system packaged together with all its dependencies. It's literally just the application compiled into bytecode. After the Spin app is applied to its designated cluster, the Spin operator provisions the app with the foundation, accompanying pods, services, and underlying dependencies that the app needs to function as a container. This way, Spin re-defines the WASM artifact as a native Kubernetes resource.

Once running, the Spin app behaves like a serverless microservice — meaning, it doesn't have to be addressed by its network location just to serve its core function. Yet Spin accomplishes this without the need to add extra overhead to the WASM artifact — for instance, to make it listen for event signals. The shim component takes care of the listening role. Spin adapts the WASM app to function within a Kubernetes pod, so the orchestration process doesn't need to change at all.

For its demonstration, ZEISS developed three Spin apps in WASM: a distributor and two receivers. A distributor app receives order messages from an ingress queue, then two receiver apps process the orders, the first handling simpler orders that would take less time, and the second handling more complex orders. The Fermyon Platform for Kubernetes manages the deployment of WASM artifacts with the Spin framework. The system is literally that simple.

In practice, according to Kai Walter, Distinguished Architect with ZEISS Group, a SpinKube-based demonstration system could process a test data set of 10,000 orders at approximately 60% less cost for Rust and TypeScript sample applications by running them on Ampere-powered Dpds v5 instances on Azure.

Migration without Relocation

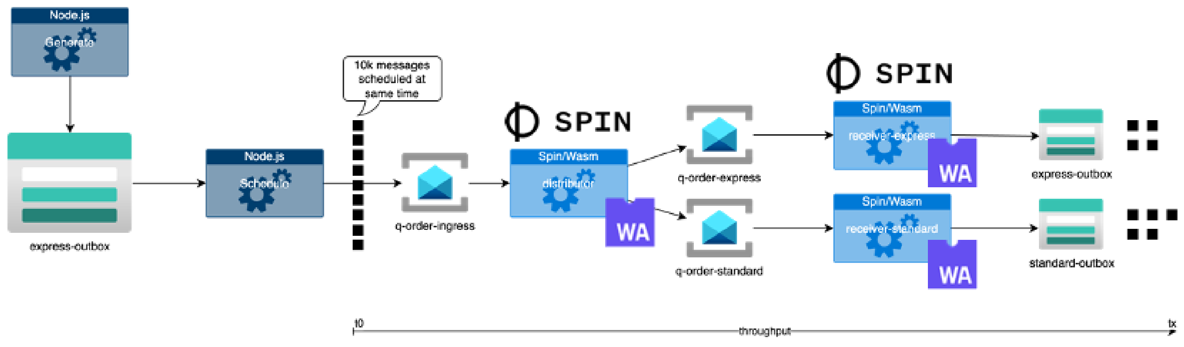
Working with Microsoft and Fermyon, ZEISS developed an iterative migration scheme enabling it to deploy its Spin apps in the same Ampere arm64-based node pools ZEISS was already using for its existing, conventional Kubernetes system. The new Spin apps would then run in parallel with the old apps without having to first create new, separate network paths, and then devise some means of A/B splitting ingress traffic between those paths.

"We would not create a new environment. That was the challenge for the Microsoft and Fermyon team. We expected to reuse our existing Kubernetes cluster and, at the point where we see fit, we will implement this new path in parallel to the old path. The primitives that SpinKube delivered allows for that kind of co-existence. Then we can reuse Arm node pools for logic that was not allowed on Arm chips before."

Kai Walter

Distinguished Architect, ZEISS Group

Generalized Order Processing System



WASM apps use memory, compute power, and system resources much more conservatively.
(Remember, WASM was created for web browsers, which have minimal environments.) As a result, the entire order processing system can run on two of the smallest, least expensive instance classes available in Azure: Standard DS2 (x86) and D2pds v5 (Ampere Altra 64-bit), both with just 2 vCPUs per instance. However, ZEISS discovered in this pilot project that by moving to WASM applications running on SpinKube, it could transparently change the underlying architecture from x86 instances to Ampere-based D2pds instances, reducing costs by approximately60 percent.

SpinKube and Ampere Altra make it feasible for global organizations like ZEISS to stage commodity workloads with high scalability requirements, on dramatically less expensive cloud computing platforms, potentially cutting costs by greater than one-half without impacting performance.

Additional Resources

For an in-depth discussion on ZEISS' collaboration with Ampere, Fermyon, and Microsoft, see this video on Ampere's YouTube channel: How ZEISS Uses SpinKube and Ampere on Azure to Reduce Costs by 60%.

To find more information about optimizing your code on Ampere CPUs,
To find more information about optimizing your code on Ampere CPUs,
check out our tuning guides in the Ampere Developer Center. You can also get updates and links to more insightful content by signing up for Ampere's monthly developer newsletter.

If you have questions or comments about this case study, join the Ampere Developer Community, where you'll find experts in all fields of computing ready to answer them. Also, be sure to subscribe to Ampere Computing's YouTube channel for more developer-focused content.

References

- It's Time to Reboot Software Development by Matt Butcher, CEO, Fermyon
- Introducing Spin 3.0 by Radu Matei and Michelle Dhanani, Fermyon blog
- Building a Serverless Python WebAssembly App with Spin by Matt Butcher, CEO of Fermyon
- Taking Spin for a spin on AKS by Kai Walter, Distinguished Architect, ZEISS Group
- Cloud Native Processors & Efficient Compute — Ampere Developer Summit session featuring Ampere chief evangelist Sean Varley, ScyllaDB CEO Dor Laor, and Fermyon senior software engineer Kate Goldenring, conducted September 26, 2024
- Integrating serverless WebAssembly with SpinKube and cloud services — video featuring Sohan Maheshwar, Lead Developer Advocate, AuthZed

Disclaimer