Whitepaper

FUNGIBLE

Fungible's Vision for the Data-Centric Era

This white paper has three main goals: to introduce the trends that underlie Fungible's foundational assumptions; to define a new category of workloads we call data-centric workloads and explain why they are an increasingly important part of modern applications; and to describe data-centric infrastructure—one designed specifically for executing modern applications efficiently.

The Data Centric Era Comes of Age

From the earliest days of computing, there has been ***unrelenting pressure to improve the performance, reliability, and economics of the underlying infrastructure*** while also making it more agile. This pressure is a reflection of the increasing applicability of computers, especially once they were built to be programmable, and then later when they were interconnected in increasing numbers over ever larger geographies. Every order of magnitude improvement in performance and connectivity led to new applications, and these in turn spurred additional improvements.

All information infrastructure can be thought of as a recursive elaboration of three fundamental building blocks: *universal computing engines, random access storage, and any-to-any networks that connect the first two*. We assert that the unrelenting pressure to improve these building blocks is an inevitable consequence of their existence, and *entirely independent of the technology used to implement them*. Another way to say this is that we find the functionality provided by these building blocks to be so compelling that we will always be looking for ways to improve them.

The primary driver of infrastructure improvements over the last five decades has been the exponentially increasing density of CMOS integrated circuits—a phenomenon referred to as *Moore's Law*. Disk drives have historically been the main technology used for secondary storage, but CMOS-based flash memory is slowly but surely also beginning to replace disks. Thus CMOS is the technology driver for all three building blocks.

While the shrinking of transistors with each generation of CMOS has served us well for many years, we are now approaching physical limits that make it difficult to maintain the historical density increases. ***Experts predict smaller and smaller increases with each technology generation and an eventual flat-lining of Moore's Law.*** The first signs of a slowdown in performance were visible in the early 2000s; the response to this was the invention of *multi-core* processors and subsequently of *scale-out* computing to replace the performance increases that had been provided by scaling up individual computers. As Moore's Law slowed down further, it became inevitable that data centers would need to adapt by

About the Author

Pradeep Sindhu
Founder and CEO

Pradeep is the Co-Founder and CEO of Fungible. Fungible is developing fundamental technology that aims to revolutionize the performance, reliability and economics of data centers.

Pradeep founded Juniper Networks in February 1996, where he has held several key roles in shaping the company over the years, including being the first CEO and Chairman, then becoming Vice Chairman and CTO, and now Chief Scientist. Pradeep had a hand in the inception, design, and development of virtually every product Juniper shipped from 1996 through 2015. Prior to that, he worked in the Computer Science Lab at Xerox PARC for 11 years.

Pradeep holds a Bachelors in Electrical Engineering from the Indian Institute of Technology in Kanpur, as well as a Masters in Electrical Engineering from the University of Hawaii. In addition, Pradeep holds both a Masters and a Doctorate in Computer Science from Carnegie Mellon University.

In his spare time, Pradeep enjoys reading and traveling.

building silicon specialized for particular types of workload. Thus the ***scale-out of homogeneous computers built using general purpose microprocessors gave way to scale-out of heterogeneous programmable and fixed-function compute elements***.

Only structures with the strongest foundations stand the test of time...

A key aspect of scale-out infrastructure was the ***co-invention of a microservices based application architecture***. Here each application is broken up into a set of services that are written, maintained, and operated independently of other services. Each application uses a subset of the services provided by a data center, with different applications using different subsets. Additionally, each service is implemented by a set of identical servers in such a way that the number of servers may be scaled up or down as performance needs dictate. An important consequence of a microservices architecture is a ***dramatic increase in network bandwidth between servers in a given data center***.

Workloads Become Increasingly Data-Centric

The last decade and a half has seen an explosion in the amount of data being collected. This began with the realization that running relatively simple algorithms on complete data sets gave significantly better insights than running sophisticated algorithms on a subset of data—a phenomenon now known as "big data". Subsequently, the availability of large data sets combined with advances in machine learning created the need for gathering even more data to feed the learning algorithms. This virtuous cycle is what created the data explosion so clearly visible today. The main consequence of this explosion is the ***need to store, retrieve, move, and process very large data sets, which must necessarily be spread across large numbers of machines***. Most useful computations on such data sets involve moving data over the network interconnecting the servers. An important consequence of this is that the *ratio of IO to arithmetic is substantially higher* than in earlier "compute-centric" models where *arithmetic dominated IO*.

These trends indicate a clear transition from a compute-centric world to a data-centric one. In this new data-centric era, ***moving data and performing lightweight computations on it is at least as important as doing traditional arithmetic and logic operations***.

We can now provide a precise definition for a ***data-centric workload*** by identifying four characteristics, all of which must be present for the workload to qualify as data-centric:

1. All work arrives at a server in the form of ***packets*** on a packet-switched interface such as Ethernet or PCIe. Given the fundamental nature of statistical multiplexing gains provided by packet switching, this characteristic is likely present in most if not all workloads.

2. The packets on a given interface typically belong to a large number of *flows*, with packets from different flows being interleaved arbitrarily in time. Thus ***multiplexing*** amongst flows at the granularity of packets is a central feature of data-centric workloads.

3. The computation performed by data-centric workloads is ***stateful*** in the sense that processing a given packet requires reading and writing state belonging to the packet's flow. Additionally, this state may be large enough that it must be kept in external memory.

4. Finally, the ***IO to arithmetic ratio*** is medium to high. Here, IO refers to any activity that requires traversing the pins of silicon chips executing the workload. Specifically, references to external memory are counted as "IO" for this purpose.

Given the prevalence of scale-out infrastructure imposed by Moore's Law limitations, the pervasive use of a micro-services application architecture, and the desire to process large data sets, we assert that ***data-centric workloads are likely to become an increasingly important component of the overall workload in data centers at all scales***.

So, how do today's data centers cope with data-centric workloads? The answer is not very well! There are two fundamental problems that confront existing data centers.

First, data-centric workloads are executed either by general-purpose CPUs or by appliances[1] such as load balancers and firewalls. Unfortunately, ***the architecture of general-purpose CPUs is fundamentally at odds with the characteristics of data-centric workloads***. The reason for this is that a CPU designer's main imperative is to maximize the instructions per second delivered by each core for a general workload; this is done by operating at as high a clock rate as is permissible by power constraints and by maximizing the instructions per clock (IPC) for each core. Achieving a high IPC though is *fundamentally at odds* with operating efficiently on a fine-grain multiplexed workload. The techniques used to achieve high IPC—caching of code and data, deep pipelines, register renaming, and branch prediction, to name just a few—are all detrimental to frequent context switching. What is worse is that general-purpose CPUs cannot execute compute-centric workloads efficiently in the presence of data-centric workloads either. This is because the multiplexing caused by a data-centric workload destroys the performance of compute-centric workloads. This interference effect is maximum if a given core is being used to run both types of workloads, but it is also present when the workloads are run on separate cores because cores still share the memory system.

The second problem confronting existing data centers is that modern workloads place extreme pressure on the packet-switched network interconnecting servers in a building. The pressure to increase performance and simultaneously decrease cost and power per bit forces network elements to be as simple as possible. In particular, ***network elements cannot afford to do stateful computations at the speeds they are required to operate***. They can either operate at speed with a small amount of state or operate slowly with the amount of state required by aggregation, but not both! The particular stateful computation that is critical to all workloads is some form of session termination, which not only provides end-to-end reliability, flow-control, and security but also permits the network to be utilized efficiently. TCP, RDMA, and RPC are examples of protocols that require such termination. Today there is no efficient and scalable way to do this.

Introducing the Data Processing Unit (DPU)

We believe that a data-centric world demands data-centric infrastructure. Executing data-centric workloads on infrastructure designed for an older compute-centric world results in economic inefficiency, unreliability, and insecurity —the exact opposite of the demands users are making on computational infrastructure! Data center operators have to resort to frequent, careful optimizations to achieve reasonable levels of performance. These optimizations are frequently fragile in the face of small changes and often become a source of security problems.

Our view is that data-centric infrastructure requires a new kind of microprocessor called the ***Data Processing Unit***, or DPU, that Fungible has invented. Unlike the CPU which is designed to provide high IPC for general-purpose applications, or the GPU which is designed to perform floating-point vector operations, the DPU focuses exclusively on executing data-centric workloads efficiently.

In particular, the DPU is designed from the outset to do two things well, *and to do them in a programmable way*:

1. ***The DPU performs data-centric computations close to an order of magnitude more efficiently than general-purpose CPUs***. It does this by using three independent techniques: creating an architecture specialized for handling heavily multiplexed workloads efficiently; designing specialized accelerators for the most frequently executed operations; and by providing an efficient model for handling flows of data while performing lightweight computations on them. The DPU provides standard IP over Ethernet (IPoE) interfaces on the network side and standard PCIe interfaces on the local bus side to connect compute or storage elements. It is easy to see that the DPU is therefore ***complementary to existing building blocks***, not competitive.

2. ***The DPU serves as the end-point of stateful network connections at large scale***. In doing so, it converts a standard IPoE network into a fabric that provides full any-to-any cross-section bandwidth, fine-grain end-to-end flow control, end-to-end error control, end-to-end security, low latency, as well as tight control over tail-latency. Most importantly, these properties can be provided across a single flat

[1] These appliances are typically built using general-purpose CPUs enhanced with accelerators.

network built with standard IPoE switches spanning small deployments of 10s of KW all the way to a massive deployment of 40MW containing over 100,000 physical servers in a single building.

A ***data-centric infrastructure*** then, is one in which ***compute and storage servers contain DPUs to perform data-centric computations efficiently and to connect the servers to a standard IP over Ethernet network in a way that enables resources to be disaggregated and pooled***. In most configurations, a single-tier (Spine alone) or two-tier network of Top-of-Rack (TOR) and Spine switches is largely sufficient, especially if the Spines are permitted to have high radix.

For modern applications, we expect ***DPU-enabled infrastructure to significantly increase performance while reducing cost and power consumption when compared with existing compute-centric infrastructure***. The higher the proportion of data-centric computations within the overall workload, the greater we expect the gains to be. It is important to note that the gains could be taken in the form of increased performance, decreased infrastructure cost (both CAPEX and OPEX), or a combination of the two.

We note that our data-centric architecture also provides the capability to *disaggregate*, *pool*, and *virtualize* data center resources for which device access latency is large[2] compared with fabric latency. Resources that fall into this class are persistent storage (HDDs, SSDs, phase-change memory), and streaming accelerators like GPUs, ML engines, and Inference engines which operate on large streams of data without requiring fine-grain interaction.

The benefits of resource pooling can be quite significant. The ***resource pooling principle*** was first quantified by Peter J. Denning in his PhD thesis. Denning's result showed that by ***combining N separate pools of a resource of 1 unit each into a single pool allows us to provide the same service level with just √N units of the resource instead of N units***. For example, if we have 36 servers in a rack, each with 4TB of storage (for a total of 144TB), the pooling principle argues that we can provide the same level of service with a single pool of 24TB of storage—a reduction of 6X in the total amount of storage needed. Pooling resources across multiple racks would provide even greater levels of reduction.

In contrast, today's compute-centric infrastructure is decidedly *anti*-pooling. This is because resources tied directly to the PCIe bus of a general-purpose processor cannot be used efficiently by processors in other servers. As a result the resources remain stranded and therefore poorly utilized.

Data-Centric Computing is the Next Frontier

We believe that within the next five years the ***vast majority of data centers will be converted from compute-centric to data-centric, and the DPU will play a central role in this transformation***.

In the coming months, Fungible will add depth and breadth to this important transformation in computing.

[2] A good rule of thumb is resource latency is > 10x fabric latency.

FUNGIBLE

Fungible, Inc.
3201 Scott Blvd.
Santa Clara, CA 95054
669-292-5522