

\section{Overview}

Serverless compute is emerging as an attractive cloud computing model that lets developers focus only on the core applications, building them as small, fine-grained workloads (i.e., lambdas), without having to worry about building and/or managing the infrastructure they run on. Cloud providers dynamically provision, deploy, patch, and monitor the infrastructure and its resources (e.g., compute, storage, memory, and network) for these workloads; with tenants only paying for the resources they consume at millisecond increments. The cloud providers generally put a strict limit on the compute time and resource that can be consumed by a single workload, in order to ensure that they can easily deploy and scale each workload without impacting the availability of other workloads. Thus, the workloads are short-lived with strict compute time and memory limits (up to 15 minutes and 3 GB, respectively, for Amazon Lambda \cite{2}) and are often latency sensitive. Some examples of these workloads include real-time stream processing and generic API endpoints.

Today, all major cloud vendors offer some form of serverless frameworks (Figure 1), such as Amazon Lambda \cite{3}, Google Cloud Functions \cite{9}, and Microsoft Azure Functions \cite{7}, along with open-source developments like OpenFaaS \cite{13} and OpenWhisk \cite{2}. These frameworks rely on virtualization and containers \cite{10} to execute and scale tenants’ lambdas. These technologies were designed to maximize utilization of the providers’ physical infrastructure, while presenting each tenant with its own view of a completely isolated machine. With serverless computing, where server management is hidden from tenants, these virtualization technologies become redundant, unnecessarily bloating the code size of serverless workloads, and causing processing delays (of hundreds of milliseconds) and memory overheads (of tens of megabytes) \cite{18}. The increased overheads also limits the concurrent execution (less than hundred or so) of these workloads on a single server, hence, raising the overall cost of running such workloads in a data center. At the same time, high latencies and limited concurrency in modern serverless compute frameworks prohibit many interactive workloads (e.g., web servers and database clients) from taking advantage of serverless compute. The industry is starting to realize these issues and some providers, such as Google and CloudFlare, have recently started developing alternative frameworks (like Isolate \cite{1}) that remove these technology layers (e.g., containers) and run serverless workloads directly on the bare-metal server \cite{12}. However, CPU-based alternatives are inherently limited by their architecture design, which is not designed to run thousands of small functions in parallel due to high cost of context switching \cite{15}.

Recently, public cloud providers are deploying SmartNICs in an attempt to reduce load on host CPUs \cite{14}. So far, these attempts have been limited to offloading ad-hoc tasks (like TCP offload, VXLAN tunneling, and overlay networking) to accelerate network processing of the hosts. However, modern SmartNICs, more specifically ASIC-based NICs, consist of hundreds of RISC processors (i.e., NPUs) \cite{4}, each with their own instruction store and local memory. These SmartNICs can run many discrete programs in parallel at high speeds and low latencies, unlike CPUs and FPGAs, which are optimized to accelerate a specific workload \cite{11, 14, 16}.

Thus, we present $\lambda$−NIC, an open-source framework for running interactive serverless workloads on ASIC-based SmartNICs. $\lambda$−NIC leverages SmartNIC’s close proximity to the network and a large number of NPU cores to simultaneously run thousands of serverless workloads on a single NIC with predictable latency. To ease development and deployment of serverless compute workloads, $\lambda$−NIC exposes a new event-based programming abstraction, \texttt{match+lambda}, that allows developers to easily compose and execute serverless compute workloads on SmartNICs.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Overview of a general serverless compute framework. $R_i$ is the request for workload $i$ ($W_i$).}
\end{figure}
2 PRELIMINARY RESULTS

To compare the performance of $\lambda$-NIC versus existing serverless compute frameworks, we select OpenFaaS [3] as our baseline framework, as it is the most favored open-source serverless framework and closely resembles existing serverless infrastructure. By default, OpenFaaS run user workloads within Docker [10] containers via Kubernetes [17]. $\lambda$-NIC is built as an extension to the OpenFaaS, running users’ custom serverless workloads on SmartNICs [4], thereby naturally inheriting all of OpenFaaS’s features with the added support. In addition, to evaluate emerging frameworks like Isolate, we add support for running the workloads on a bare-metal backend, which runs the workloads as Linux process directly on the host OS. The overall architecture is in Figure 1.

**Workloads.** We evaluate $\lambda$-NIC on three types of workloads, reflecting popular serverless compute usage patterns [5, 8].

a. **Short workloads with no dependency.** These workloads involve relying self-contained contents, such as a static web page. We evaluate a simple server that responds various static contents.

b. **Short workloads with external dependencies.** These workloads request data from external data sources (e.g., database clients). These workloads generate extensive intra-data center requests and typically have strict tail-latency requirements. We evaluate memcached client workloads that each make SET and GET requests.

c. **Longer workloads.** These workloads involve processing on larger piece of data (e.g., image or stream processing). The data required for such workloads is generally larger than a single packet, and is stored in the DRAM. While such workloads often do not have low latency requirements, they require higher throughput, which can benefit from more cores available on SmartNICs. We evaluate an image grayscaler to emulate a compute intensive workload.

**Evaluation Results.** $\lambda$-NIC is more efficient in many aspects when compared to the baseline system. The $\lambda$-NIC workload optimizer efficiently coalesces workloads to reduce the executable binary’s code size to fit in a single NIC core. After optimizations, $\lambda$-NIC is capable of achieving up to 880x improvements in workload response latency (Table 1) and 736x improvements in work completion time across 56 threads while context switching.

**ACKNOWLEDGMENTS**

We thank the members of The Stanford Platform Lab, Neeraja Yadawdakar, and the anonymous SIGCOMM reviewers for their valuable feedback that helped improve the quality of this paper. This research was supported by The Stanford Platform Lab. Muhammad Shahbaz was also supported by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA6500-18-2-7865. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL and DARPA or the U.S. Government.

REFERENCES

[1] Isolate Class Reference. https://v8docs.nodesource.com/node-0.8/.