FLOWMILL

Network Reliability and Performance in Distributed Applications

www.flowmill.com

Introduction

Modern applications are quickly shifting towards service-based, distributed architectures. Many of these applications are moving into containers managed by orchestration platforms such as Kubernetes as well. While this has introduced huge operational advantages in agility and scalability, it has at the same time created new challenges in observability.

Most conversations about observability center around the collection and analysis of metrics, logs, and traces. While each of these technologies can play an important role in an observability strategy, they come with some important limitations. Each approach requires engineer time to implement, andt more importantly, they focus only on application layer data.

However, distributed applications consume and rely on the network more than ever before. As these applications have shifted into containers, function calls have essentially migrated to network-based API calls, adding hundreds or even thousands of internal communication touchpoints within an application. Teams have rapidly adopted external cloud-based services for everything from databases, message queues, CDNs, and communication platforms and services outside of their control have become critical to their application's behavior. Public cloud providers, many of which operate complex software-defined networks, provide limited and expensive tooling that operates with no knowledge of the workload running in each instance. Some teams have even begun operating hybrid cloud scenarios with multiple cloud providers or an on-premise datacenter and a cloud provider.

As these complexities and layers of abstraction have compounded, the network has become a significant visibility gap for many SRE teams when it should instead be a source of truth about the behavior of a distributed system.

Business Impact

While network problems represent a minority of issues that SRE teams encounter, these problems can have outsized impact. In any digital business, network performance and reliability problems can have a significant negative impact on customer experience. Users equate slow networks with slow applications and this drag on performance may manifest itself in lower customer conversion rates, abandoned transactions, and lost revenue.

Network performance and reliability problems may also manifest to users as intermittent application failures. Depending on the architecture, a subset of users may see consistent failures or a larger set may experience intermittent problems. In either case, these user visible failures may impact customer conversion, engagement, and retention in an application.

Finally, the most drastic business risk with network problems stems from prolonged downtime. With limited tooling available, network issues frequently take hours or sometimes days to properly root-cause as teams investigate the resulting stream of downstream application-level failures . This sort of long term issue can violate customer SLAs, breach contracts, and put the entire business at risk.

Challenges

Network-related issues can have a broad impact on distributed applications. For many teams, just knowing whether a string of alerts was caused by the network infrastructure can save hours of troubleshooting their internal services. However, deeper visibility is necessary to identify specific classes of problems As part of any triage process, it is critical to understand if a problem is caused by the network infrastructure and how different services are affected. Let's examine some of the specific challenges SRE teams face.

Identifying Services Impacted by Packet Loss in the Infrastructure

SRE teams need to understand where and when packet loss and connection failures occur and what services are affected. This data is frequently obscured by the cloud provider entirely and even when it is not, it can be difficult to map the behavior of a distributed service consisting of an ever-changing set of containers or instances to network-level statistics.

When API calls are slow or unreliable because of a true problem in the network, it is important to know which services are affected and which hosts or instances are involved.

Having accurate, real-time service-aware network reliability data can also accelerate troubleshooting and incident response. Development teams may blame the network for application level issues that they cannot otherwise explain. Without ground truth data on network reliability, these problems may fail to get the appropriate amount of developer attention to ultimately resolve them.

It is worth noting that many teams are currently evaluating or implementing service meshes based on proxies such as Envoy. Since these proxies operate at Layer 7 in the network stack, they will not report, and in fact, may even obscure underlying network reliability issues.

Identifying Operating System Performance Problems

In addition to problems occurring in the infrastructure, a range of networking correctness and performance issues can be traced back to the operating system as well. Overloaded network address translation tables, CPU or memory bottlenecks, NIC saturation, and a wide array of other issues can cause network issues at the operating system level.

Measuring DNS Reliability

DNS (Domain Name System) is a critical service in most distributed applications since it serves as the backbone of service discovery. When DNS fails, it can lead to a wide array of downstream, application level failures that can take substantial time to debug.

On a platform such as Kubernetes, there are a number of places DNS can fail, including the operating system, the coreDNS service, and the upstream DNS server. It's important to have visibility into the end to end DNS request and response sequence and, ideally be able to identify any individual component that may be failing.

Examples of DNS failures:

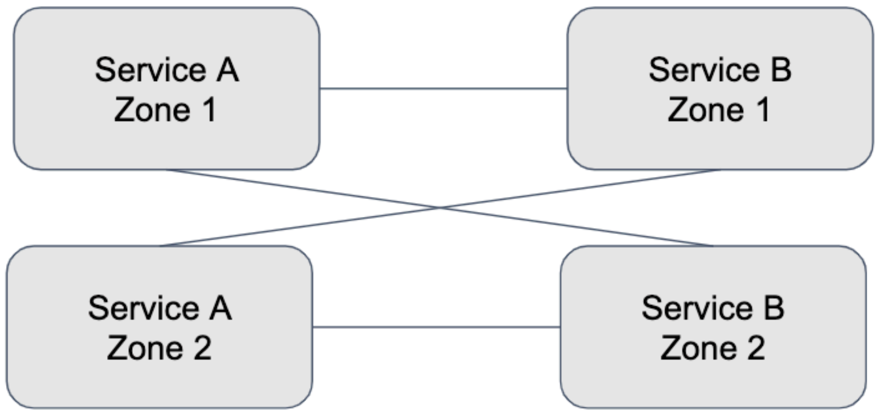
- Race conditions in Linux NAT translation.
- Underprovisioned DNS containers (in Kubernetes)
- Failure of cloud provider DNS
- ... many more...

Managed or Cloud Services

It is increasingly common to design applications to include one or more external cloud services such as a databases, message queues, object storage, or communication tools. Some of these services, like RDS (AWS's Relational Database Service) are hosted by the cloud provider on a region-basis. Others, like Slack, Stripe, Salesforce, or Twilio are operated by 3rd parties (although sometimes hosted in a public cloud provider). Network reliability and performance issues may emerge between your team's services and these external providers. It is important to get an accurate view of the network behavior your services perceive in their communication to these external interfaces so you can properly diagnose issues. In some cases, drilling down to specific remote IP addresses may allow you to assess the scope of a provider's problems and shift traffic to other endpoints.

Cross Zone / Region / Datacenter Traffic

Availability architectures sometimes dictate that services be run in multiple zones or regions. This can lead to complex traffic patterns within and between services. In some cases, traffic may stay within a zone and travel across a stable datacenter network. In other cases, it may be directed outside of a physical building, traversing multiple routers and potentially leaving the cloud providers backbone as well. This can affect the likelihood of network performance issues, connection failures, and packet loss. The ratios of intrazone and interzone traffic may even vary with times of day as services autoscale on less heavily loaded infrastructure. It is critical that SRE teams understand how their services communicate across zone or region boundaries so they can identify errors the network introduces.



Avoiding “Sledgehammer” Mitigations

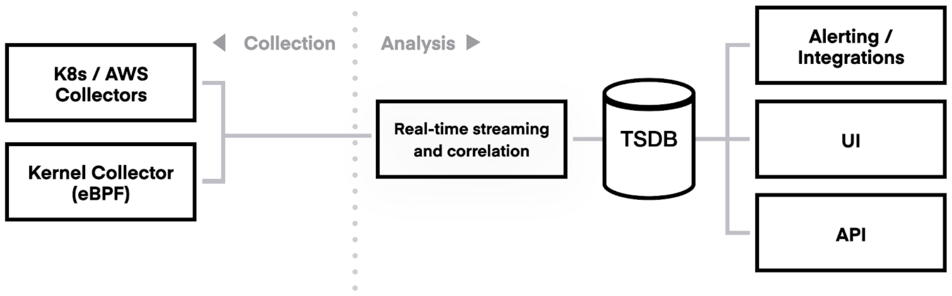
When a team is able to identify a network reliability or performance issue, a new challenge emerges around mitigation. Without a means of assessing the scope of the problem, teams are often left only with very aggressive options, such as vacating an entire zone of a public cloud. While turning off wide swaths of infrastructure will be effective, it is also incredibly expensive both in terms of engineering time and customer disruption.

It is important to measure network reliability and performance issues by service, zone, and host so it is possible to identify the subset of hosts or containers running a specific traffic flow that exhibit a network problem. This sometimes allows a lightweight mitigation strategy such as turning off a handful of instances and allowing them to relocate to another part of the datacenter.

Flowmill’s Solution

Flowmill monitors network connections from within the operating system and aggregates them to provide a service-aware view of the network.

Flowmill consists of an eBPF-based "kernel" collector on each host (cloud instance / VM / bare metal) that sends a detailed summary of communication collected directly from the operating system to the cloud-based Flowmill service. This agent leverages eBPF, an interface exposed by the Linux kernel, to automatically instrument the network stack and collect real-time data on every socket, along with associated process and container metadata. This targeted approach allows the Flowmill agent to operate with negligible overhead, typically 0.25% / CPU core.   Flowmill can collect a broad range of network data such as TCP traffic, latency, retransmission rates, UDP traffic, DNS response and error rates, HTTP response codes.



The Flowmill agents encrypt and compress data before sending it to the Flowmill service. This 'raw' data must be matched across both sides of the connection, labeled and enriched with service names, and aggregated and analyzed in real time.   This matching and enrichment is a critical part of the analysis process since it turns low level socket metrics into service and host-oriented views of system behavior.

This approach has several properties that make it ideal for network observability in distributed applications:

- This is 'ground truth' data collected passively from every process and provides a very accurate picture of what your application experiences.   It does not generate additional active, synthetic traffic.

- Unlike a traditional network tool that captures packets or IP addresses, enriching and aggregating connection data by service provides an application-aware picture of the system. In container environments like Kubernetes, pod IP addresses may recycle frequently so it's essential to provide a service-oriented view.

- By collecting and analyzing flows in real time, Flowmill enables correlatation and drilldown on physical dimensions such as zone, host, or process and logical dimensions such as service or namespace. For example, it is possible to tell whether a specific service or service to service connection was affected by a network reliability issue and then determine that problem exists only in a single zone or a limited set of hosts.

- It is also possible to observe connections to remote locations such as cloud services. In this case, Flowmill uses a combination of metadata gathered from cloud provider and passively observed DNS traffic to automatically label remote services and provide insights into their reliability.

Real World Examples

There are a wide array of real-world applications of how network visibility can improve overall observability in distributed applications. Here are a handful of real world examples.

Detection and Mitigation of cloud provider outages

A large advertising exchange running on a cloud-provider's hosted Kubernetes service was impacted by a rolling outage that affected networking within several regions in Asia.   The issue caused connection failures to a number of their advertising partners.   The cloud provider posted the issue on their status page ~33 minutes after it began and it ultimately persisted for several hours. The posting mentioned a number of regions and services affected without offering specifics on how specific services or nodes were impacted.

Flowmill was able to real-time data per service on network connections to identify the issue and avoid unnecessary investigation into potential internal causes. It allowed the team to scope the issue to identify the services, instances, and containers affected to both measure the impact and reallocate capacity to regions that have recovered.

Troubleshooting Service Performance

A large service provider who had recently moved their workloads to Kubernetes was trying to track down a performance problem between services. They had set up an Envoy-based service mesh and APM tooling. They observed extremely long tail latencies on their API calls but they could not tell if this latency was occurring in the application frameworks, the service mesh, or the physical network.

Flowmill helped explicitly break out network round trip times as measured by the Linux kernel both before and after Envoy. This provided definitive data on true network latency and helped them direct analysis away from the network infrastructure.

Observing Internet Reliability

A major entertainment provider operates numerous points of presence (POPs) throughout the country as well as several large datacenters, each running a new cluster of containerized, scale-out services. Users connect to each POP but this initiates numerous connections back to containerized services in the main datacenter. The team observed intermittent connection failures between the POPs and specific containers but could not tell if the problems were caused by true physical network issues or application level issues in their containers or orchestration layer.

Flowmill can help here by tracking every connection to each container with negligible overhead, measuring round trip time, retransmission rate, and SYN-timeout rates in real time. This provided a service-aware view of how the network is behaving and helped the team discern true infrastructure issues from application-level problems in their containers.

About Flowmill

Flowmill is a Sunnyvale, CA venture-backed company founded in 2016.

Interested in learning more?

http://www.flowmill.com

http://docs.flowmill.com

sales@flowmill.com