DEVELOPER STORY

Momento Migrates Object Cache as a Service to Ampere® Altra®

By Dave Neary

SNAPSHOT

Organization - Momento caching infrastructure for cloud applications is complex and time-consuming. Traditional caching solutions require significant effort in replication, fail-over management, backups, restoration, and lifecycle management for upgrades and deployments. This operational burden diverts resources from core business activities and feature development.

Solution - Momento provides a serverless cache solution, utilizing Ampere-based Google Tau T2A instances, that automates resource management and optimization, allowing developers to integrate a fast and reliable cache without worrying about the underlying infrastructure. Based on the Apache Pelikan open-source project, Momento's serverless cache eliminates the need for manual provisioning and operational tasks, offering a reliable API for seamless results.

Key Features -

1. **Serverless Architecture:** No servers to manage, configure, or maintain.
2. **Zero Configuration:** Continuous optimization of infrastructure without manual intervention.
3. **High Performance:** Maintains a service level objective of 2ms round-trip time for cache requests at P99.9, ensuring low tail latencies.
4. **Scalability:** Uses multi-threaded storage nodes and core pinning to handle high loads efficiently.
5. **Additional Services:** Expanded product suite includes pub-sub message buses.

Technical Innovations - Context Switching Optimization: Reduced performance overhead by pinning threads to specific cores and dedicating cores for network I/O, achieving over one million operations per second on a 16-core instance.

Impact - Momento's serverless caching service, powered by Ampere-based Google Tau T2A, accelerates the developer experience, reduces operational burdens, and creates a cost-effective, high-performance system for modern cloud applications.

Background: Who and what is Momento?

Momento is the brainchild of cofounders Khawaja Shams and Daniela Miao. They worked together for several years at AWS as part of the DynamoDB team, before starting Momento in late 2021. The driving principle of the company is that commonly used application infrastructure should be easier than it is today.

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.

All cloud applications use caching in some form or other. A cache is a low-latency store for commonly requested objects, which reduces service time for the most frequently used services. For a website, for example, the home page, images or CSS files served as part of popular webpages, or the most popular items in a web store, might be stored in a cache to ensure faster load times when people request them.

The operationalization of a cache involved managing things like replication, fail-over when a primary node fails, back-ups and restoration after outages, and managing lifecycle for upgrades and deployments. All these things take effort, require knowledge and experience, and take time away from what you want to be doing. As a company, Momento sees it as their responsibility to free their customers from this work, providing a reliable, trusted API that you can use in your applications, so that you can focus on delivering features that generate business value. From the perspective of the Momento team, "provisioning" should not be a word in the vocabulary of its cache users – the end-goal is to have a fast and reliable cache available when you need it, with all the management concerns taken care of for you.

The Deployment: Ease of Portability to Ampere Processor

Initially, Momento's decision to deploy their serverless cache solution on Ampere-powered Google T2A instances was motivated by price/performance advantages, and efficiency.

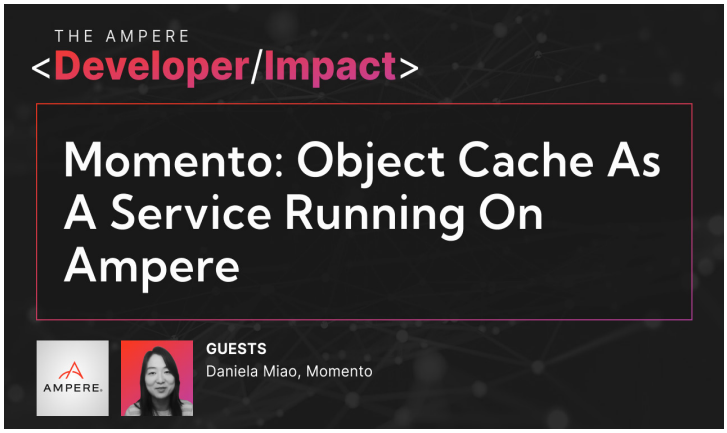
Designed from the ground up, the Ampere-based Tau T2A VMs deliver predictable high performance and linear scalability that enable scale-out applications to be deployed rapidly and outperform existing x86 VMs by over 30%.

However, during a recent interview, Daniela Miao, Momento Co-Founder and CTO, also noted the flexibility offered with the adoption of Ampere as it was not an all-or-nothing proposition: "it's not a one-way door [...] you can run in a mixed mode, if you want to ensure that your application is portable and flexible, you can run some of [your application] in Arm64 and some in x86".

In addition, the migration experience to Ampere CPUs went much more smoothly than the team had initially expected.

"The portability to Ampere-based Tau T2A instances was really amazing – we didn't have to do much, and it just worked"

Checkout the full video interview to hear more from Daniela as she discusses what Momento does, what their customers care about, how working with Ampere has helped them deliver real value to customers as well as some of the optimizations and configuration changes that they made to squeeze maximum performance from their Ampere instances.



Click to view video

The Results: How does Ampere help Momento Deliver a Better Product

Momento closely watches tail latencies – their key metric is P99.9 response time – meaning 99.9% of all cache calls return to the client in that time. Their goal is to maintain a service level objective of 2ms round-trip time for cache requests at P99.9. Why care so much about tail latencies? For something like a cache, loading one web page might generate hundreds of

API requests behind the scenes, which in turn might generate hundreds of cache requests – and if you have a degradation in P99 response time, that can end up affecting almost all your users. As a result, P99.9 can be a more accurate measure of how your average user experiences the service.

"Marc Brooker, who we follow religiously here at Momento, has a great blog post that visualizes the effect of your tail latencies on your users", says Daniela Miao, CTO. "For a lot of the very successful applications and businesses, probably 1% of your requests will affect almost every single one of your users. [...] We really focus on latencies for P three nines (P99.9) for our customers."

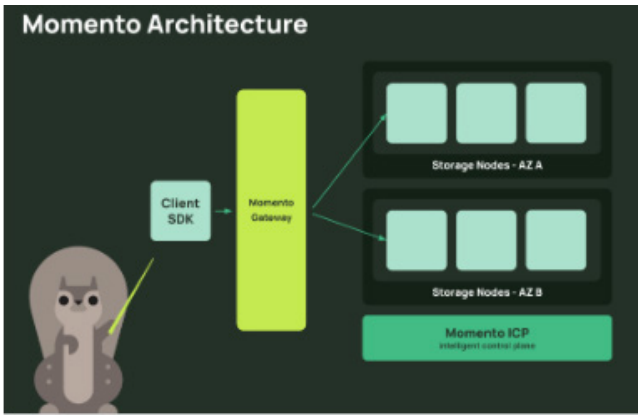
Context Switching Optimization

As part of the optimization process, Momento identified performance overhead due to context switching on certain cores. Context switching occurs when a processor stops executing one task to perform another, and it can be caused by:

1. **System Interrupts**: The kernel interrupts user applications to handle tasks like processing network traffic.
2. **Processor Contention:** Under high load, processes com pete for limited compute time, leading to occasional "swapping out" of tasks.

In Momento's deep-dive into this topic, they explain that context switches are costly because the processor loses productivity while saving the state of one task and loading another. This is like how humans experience a loss of productivity when interrupted by a phone call or meeting while working on a project. It takes time to switch tasks and then additional time to regain focus and become productive again.

By minimizing context switching, Momento enhanced processor efficiency and overall system performance.



Momento's Architecture

Momento's architecture separates API gateway functionality from the data threads on storage nodes. The API gateway routes requests to the optimal storage node, while each storage node has multiple worker threads to handle cache operations.

- **Scalability**: On a 16-core T2A-standard-16 VM, two instances of Pelikan run with 6 threads each.
- **Core Pinning**: Threads are pinned to specific cores to prevent interruptions from other applications as load increases.
- **Network I/O Optimization**: Four RX/TX (receive/transmit) queues are pinned to dedicated cores to avoid context switches caused by kernel interrupts. While it is possible to have more cores process network I/O, they found that with four queue pairs, they were able to drive their Momento cache at 95% load, without network throughput becoming a bottleneck

Getting Started with Momento

Momento focuses on performance, especially tail latencies, and manually curates all client-side SDKs on GitHub to prevent version mismatch issues.

1. **Sign Up**: Visit Momento's website to sign up.
2. **Choose an SDK**: Select a hand-curated SDK for your preferred programming language.
3. **Create a Cache**: Use the simple console interface to create a new cache.
4. **Store/Retrieve Data**: Utilize the set and get functions in the SDK to store and retrieve objects from the cache.

```
const { CacheClient, CredentialProvider } =
require('@gomomento/sdk');

const cacheClient = await CacheClient.create({
    credentialprovider: CredentialProvider.fromString(TOKEN),
    defaultTtlSeconds: 60
});

await cacheClient.set('my-cache', 'foo', 'bar');
```

Additional Resources

To learn more about Momento's experience with Tau T2A instances powered by Ampere CPUs, check out "Turbocharging Pelikan Cache on Google Cloud's latest Arm-based T2A VMs".

To find more information about optimizing your code on Ampere CPUs, checkout our tuning guides in the Ampere Developer Center. You can also get updates and links to more great content like this by signing up to our monthly developer newsletter.

Lastly, if you have questions or comments about this case study, there is an entire community of Ampere users and fans ready to answer at the Ampere Developer community. And be sure to subscribe to our YouTube channel for more developer-focused content in the future.

References

- 2015 StrangeLoop presentation of Pelikan by author Yao Yue: https://www.youtube.com/watch?v=pLRztKYvMLk preferred programming language.
- "Making Pelikan fly on Arm": https://www.gomomento.com/blog/making-pelikan-fly-on-arm-diving-deeper-into-adventures-with-tau-t2a-vmsStore/Retrieve Data: Utilize the set and get functions in the SDK to store and retrieve objects from the cache.
- "Turbo-charging Pelikan cache with Tau T2A VMs on Google Cloud": https://www.gomomento.com/blog/turbocharging-pelikan-cache-on-google-cloud-arm-based-t2a-vm
- "Tail Latencies Might Matter More Thank You Think", by Marc Brooker: https://brooker.co.za/blog/2021/04/19/latency.html
- "Why Tail Latencies Matter": https://www.gomomento.com/blog/why-tail-latencies-matter