Debugging Live Applications in Kubernetes

Joe Elliott

github.com/joe-elliott/netcore-kubernetes-profiling

Grafana Labs

What This Is About

- Surveying native Linux debugging tools and technologies
  - perf - CPU Profiling
  - LTTng - Userspace Static Tracepoints
  - BCC (BPF) - Dynamic Tracing/Uprobes
- Increasing knowledge of applications in production environments
  - Low impact
- Performing analysis from a sidecar

References

- Sasha Goldstein
  - http://blogs.microsoft.co.il/sasha/
- Brendan D. Gregg
  - http://www.brendangregg.com/
- Others
  - https://jvns.ca/blog/2017/07/05/linux-tracing-systems/
  - https://www.joyfulbikeshedding.com/blog/2019-01-31-full-system-dynamic-tracing-on-linux-using-ebpf-and-bpftrace.html
  - https://github.com/iovisor/bcc

github.com/joe-elliott/netcore-kubernetes-profiling

From a Sidecar!

- Doesn’t require host access
  - Preserves immutability of host
- “Easy” to build a complete toolset
- Can dynamically add tools on the fly
- Supports development diversity

“Easy”

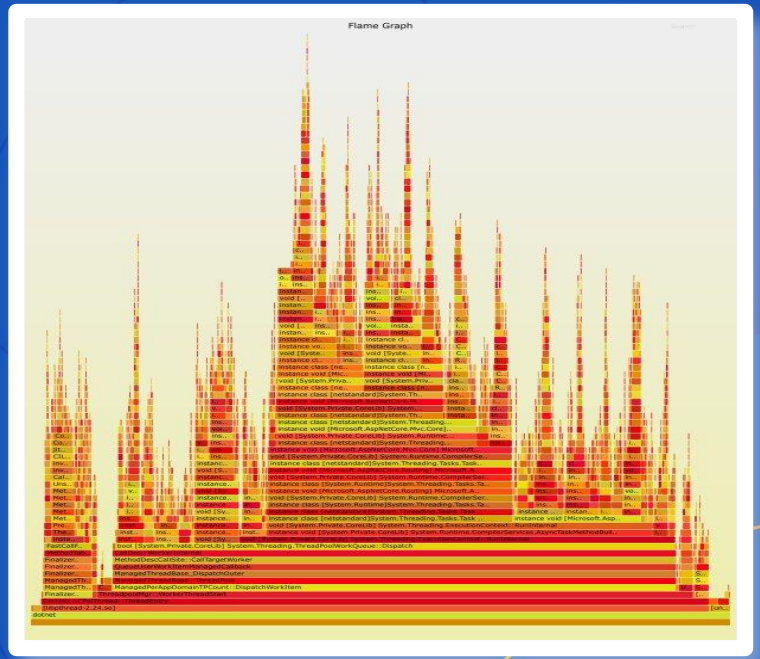
- Finding tools/resources that work with your kernel in your sidecar
  - Pull after deployment
  - Bake in tooling
  - Mount from host
- Sidecar image can be very large
- Sidecars can’t be added dynamically

Pod Features

- shareProcessNamespace
- Sharing mounted volumes
- Mounting host paths
- securityContext.privileged

CPU Profiling

By sampling and recording the stack many times a second, we can determine which methods our application spends most of its time in.



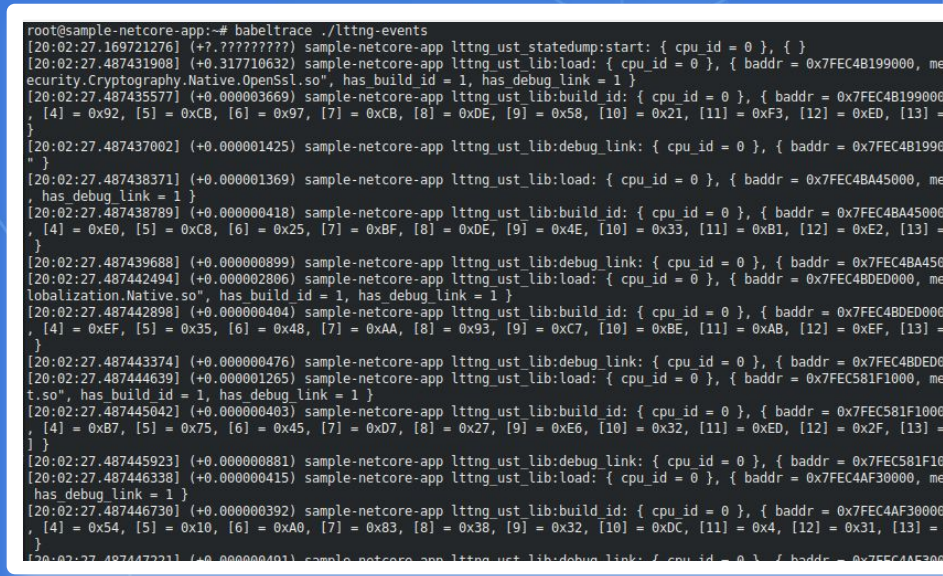
CPU Profiling

- Tools Used
  - Perf
  - flamegraphs
- Information Gathered
  - Where is my application spending most of its time?
  - Why does it have intermittent performance issues?
  - What is it doing when the CPU spikes?

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-netcore-app
spec:
  shareProcessNamespace: true
  Containers:
  - name: sample-netcore-app
    image: joeelliott/sample-netcore-app:v1.0.0-2.2.5
    volumeMounts:
    - mountPath: /tmp
      name: tmp
  - name: profile-sidecar
    image: joeelliott/netcore-debugging-tools:v0.0.7-2.2.5
    securityContext:
      privileged: true
    volumeMounts:
    - mountPath: /tmp
      name: tmp
  volumes:
  - name: tmp
    emptyDir: {}
```

Static Tracepoints

Pre-instrumented events can be captured and stored for later analysis.



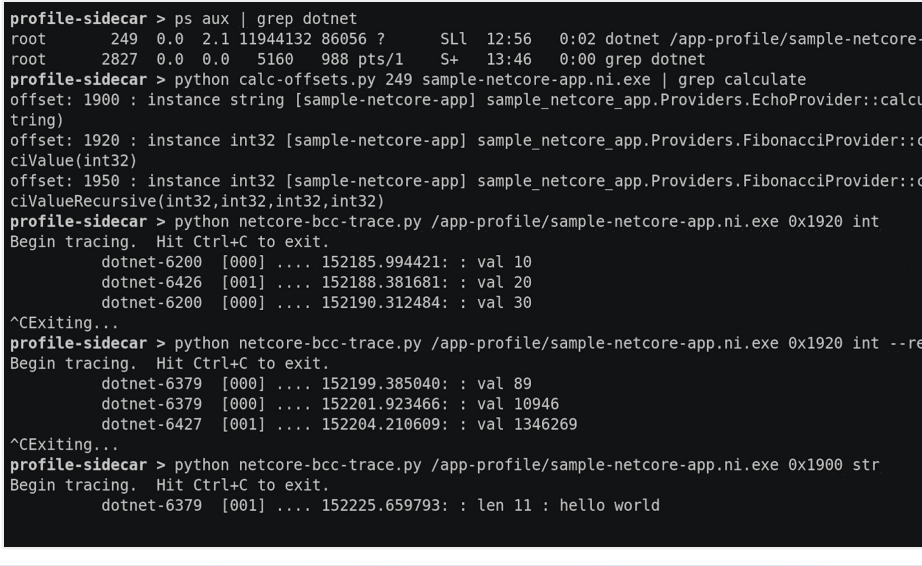
Static Tracepoints (LTTng)

- Tools Used
  - LTTng
  - Babeltrace
  - Trace Compass
- Information Gathered
  - When and how often do pre-instrumented events occur?

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-netcore-app
spec:
  shareProcessNamespace: true
  Containers:
  - name: sample-netcore-app
    image: joeelliott/sample-netcore-app:v1.0.0-2.2.5
    volumeMounts:
    - mountPath: /var/run/lttng
      name: lttng
  - name: profile-sidecar
    image: joeelliott/netcore-debugging-tools:v0.0.7-2.2.5
    volumeMounts:
    - mountPath: /var/run/lttng
      name: lttng
  volumes:
  - name: lttng
    emptyDir: {}
```

Dynamic Tracing

Attach custom tracepoints to uninstrumented code and dynamically record when and how they are executed.



Dynamic Tracing

- Tools Used
  - Perf
  - BCC/BPF
- Questions Gathered
  - When is an arbitrary function called?
  - What arguments are passed and what does it return?
  - https://github.com/iovisor/bcc

```
- image: joeelliott/sample-netcore-app:v1.0.0-2.2.5
  name: sample-netcore-app
  command: ["/run-native/runNative.sh"]
  args: ["/app/sample-netcore-app.dll"]
  volumeMounts:
  - mountPath: /run-native
    name: run-native-volume
  - mountPath: /app-profile
    name: app
  - mountPath: /tmp
    name: tmp
- image: joeelliott/netcore-debugging-tools:v0.0.10-2.2.5
  name: profile-sidecar
  securityContext:
    privileged: true
  volumeMounts:
  - mountPath: /app-profile
    name: app
  - mountPath: /tmp
    name: tmp
  - mountPath: /sys
    name: sys
  - mountPath: /usr/src
    name: src
    readOnly: true
  - mountPath: /lib/modules
    name: headers
    readOnly: true
```

Questions?