УДК 004.77, 004.62

# Исследование производительности различных имплементаций Ingress-контроллеров в кластере Kubernetes

**В статье представлен анализ производительности базовых имплементаций Ingress-контроллеров Traefik, Envoy (Contour), HAProxy, NGINX и NGINX Inc. при равных условиях. Анализ производительности выполнялся по предложенной авторами методике и алгоритмам, исходя из опыта работы с высоконагруженными системами. Результатом исследования является выбор наиболее эффективной имплементации, которая лучше всего подходит для широкого применения в высоконагруженных системах без дополнительной конфигурации.**

**А. В. ШУЛЯК,**
ведущий инженер по стабильности и производительности платформы, Divido Financial Services Ltd. (Кембридж, Великобритания)

**А. Г. САВЕНКО,**
аспирант, магистр технических наук, старший преподаватель кафедры информационных систем и технологий Института информационных технологий

Белорусский государственный университет информатики и радиоэлектроники

***Ключевые слова:***
*производительность высоконагруженных систем, система оркестрации Kubernetes, развертывание контейнеризированных приложений, масштабирование контейнеризированных приложений, обратный прокси-сервер.*

**Введение.** В настоящее время тенденции автоматизации и цифровизации всех сфер человеческой деятельности являются едва ли не самыми главными в экономиках всех стран мира. Отрасль информационно-коммуникационных технологий в этом плане является передовой. Вполне логичным является автоматизация производственных и бизнес-процессов компаний, продуктами которых является программное обеспечение для автоматизации других отраслей народного хозяйства. Ввиду большого спроса на автоматизацию жизненного цикла программных продуктов возникло и огромное количество предложений среди ведущих мировых компаний.

Наиболее популярным и развивающимся программным обеспечением (ПО) для автоматизации развертывания, масштабирования и управления контейнеризированными приложениями является Kubernetes, поддерживающее технологии Docker, RKT, а также аппаратную виртуализацию [1], изначально разработанное для собственных нужд компанией Google. Kubernetes Ingress является собственным ресурсом, который обеспечивает реализацию балансировки нагрузки, установление SSL-соединения и маршрутизации запросов на основе доменных имен. Ingress-контроллер, в свою очередь, осуществляет типизацию и реализацию абстракций ресурса Ingress (рис.1).

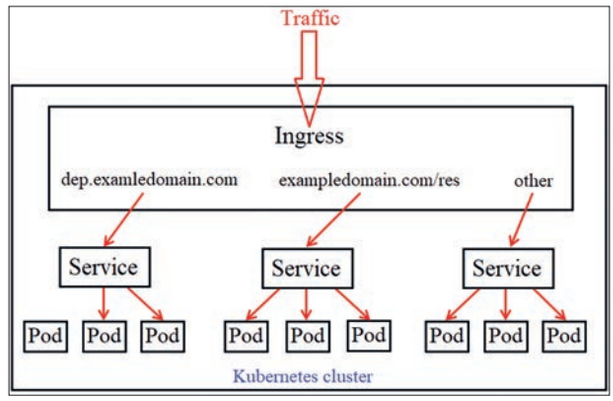



**Рисунок 1 – Кластер Kubernetes**

В связи с тем, что Kubernetes является инструментом с открытым исходным кодом, существует достаточно большое количество реализаций Ingress-контроллеров. В результате чего возникает проблема анализа производительности обратных прокси-серверов, на основе которого можно было бы осуществить выбор в сторону одной или другой реализации [2]. Исследование, описанное в данной работе, позволяет определить явные отличия в производительности различных имплементаций наиболее популярных Ingress-контроллеров.

**Методика проведения исследования.** В ходе исследования была проанализирована производительность следующих

наиболее перспективных и популярных Ingress-контроллеров: Traefik, NGINX, NGINX Inc., Envoy (управляющий слой Contour), HAProxy.

Общие исходные данные для проведения анализа производительности:

– одинаковые наборы тестов каждого Ingress-контроллера выполняются на протяжении 3600 секунд;

– реализация алгоритмов тестирования осуществляется через 30 секунд после начала тестирования;

– тестирование производится на мощностях провайдера облачных вычислительных ресурсов Linode;

– конфигурация кластера Kubernetes для тестирования: 2 мастер-узла и 6 вычислительных узлов, отсутствие любой побочной нагрузки [3] на вычислительном узле, на котором располагается тестируемый Ingress-контроллер;

– конфигурация мастер-узла: 1 ядро центрального процессора и 2 ГБ оперативной памяти;

– конфигурация вычислительного узла: 4 ядра центрального процессора и 8 ГБ оперативной памяти;

– конфигурация настроек Ingress-контроллеров не модифицируется и находится в состоянии, поставляемом разработчиком;

– все тестирование производится только для протокола HTTPS, то есть для каждого соединения устанавливается TLS-соединение (поскольку в современном мире не выполняется передача данных в открытом виде без шифрования);

– в качестве системы назначения всех запросов используется эхо-сервер, возвращающий в качестве ответа информацию, содержащуюся в запросе. Это позволяет исключить фактор дополнительной нагрузки на процессор, создаваемой обработкой содержимого запроса [4];

– нагрузка создается при помощи инструмента hey с открытым исходным кодом для нагрузочного тестирования;

– каждый набор тестов производится в следующих двух вариантах:

1) одногенераторный – запускается один генератор трафика, который создает 250 одновременных соединений;

2) пятигенераторный – запускается пять генераторов трафика, которые создают по 50 соединений каждый.

Анализируемыми данными при проведении тестирования производительности Ingress-контроллеров являются:

• Средняя нагрузка на процессор в течение тестирования (загрузка CPU).

• Количество HTTP-кодов состояний «успешно» и «ошибка сервера».

• Перцентиль (задержка ответа на запрос).

• Количество выполняемых запросов в секунду.

Ниже приведены алгоритмы выполнения тестов производительности.

*1) Тест изменения количества экземпляров обратных прокси-серверов.*

шаг 1: производится увеличение количества экземпляров с 5 до 7;

шаг 2: производится замер производительности на протяжении 30 секунд;

шаг 3: производится уменьшение количества экземпляров с 7 до 5;

шаг 4: производится замер производительности на протяжении 30 секунд;

шаги 1–4 повторяются три раза.

*2) Тест изменения конфигурации обратных прокси-серверов под нагрузкой.*

шаг 1: ожидание завершения последнего теста по изменению количества прокси-серверов;

шаг 2: производится замер производительности на протяжении 30 секунд;

шаг 3: добавляются CORS-заголовки;

шаг 4: производится замер производительности на протяжении 30 секунд;

шаг 5: удаление CORS-заголовков;

шаг 6: производится замер производительности на протяжении 30 секунд;

шаг 7: добавление маршрута перенаправления запроса;

шаг 8: производится замер производительности на протяжении 30 секунд;

шаг 9: удаление маршрута перенаправления запроса.

**Результаты исследования и их обсуждение.** В результате проведенного исследования были получены данные о производительности Ingress-контроллеров в двух вариантах тестирования: одногенераторном и пятигенераторном.

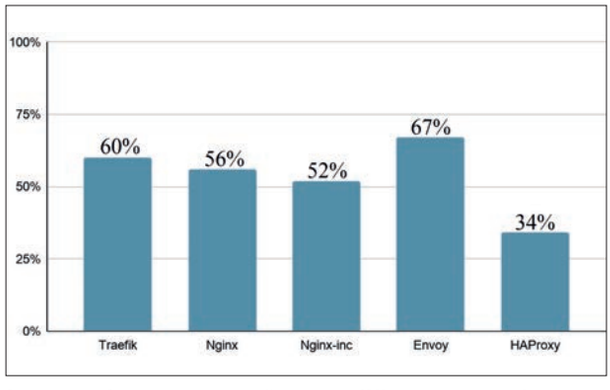



**Рисунок 2 – Результаты средней загрузки CPU для различных Ingress-контроллеров**

**Таблица 1 – Результаты тестирования на количество HTTP-кодов состояний**

| Название Ingress-контроллера | HTTP-код состояния | | | |
|---|---|---|---|---|
| | Код 200* | Код 502** | Код 503*** | Код 504**** |
| | Количество полученных кодов состояний | | | |
| Envoy | 999902 | 0 | 98 | 0 |
| HAProxy | 1000000 | 0 | 0 | 0 |
| Nginx-inc | 999962 | 34 | 0 | 4 |
| Nginx | 999951 | 42 | 0 | 7 |
| Traefik | 994863 | 5137 | 0 | 0 |

*Результаты одногенераторного тестирования.* Полученные результаты средней загрузки CPU представлены на рис. 2.

Как видно из полученных данных, при равных условиях наилучшую производительность с точки зрения средней загрузки CPU имеет Ingress-контроллер HAProxy.

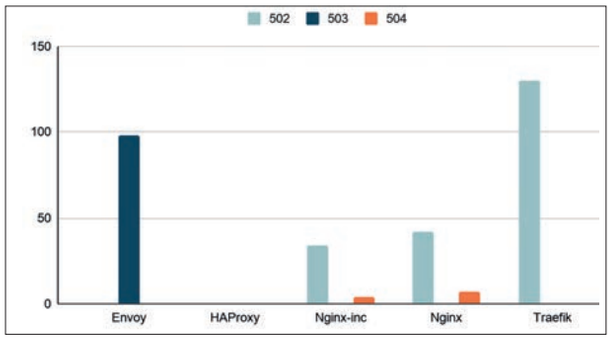



**Рисунок 3 – Количество и тип ошибок сервера для различных Ingress-контроллеров**

Результаты тестирования на количество HTTP-кодов состояний «успешно» и «ошибка сервера» представлены в таблице 1.

Диаграмма, иллюстрирующая количество и тип ошибок сервера, представлена на рис. 3.

Незначительное количество ошибок относительно количества успешных запросов (код ошибки 504 для Ingress-контроллеров Nginx и Nginx-inc) можно не учитывать как статистическую погрешность.

Результаты тестирования задержки на запрос (перцентиль) представлены в таблице 2.

Задержка (перцентиль), как статистическая характеристика, средние значения которой (включая среднее арифметическое значение) в практическом применении имеют множество достоинств [5].

Диаграмма, иллюстрирующая перцентиль с разбиением на долю от всех задержек для различных Ingress-контроллеров, представлена на рис. 4.

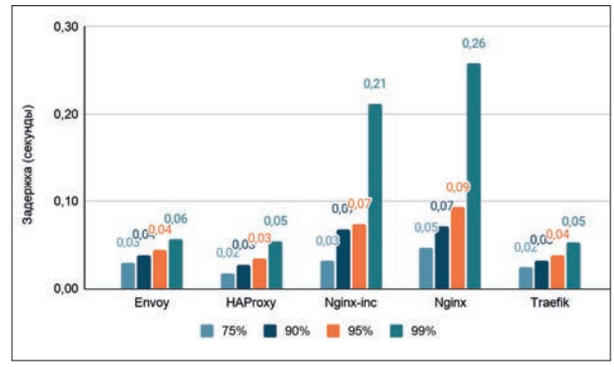



**Рисунок 4 – Перцентиль различных Ingress-контроллеров**

Как видно из полученных результатов, наименьшие задержки имеют Ingress-контроллеры HAPRoxy и Traefik. Выделяются относительно высокой задержкой только контроллеры на базе обратных прокси Nginx, которая обуславливается общей стандартной конфигурацией.

Результаты тестирования по количеству выполняемых запросов в секунду представлены на рис. 5.

**Таблица 2 – Результаты тестирования задержки на запрос (перцентиль)**

| Название Ingress-контроллера | Доля задержек, % | | | |
|---|---|---|---|---|
| | 75 | 90 | 95 | 99 |
| | Время задержки, с | | | |
| Envoy | 0,0293 | 0,0385 | 0.0443 | 0.0569 |
| HAProxy | 0,0179 | 0,0270 | 0.0344 | 0.0539 |
| Nginx-inc | 0,0317 | 0,0682 | 0.0740 | 0.2116 |
| Nginx | 0,0467 | 0,0712 | 0.0930 | 0.2576 |
| Traefik | 0,0243 | 0,0326 | 0.0386 | 0.0534 |

**Примечание:**
* Код 200 – запрос клиента обработан успешно и ответ сервера содержит затребованные данные.
** Код 502 – сервер, действуя как шлюз или прокси, получил неверный ответ от восходящего сервера.
*** Код 503 – сервер не готов обработать данный запрос. Часто причиной этого оказывается закрытие сервера для технических работ или его перегрузка.
**** Код 504 – ошибка возникает, когда один сервер не получает своевременный ответ от другого сервера, который действует как шлюз или прокси. То есть сервер не смог выполнить запрос в течение заданного периода времени.

**Таблица 3 – Результаты тестирования на количество HTTP-кодов состояний**

| Название Ingress-контроллера | HTTP-код состояния | | | |
|---|---|---|---|---|
| | Код 404* | Код 502 | Код 503 | Код 504 |
| | Количество полученных кодов состояний | | | |
| Envoy | 0 | 0 | 136 | 0 |
| HAProxy | 439 | 0 | 0 | 0 |
| Nginx-inc | 0 | 106 | 0 | 6 |
| Nginx | 0 | 117 | 0 | 6 |
| Traefik | 0 | 660 | 0 | 0 |

Примечание: *Код 404 – ошибка ответа сервера назначения – «страница не найдена».

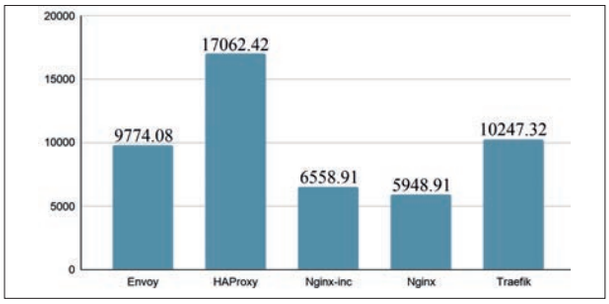



**Рисунок 5 – Количество выполняемых запросов в секунду для различных Ingress-контроллеров**

Из полученных результатов следует, что существенно большее число запросов в секунду выполняет Ingress-контроллер HAPRoxy.

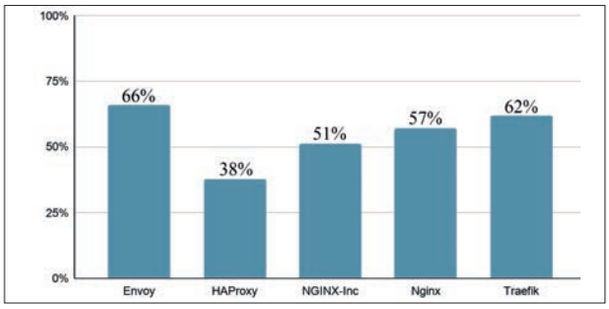



**Рисунок 6 – Результаты средней загрузки CPU для различных Ingress-контроллеров**

*Результаты пятигенераторного тестирования.* Полученные результаты средней загрузки CPU представлены на рис. 6.

Полученные данные свидетельствуют, что при равных условиях наилучшую производительность с точки зрения средней загрузки CPU имеет Ingress-контроллер HAProxy (38 %).

Результаты тестирования на количество HTTP-кодов состояний «успешно» и «ошибка сервера» представлены в таблице 3.

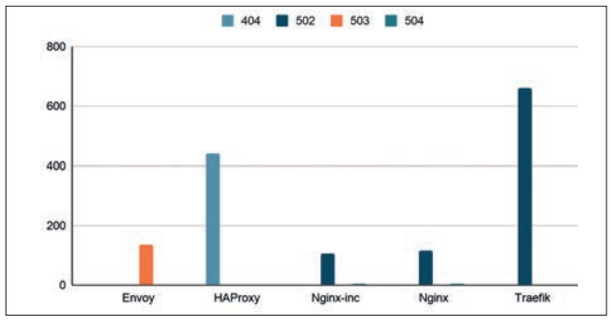



**Рисунок 7 – Количество и тип ошибок сервера для различных Ingress-контроллеров**

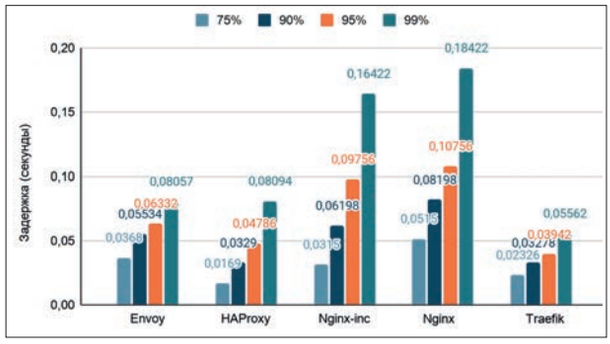



**Рисунок 8 – Перцентиль различных Ingress-контроллеров**

Диаграмма, иллюстрирующая количество и тип ошибок сервера, представлена на рис. 7.

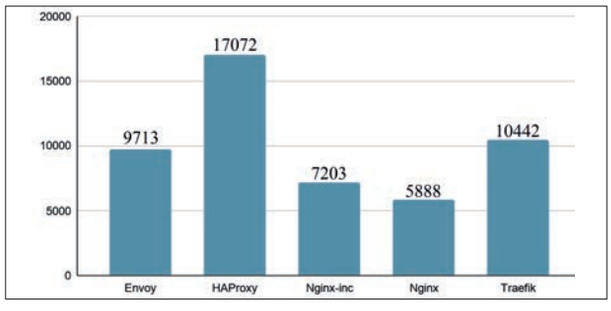
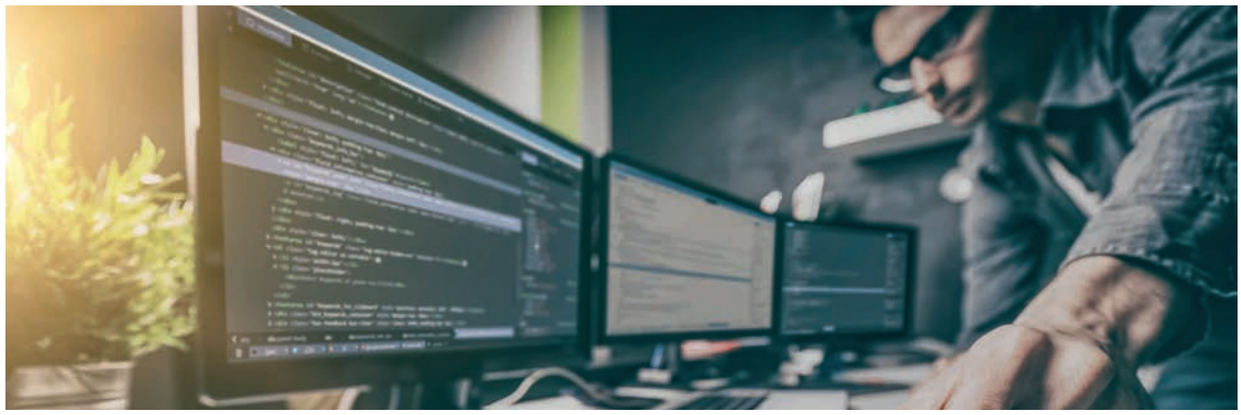



**Рисунок 9 – Количество выполняемых запросов в секунду для различных Ingress-контроллеров**

**Таблица 4 – Результаты тестирования задержки на запрос (перцентиль)**

| Название Ingress-контроллера | Доля задержек, % | | | |
|---|---|---|---|---|
| | 75 | 90 | 95 | 99 |
| | Время задержки, с | | | |
| Envoy | 0.0368 | 0.05534 | 0.06332 | 0.08057 |
| HAProxy | 0.0169 | 0.0329 | 0.04786 | 0.08094 |
| Nginx-inc | 0.0315 | 0.06198 | 0.09756 | 0.16422 |
| Nginx | 0.0515 | 0.08198 | 0.10756 | 0.18422 |
| Traefik | 0.02326 | 0.03278 | 0.03942 | 0.05562 |



Результаты тестирования задержки на запрос (перцентиль) представлены в таблице 4.

Диаграмма, иллюстрирующая перцентиль с разбиением на долю от всех задержек для различных Ingress-контроллеров, представлена на рис. 8.

Результаты тестирования по количеству выполняемых запросов в секунду представлены на рис. 9.

**Выводы.** В ходе исследования была проанализирована производительность наиболее перспективных и популярных Ingress-контроллеров. Выполнены одногенераторное и пятигенераторное нагрузочное тестирование. На основании полученных результатов можно сделать вывод, что Ingress-контроллер построенный на обратном прокси-сервере HAProxy, имеет наилучшую производительность с точки зрения количества запросов в секунду, а также наименьшую нагрузку на центральный процессор, что обеспечивает дополнительную емкость по ресурсам для большего количества запросов. С точки зрения задержки в обработке запросов наилучшим образом показывает себя Ingress-контроллер на базе обратного прокси-сервера Traefik. Зарекомендовавшие себя на рынке Ingress-контроллеры на базе обратных прокси-серверов NGINX имеют наихудшие показатели в стандартной конфигурации без модификаций. Однако разница между стандартной конфигурацией от разработчика и модификациями конфигураций от специалистов в данной области показывает значимость настроек для значений показателей производительности данного сервера, что означает достаточно большую емкость для модификаций [6].

**ЛИТЕРАТУРА**

1. **Potdar, A. M.** et al. Performance evaluation of docker container and virtual machine // Procedia Computer Science. – 2020. – Т. 171. – P. 1419–1428. – Режим доступа: https://doi.org/10.1016/j.procs.2020.04.152. – Дата доступа: 05.05.2021.
2. **Pal, H., Nagar, C.** Hybrid algorithm for performance improvement in Cloud load balancing // Materials Today: Proceedings. – 2021. – Режим доступа: https://doi.org/10.1016/j.matpr.2021.01.876. – Дата доступа: 05.05.2021.
3. **Hu, Y.** et al. Concurrent container scheduling on heterogeneous clusters with multi-resource constraints //Future Generation Computer Systems. – 2020. – Т. 102. – P. 562–573. – Режим доступа: https://doi.org/10.1016/j.future.2019.08.025. – Дата доступа: 06.05.2021.
4. **Weiqing, C.** et al. Measuring web page complexity by analyzing TCP flows and HTTP headers // The Journal of China Universities of Posts and Telecommunications. – 2017. – Т. 24. – №. 6. – P. 1–13. – Режим доступа: https://doi.org/10.1016/S1005-8885(17)60237-1. – Дата доступа: 12.05.2021.
5. Frequency Trails: What the Mean Really Means [Электронный ресурс]. – Режим доступа: http://www.brendangregg.com/FrequencyTrails/mean.html. – Дата доступа: 27.04.2021.
6. **Vayghan, L. A.** et al. A Kubernetes controller for managing the availability of elastic microservice based stateful applications // Journal of Systems and Software. – 2021. – Т. 175. – Article 110924. – Режим доступа: https://doi.org/10.1016/j.jss.2021.110924. – Дата доступа: 17.05.2021.

***The article presents an analysis of the performance of basic implementations of Ingress controllers Traefik, Envoy (Contour), HAProxy, NGINX and NGINX Inc under equal conditions. The performance analysis was carried out according to the methodology and algorithms proposed by the authors, based on the experience of working with highly loaded systems. The result of the research is the selection of the most effective implementation, which is best suited for widespread use in high-load systems without additional configuration.***

***Keywords:*** *high-load system performance, Kubernetes orchestration system, containerized application deployment, containerized application scaling, reverse proxy.*

*Получено 19.05.2021.*