Supercharging Scientific Serverless: Slashing Cold Starts with Python UniKernels

ILLINOIS TECH
uchicago

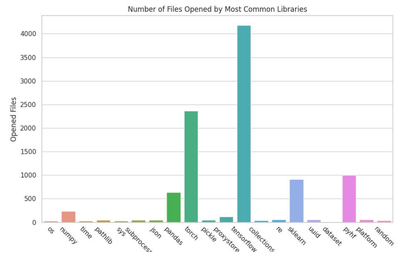
Jamison Kerney, Kyle Hale, Valerie Hayot-Sasson, Kyle Chard

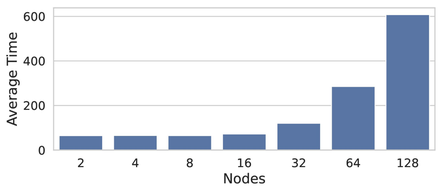
HExSA Lab
globus labs

1 ABSTRACT

- Serverless computing platforms use containers to create environments.
- Function as a Service paradigm, is dependent on the time to load the necessary container.
- We focus on the problem of rapidly loading Python environments in Globus Compute platform.
- Globus Compute is deployed on HPC systems and suffers from costs of shared file systems.
- We evaluate existing solutions such as containers and microvmms (Docker and Firecracker)
- We propose a new approach using lightweight Python Unikernels.
- We show considerable speed up in cold-start times using Unikernels

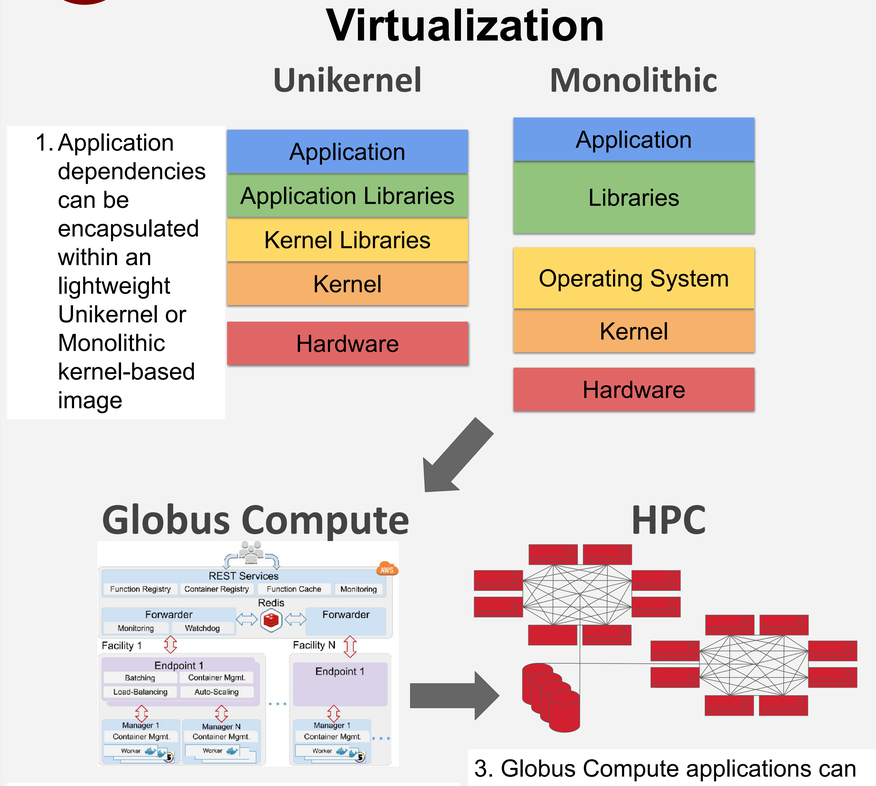
2 BACKGROUND





- The critical component of HPC systems that affects performance for containers is the shared filesystem.
- Small files have less potential for parallelization
- Figure above shows the number of files open in common libraries in Globus Compute.
- Figure below shows the time taken to load Tensorflow as we increase the number of nodes on the Theta supercomputer

3 SYSTEM ARCHITECTURE



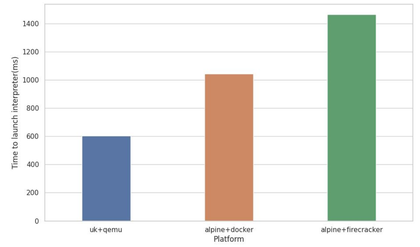
2. Globus Compute, a federated FaaS platform, can leverage virtualization to improve portability of applications

3. Globus Compute applications can be executed on various infrastructures, including HPC, which is commonly used by scientific applications

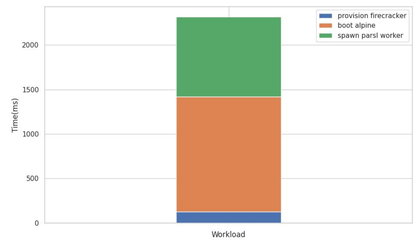
4 EXPERIMENTS

- Experiment 1: We measure the boot time of the Python Unikernel running on qemu-kvm, and compare it with the boot time of Alpine Linux running on Firecracker, and Docker. Boot times are measured within the Python interpreter
- Experiment 2: We further explore where time is spent when booting Firecracker VMs
- Experiment 3: we conducted a scaling experiment with Firecracker and Docker integrated with Parsl. We measure the time it takes to create 240 workers across 16 nodes.

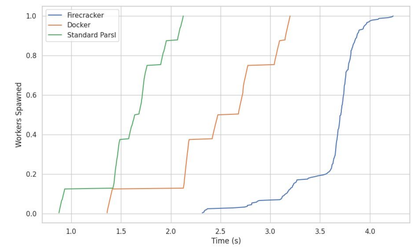
5 RESULTS



- Python Unikernel boots 1.6x faster than Docker and 2.4x faster than Firecracker



- Alpine Boot consumes a plurality of time
- Provisioning Firecracker consumes little time
- Spawning FuncX worker consumes almost as much time as boot



- Standard Parsl spawns all 240 workers before Firecracker boots 1 worker
- Docker's boot time mirrors Standard Parsl

6 CONCLUSIONS

- We investigate methods to improve cold start times with Python unikernels.
- Unikernels move the entire execution environment into memory turning many small reads into one big read.

7 REFERENCES

Kamatar, A., Sakarvadia, M., Hayot-Sasson, V., Chard, K., & Foster, I. (2023). Lazy Python Dependency Management in Large-scale Systems. *To Appear in Proceedings of the International Conference on eScience*.

Chard, R., Babuji, Y., Li, Z., Skluzacek, T., Woodard, A., Blaiszik, B., Foster, I., & Chard, K. (2020). Funcx: A federated function serving fabric for science. In *Proceedings of the 29th International symposium on high-performance parallel and distributed computing* (pp. 65–76).

Babuji, Y., Woodard, A., Li, Z., Katz, D., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J., Foster, I., & others (2019). Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing* (pp. 25–36).

Wanninger, N., Bowden, J., Shetty, K., Garg, A., & Hale, K. (2022). Isolating functions at the hardware limit with virtines. In *Proceedings of the Seventeenth European Conference on Computer Systems* (pp. 644–662).

Agache, A., Brooker, M., Iordache, A., Liguori, A., Neugebauer, R., Piwonka, P., & Popa, D.M. (2020). Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX symposium on networked systems design and implementation (NSDI 20)* (pp. 419–434).

Oakes, E., Yang, L., Zhou, D., Houck, K., Harter, T., Arpaci-Dusseau, A., & Arpaci-Dusseau, R. (2018). $\$SOCK$\$: Rapid task provisioning with $\$Serverless-Optimized$\$ containers. In *2018 USENIX annual technical conference (USENIX ATC 18)* (pp. 57–70).

Cadden, J., Unger, T., Awad, Y., Dong, H., Krieger, O., & Appavoo, J. (2020). SEUSS: skip redundant paths to make serverless fast. In *Proceedings of the Fifteenth European Conference on Computer Systems* (pp. 1–15).

Kuenzer, S., Bădoiu, V.A., Lefeuvre, H., Santhanam, S., Jung, A., Gain, G., Soldani, C., Lupu, C., Teodorescu, ., Raducanu, C., & others (2021). Unikraft: fast, specialized unikernels the easy way. In *Proceedings of the Sixteenth European Conference on Computer Systems* (pp. 376–394).