IT'S TIME FOR NETWORK ATTACHED MEMORY

pivotnine

SPONSORED BY: MemVerge

Author: Justin Warren

May 2021

WP

NETWORK ATTACHED MEMORY

Recent market developments lead us to believe that we are on the cusp of the creation of a massive new infrastructure market that we are calling Network Attached Memory.

Network switches and routers were once just servers, and then the demands of networking became too large to remain contained within the tight confines of server infrastructure. They broke out into what is now a massive market with its own massive sub-markets like wireless networking.

Dedicated SAN and NAS devices were once just servers with internal disks. Again, the demands on storage became too large to be satisfied by the relatively simple matter of placing disks in servers, and another massive dedicated market came to life.

We see the same dynamics playing out with memory. AI/ML, analytics, graphic processing, and several other memory-intensive workloads are pushing the limits of current memory infrastructure, and new infrastructure techniques have arisen that, we believe, will combine to create an entirely new infrastructure market: Network Attached Memory.

Why SAN and NAS Exist

The benefits of externally attached storage comes from their centralized, pooled and shared nature. Without a Storage Area Network (SANs) or Network Attached Storage (NAS), storage has to live inside the server you want to use. Direct Attached Storage (DAS) has the advantage of being simple. It's just a disk (or these days a bunch of flash) inside your laptop that you can use.

And that's fine, until you need to add more, or want to replace a disk that's worn out. You can't use your laptop while it's in pieces and you're replacing the drive with something bigger. An external disk array helps, in that you can use fancy RAID software to keep things running while you swap out a disk, but notice the cable connecting your laptop to the disk array? What if you could connect lots of computers to a big pool of storage that you manage centrally?

And that's what SAN and NAS fundamentally is. A SAN provides a pooled set of block-addressable disk, just like DAS, but in an external array that can connect to multiple computers over a network. A NAS is a SAN with a filesystem so you can use a higher-level abstraction (files) to access your storage.

The central pooling can be a lot more efficient than hundreds of servers with internal disks that are mostly empty. Pooling all that empty space means it's now usable by other servers. Backing up data can be done in bulk from the storage, often over a dedicated network connection to tape, or another storage array. You can replicate the storage to protect against disasters without placing additional load on the front-end servers.

Cloning datasets can be done within the array instead of copying data around at much slower speeds. Deleting copies when you're done is fast and they get instantly returned to the pool so the storage is ready to be used for other things. Modularity and separation of concerns means you can divide and conquer the work instead of requiring admins to be experts in everything.

And all of this can be automated with software. Provisioning storage to a new system doesn't require physically installing disk drives into a server. Servers can receive an allocation of storage automatically when they appear within a given network segment, for example. Rather than dealing with the physical reality, we can instead deal with software abstractions that decouple the processes of physical storage from the logical use of storage. Software defined infrastructure has been extremely beneficial.

There are lots of very good reasons that SAN and NAS remains a popular and useful design pattern, though it's not the only way to achieve this.

Detour via HCI

Hyper-converged infrastructure (HCI) burst forth into the public consciousness mostly due to the efforts of Nutanix, which brought the ideas developed inside Google out into the wider world. Its central idea was that, instead of using big, expensive and usually proprietary storage arrays, why not take your storage unit up a level of abstraction from the disks themselves to relatively cheap, disposable servers with disks in them, running Linux?

At the scale of Google, something is constantly failing and needs to be replaced. What if you pooled more than just disk together, and pooled servers together and shared all of their resources? Using virtualization and clustering, that's essentially what HCI did and, I would argue, is part of Kubernetes' appeal.

By communicating over a network as a multi-dimensional pool of resources, the server becomes the unit of scale. Rather than scaling up a central core to become every bigger and more powerful, the distributed nature of HCI scales out in a swarm-like fashion. This scale-out choice suits highly parallel workloads best, where the total work can be split into sub-units that can be farmed out to the nodes in the wider cluster for processing on their local resources. It's not very good for workloads that genuinely require a lot of centralised power to work on one thread really quickly.

This is an oversimplification, but you can see that there were good reasons for adopting HCI at the time.

However, as time moved on, the limitations of HCI became clearer. HCI clusters scaled best in uniform increments of the chosen server dimensions of CPU, memory, and storage. If scale was required in only one dimension rather than all three, the other two quantities would still be purchased only to sit idle. Capacity management and tuning became a complex multi-dimensional non-polynomial planning exercise, and required a certain minimum scale to be cost-effective.

WP

Flexibility to Scale

It turns out that having the flexibility to scale CPU, memory, and storage independently of each other is quite valuable. It's particularly valuable in a field that changes as rapidly as information technology where depreciation schedules and hardware replacement cycles are years longer than market dynamics. Spending big on a choice that's hard to change later magnifies risk, rather than assisting with responsiveness.

But scaling memory remains a challenge. Memory access is still mediated by CPUs over communications buses that live within a server. If we exceed the amount of DRAM we can stuff into a server, we're forced to add another server, and some kind of clustering mechanism and server-to-server networking to move data around.

Persistent memory helps, in that we can add vastly more memory into a server (and much cheaper than the equivalent DRAM) than using DRAM DIMMs, but we're still limiting access to this memory to processes that run within the server itself. We can't access this memory from CPUs (or GPUs, or TPUs) that are physically in other servers. We can only scale up.

And we can't pool our resources to make memory a dynamic quantity, able to be accessed dynamically as and when different workloads require it. We are stuck using whole servers (though they are usually virtual machines) or containers deployed into a Kubernetes cluster that can only make use of part of the memory that is physically installed in the node our container is deployed to run on.

But what if we could use the same design pattern as a SAN or NAS, but for memory? What if we could use Network Attached Memory[1] ?

The Rise of CXL

Compute Express Link[2] is an open standard interconnection for high-speed CPU-to-device and CPU-to-memory based on the PCI Express (PCIe) physical and electrical interface. Using CXL (or competing standards like NVIDIA's NVLink[3], or Gen-Z[4]) we could decouple compute from memory and allow physical compute devices and memory devices to be pooled and managed in bulk, much as we do now with SANs.

The CXL2.0 specification released in 2020 added support for single-level switching, showing a direct parallel to the development of network and storage switching devices. Switching allows for the creation of a network layer between host nodes with CPUs and remote devices such as GPUs and memory, including persistent memory.

It is not difficult to predict the likely trajectory of this market, based on what we know of the SAN and NAS markets that came before. We can easily predict many of the core customer needs that will arise. But there are also some marked differences that are important to note.

While there are multiple alternative standards, there isn't the same degree of ruthless competition we've seen in previous markets. The CXL consortium is supported by a wide array of industry vendors, including all of the major and powerful vendors such as Intel, AMD, NVIDIA, Arm, Cisco, Broadcom, Huawei, Xilinx and more. No major vendor appears to have chosen to stand apart.

Importantly, in our discussions with vendors working on CXL, there is an acknowledgement that customers would not be well-served by a drawn-out battle for

[1] Or perhaps a Memory Area Network but we prefer the sound of Network Attached Memory.

[2] https://en.wikipedia.org/wiki/Compute_Express_Link

[3] https://blogs.nvidia.com/blog/2014/11/14/what-is-nvlink/

[4] https://en.wikipedia.org/wiki/Gen-Z

interconnect supremacy. Vendors seem to realize that a strong standard creates a much larger market for them to compete for share in, rather than fighting over larger pieces of a smaller pie. Indeed, capturing the whole market for Banyan Vines or Betamax is hardly the path to fortune and glory.

CXL looks poised to create a massive new market with plenty of easily predictable opportunities for smart players. We will need physical infrastructure of CXL connected devices to build the new memory network with. We will need software to pool memory and CPUs together, protect ongoing work from failures, and aid operators to manage the infrastructure itself. We will need software to abstract the physical infrastructure so that it can be orchestrated with APIs.

We will need management software to help us determine how workloads can access resources when they need to, and security software to protect resources from unwanted access. We will need software to manage pools of memory so that we can have snapshots and clones and backups and all our familiar data services.

Market Adoption

The early adopters of Memory Area Networks will be those customers with large memory footprints and numerous workloads that require large memory pools. At these scales, significant savings will be achievable through consolidation and the resultant increase in resource utilization. While these savings will likely be substantial enough to justify the purchase of additional CXL-enabled devices on their own, there are also significant productivity gains to be had.

A large central pool of basic resources that can be efficiently shared between transient workloads using software-defined-infrastructure techniques has already proven extremely useful, first with SAN and NAS and then later with cloud computing. The major cloud providers in particular will see major advantages to being able to share previously exclusive-access resources like memory in the way they are able to share compute and storage resources. It is not difficult to imagine an Elastic Memory Service being built on top of CXL (or a similar interconnect) in order to provide Memory-as-a-Service to compute instances.

WP

The same architecture makes sense for non-cloud customers who invest in large amounts of infrastructure themselves but who want to operate that infrastructure in a cloud-like manner. Cloud is not so much a location as a way of operating infrastructure. A central IT group that manages shared infrastructure for an entire enterprise, carving out slices of that infrastructure for business units via software APIs, functions very much like a public cloud provider. Network Attached Memory makes sense for both.

But the infrastructure and interconnect, while necessary, is not sufficient for this new major market to develop. We need to put the Software in -as-a-Service.

Software Control Required

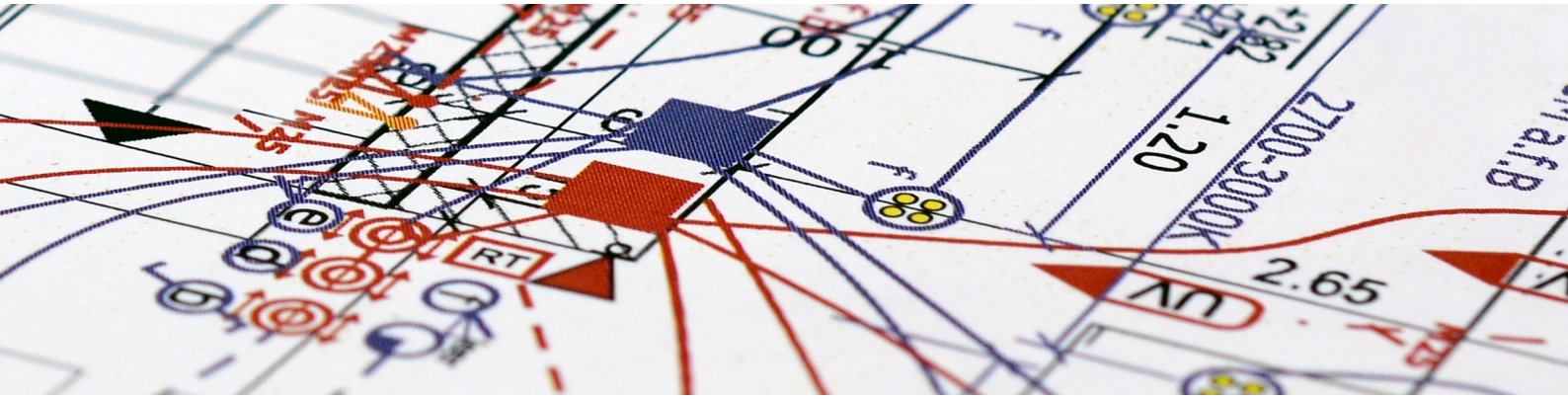
In order to make use of the physical infrastructure of Network Attached Memory, software will be required to make the memory accessible to programs. Existing software has not been written with the assumption that it will need to manage its own memory access over a network, just as most software doesn't automatically assume that data storage exists remotely at the end of a Fiber Channel network connection.

Instead, these services are provided by the operating system or hypervisor which provide the illusion of local infrastructure to programs. This illusion simplifies what developers need to take care of with their own programs. Instead of needing to write storage and network protocol code themselves, developers rely on modules elsewhere in the technology stack to take care of those details, and present to them common structures via well-known APIs. Developers then use those APIs instead of handling everything themselves.

Memory management has already operated this way for many decades. Virtual memory is the standard mechanism in all modern operating systems, and direct manipulation of memory regions is rare. The operating system is assumed to know what it is doing, and individual programs simply need to care about the amount of memory they need to access.

While it's prudent for developers to understand something about how the layers they rely on function, on a daily basis this knowledge sits in the background while developers deal with more abstract data structures like arrays and lists and dictionaries and hashmaps. For many languages, memory management happens within the interpreter or compiler and even freeing up memory after it is no longer in use isn't something the developer needs to concern themselves with (unless something starts to go very wrong).

It's clear that something will need to manage Network Attached Memory on behalf of these user programs. The question thus becomes: What will that something be?



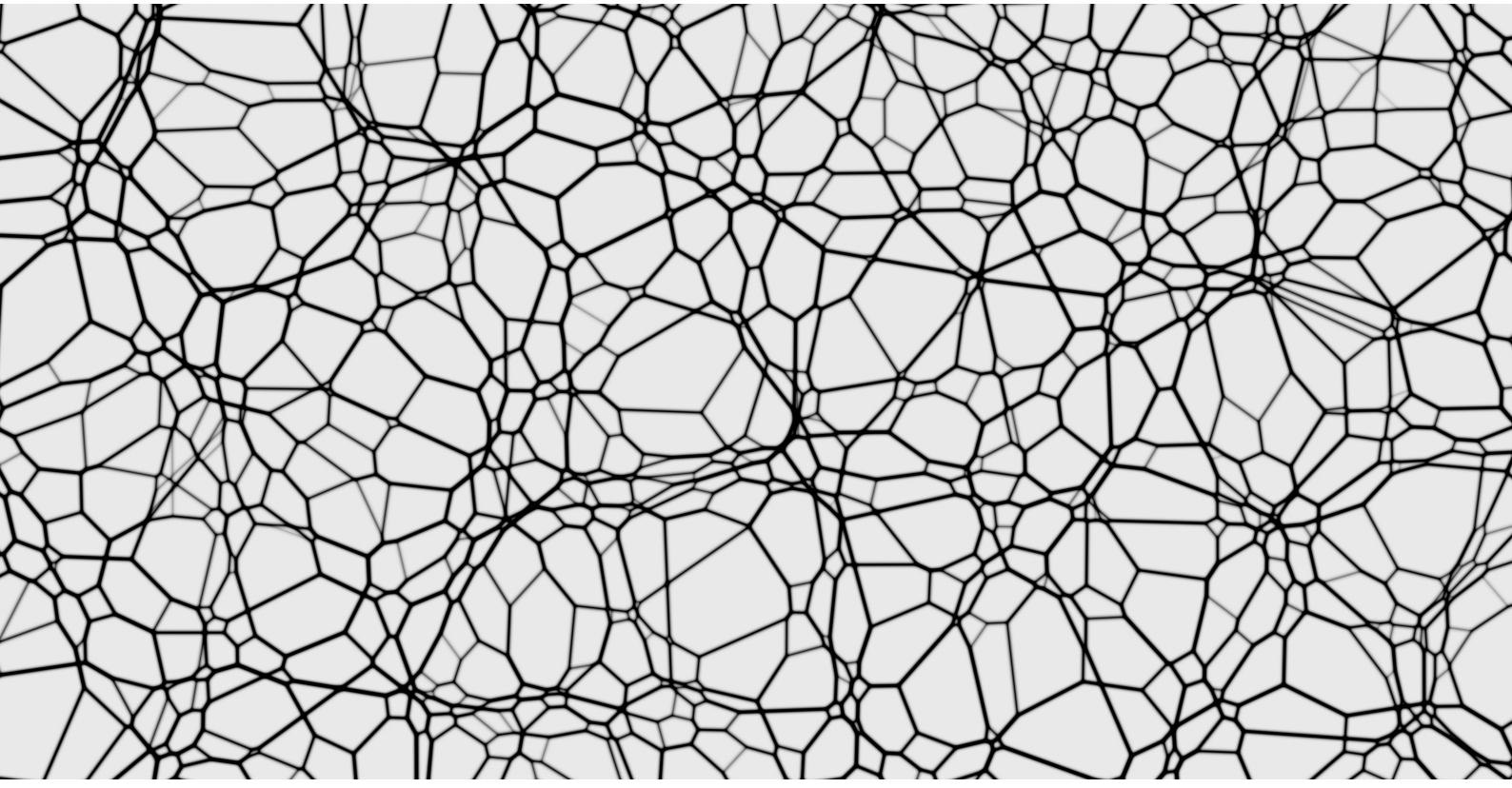
Conclusion

MemVerge is well positioned to become the memory hypervisor for Network Attached Memory. It already provides an abstraction layer between new memory concepts, like persistent memory, to the vast quantity of software that already exists and likely won't get rewritten. MemVerge already provides data services and some operational control over memory, required to build Memory-as-a-Service systems.

MemVerge also supports multi-server memory operations like teleporting a process from one server to another. This provides a solid foundation for clustered resource scheduling where processes can be moved around physical compute infrastructure without losing state. Kubernetes is useful for stateless workload scheduling, but stateful workload scheduling could provide substantial performance improvements, particularly for in-memory data intensive workloads like machine-learning and analytics.

We think the prospect of Memory Area Networks is inevitable, and we're seeing the required infrastructure components develop, and the associated standards are solidifying nicely. It's still very early in the market development, but there are encouraging signs that vendors will quickly settle on CXL as an interoperable standard. Customers are thus less likely to delay adoption through fear they may be stranded on a failed standard, as has happened in other markets before. Rapid customer adoption and widespread vendor support should create a virtuous cycle, leading to an explosion in market size and activity.

Who says storage is boring?



PN

About The Author

Justin Warren is Founder and Chief Analyst of PivotNine. He has worked with many well-known companies around the world, including ANZ, Australia Post, IBM, NetApp, Nutanix, Pure Storage, Red Hat, Suncorp, Telstra, and VMware as well as a variety of startups including Atomist, CodeSee, Cloudistics, Datera, Elastifile, Env0, Habrdata, Hasura, Illumio, Isovalent, Manifold, Mattermost, Smallstep, Spectro Cloud, and Solo.io, among others.

Justin has written for a variety of mastheads, including *Forbes.com*, *iTnews*, *CRN Australia* and *The Saturday Paper*.

Justin holds an MBA from Melbourne Business School, and is a graduate member of the Australian Institute of Company Directors.

About PivotNine

PivotNine Pty Ltd is a specialist IT consulting firm based in Melbourne, Australia.

PivotNine helps customers to evaluate and select technology products, and to implement effective organisational structures and processes.

PivotNine assists vendors with marketing positioning and messaging, with a focus on data driven marketing methods.

Contact

enquiries@pivotnine.com

https://pivotnine.com