Prometheus/Jaeger Slides

10/27/20

Prometheus Setup + Usage

- Requires Metrics Server.
  - Using release v0.3.6, v0.3.7 untested (latest release)
- If you see an error with `metrics not available`, requires additional arguments when starting metrics-server. Not needed on VM, but needed on CloudLab
  - - --kubelet-insecure-tls
  - - --kubelet-preferred-address-types=InternalIP
- Detailed Instructions to setup is in repo - /testing/monitoring folder
  - By default, build.sh script assumed a monitoring namespace so create that beforehand
  - Can get the port number for prometheus via: `kubectl get svc -n monitoring`
  - To view prometheus on local machine, navigate to web address: http://<GCM-node-public-ip>:<prometheus-port>
- Sample Prometheus Queries - note that no reason to measure jaeger pod usage
  - sum(rate(container_cpu_usage_seconds_total{pod!="~jaeger*", image!="", container_name!="POD", namespace="media-microsvc"}[1m])) by (pod)
  - sum(container_memory_usage_bytes{pod!="jaeger*", image!="", container_name!="POD", namespace="media-microsvc"}) by (pod)

Jaeger Overview + Usage

- What is Jaeger?
  - An open-source, end-to-end distributed tracing framework to track transactions between distributed services.
  - Default option in DeathstarBench to debug services. Port can be identified via: `kubectl get svc -n media-microsvc`, similar to prometheus
  - Can view on the web via jaeger console via: http://<GCM-node-public-ip>:<jaeger-port>
- How does it work?
  - Follows the path of a request through different microservices. This generates a call flow.
  - Can trace through REST API calls, which is very beneficial for us
  - Can also export traces in JSON which can be used to generate CDF/latency graphs for different services/calls
- Tutorial I used to gain familiarity:
  - https://www.scalyr.com/blog/jaeger-tracing-tutorial/
- Original Jaeger Source
  - https://www.jaegertracing.io/