Motivations behind TidalScale
Ike Nassi
2022-03-28

TidalScale was formed to address the needs of a broad class of computing problems. The problem was becoming more and more pressing since while computing speed in a server was increasing at an acceptable rate, the amount of addressable memory in those servers was not. As we began to see more and more interest in things like "in memory computing" and other high-appetite applications, we could also peer into the future and see that this problem would become increasingly acute.

We had to solve a very difficult problem: distributed virtual machines. Today we call them "software-defined servers" at TidalScale. The idea was to create a virtual machine to span a set of physical servers. The virtual machine would "see" the aggregation of all the processors, memory, storage, and networks in the combined set of physical servers.

A further constraint we imposed on ourselves was to do it such a way as to not require any changes to existing operating systems or applications. This would greatly increase speed of adoption in the marketplace.

We needed to make sure the distributed nature of the virtual machine would not appreciably slow things down. Since we couldn't predict exactly what software would run, or what data it would compute on, we came up with the idea that we could use an approach based on adaptive and dynamic machine learning through introspection. The machinery could monitor its own behavior in real time, and migrate memory and computing resources so that a large percentage of the time they would be on the same physical hardware. If the access patterns during the job's execution change, as is often the case, we'd migrate the resources to minimize performance degradation in real time. Migration would need to happen on the order of tens of microseconds.

We solved these problems, adding more and more memory, processing capability, I/O, networking, as well as memory and I/O bandwidth to keep things in balance.

But, along the way, we found that increasing the size of distributed virtual machine could have the potential of making the system less reliable. Adding more hardware could mean an increase in the absolute number of hardware errors, which could destabilize the entire virtual machine.

The advantage of the distributed virtual machine is that we have redundancy. We could add or subtract physical servers, thereby placing failing physical hardware in quarantine. This gave rise to our investing in a capability we call TidalGuard™, which allows us to identify 90% of potential hardware failures in advance, and take steps to isolate, replace, or fix these failures without disrupting the work the virtual machine was doing.

By not specifically focusing on a single class of application or operating system, we cover an extremely broad set of use cases, from mission-critical enterprise application using industry standard unmodified software like SAP and Oracle, Red Hat and SUSE, as well as scientific computations like computational genomics. By rigorously focusing only on the lowest level characteristics of the computing environment, we achieved perfect compatibility and the broadest coverage of use cases.

Read more about a specific use case here: https://doi.org/10.1049/smc2.12026