SnapFaaS: Fast Serverless Function Cold-boot through VM Snapshot Restoration

PRINCETON UNIVERSITY

Code repository: https://github.com/princeton-sns/firecracker-tools

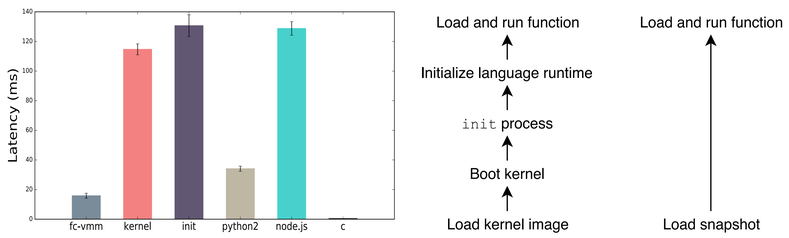
Yue Tan: yuetan@cs.princeton.edu David H. Liu: hao.liu@princeton.edu Amit Levy: aalevy@cs.princeotn.edu

Motivation

- Function-as-a-Service (FaaS), or serverless, systems allow many bursty "functions" to share relatively few datacenter resources.
- Because many of these applications are short-lived (typically hundreds of milliseconds), their cold-boot costs dominate their resource consumption.
- As a result, serverless providers must heavily over-provision resources and cache unused functions for long periods of time.
- Amazon's Firecracker providing strong isolation already cuts cold-boot latencies down to hundreds of milliseconds. However, hundreds of milliseconds is still high for typical serverless functions.

How Fast Is Firecracker

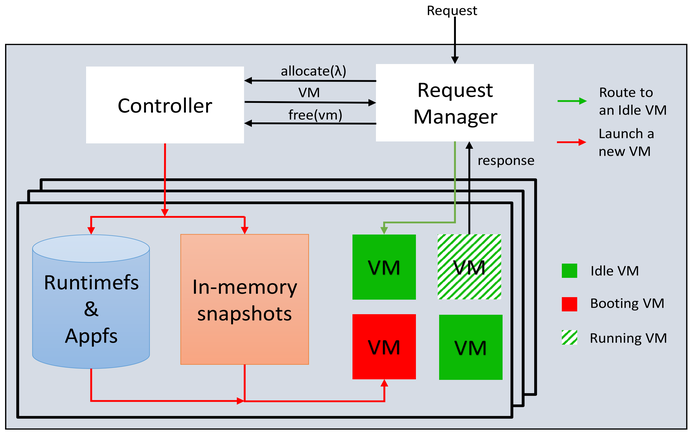
- We measure that it takes 296ms, 390ms, and 262ms on average to initialize complete Python2, Node.js, and C environments, respectively
- And that dirty memory is 37MB for Python2 and 40MB for Node.js after the language runtime initialization.
- Taken together, we note that on modern hardware, loading a 37MB file from the page cache into memory takes only a few milliseconds. Even loading it from disk at 500MB/s takes only 78ms – a quarter of the time it takes the VM to initialize the same memory.



- `fc_vmm` includes launching a VMM and configuring a VM. `init` runs OpenRC init and starts a minimum set of system services.
- Booting from a snapshot turns kernel booting, `init` process, and language runtime initialization into memory restoration.

SnapFaaS Design

1. **Provisioning through snapshot restoration:** we create a function environment by restoring from a VM snapshot. A restored VM already has the kernel and language runtime initialized and running.
2. **Snapshot generality:** Snapshots in SnapFaaS are taken after language runtime initialization to maximize the application stack's commonality for a whole category of functions. A Node.js snapshot can run all Node.js apps.
3. **In-memory snapshot store:** SnapFaaS VM launch latency is largely determined by how fast snapshots are loaded into guest memory. Thus, we keep snapshots in memory for fast copying to maximize latency reduction.



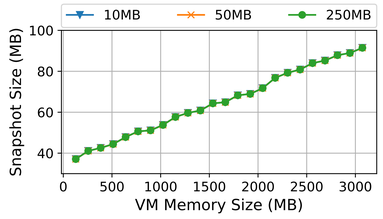
- **A VM snapshot** consists of registers, device (e.g. block device) states and dirty memory.
- **Runtimefs** includes typical operating system utilities (e.g. `bash`, `ls`, `cat`), and other standard UNIX utilities, common libraries (e.g. `libc`, `libm`, `libgmp`, etc), as well as the language interpreter and standard libraries (e.g. Python or Node.js).
- **Appfs** contains all function specific files — including the handler code and any necessary libraries or executables not provided by the `runtimefs`.

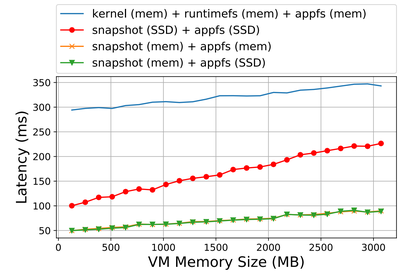
Python2 Wrapper

```
from subprocess import call, Popen
import multiprocessing as mp
import imp, sys
# send out snapshot signal from each vcpu
for i in range(1, mp.cpu_count()):
  Popen('taskset -c %d outl 124 0x3f0'%(i), shell=True)
call('taskset -c 0 outl 124 0x3f0', shell=True)
# mount appfs
call('mount -r /dev/vdb /srv', shell=True)
# signal boot completion
call('outl 126 0x3f0', shell=True)
# load app dependencies
sys.path.append('/srv/package')
# load app
app = imp.load_source('app', '/srv/workload')
# invoke app.handler upon requests...
```

Results

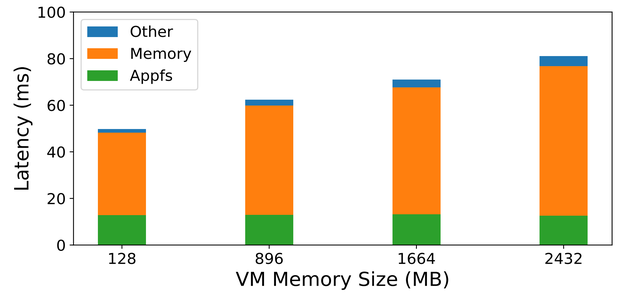
1. **6x speed-up:** SnapFaaS currently restores memory through memory copying and achieves 50ms instead of 296ms boot latency for Python2 VMs with 128MB memory.
2. **In-memory snapshots are justifiable and desirable:** In total, it takes about 1.5GB memory to store all Python2 snapshots. All snapshots for ~10 language runtimes will take about 15GB. Booting from in-memory snapshots is at least **2x** faster than from on-disk snapshots.
3. **The size or location of `appfs` has no impact:** on snapshot sizes or boot latencies.





- VM memory size grows from 128MB to 3072MB (i.e. 3GB) with step size 128MB.

Further Work



- Snapshot boot latencies consists of three parts: memory restoration, mounting appfs, and other (including launching a VMM and configuring a VM)

- Memory restoration can be further sped up through on-demand restoration.
- Mounting appfs costs 10ms. We think this latency can be cut down.
- We need to add support for network devices.