

Debugging Live Applications in Kubernetes

Joe Elliott

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



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”

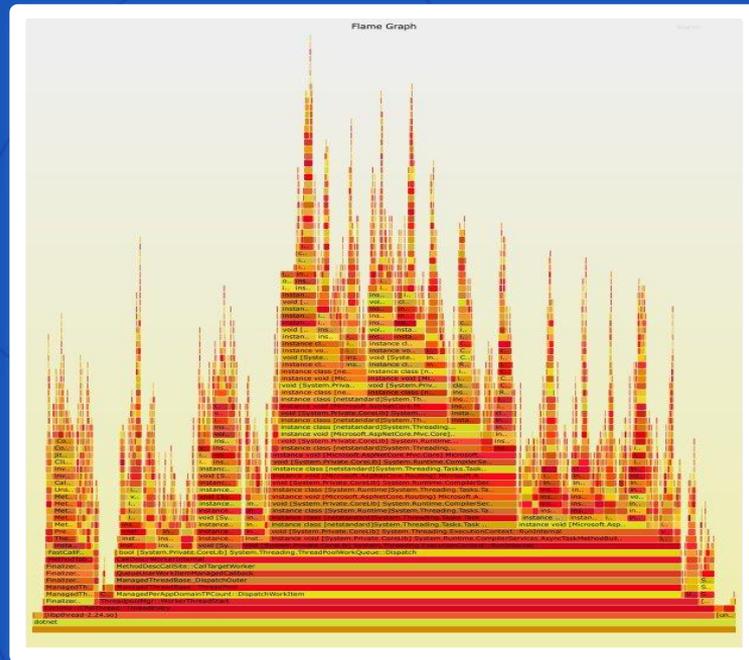
- Finding tools/resources that work with your kernel in your sidecar
 - Pull after deployment
 - Bake in tooling
 - Mount from host
- Sidecar image can be very large
- Sidecars can't be added dynamically

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: joelliott/sample-netcore-app:v1.0.0-2.2.5
    volumeMounts:
    - mountPath: /tmp
      name: tmp
  - name: profile-sidecar
    image: joelliott/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.

```
root@sample-netcore-app:~# babeltrace ./lttng-events
[20:02:27.169721276] (+7.?????????) sample-netcore-app lttng_ust_statedump:start: { cpu_id = 0 }, { }
[20:02:27.487431908] (+0.317710632) sample-netcore-app lttng_ust_lib:load: { cpu_id = 0 }, { baddr = 0x7FEC4B199000, me
security.Cryptography.Native.OpenSsl.so", has_build_id = 1, has debug_link = 1 }
[20:02:27.487435577] (+0.000003669) sample-netcore-app lttng_ust_lib:build_id: { cpu_id = 0 }, { baddr = 0x7FEC4B199000
, [4] = 0x92, [5] = 0xCB, [6] = 0x97, [7] = 0xCB, [8] = 0xDE, [9] = 0x58, [10] = 0x21, [11] = 0xF3, [12] = 0xED, [13] =
}
[20:02:27.487437002] (+0.000001425) sample-netcore-app lttng_ust_lib:debug_link: { cpu_id = 0 }, { baddr = 0x7FEC4B1990
}
[20:02:27.487438371] (+0.000001369) sample-netcore-app lttng_ust_lib:load: { cpu_id = 0 }, { baddr = 0x7FEC4BA45000, me
, has debug_link = 1 }
[20:02:27.487438789] (+0.000000418) sample-netcore-app lttng_ust_lib:build_id: { cpu_id = 0 }, { baddr = 0x7FEC4BA45000
, [4] = 0xE0, [5] = 0xCB, [6] = 0x25, [7] = 0xBF, [8] = 0xDE, [9] = 0x4E, [10] = 0x33, [11] = 0xB1, [12] = 0xE2, [13] =
}
[20:02:27.487439688] (+0.000000899) sample-netcore-app lttng_ust_lib:debug_link: { cpu_id = 0 }, { baddr = 0x7FEC4BA450
}
[20:02:27.487442494] (+0.000002806) sample-netcore-app lttng_ust_lib:load: { cpu_id = 0 }, { baddr = 0x7FEC4BDE0000, me
lobalization.Native.so", has_build_id = 1, has debug_link = 1 }
[20:02:27.487442898] (+0.000000404) sample-netcore-app lttng_ust_lib:build_id: { cpu_id = 0 }, { baddr = 0x7FEC4BDE0000
, [4] = 0xEF, [5] = 0x35, [6] = 0x48, [7] = 0xAA, [8] = 0x93, [9] = 0xC7, [10] = 0xBE, [11] = 0xAB, [12] = 0xEF, [13] =
}
[20:02:27.487443374] (+0.000000476) sample-netcore-app lttng_ust_lib:debug_link: { cpu_id = 0 }, { baddr = 0x7FEC4BDE00
}
[20:02:27.487444639] (+0.000001265) sample-netcore-app lttng_ust_lib:load: { cpu_id = 0 }, { baddr = 0x7FEC581F1000, me
t.so", has_build_id = 1, has debug_link = 1 }
[20:02:27.487445042] (+0.000000403) sample-netcore-app lttng_ust_lib:build_id: { cpu_id = 0 }, { baddr = 0x7FEC581F1000
, [4] = 0xB7, [5] = 0x75, [6] = 0x45, [7] = 0xD7, [8] = 0x27, [9] = 0xE6, [10] = 0x32, [11] = 0xED, [12] = 0x2F, [13] =
}
[20:02:27.487445923] (+0.000000881) sample-netcore-app lttng_ust_lib:debug_link: { cpu_id = 0 }, { baddr = 0x7FEC581F10
}
[20:02:27.487446338] (+0.000000415) sample-netcore-app lttng_ust_lib:load: { cpu_id = 0 }, { baddr = 0x7FEC4AF30000, me
has debug_link = 1 }
[20:02:27.487446730] (+0.000000392) sample-netcore-app lttng_ust_lib:build_id: { cpu_id = 0 }, { baddr = 0x7FEC4AF30000
, [4] = 0x54, [5] = 0x10, [6] = 0xA0, [7] = 0x83, [8] = 0x38, [9] = 0x32, [10] = 0xDC, [11] = 0x4, [12] = 0x31, [13] =
}
[20:02:27.487447221] (+0.000000403) sample-netcore-app lttng_ust_lib:debug_link: { cpu_id = 0 }, { baddr = 0x7FEC4AF30000
```

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.

```
profile-sidecar > ps aux | grep dotnet
root      249  0.0  2.1 11944132 86056 ?        Sll  12:56   0:02 dotnet /app-profile/sample-netcore-
root      2827 0.0  0.0   5160   988 pts/1    S+   13:46   0:00 grep dotnet
profile-sidecar > python calc-offsets.py 249 sample-netcore-app.ni.exe | grep calculate
offset: 1900 : instance string [sample-netcore-app] sample_netcore_app.Providers.EchoProvider::calcul
tring)
offset: 1920 : instance int32 [sample-netcore-app] sample_netcore_app.Providers.FibonacciProvider::c
ciValue(int32)
offset: 1950 : instance int32 [sample-netcore-app] sample_netcore_app.Providers.FibonacciProvider::c
ciValueRecursive(int32,int32,int32,int32)
profile-sidecar > python netcore-bcc-trace.py /app-profile/sample-netcore-app.ni.exe 0x1920 int
Begin tracing. Hit Ctrl+C to exit.
dotnet-6200 [000] .... 152185.994421: : val 10
dotnet-6426 [001] .... 152188.381681: : val 20
dotnet-6200 [000] .... 152190.312484: : val 30
^CExiting...
profile-sidecar > python netcore-bcc-trace.py /app-profile/sample-netcore-app.ni.exe 0x1920 int --re
Begin tracing. Hit Ctrl+C to exit.
dotnet-6379 [000] .... 152199.385040: : val 89
dotnet-6379 [000] .... 152201.923466: : val 10946
dotnet-6427 [001] .... 152204.210609: : val 1346269
^CExiting...
profile-sidecar > python netcore-bcc-trace.py /app-profile/sample-netcore-app.ni.exe 0x1900 str
Begin tracing. Hit Ctrl+C to exit.
dotnet-6379 [001] .... 152225.659793: : len 11 : hello world
```

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?