istio open source community document

Istio Performance & Scalability

Summary
How to measure and ensure Istio has the desired performance and scalability characteristics.

Owner: ldemailly@google.com
Work-Group: Performance & Scalability
Contributors: suryadu@us.ibm.com, ozben@google.com, bochun@google.com
Short self link: https://goo.gl/CXewuj

Status: WIP
Created: Oct 3, 2017
Target Release Version: v0.3

Performance
- Scenario 0:
- Scenario 1 (minimal/best case/perf smoke test)
- Scenario 2 ('realistic' medium size)
- Scenario 3 ('realistic' large size)

Scalability

Integration with build/release/testing

Tracking

Current Status / What is next

Out of scope

See also: https://goo.gl/ENFQWb (Perf WG meeting notes) and https://goo.gl/DauRxi (Perf SWAT team doc)

Performance

There are many interesting dimensions along which performance will vary, for instance:

- QPS for single service
- Number of services
- Number of rules
- Rate of change in the system
- mTLS on/off
- iptables/containers vs just process running on VM
- Network latency and throughput
- Cached vs not cached mixer query

We want to find when each component becomes - on the proxy container, ideally figure out which part is due to Envoy vs MixerClient vs making the Mixer call(s)

- CPU bound
- Memory bound
- IO (network) bound

And determine:

- Throughput sustainable
- With an acceptable p99 latency

We also want to figure out both vertical scalability (does say, mixer, scale almost linearly with the number of CPU, up to which point) and horizontal scalability (if we have K machines how close to K x the throughput of 1 machine can we achieve).

We need to select a small number of scenarios and establish a baseline for each and track the change over time (automatically).

Scenario 0:

Vm -> Ingress -> Svc1
Svc1 -> Svc2 (no caching):
https://fortio.istio.io/browse?url=2018-01-29-222530_400qps_no_cache.json

Scenario 1 (minimal/best case/perf smoke test)

Current minimal test (Setup instructions): [todo: move to its own page]
Client (load generating) machine: running fortio load

Server (measured) machine: envoy + mixer + echosrv

This gives a “best case” kind of data but does help with basic performance regressions

TODO: similar “smoke” for Pilot

Scenario 2 (‘realistic’ medium size)

12 services on GKE, induced changes in rules and pods/deployments, multi layered calls

+ 2 VMs expanding the mesh

Scenario 3 (‘realistic’ large size)

100 services

Scalability

Goal: Measure and ensure that having N Pilot, Mixer scale near linearly

Environment/Scenario:
After we establish the single instance / per core max throughput for each component, add 1, 2, N and checking the expected increased throughput.

Integration with build/release/testing

Searchable keys for the result:

- A general type string for the test run (e.g. “master” for building from head of master, “release/0.2.12” for release candidate run, etc.)
- Date and time for the Run
- Target QPS
- Target Duration (how long the load test run)
- Whether mTLS is enabled
- Num Connections (called NumThreads in JSON)
- URL used (contains some of the setup information)
- Labels/config/setup: for now I am adding just a single string “Labels” of unspecified format, but we probably should use a comma separated list of labels: e.g “`vm,4proc,notls,mixer+envoy,v0.3.7`” or some such [TBD vm/k8s, tls/notls, machine size(s), version/sha, how many layers are tested...]

Ideally [P1] we need to track over time changes in performance to catch regressions early

- We need a system (db like) to record the results for each build being evaluated: Google Cloud Datastore seems like a good fit.
- Nice to have : graph over time/releases - export to https://velodrome.istio.io/ and/or fortio self graphing

Tracking

Epic on github: https://github.com/istio/istio/issues/369

Current Status / What is next

- Investigate Pilot/Envoy response with 100s of services and routes
    - Start up time
    - Memory usage
    - CPU usage
- Check on High CPU on envoy+mixerclient for simple rules
    - Extra 9ms latency for 1 && using headers
- Cache short circuiting bug : appears fixed on latest daily!
- VM setup vs GKE/Bluemix vs Minikube
    - And for VM: raw processes vs kubeadm
    - Need to update https://github.com/istio/istio/blob/master/tools/setup_run
- Ingress vs Svc2Svc

Out of scope

- Longevity testing
- Stability testing - this is fairly short running tests (up to 1h - it may be enough to step on stability problems but it isn't the focus)
- Interoperability or Upgradability (ability to run mix of old/new istio release, api backward compatibility checks, protocol negotiation checks etc)