DOI - 10.32743/UniTech.2024.128.11.18554

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ СЕТЕВЫХ ПЛАГИНОВ ОРКЕСТРАТОРА KUBERNETES

Бойдадаев Мухаммадшахзод Нодирович
ведущий инженер по инфраструктуре,
Национальный исследовательский университет
«Высшая школа экономики,
РФ, г. Москва
E-mail: mboidadaev@gmail.com

COMPARATIVE PERFORMANCE ANALYSIS OF KUBERNETES ORCHESTRATOR NETWORK PLUG-INS

Mukhammadshakhzod Boidadaev
Lead infrastructure engineer,
National Research University Higher School of Economics,
Russia, Moscow,

АННОТАЦИЯ

В статье анализируются сетевые плагины (CNI) в Kubernetes: Flannel, Cilium, Calico и Canal. Представлены описания каждого плагина, их принцип работы. Целью является сравнительный анализ различных параметров, таких как производительность, безопасность и потребление ресурсов (ОЗУ, ЦП). На основе данных, полученных в результате анализа, в статье делается вывод о том, какой плагин является лучшим выбором среди изученных.

ABSTRACT

The article analyzes network plugins (CNI) in Kubernetes: Flannel, Cilium, Calico and Canal. Descriptions of each plugin and their principle of operation are presented. The goal is a comparative analysis of various parameters, such as performance, security, and resource consumption (RAM, CPU), with performance being the central parameter being studied. Based on the data obtained as a result of the analysis, the article concludes which plugin is the best choice among those studied.

Ключевые слова: Kubernetes, CNI, Flannel, Cilium, Calico, Canal, бенчмарк, производительность.
Keywords: Kubernetes, CNI, Flannel, Cilium, Calico, Canal, benchmark, performance.

Введение

Kubernetes – платформа, которая обеспечивает эффективную оркестрацию контейнеров, декларативный и модульный подход к развертыванию и управлению контейнерными рабочими нагрузками, упрощая разработчикам и операторам развертывание и управление своими приложениями в масштабе. Позволяет легко управлять контейнерными приложениями, автоматизировать развертывание и масштабирование, а также обеспечивать высокую доступность и отказоустойчивость приложений.

Kubernetes также предлагает динамичную экосистему инструментов и интеграций, которые делают ее мощной платформой для создания современных облачных приложений. О некоторых инструментах (плагинах) мы поговорим в нашей статье, разберем плюсы и минусы, сравним их по ряду параметров: производительности, безопасности, потреблению ресурсов.

Также в статье приведен пример мета-плагина - Multus, который поддерживает подключение всех исследуемых плагинов.

Материалы и методы исследования

Для сравнения использовались следующие сетевые плагины (CNI): Flannel, Cilium, Calico, Canal (комбинация Flannel + Calico). Анализ их параметров производился на основе данных бенчмарка этих плагинов. Для бенчмарка использовался ряд протоколов в 10 ГБитной сети: TCP, UDP, HTTP, FTP и SCP.

Тестирование проводилось на трех серверах Supermicro, подключенных через коммутатор Supermicro 10 Гбит/с. Серверы были напрямую подключены к коммутатору с помощью пассивных кабелей SFP+ DAC и были настроены в одной и той же VLAN с включенными Jumbo-кадрами (MTU 9000). В установке использовался Kubernetes 1.14.0 на Ubuntu 18.04 LTS, который работал с Docker 18.09.2.

Чтобы повысить воспроизводимость, мастер размещен на первом узле, серверная часть теста на втором сервере, а клиентская часть на третьем. Для достижения этого использовался nodeSelector в развертываниях Kubernetes.

Библиографическое описание: Бойдадаев М.Н. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ СЕТЕВЫХ ПЛАГИНОВ ОРКЕСТРАТОРА KUBERNETES // Universum: технические науки : электрон. научн. журн. 2024. 11(128). URL: *https://7universum.com/ru/tech/archive/item/18554*

Для лучшего восприятия сравнение представлено в виде линейных диаграмм, за эталон взято «голое железо».

Описание плагинов, их принцип работы

Плагин Flannel — это сетевое решение для Kubernetes, которое обеспечивает простой и эффективный способ подключения контейнеров к разным узлам в кластере.

Flannel можно использовать в различных облачных средах, таких как AWS, GCP, Azure и в локальных центрах обработки данных.

Принцип работы: назначает уникальный IP-адрес каждому модулю в кластере, позволяя им взаимодействовать друг с другом. Он работает путем создания наложения виртуальной сети поверх базовой физической сети, используя протокол VXLAN для инкапсуляции и маршрутизации сетевого трафика между модулями. Flannel также предоставляет простой API для настройки виртуальной сети, позволяя администраторам настраивать топологию сети и правила маршрутизации по мере необходимости.

Flannel использует распределенное хранилище ключей и значений, такое как etcd или Consul, для поддержания состояния сети и обеспечения согласованного представления топологии сети всеми узлами. На каждом узле работает небольшой агент, называемый «flanneld», который взаимодействует с хранилищем «ключ-значение» для получения информации о конфигурации и управления сетевым интерфейсом на хосте.

Когда контейнер запускается, он запрашивает IP-адрес у агента flanneld, работающего на хосте. Агент назначает доступный IP-адрес из сетевого пула Flannel и создает виртуальный интерфейс на узле, который соединяет пространство имен сети контейнера с оверлейной сетью Flannel.

Когда контейнер хочет связаться с другим контейнером на другом узле, он отправляет трафик на IP-адрес шлюза Flannel, который является виртуальным IP-адресом, назначенным каждому узлу в кластере. Агент Flannel на принимающем узле направляет трафик на виртуальный интерфейс целевого контейнера, а сетевой стек контейнера обрабатывает все остальное.

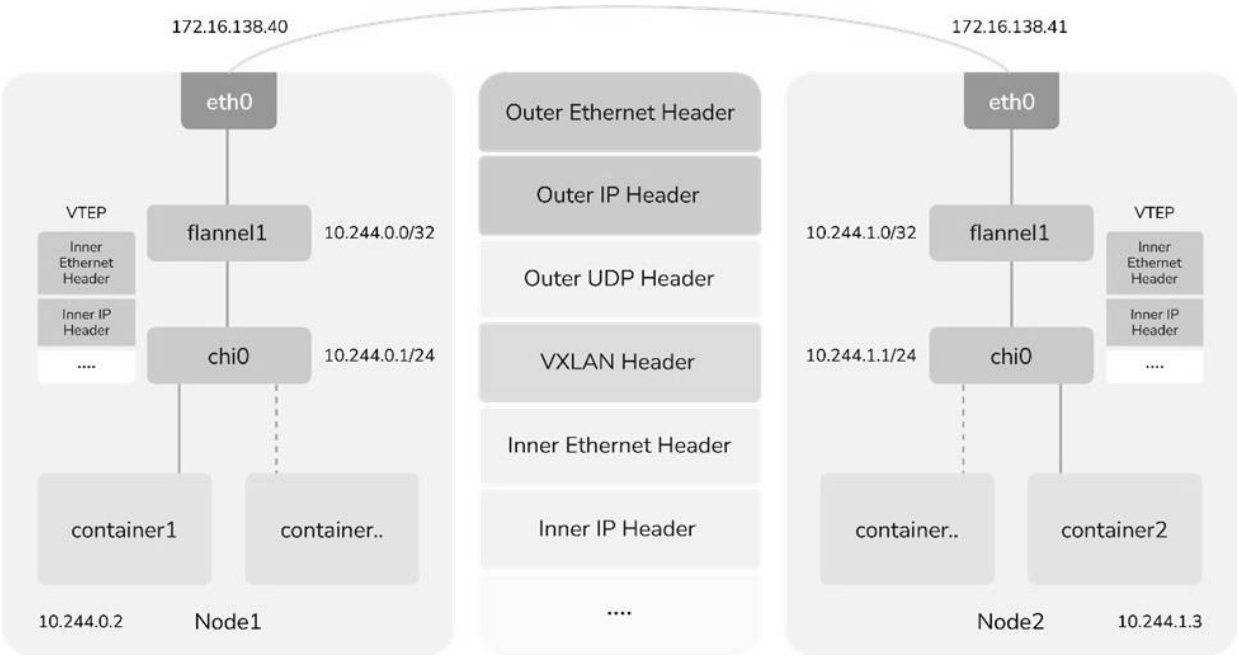

Рисунок 1. Принцип работы плагина Flannel

Cilium — это подключаемый модуль сети и безопасности для контейнерных приложений. Он предоставляет прозрачные сетевые службы и службы безопасности на основе намерений для контейнеров, упрощая применение политик и повышая прозрачность в крупномасштабных развертываниях контейнеров. Cilium использует технологии ядра Linux, такие как eBPF, для эффективной обработки сетевого трафика и применения политик на сетевом и прикладном уровнях. Он также интегрируется с Kubernetes для обеспечения динамического обнаружения сервисов и автоматического обновления политик.

Обычно он используется в облачных средах, таких как Amazon Web Services (AWS), Google * Cloud Platform (GCP) и Microsoft Azure для обеспечения безопасной связи в сети и микросервисов.

Принцип работы: обеспечивает безопасное сетевое подключение между службами приложений с использованием технологии eBPF (расширенный фильтр пакетов Berkeley). Он использует комбинацию политик уровня 3/4 и уровня 7 для принудительной изоляции и сегментации сети, а также обеспечивает видимость сетевого трафика и может использоваться для применения политик детального контроля доступа на основе меток и пространств имен Kubernetes.

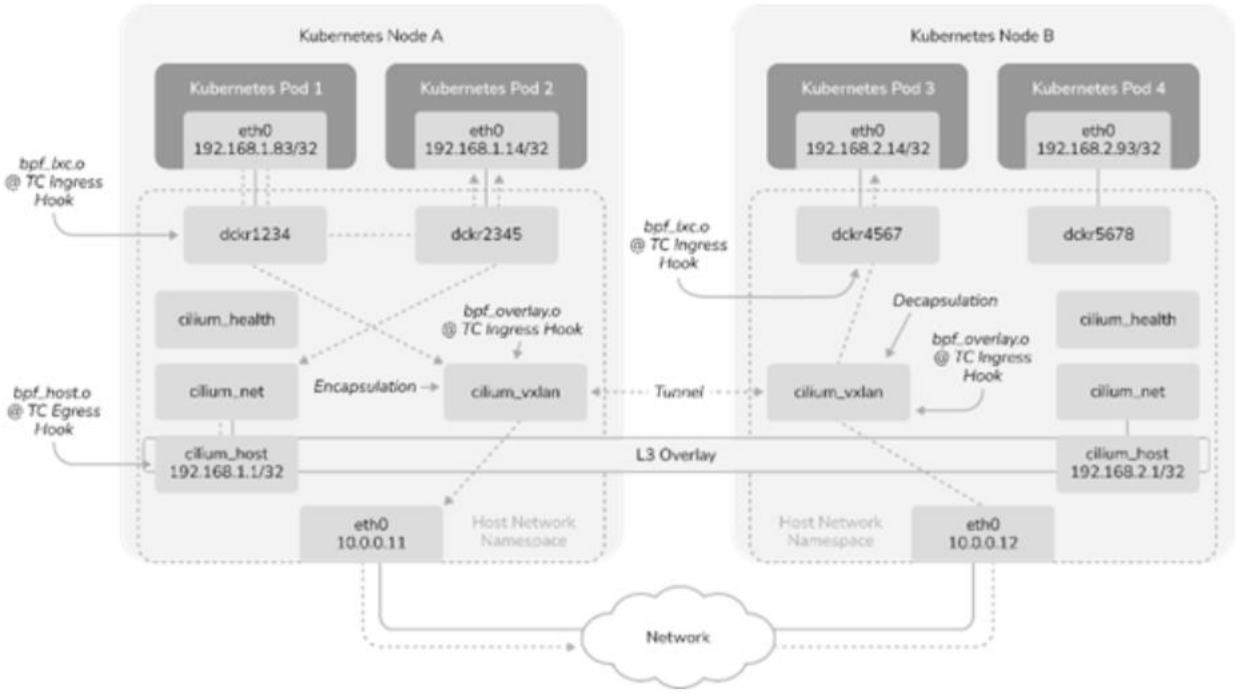

Рисунок 2. Принцип работы плагина Cillium

Calico – плагин, используемый для обеспечения сетевого подключения и безопасности для контейнерных приложений в среде Kubernetes. Он использует чистый подход уровня 3 к сети, что означает, что он избегает сложности и накладных расходов, связанных с оверлейной сетью. Calico также предлагает расширенные функции безопасности, такие как принудительное применение сетевых политик, которые позволяют администраторам определять и применять подробные правила о том, как различные наборы модулей могут взаимодействовать друг с другом.

Он, так же, как и предыдущие плагины, используется в облачных средах, таких как AWS, GCP и Azure, а также в локальных центрах обработки данных.

Принцип работы: Calico обеспечивает связь между контейнерами на нескольких узлах в кластере путем создания наложения виртуальной сети.

Он использует BGP (протокол пограничного шлюза) для распределения маршрутов и поддерживает несколько сетевых режимов, таких как IP-контейнер и хост-контейнер. Calico также обеспечивает применение сетевых политик, позволяя администраторам определять и применять правила брандмауэра и элементы управления доступом.

Когда контейнер запускается на узле Kubernetes, плагин Calico создает виртуальный Ethernet-интерфейс для контейнера и назначает ему IP-адрес. Calico определяет сетевые политики с помощью объектов Kubernetes NetworkPolicy, которые могут указывать разрешенный или запрещенный трафик на основе исходного и конечного IP-адресов, портов и протоколов.

Calico использует алгоритм распределенной маршрутизации, чтобы обеспечить эффективную маршрутизацию трафика между контейнерами, даже если они работают разных узлах. Плагин также предоставляет расширенные сетевые функции, такие как применение сетевой политики, балансировка нагрузки и формирование трафика.

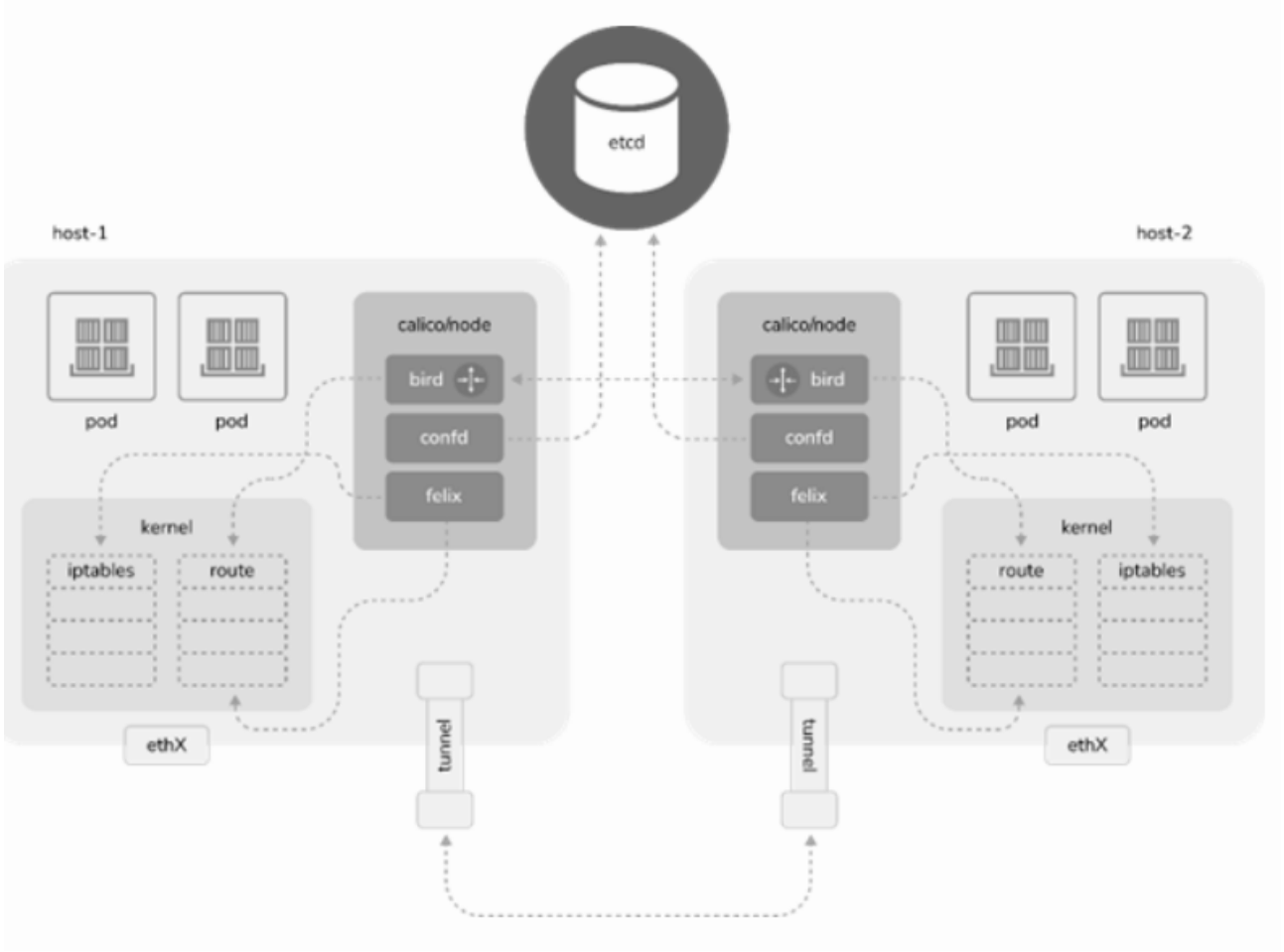

Рисунок 3. Принцип работы плагина Calico

Плагин Canal использует сетевые технологии Calico и Flannel для обеспечения безопасной, масштабируемой и высокопроизводительной сети. Canal позволяет модулям в кластере Kubernetes беспрепятственно и безопасно взаимодействовать друг с другом независимо от базовой инфраструктуры. Он также обеспечивает межкластерную связь и поддерживает сетевые политики для обеспечения контроля доступа и изоляции.

Так же, как и другие плагины используется в различных облачных средах и локальных центрах обработки данных, включая AWS, GCE, Azure, OpenStack, а также на «голом железе». Плагин позволяет определять и применять сетевые политики, помогая повысить безопасность и обеспечить сетевую изоляцию между модулями.

Принцип работы: использует компоненты Flannel и Calico для обеспечения возможностей работы в сети и применения сетевых политик. Flannel обеспечивает оверлейную сеть, а Calico обеспечивает применение сетевой политики. Плагин Canal устанавливается как набор демонов на каждом узле в кластере Kubernetes и отвечает за настройку сетевых политик и политик сетевой безопасности для модулей, работающих в кластере.

Canal устанавливает виртуальный сетевой интерфейс на каждом узле Kubernetes и назначает ему IP-адрес. Затем он создает виртуальный сетевой интерфейс на каждом модуле и назначает ему IP-адрес из сети Flannel.

Подключаемый модуль Canal также обеспечивает применение сетевой политики с помощью объектов Kubernetes NetworkPolicy. Он позволяет вам определять правила, которые контролируют входящий и исходящий сетевой трафик к модулям на основе различных критериев, таких как IP-адреса, порты и протоколы.

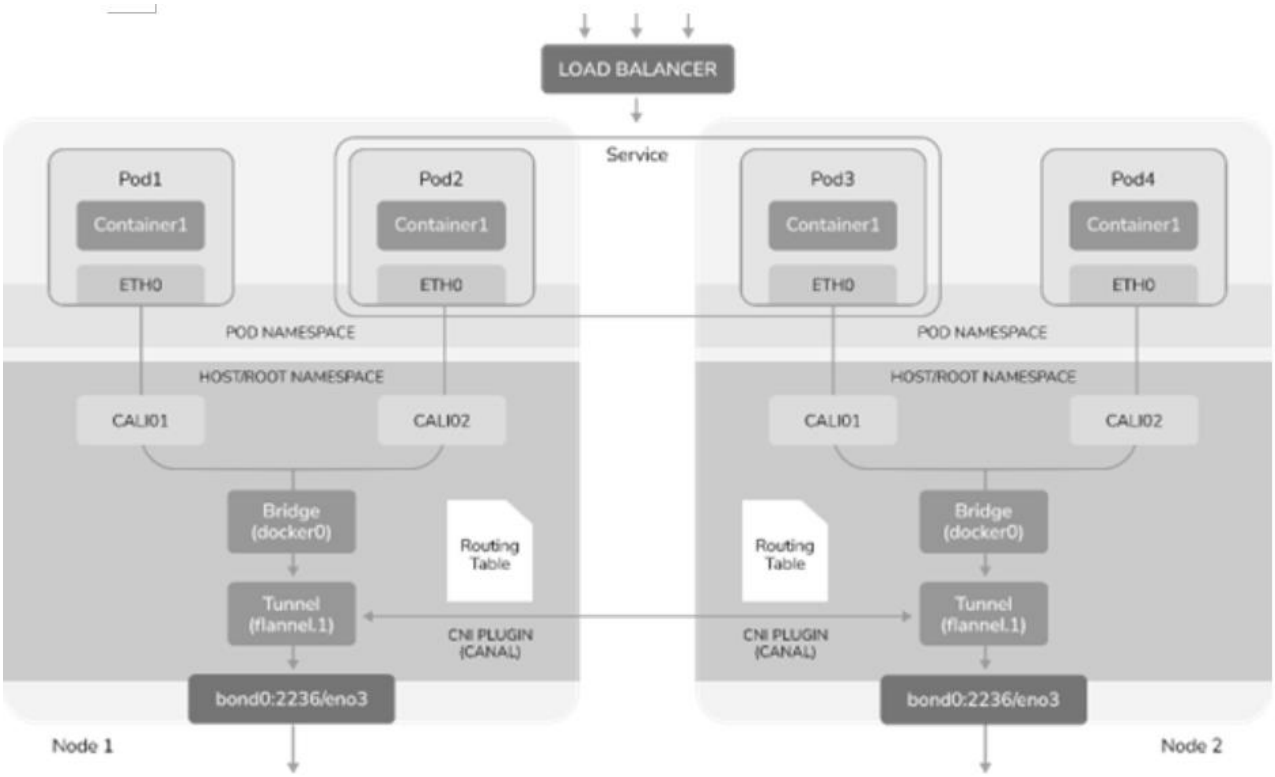

Рисунок 4. Принцип работы плагина Canal

Плагин Multus обеспечивает возможности работы с несколькими сетями в одном кластере Kubernetes. Это позволяет модулям одновременно подключаться к нескольким сетям, что упрощает развертывание сложных приложений, требующих доступа к нескольким сетям.

Multus позволяет создавать и управлять несколькими сетевыми интерфейсами для каждого модуля, предоставляя больше гибкости и полномочий при настройке сети.

Кроме того, Multus поддерживает широкий спектр сетевых плагинов, что позволяет пользователям выбирать лучший плагин для каждого сетевого интерфейса.

Multus можно использовать в различных облачных средах и локальных центрах обработки данных, поддерживающих Kubernetes.

Принцип работы:

Принцип работы Multus заключается в использовании API Kubernetes Network Attachment Definition (NAD) для определения дополнительных сетевых интерфейсов. Каждый подключенный сетевой интерфейс реализован в виде отдельного плагина CNI, который можно настраивать независимо.

При создании модуля подключаемый модуль Multus создает контейнер, в котором выполняются все подключаемые модули CNI для каждого сетевого интерфейса. Эти подключаемые модули отвечают за настройку сетевых интерфейсов, назначение IP-адресов и управление сетевыми пространствами имен и таблицами маршрутизации.

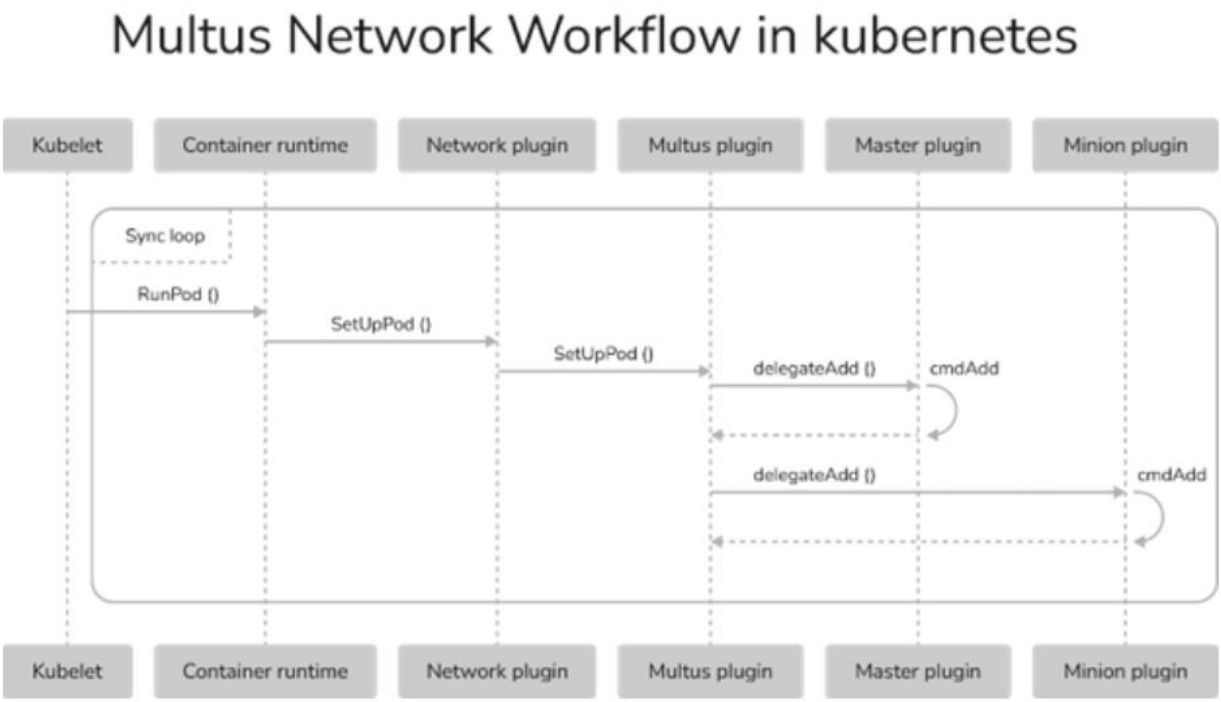

Рисунок 5. Принцип работы плагина Multus

Исследование производительности сетевых плагинов (CNI) Kubernetes

Тестирование производительности проводилось по ряду протоколов: TCP, UDP, HTTP, FTP и SCP. Для эталонного сравнения использовался бенчмарк на «голом железе». Числовые данные на диаграмме отображены в виде Мбит/с.

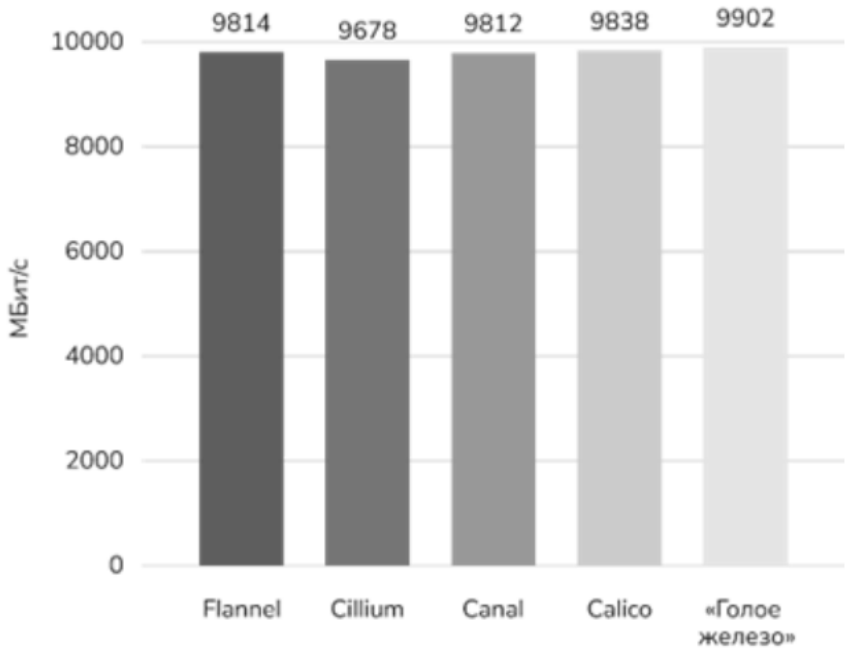

Рисунок 6. Бенчмарк CNI в 10Гбитной сети по протоколу TCP

По результатам данного тестирования видно, что все CNI примерно одинаково справились с задачей. Лидирует в данном бенчмарке плагин Calico, хоть и с небольшим отрывом от Canal и Flannel. Cillium немного отстает на 134 Мбит/с от ближайшего от него результата у Canal.

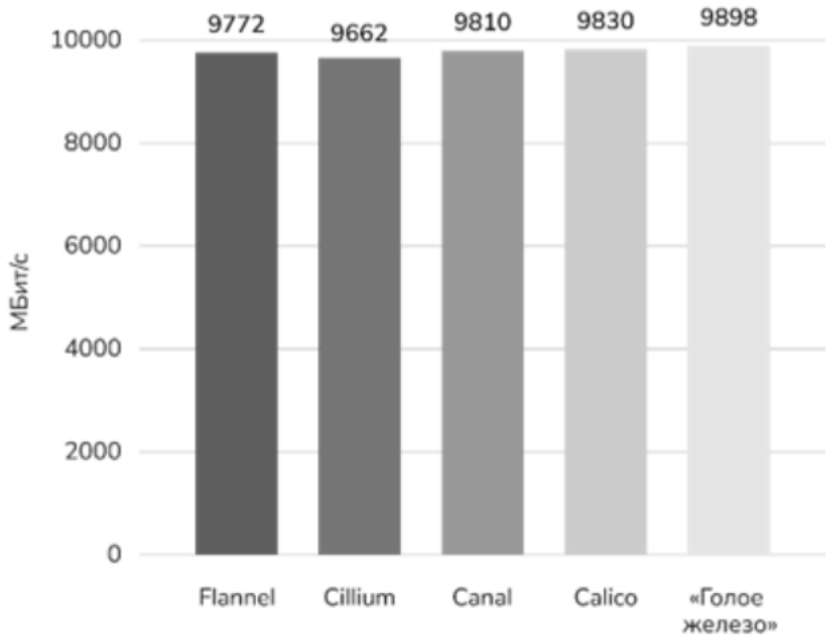

Рисунок 7. Бенчмарк CNI в 10Гбитной сети по протоколу UDP

В бенчмарке по протоколу UDP сохраняется почти такая же картина, как и при тестировании по протоколу TCP: сохраняется лидерство сетевого плагина Calico, а Cillium отстает, общая производительность в сравнении с «голым железом» также показывает хорошие результаты.

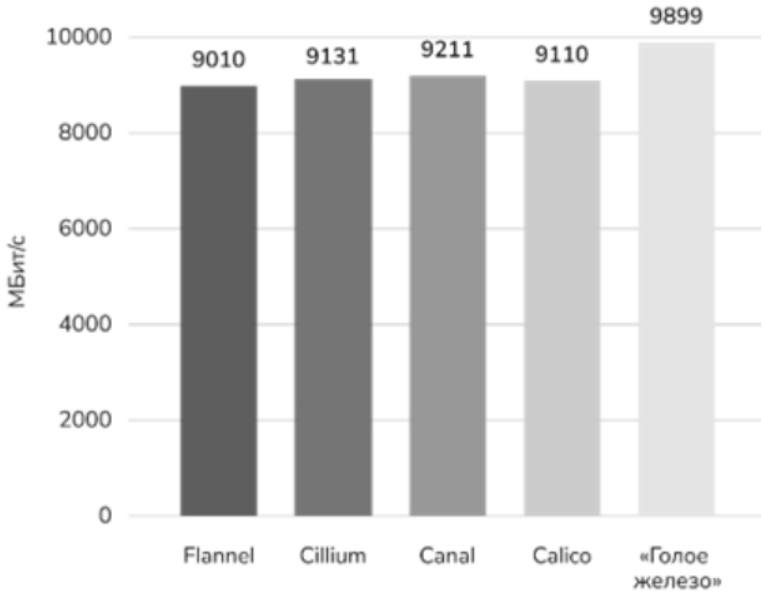

Рисунок 8. Бенчмарк плагинов Kubernetes в 10Гбитной сети по протоколу HTTP

При переходе на тестирование по протоколу HTTP ситуация меняется, общая производительность немного просела, в лидеры выбивается гибридный плагин Canal, а его составляющие Calico и Flannel располагаются на третьем и четвертом месте по производности соответственно. Прежде располагающийся в отстающих Cillium занимает второе место по производительности.

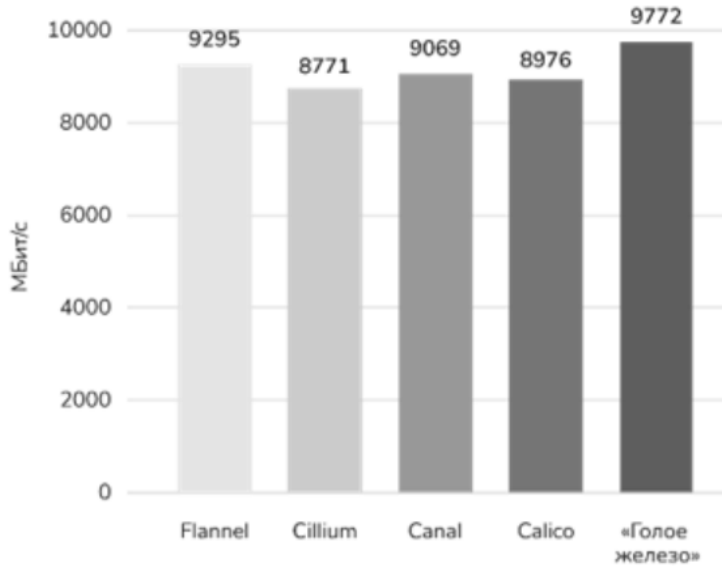

Рисунок 9. Бенчмарк плагинов Kubernetes в 10Гбитной сети по протоколу FTP

Бенчмарк протокола FTP показывает следующие результаты: есть проседание в общей производительности, а также локально сильно просел плагин Cillium. В лидерах по производительности находится CNI Flannel.

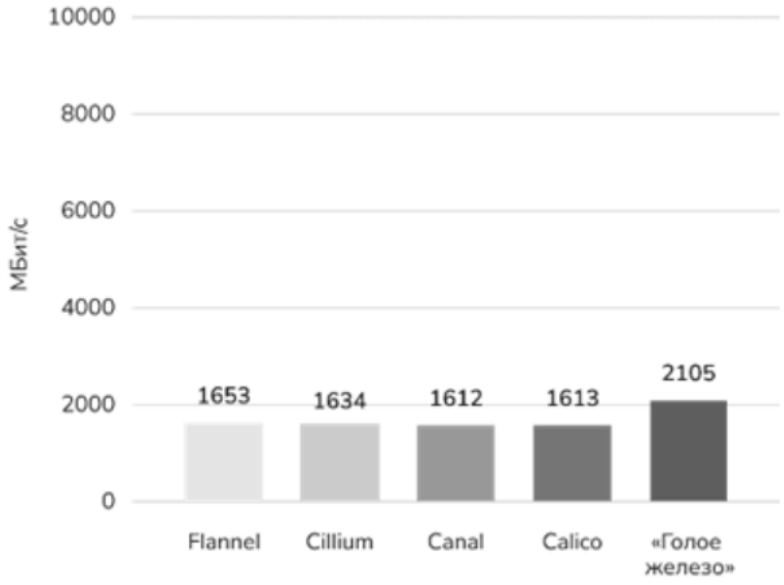

Рисунок 10. Бенчмарк плагинов Kubernetes в 10Гбитной сети по протоколу SC

Результаты данного бенчмарка нам показывает, как сильно падает производительность не только плагинов, но и «голого железа» при использовании протокола SCP, который использует защищенную оболочку SSH для передачи данных. В целом все плагины одинаково справились с задачей, но Flannel справился немного лучше всех.

Тестирование CNI по потребности в ресурсах и безопасности

Безопасность оценивалась по двум параметрам: способности к шифрованию данных и реализации сетевой политики. Среди исследуемых плагинов, только Cillium шифрует данные. Он обеспечивает шифрование данных при передаче с использованием протокола Transport Layer Security (TLS). Cilium использует взаимный TLS (mTLS) для безопасной связи между микросервисами.

Если говорить про реализацию сетевой политики, то все исследуемые CNI справляются с задачей настройки правил как Ingress, так и Egress, кроме плагина Flannel, у которого вообще нет сетевых политик.

Перейдем к сравнению CNI по потреблению ресурсов: для анализа возьмем потребление RAM и CPU. Для сравнения кроме «голого железа» в диаграмме также приведены показатели потребления оркестратором Kubernetes без использования плагинов.

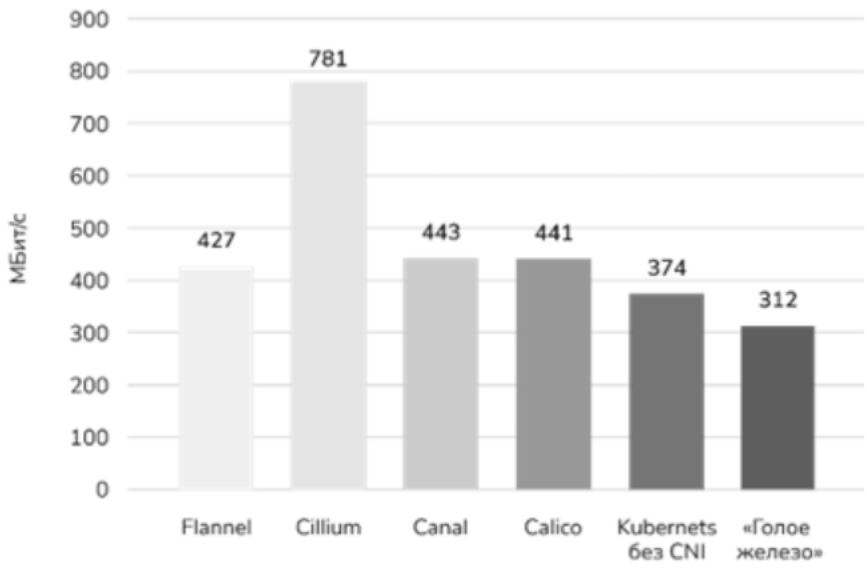

Рисунок 11. Бенчмарк потребления RAM плагинов Kubernetes в 10Гбитной сети

Среди всех наименее затратным оказался Flannel. Плагины Calico и Canal имеют потребление RAM так же на хорошем уровне. Cillium потребляет почти в 1,83 раза больше, чем лидер данного бенчмарка Flannel.

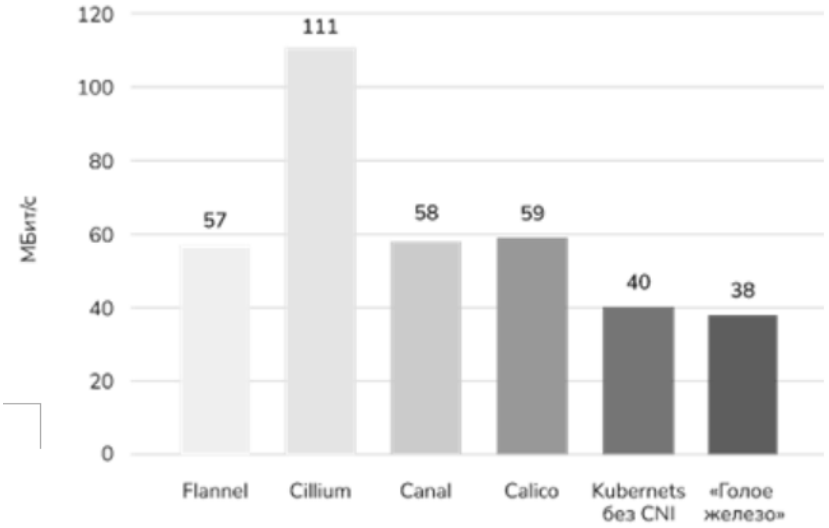

Рисунок 12. Бенчмарк потребления CPU плагинов Kubernetes в 10Гбитной сети

Результаты по тестированию потребления CPU показывают похожие соотношения. Лидирует Flannel, с показателем в 57 промилле. Cillium в этот раз потребляет в 1,95 раз больше, чем Flannel. Остальные два плагина приближены имеют хорошие результаты потребления.

Заключение

На основании проведенных бенчмарков можно сделать вывод, что наиболее производительными можно считать CNI Flannel и Calico, так как оба в части тестов показали лучшие показатели производительности. Так же эту точку зрения подкрепляют результаты анализа потребления RAM и CPU.

Высокая производительность плагина Flannel вероятно связана с его компактностью, а также его совместимостью с большим количеством архитектур. Так же к вероятным причинам стоит добавить способность этого плагина автоматически определять MTU.

Что касается безопасности из двух лучших стоит использовать плагин Calico. Высокая производительность этого плагина обусловлена тем, что он пропускает только тот трафик, который вы устанавливаете сами.

**По требованию Роскомнадзора информируем, что иностранное лицо, владеющее информационными ресурсами Google является нарушителем законодательства Российской Федерации – прим. ред.*

Список литературы:

1. Ковалев М.Г. ТРАССИРОВКА СЕТЕВЫХ ПАКЕТОВ В ЯДРЕ LINUX С ИСПОЛЬЗОВАНИЕМ EBPF // Труды ИСП РАН. 2020. №3. URL: https://cyberleninka.ru/article/n/trassirovka-setevyh-paketov-v-yadre-linux-s-ispolzovaniem-ebpf
2. Липатова Софья Евгеньевна, Белов Юрий Сергеевич ПРАКТИКИ ОБЕСПЕЧЕНИЯ КИБЕРБЕЗОПАСНОСТИ В KUBERNETES // E-Scio. 2022. №1 (64). URL: https://cyberleninka.ru/article/n/praktiki-obespecheniya-kiberbezopasnosti-v-kubernetes
3. Урманцева Н.Р., Хитрень Д.В. ПРИМЕНЕНИЕ ВИРТУАЛЬНЫХ КОНТЕЙНЕРОВ ПРИ СОЗДАНИИ МЕДИЦИНСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ // ВК. 2021. №2 (42). URL: https://cyberleninka.ru/article/n/primenenie-virtualnyh-konteynerov-pri-sozdanii-meditsinskih-informatsionnyh-sistem
4. Фаррахов И.Г., Якупов И.М. АВТОМАТИЗИРОВАННЫЙ ИНСТРУМЕНТАРИЙ РАЗВЕРТЫВАНИЯ ОБЛАЧНЫХ СЕРВИСОВ // Мировая наука. 2021. №2 (47). URL: https://cyberleninka.ru/article/n/avtomatizirovannyy-instrumentariy-razvertyvaniya-oblachnyh-servisov
5. https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-april-2019-4a9886efe9c4 - Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network
6. https://kubernetes.io/docs/concepts/overview/ - официальный сайт платформы Kubernetes