



Computational Acceleration: Offense in Depth

Mark Campbell, EVOTEK

As budgets tighten, customer demands increase, and the historic computational growth trends flatten, doing more with what you have will become the hallmark of an elite enterprise IT organization. Operating system, environment, application, and operations acceleration are emerging as the key enablers to span this compute gap.

For the past several decades, companies have been driven by a steady customer demand for increased digital engagement. To feed this insatiable hunger, underlying applications have grown in complexity, functionality, and interdependence to fuel a geometrically increasing need for more compute horsepower. Moore's law had provided a ready answer, with new compute platforms

rolling onto the market with double the horsepower every few years. Companies kept pace with growing application performance demand by embracing a three- to five-year "tech refresh" cycle that allowed them to maintain modest budget and infrastructure growth. However, with the long-heralded end of Moore's law looming,¹ periodic tech refreshes to newer, faster systems will no longer keep pace with growing application demand, so enterprises must find ways to fill the compute gap.²

TRADITIONAL APPROACHES

Companies tend to evaluate a combination of five alternatives to close the compute gap.

1. *Throw hardware at it:* The default solution is to increase compute capacity by turning up more cloud or on-premises compute capacity. However, compared to past decades where compute power doubles for the same budget every several years, we increasingly see a linear effect today where doubling the compute requires doubling the cost. So, embarking on a hardware scaling race provides a short-term fix, not a long-term solution.
2. *Rewrite applications:* Companies can rewrite their applications in more efficient language, use faster

algorithms, update or replace external dependencies, and optimize software designs. However, this is costly, time-consuming, error-prone, and likely relies on hard-to-find engineering skills. Companies that can surmount these impediments prefer to focus their scarce software engineering resources

4. *Rehost applications:* Some port their application to higher performance compute architectures like GPUs or task-specific hardware like tensor processing units. Some companies invest in custom hardware for special applications (for example, MapReduce and Inference). However, not all applications benefit

As application features, user behavior, and interaction with other applications change, the drift is quantified and parameters adjusted to keep things running smoothly.

on releasing new features instead of sprucing up existing services—thus perpetuating the compute gap cycle.

3. *Re-architect infrastructure:* Companies can examine their enterprise architecture for redundant components, design inefficiencies, and simplifications to boost the effectiveness of their existing servers, networks, and storage infrastructure. While this was effective a decade ago, most modern architectures are built on industry best practices and frameworks, so this typically yields only modest improvements.
5. *Manually tune parameters:* Others employ system and software engineers to analyze underlying compute systems and experimentally tune them for increased performance. However, this is a manual needle hunt in a haystack of tens of thousands of tunable parameter combinatorics. Any modest improvements they may find are often transitory as application demands change with evolving customer and system behavior.

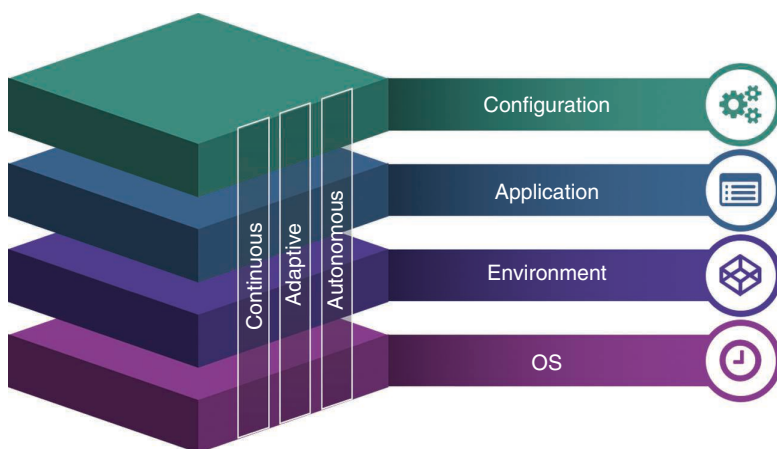


FIGURE 1. The four performance optimization layers.

EMERGING APPROACHES

All is not lost, however. Four layers of compute acceleration solutions—1) configuration, 2) application, 3) environment, and 4) operating system (OS)—are startlingly efficient without requiring new hardware, re-architecting, recoding, or manual tweaking. These solutions share three common attributes.

- *Autonomous:* As mentioned, traditional approaches often flounder because of the colossal human legwork and specialized skills required. New solutions automate manual tasks through programmatic tools and techniques that operate without human intervention.
- *Continuous:* Many direct approaches look to optimize performance on a given application version and retune on each subsequent release. However, in today's continuous integration/continuous deployment DevOps world, application versions seldom linger very long. New functionality, bug fixes, updated models, and new interdependencies are constantly rolling out and require a commensurately continuous performance optimization process to keep pace.
- *Adaptive:* Once applications are deployed, how they're consumed by customers, end users, and other systems changes almost immediately. These usage variances directly affect an application's computational consumption. New solutions leverage artificial intelligence (AI) to learn "normal" behavior, detect usage drift, and make adaptive performance adjustments.

Today, four distinct performance optimization layers (operations, application, environment, and OS) deliver these three common attributes (see Figure 1).

Operations acceleration

Operations acceleration (also known as configuration acceleration or software acceleration) uses AI to observe running applications, learn their behavior, and tune accelerators (for example, memory allocators), libraries, and configuration settings to drive optimal systemic performance. Operations acceleration is easy to implement, verify, and deploy because it uses changes outside the application that don't require any recoding or re-implementing. Once deployed, the operations acceleration agent continuously monitors the application. As application features, user behavior, and interaction with other applications change, the drift is quantified and parameters adjusted to keep things running smoothly.

One example of an operations acceleration product is Concertio. Its AI-based solution samples runtime data, classifies this into characteristics, predicts the impact of a parameter adjustment, selects the optimal correction, applies the change, and repeats the process. Concertio optimizes configuration settings throughout the stack, such as Linux's sysctl settings, processor settings, network settings, and application settings. Concertio also uses runtime features such as Linux's LD_PRELOAD to overload and replace memory allocators (for example, jemalloc, tcmalloc, and mimalloc) without changing the application binary. Other incremental performance tricks such as batching and caching syscalls to reduce user space to kernel space transition overhead, adjusting language specific setting such as Java and Go garbage collection parameters, right-sizing Kubernetes cluster pods, optimizing interprocess communication via shared memory, database parameter tuning, and exhaustive compiler flag mining allow Concertio to cumulatively deliver significant performance improvements.³

Application acceleration

Application performance management (APM) has been a mainstream

method for many enterprises since the late 1990s. APM has evolved from a bolt-on performance monitor for the Internet's first three-tier applications to a distributed, containerized, hybrid cloud service platform. However, in the past few years, the space has been modified again from a reactive, rules-based tool suite into an autonomous, adaptive, and proactive watchdog of the development pipe-

"Our ultimate goal is to scrutinize the next line of code the developer writes and give them immediate feedback on its performance impact," touts Andreas Grabner, self-described DevOps activist at Dynatrace.⁴

Another notable addition to this space is the Keptn open source initiative that APM solutions like Dynatrace use to automate remediation of performance, scalability, and availability

APM has evolved from a bolt-on performance monitor for the Internet's first three-tier applications to a distributed, containerized, hybrid cloud service platform.

line and deployed application stack. This next generation of APM tools not only alerts operators and developers to performance issues in running applications; it also can determine root causes and autonomously trigger actions that adapt the running software to remediate underlying issues rather than just symptoms.

Dynatrace is one shining example of application acceleration delivered by next-generation APM. Agents are installed on hosts, virtual machines, and Kubernetes nodes and pods or injected into serverless functions. They monitor and report user behavior, logs, events, and system telemetry trends back to the APM, giving a holistic view of the entire application stack. For key processes requiring greater granularity than system agents can provide, Dynatrace can inject on-the-fly instrumentation at the process level that allows distributed transaction round-trip tracing.

Dynatrace not only reactively reports on deployed applications; it's also involved in the initial steps of the development pipeline. As the coder commits newly written code, the APM notifies them of potentially adverse performance, resilience, stability, and business impacts, asks the coder if this is intentional, and highlights the problematic code or configuration change.

Keptn acts as a central control plane that consumes simple, declarative specifications to generate automation flows. Although primarily focused on DevOps automation, Keptn can be leveraged by APMs to perform autonomous application performance remediations.

Environment acceleration


The newest of the acceleration layers, environment acceleration, squeezes performance out of the cluster, container, virtual machine, or function-as-a-service environment level. The most common environment acceleration targets today are Kubernetes cluster and pod acceleration because of the rapid adoption and proliferation of containers and microservices. Kubernetes clusters are notoriously inefficient for container startup and underlying compute infrastructure allocation, resulting in a great opportunity for tuning and acceleration.

One example of an emerging environment acceleration solution is Kontain. "Programming languages and frameworks like Kubernetes make the life of a software developer much easier, but the price we pay is low compute efficiency and therefore higher costs," observes Mark Davis, CEO and founder of Kontain. The Kontain solution

provides container isolation inside Kubernetes pods for efficient multitasking without the typical, and inefficient, practice of standing up new pods for each tenant.

Beyond the efficiencies offered by container isolation, Kontain also “prewarms” high demand microservices by loading them into memory, initializing them, and readying them

general access and control services, making it difficult to effectively specialize the OS. Likewise, GPU-based applications leverage specialized hardware optimized for a particular type of operation (for example, GPUs and vector mathematics) so an optimized OS is minimally effective. Beyond these edge examples, however, most applications experience dramatic speedup from OS

layers of computational acceleration, but targeting just one tier can double the transactional throughput of a given compute platform. As budgets tighten, customer demands increase, and historic computational growth trends flatten, doing more with what you have will become the hallmark of an elite enterprise IT organization. OSs, environments, applications, and operations acceleration are emerging as the key enablers to span this compute gap. 

Keptn acts as a central control plane that consumes simple, declarative specifications to generate automation flows.

to run on request. This results in single-digit millisecond instantiation times between service request arrival and the execution of the first payload instruction. Kontain can dynamically determine where in the function stack initialization is complete, while it also allows developers to narrow this to an exact line of code if needed.⁵

OS acceleration

OSs, by design, are general-purpose resource and process schedulers capable of handling every type of application—from critical, real-time systems to mathematically complex number crunchers to wide-ranging user interaction systems. Yet each application that runs on an OS is designed for a specific task using a specific set of resources and processes. It’s analogous to using a Swiss Army knife to tighten a screw versus an electric screwdriver. In typical systems, processes and threads spend almost all their time waiting on locks, resources, data, and other threads. OS acceleration tools autonomously adjust the thousands of parameters that control how the kernel manages memory, processes, threads, locks (especially mutex locks), swapping, and interprocess communications to find the optimal combination for the specific application.

Not all applications and hardware platforms benefit from OS acceleration. Databases, for example, provide

acceleration, especially systems hosting a single specialized application.

One prominent OS acceleration tool is Granulate. It inserts a new layer into the Linux kernel that continuously and autonomously optimizes memory access, process swapping, thread scheduling, storage access, network communications, and mutex management. Because languages impart characteristics on applications as they are written, compiled, and linked, Granulate provides specialized kernel modules for the most common programming languages and frameworks, such as Java, Go, Node, Presto, and Spark, that deliver immediate performance enhancement. From there, Granulate watches the running applications and learns their idiosyncrasies. After establishing this performance baseline, it begins slewing kernel parameters toward the optimal. After tuning, Granulate continues to monitor the system, adjusting its baseline and kernel parameters continuously as application features and usage patterns drift over time. This optimization can be applied in or across OS instances, segmented into multitenant partitions and span Kubernetes pods.⁶

Applications stacks may reach diminishing performance returns by implementing all four

REFERENCES

1. D. Rotman, “We’re not prepared for the end of Moore’s Law,” *MIT Technol. Rev.*, Feb. 24, 2020. [Online]. Available: <https://www.technologyreview.com/2020/02/24/905789/were-not-prepared-for-the-end-of-moores-law/>
2. D. J. Barry, “As Moore’s Law ends, hardware acceleration takes center stage,” *RCR Wireless News*, Oct. 1, 2019. [Online]. Available: <https://www.rcrwireless.com/20191001/opinion/as-moores-law-ends-hardware-acceleration-takes-center-stage-part-1-reality-check>
3. M. Campbell, interview with T. Morad, Jan. 8, 2021.
4. M. Campbell, interview with A. Grabner, Jan. 20, 2021.
5. M. Campbell, interview with M. Davis and S. Pashenkov, Jan. 21, 2021.
6. M. Campbell, interview with A. Ezra, Jan. 11, 2021.
7. M. McGuinness, “A brief history of Application Performance Management (APM),” *Application Performance.com*, Jan. 31, 2017. [Online]. Available: <https://blog.applicationperformance.com/a-brief-history-of-application-performance-management-apm>

MARK CAMPBELL is the chief innovation officer for EVOTEK, San Diego, California, 92121, USA. Contact him at mark@evotek.com.