User Guide

Service Mesh - Envoy Regular Expression Matching Acceleration with Hyperscan

Authors

Zhihao Xie

Ramesh Masavarapu

1 Introduction

One of the key values within Envoy is to perform routing and filtering requests. Regular expression matching is widely used during filter selection, access control, and routing operations by Envoy as an ingress gateway within a service mesh environment. For customers who manage Envoy routing tables that have entries in thousands, there is an impact on the performance and latency. The reason is that the default regular expression library within Envoy is CPU intensive, leading to impact on latencies.

Hyperscan is a high-performance regular expression matching library. By using Hyperscan, the regular expression operations have significantly improved the latency compared to the default regular expression engine within Envoy in specific scenarios. Hyperscan supports most of the PCRE (Perl Compatible Regular Expressions) syntax as a baseline and it delivers an extensive set of features and orders of magnitude better performance than PCRE. Unlike the default regular expression matcher, Hyperscan is introduced as an input matcher and the user needs to change the matching method from safe regex to custom matcher. Because of this, Hyperscan matcher provides more features than the default regular expression matcher.

Hyperscan matcher allows users to enter multiple matching patterns at the same rule, and users can merge multiple regular expression matching rules that have the same action, providing faster matching capabilities while reducing lengthy configuration. For situations when a user requires behavior that depends on the presence or absence of matches from groups of patterns, Hyperscan matcher also provides the ability to perform logical combinations.

In use cases of Envoy being used as an ingress gateway managing a routing table that runs in thousands within a Service Mesh deployment, customers stand to benefit with Hyperscan optimizations. Without Hyperscan optimizations, there is an impact on the high latencies impacting end customer experience.

This document is intended for cloud service providers, telco customers, enterprise customers, and others who are using Envoy as an ingress gateway server that requires large amounts of regular expression matching on 3rd Gen and newer Intel® Xeon® Scalable processors.

This document is part of the Network Transformation Experience Kits.

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	January 2023	Initial release.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
PCRE	Perl Compatible Regular Expressions
RBAC	Role Based Access Control
RE2	Regular Expression Engine used in Envoy developed by Google
WAF	Web Application Firewall

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Introduction to Hyperscan	https://www.hyperscan.io/
Envoy	https://www.envoyproxy.io/
Service Mesh – Istio and Envoy Optimizations for Intel® Xeon® Scalable Processors Solution Brief	https://networkbuilders.intel.com/solutionslibrary/service-mesh-istio-envoy-optimizations-intel-xeon-sp-solution-brief
Service Mesh - Crypto Accelerations in Istio and Envoy with Intel Xeon Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/service-mesh-crypto-accelerations-istio-envoy-intel-xeon-sp-user-guide
Service Mesh - TCP/IP eBPF Bypass in Istio and Envoy with Intel® Xeon® Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/service-mesh-tcp-ip-bypass-istio-envoy-intel-xeon-sp-user-guide
Service Mesh – mTLS Key Management in Istio and Envoy for Intel® Xeon® Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/service-mesh-mtls-key-mgmt-istio-envoy-intel-xeon-sp-user-guide

2 Overview

Envoy performs filtering, access control, and routing operations within a service mesh environment. During filtering operations, it selects different filters for different HTTP requests. For access control operations, Envoy uses access control to block suspicious requests by matching their characteristics with security policies. The routing actions involve parsing the URL paths and other components of HTTP requests that need to be routed upstream to different clusters or services.

In the operations mentioned, matching is the basic but core module, which helps Envoy to decide where a request can be redirected. regex matching is one of the most expensive methods; it produces significantly more CPU utilization compared to prefix and exact matching.

This user guide describes the technical details of using Hyperscan in Envoy when Envoy is used as an ingress gateway within a Service Mesh environment.

3 Topology

Figure 1 shows an example using Envoy as an ingress gateway. Envoy is configured with multiple HTTP filters along with RBAC and a router. In the example, all requests pass through different filters for TLS encryption, content modification, statistics generation, access control, and finally arrive at the router for routing and forwarding.

During each step of the pipeline, the regular expression is invoked to perform pattern matching. The first code block is the configuration of Extension with Matcher, which can be used for filter selection. In the code block, requests are matched upon their path in request headers to be handled by the filters. The second code block is the configuration of RBAC rules, where the user specified that requests via GET with a path need to follow the corresponding rules. The third code block is the HTTP router configuration, where the user routes the request with the specified path to the service-https cluster.

Each request may be filtered across these multiple cascaded extensions, each of which can have numerous regular expression rules for matching. Some community access control rules used in IDS/IPS or WAF have more than 1,000 regular expressions rules. In addition, the ingress gateway for complicated services can have a large number of routing table rules. Expensive regular expression matching consumes large amounts of CPU resources, making Envoy the bottleneck, which is unacceptable in a latency-sensitive service mesh environment.

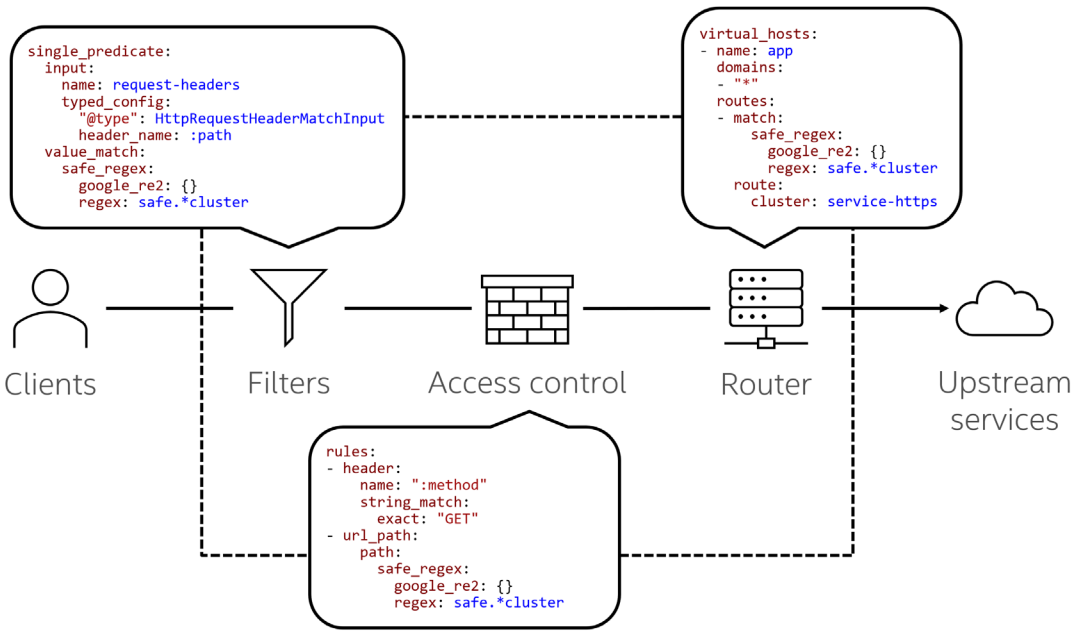

Figure 1. Architecture Overview when Envoy is Used as an Ingress Gateway

Figure 2 shows the relationship between Envoy and the various components of matching. Every request is handled by the foundational events of Envoy, which includes a listener, network filters, HTTP filters, and a router. Filter chain in the listener populates all filters through which requests should pass. The filter chain match is responsible for distributing different connections to different filter chains according to their network information. The HTTP connection manager, RBAC, and the HTTP router are components of the network filters, HTTP filters, and the router. In Figure 2, we create a gateway from the filters, which means a request is examined by the ACL and be redirected to different upstream services.

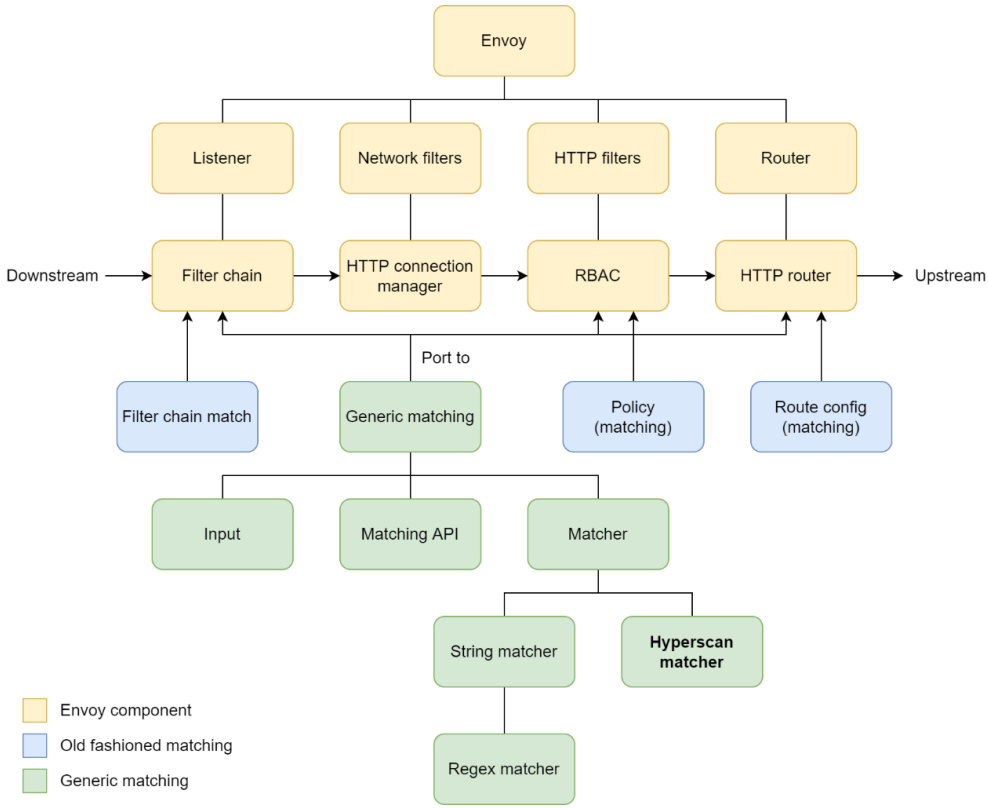

Figure 2. Relationship Between Envoy Components and Matching

Generic matching is the latest matching mechanism of Envoy, which is designed to replace the separated, coupled, and old-fashioned matching owned by every extension, such as the policy of RBAC and the routing configuration of the HTTP router. The API of generic matching is called the matching API, which is still in the early alpha stages and thus may be subject to a large number of impending changes. Currently, all matching interpreted by Istio is based on older matching methods, so generic matching cannot be deployed with Istio. Generic matching is constructed with three parts, namely: inputs, the matching API, and matchers. The input defines what kind of information should be matched on, like destination IP and HTTP request headers. The matching API provides both sublinear (for example, hash table and trie matching) and linear (for example, list-based matching) support, logical combination, and orchestratable actions, which means the action that should be taken when the target data is matched. The matching API is now part of the xDS API. The matcher is the real implementation of generic matching, which provides a matching algorithm against different data. String matcher is the default matcher of generic matching and regex matcher, which is based on the default regular expression engine RE2 is a kind of string matcher. Hyperscan matcher is the matcher that we developed, based on Hyperscan. Since generic matching has been ported to filter chain, RBAC, and HTTP router, we can benefit from Hyperscan directly with Hyperscan matcher.

4 Ingredients

4.1 Hardware Bill of Materials

3rd Gen Intel Xeon Scalable processor or 4th Gen Intel Xeon Scalable processor.

4.2 Software Bill of Materials

Envoy Contrib 1.22.0 or later.

5 Deployment Details

Beginning with Envoy 1.22.0, Hyperscan is integrated into Envoy Contrib. Envoy Contrib can be built from the source code or launched directly via the official Docker image.

```
# Build from the source code and run
git clone https://github.com/envoyproxy/envoy
cd envoy
ci/run_envoy_docker.sh 'ci/do_ci.sh bazel.release'
./linux/amd64/build_envoy-contrib_release_stripped/envoy

# Run the official Docker image
docker run envoyproxy/envoy-contrib:v1.22-latest
```

Here is a configuration route, which allows requests from port 8080 to 80. The router routes requests if their path matches the regular expression pattern `safe.*cluster`, and all unmatched requests are dropped immediately.

```
static_resources:
  listeners:
  - address:
      socket_address:
        address: 0.0.0.0
        port_value: 80
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type":
type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionM
anager
          codec_type: AUTO
          stat_prefix: ingress_http
          route_config:
            name: local_route
            virtual_hosts:
            - name: app
              domains:
              - "*"
              matcher:
                matcher_list:
                  matchers:
                  - predicate:
                      single_predicate:
                        input:
```

```
                          name: request-headers
                          typed_config:
                            "@type":
type.googleapis.com/envoy.type.matcher.v3.HttpRequestHeaderMatchInput
                            header_name: :path
                        custom_match:
                          name: hyperscan
                          typed_config:
                            "@type":
type.googleapis.com/envoy.extensions.matching.input_matchers.hyperscan.v3alpha.Hyperscan
                            regexes:
                            - regex: safe.*cluster
                    on_match:
                      action:
                        name: route
                        typed_config:
                          "@type": type.googleapis.com/envoy.config.route.v3.Route
                          match:
                            prefix: /
                          route:
                            cluster: service-http
          http_filters:
          - name: envoy.filters.http.router
            typed_config:
              "@type": type.googleapis.com/envoy.extensions.filters.http.router.v3.Router

  clusters:
  - name: service-http
    type: STRICT_DNS
    lb_policy: ROUND_ROBIN
    load_assignment:
      cluster_name: service-http
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: 127.0.0.1
                port_value: 8080
```

6 Results

Hyperscan and Hyperscan matcher can be benchmarked in Envoy with the following commands.

```
# Benchmark Hyperscan
bazel run -c opt //contrib/hyperscan/matching/inputs_matchers/test:hyperscan_speed_test

# Benchmark Hyperscan matcher
bazel run -c opt //contrib/hyperscan/matching/inputs_matchers/test:matcher_speed_test
```

These benchmarks are constructed with the default regular expression benchmark inputs and pattern and are based on the Google Benchmark. The regular expression engine takes the regular expression pattern `^cluster\.((.*?)\.)` and matches against inputs `cluster.no_trailing_dot`, `cluster.match.`, `cluster.match.normal` and `cluster.match.and.a.whole.lot.of.things.coming.after.the.matches.really.too.much.stuff` in sequence.

7 Summary

This document describes the architecture, configurations, and benefits of Hyperscan matcher, and demonstrates how to accelerate regular expression matching in Envoy with Hyperscan. In a scenario with numerous regular expression patterns, the method using Hyperscan has a significant performance improvement in latency reduction compared to the default apparatus using the default regular expression engine RE2. For customers who have a use case of using Envoy as an ingress gateway or a WAF, we recommend using the Hyperscan matcher, especially in cases that require more than 1,000 rules.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

0123/DN/WIT/PDF 761773-001US