Fluentd performance comparison: Docker and CRI-O runtimes

Context

Testing logging-fluentd with a CRI-O runtime for the pods creating logs was, until recently, blocked by https://bugzilla.redhat.com/show_bug.cgi?id=1517605. Now that it is functional, some early performance and scalability testing has begun.

Fluentd shows a distinctly different performance profile when running on a node where the pod container runtime is CRI-O vs. when the container runtime is Docker. The CRI-O pod log write behavior initially appears to be responsible for higher fluentd CPU utilization and (currently) lower sustainable throughput.

Configuration

1. 2 clusters (1 Docker, 1 CRI-O) with cluster spec:
    a. 1 master/etcd - m4.xlarge
    b. 3 infra nodes - m4.xlarge
    c. 2 compute nodes - m4.xlarge
2. Docker 1.13 nodes configured with json-file logging. 3 files x 100MB.   CRI-O 3.9 configured with OOTB log configuration. The only tunable found in crio.conf was log_size_max = 52428800
3. OpenShift aggregated logging v3.9.2 or later
4. SVT ocp_logtest tool is used to log pod messages to pod stdout (see https://github.com/openshift/svt/blob/master/openshift_scalability/content/logtest/ocp_logtest-README.md)

Test Case

Repeat on clusters with Docker 1.13 nodes and CRI-O 3.9 nodes

```
projects:
  - num: 12
    basename: logtest
    ifexists: delete
    tuning: default
    templates:
```

```
    - num: 1
      file: ./content/logtest/logtest-rc.json
      parameters:
       - REPLICAS: "1"
       - INITIAL_FLAGS: "--num-lines 20000 --line-length 1024 --word-length 9 --rate 1000
--fixed-line\n"
```

This creates 12 projects each with 1 pod - 6 pods per compute node. Each pod runs 1000 1Kb messages/minute for 20000 messages (20 minutes). This is an aggregate message rate of 100 messages per second per node (fluentd).

- Collect pbench data for the duration of the run
- Verify message counts at the end of the run. 2 bz can cause incorrect message counts
  - https://bugzilla.redhat.com/show_bug.cgi?id=1491401
  - (CRI-O specific) https://bugzilla.redhat.com/show_bug.cgi?id=1552304

Repeat the tests for higher message rates (200mps/node, 500mps/node, 750mps/node) until fluentd is no longer able to keep up. This shows up as 100% utilization of a single core which is all fluentd can use as configured for OCP.

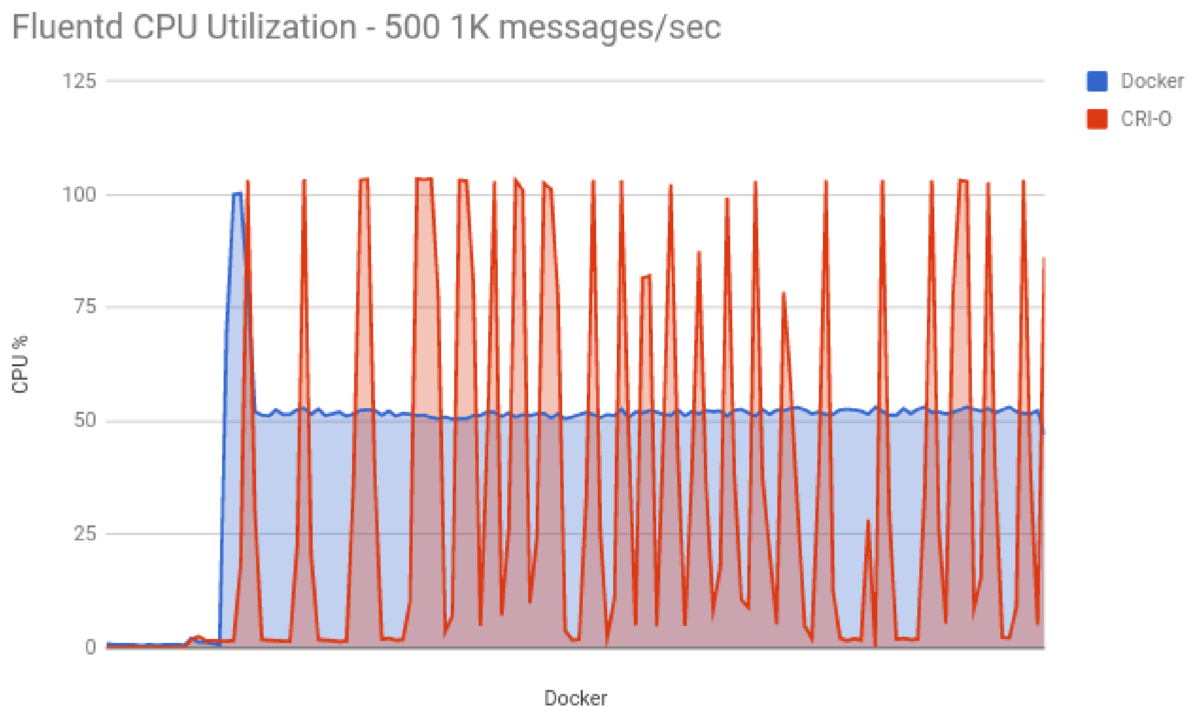
Results

Data for various message rates are available here:

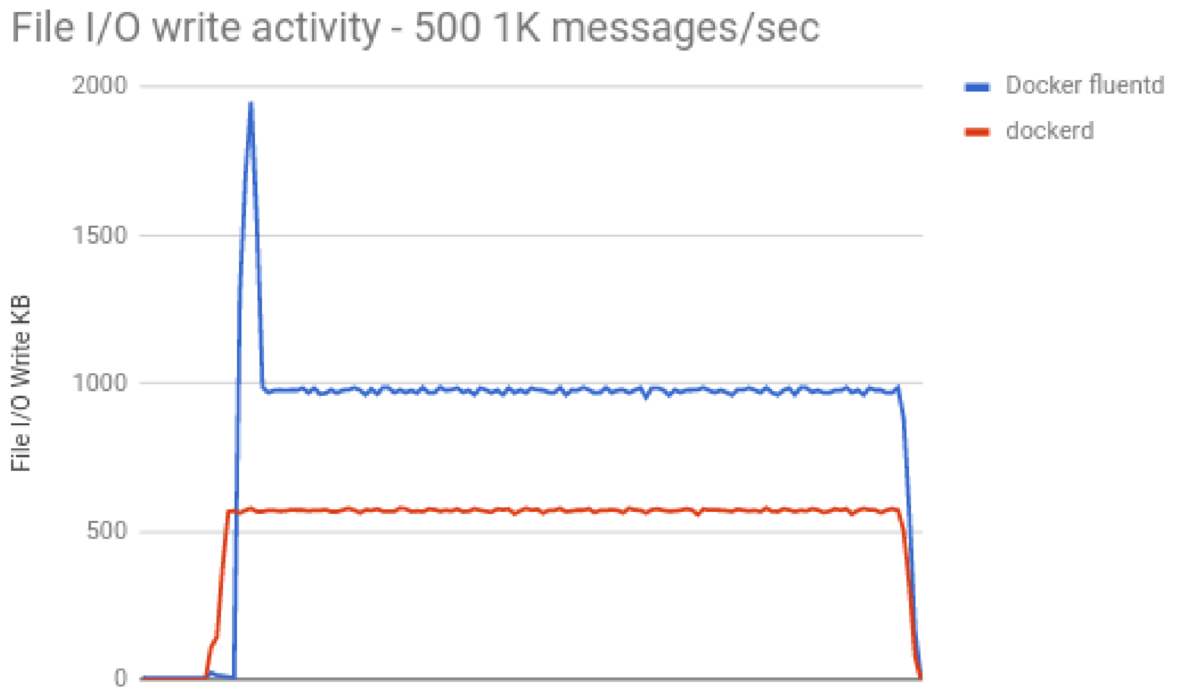
Docker runs: http://pbench.perf.lab.eng.bos.redhat.com/results/EC2::ip-172-31-0-119/
CRI-O runs: http://pbench.perf.lab.eng.bos.redhat.com/results/EC2::ip-172-31-46-30/

The data and the observations below are for 500 aggregated messages/second per fluentd (across all pods) and the patterns are representative of all message rates.

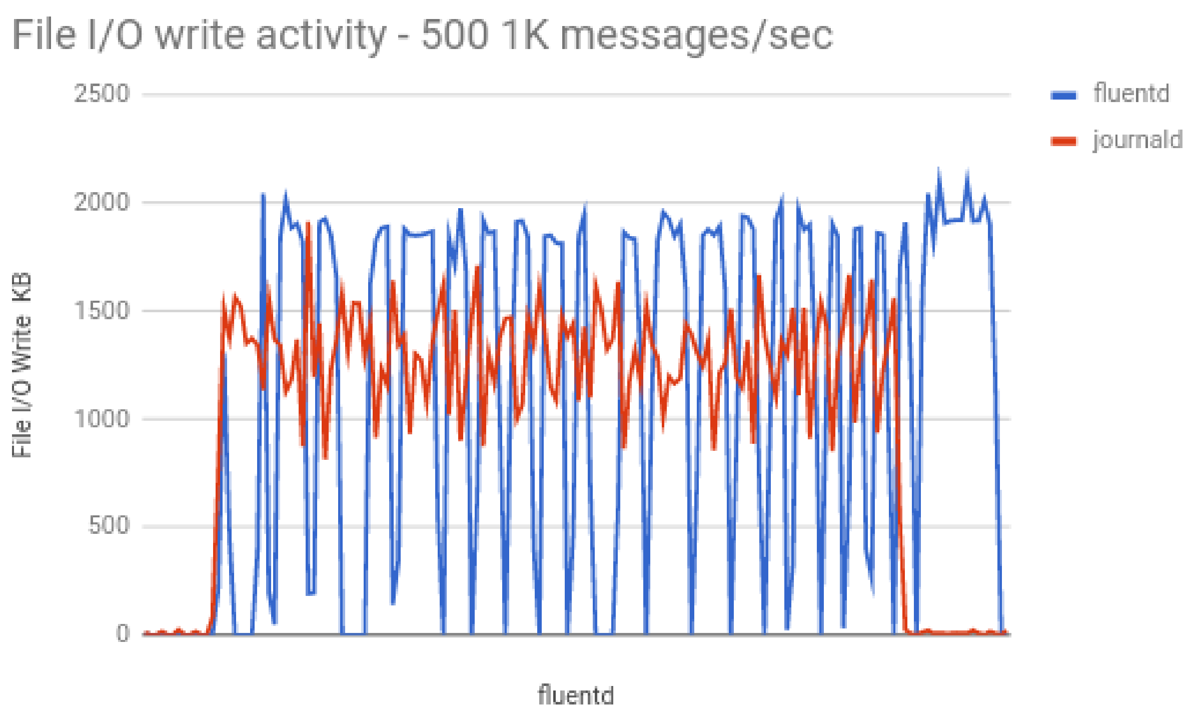
Fluentd CPU at 500 messages/sec for each runtime:



File write KB dockerd and fluentd at 500 messages/sec

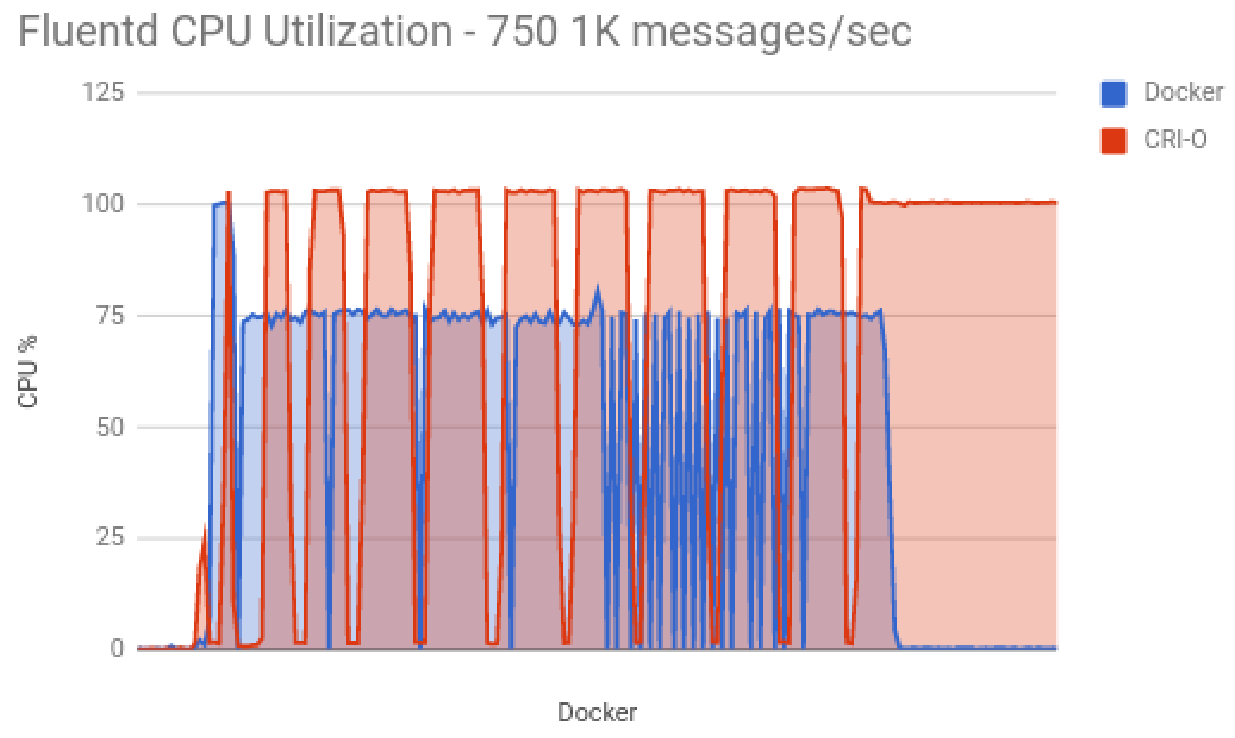


File write KB journald and fluentd at 500 messages/second



Observations

- Docker writes the pod log output at a constant rate that allows fluentd to consume it an write to elasticsearch at a constant rate
    - Watching ES indices during a run, the Docker runs show smoothly increasing numbers of index entries. The CRI-O runs show the number of entries periodically ratcheting up. <Need a graph for this>
- In the CRI-O case, it takes 10 minutes after the last journald entry for the fluentd to finally index all entries. This is evidence of fluentd falling behind in this scenario which is evidence that the rate is not sustainable long term
- In previous tests similar to this with Docker runtimes, fluentd could sustain a rate of 750 - 1000 1K messages to second at a relatively constant cpu utilization of 80%. With a CRI-O runtime it appears event 500 1K messages/sec is not sustainable. One final graph comparing CPU at 750 messages/second:



Note: The pbench collection in the docker runs is missing some intervals - those are not true spikes - fluentd was flat at 75%

Summary

More investigation has to be done and at least 1 bug files. Initial data shows that logging-fluentd on a CRI-O runtime cannot maintain the same sustained rates as on a Docker runtime due to the way a) CRI-O is writing pod logs and b) the way fluentd is configured to read them.