The Wandering CPU & Why TidalScale is Unique

TidalScale

Software-Defined Servers

Contents

1 Introduction

TidalScale is the leading provider of Software-Defined Servers that bring flexibility to modern data centers by right-sizing servers on the fly to handle any workload. Designed to address the growing need for more flexibility and scalability in today's data centers, TidalScale's Software-Defined Servers combine multiple commodity servers into one or more virtual servers—all without a single change to applications or operating systems.

The new WaveRunner release also features significant improvements to TidalScale's HyperKernel software, the foundation of its Software-Defined Server platform. The TidalScale HyperKernel delivers up to twice the CPU scalability and overall performance across the distributed cluster, besting AWS performance on popular applications such as LogicBlox. Meanwhile, more efficient machine learning for thread locality means resources such as cores, memory and storage automatically migrate to where they're needed within a Software-Defined Server. The ingenious wandering CPU allows the TidalPod to achieve optimal performance and deliver results sooner.

In this paper we will examine the advantages of the TidalScale platform by first drawing an analogy and then digging into the platform capabilities that make the TidalScale architecture a computing breakthrough.

2 An Analogy: The Wandering Detective

Imagine a Police Investigator, Detective Walter Swope, who needs to read all of the cold case files at the East Village 9th precinct Police station to find out if any prior crimes in the area match the profile of a current case he's investigating.

He looks a bit like Clint Eastwood, but meaner, and packs a Colt .38 snubnose under his arm.

Figure 1: Detective Swope Going Through Cold Case Files

In the records room, Detective Swope faces a room full of case files sitting on shelves. There is no electronic search but he can scan each case file in a few seconds.

How should Detective Swope go about looking at each and every case?

- Option 1, he could set up a table and bring each folder to the table to read through the cases. The table only holds a limited number of files. We can imagine that he has two assistants, one to retrieve a file and another to put it back.

- Option 2, he could wander the aisles and simply pick up and scan through each folder while standing right next to where it sits on the shelf.

Let's call Option 1 the "Sitting Detective." Let's call Option 2 the "Wandering Detective."

It is intuitively obvious that the most physically efficient way to read these case files is Option 2 because you save the time and energy required to move the folders from the shelf to the table and back. In fact, with Option 2 those two assistants could be used to read more case folders in parallel instead of shuttling back and forth carrying folders to Detective Swope. In this case, with a small order of data, Option 1 is also feasible and would not expend much more energy.

Now, however, imagine that the scale of Detective Swope's problem increases by several orders of magnitude: he now has to search through every cold case file in every police office in the continental United States. Option 1, the Sitting Detective, now consumes absolutely enormous resources just physically moving the cold case files to Detective Swope's desk and thus becomes much more inefficient. As this problem gets larger, Option 2, the Wandering Detective (or detectives), becomes obviously and dramatically more productive.

3 TidalScale: The Wandering CPU

Let's first translate the analogy above into its equivalent in a computer system. The case files in this analogy represent memory, DRAM, and Detective Swope represents a CPU that needs to read that memory.

Surprisingly, modern computer systems work much like Option 1, the Sitting Detective. When there is a lot of data in today's computing architectures, most of the data is far away from the physical CPU running a program. That means that ALL of the data that a CPU needs to touch must be transported physically to be near that CPU in order to be processed. Then, since there is only a little bit of memory storage near the CPU, that data must be pushed away again to make room for the next bit of data to be processed. Caches don't improve this because, even though they are near the CPU, they still hold a relatively small amount of data.

This is a terrible architectural inefficiency. Is it possible to remove that inefficiency? Yes. The alternative is creating a "virtual" CPU that will act like our "Wandering Detective."

Virtualization is technically quite a well-known technique. Memory is virtualized all the time and ships in every working computer system available today. A page of virtual memory is bound to a physical page of memory and accessed, and then unbound from that particular physical page to go elsewhere (whether swapped out to disk or moved to another place in memory). To enable computations to be mobile, TidalScale has applied virtualization to the CPU and enabled it to migrate across physical processors.

TidalScale's key architectural insight is that, beyond a certain size of dataset, it's better to take the state of a running program (i.e. a CPU and its temporary results) and simply move that computational entity to be physically near the data it needs to process next.

The TidalScale virtual CPU (vCPU) is an abstraction of a physical CPU's processing state. A vCPU can be "bound" (attached) to, or "unbound" (detached) from a physical processor. A virtual CPU is not a "program": it is a low level computational abstraction consisting of a set of registers and instruction pointers, etc. A virtual CPU's contents are quite small (basically just the contents of its registers). In modern processors, a vCPU can be stored in no more than a page or two of memory and transmitted in a single raw Ethernet packet. This means vCPUs can move across a multinode system very fast.

A vCPU is free to wander about a computer system's physical CPUs and their caches and RAM much like the Wandering Detective is able to wander about the file storage shelves. Each vCPU in a TidalScale system moves to where it can access the data it needs, wherever that data physically sits. The larger the TidalScale system, the more physical CPUs and RAM are available for its wanderings, and the more efficient the wandering vCPU becomes compared to the other options for processing lots of data.

3.1 How to Process a Lot of Data

Suppose you wanted to add up an array of numbers that occupy P thousands of pages of data - more data than exist on a single memory controller or motherboard. There are currently two standard ways to organize processing of a lot of data either by scaling up or scaling out. The Wandering CPU suggests a third approach. Let's explore these three approaches.

3.1.1 Suffer the Memory Cliff by Scaling Out

One way to add up all the data is to bring it all to a single CPU, where the data is added to an accumulator variable that holds the sum- "scaling up". This will require all of the pages of the array to be brought to the node of the CPU from wherever they are - in memory, on disk, on SSD, etc. So P pages must be moved. However, this design might be slow as a lot of time is spent waiting for pages to be brought from slow storage into memory and then into cache.

3.1.2 Shard and Scale Out

Another way to do it is to break up array P into pieces that can then all fit in the memory of multiple small systems, essentially a lot of physical CPUs with a little data attached to each CPU so that each chunk of data is being processed by a CPU that is sitting next to it. This approach is called "scale out" and it works great if the data is statically partitioned or if its possible to automatically partition and distribute the chunks of the program. This has been called a MIMD architecture since the 1970's. "Map-Reduce" recently adopted the MIMD architecture. However, this design cannot be applied to all operations uniformly.

3.1.3 Use a Software-Defined Server

With a vCPU that can be bound dynamically to the physical CPU that sits close to the memory pages it needs, TidalScale has invented a third architectural option: create a Software-Defined Server and let vCPUs wander to the physical CPUs sitting on the memory container or motherboard that has the data it needs. Since there are thousands of pages on each memory container, the vCPU moves a thousandth of the number of times. On a sufficiently large system, this architecture wins. In fact, this is the insight behind Gustafson's Law.

As a computing problem increases in size, the Wandering CPU gets more and more efficient. Conversely, as a computing problem increases in size, the Sitting CPU gets more and more inefficient. In other words, the wandering TidalScale vCPU pays increasing dividends as a dataset scales. Like the Wandering Detective, the TidalScale vCPU is dramatically more efficient as a dataset grows in size.

4 The Benefits of Systematic Virtualization and Mobilization

TidalScale does more than just virtualize the CPU: TidalScale systematically virtualizes all compute resource types (CPU, Memory, I/O, Storage). This means that the whole compute problem floats over the physical resources in a cloud such that working sets are free to flow, form and reform very dynamically and synergistically. As each vCPU wanders, TidalScale studies its motion and regroups the data it is accessing to shorten the distance that the CPU has to travel. In other words, TidalScale uses machine learning to automatically bring vCPUs to the data they want to see and encourages the appearance and maintenance of locality sets.

For example, a vCPU can move to be physically next to an existing locality set or it can be stabilized in a spot to attract a locality set that needs to be created next to an I/O resource. The many wandering vCPUs can form teams that travel together, follow chains of performance clues and even share physical locality sets.

TidalScale's systemic virtualization allows the virtual computing resources to flow across the physical system, like the ocean tides, to automatically seek out and maintain the physical arrangement that achieves best locality for each individual resource which, in aggregate, delivers the best overall system performance at all times.

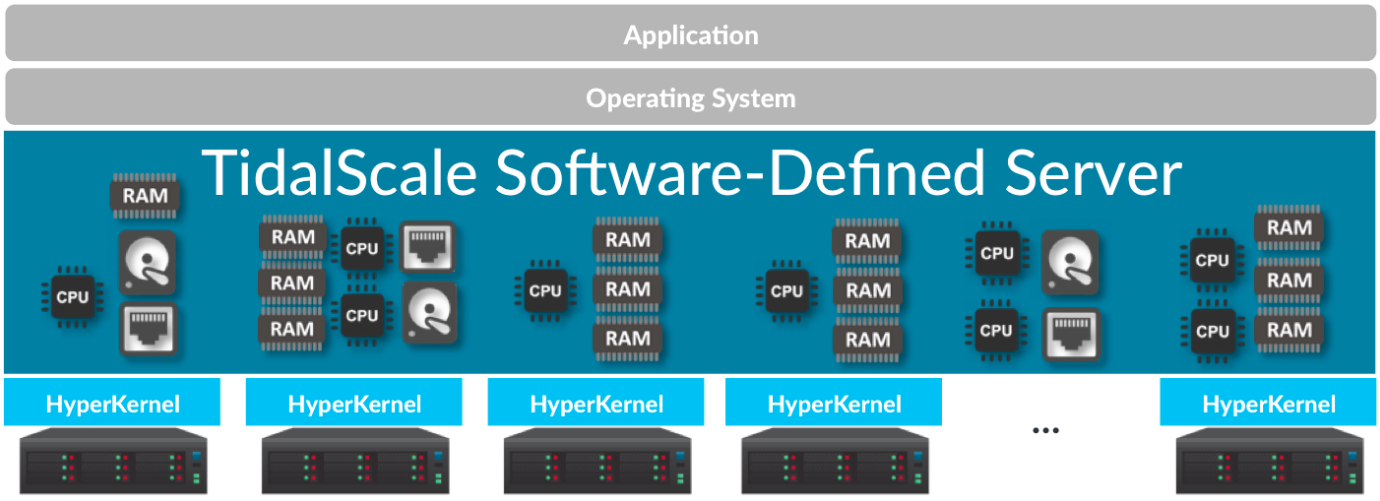


Figure 2: TidalScale Architectual Overview

5 Conclusion

TidalScale uniquely combines virtualization and mobilization to create a large, multinode Software-Defined Server that is completely self tuning. Like the Wandering Detective, the TidalScale Wandering CPU is dramatically more efficient as a dataset grows in size. By virtualizing all computing resources, the TidalScale Hyper Kernel frees them to flow across the system, like the ocean tides, to automatically seek out the optimum physical arrangement that delivers best system performance at all times.

In the words of TidalScale's first Beta customer:

> "This is revolutionary. You will make clusters a thing of the past. At school you are taught to always think of designing around non-locality: slow network, slow disks, etc… [the] Tidalscale platform takes care of all of that."

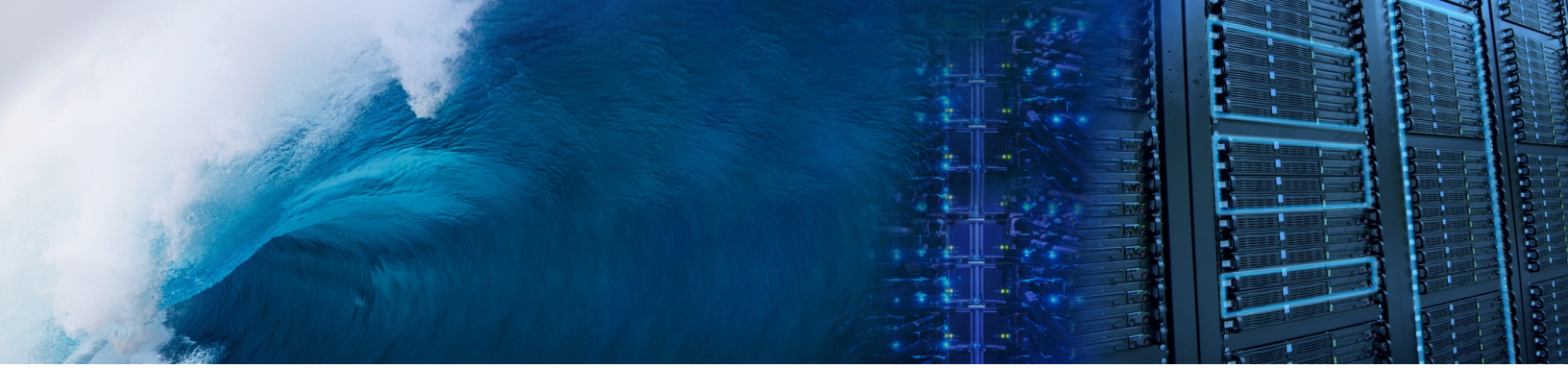
Dr. Jeff Prevost, an assistant professor of electrical and computer engineering at University of Texas San Antonio, which is also using Software-Defined Servers as a reference implementation of a modern cloud data center:

> "The new capabilities in TidalScale's software will help us achieve the flexibility and granular control we need to scale and reprovision resources on demand."

This announcement comes on the heels of TidalScale being named as both an IDC Innovator and a Gartner Cool Vendor for 2017. TidalScale also won the Red Herring Top 100 for North America award for 2017.

TidalScale is the first technology to fully support applications that follow Gustafson's law in the way it delivers scaled speedup for computations on larger and larger datasets. The TidalScale architecture means that when you scale up a problem you **can** use more and more processors. The revolution here is that TidalScale enables customers to do this very cost effectively, effectively scaling their systems to fit the size of their computing problems.

Note
The idea of a movable virtual processor is quite different from a movable virtual machine. Moving a whole virtual machine around is a heavy duty operation that involves moving large chunks of memory to a new physical location. While TidalScale can move data quite efficiently, moving virtual processors is vastly more lightweight (it's a couple of pages) and can be far more effective as the size of an application's data continues to grow. The optimal approach would involve a mixture of strategies and this is exactly what TidalScale does.



TidalScale

Software-Defined Servers

TidalScale, Inc. 1694 Dell Avenue, Campbell CA USA Tel +1-877-399-0680 Fax +1-408-628-0312
www.tidalscale.com. Date: 01-Aug-17