PEAK

Proxy **E**liminating **A**rchitecture using **K**ubernetes

A research propsal

Joar Heimonen
contact@joar.me

February 20, 2025

Contents

1 Part 1

1.1 Introduction

With the adoption of IPv6 [6] our networking landscape is about to change. Last year we proposed the architecture **PEAK** [5] which leverages the new capabilities of IPv6 to create distributed systems that are not reliant on proxies. Our last paper introduced a reference implementation of the PEAK architecture, while this is useful it does not allow us to benchmark and compare the performance of the PEAK architecture to traditional systems.

While the PEAK article [5] was a qualitative study, this paper proposes a quantitative study of the PEAK architecture. This quantitative study will be performed according to the guidelines set forth by "Research Design: Qualitative, Quantitative, and Mixed Methods Approaches" [4]. We will However not be performing a Mixed Methods study as this all ready has been done in the PEAK article [5].

In this paper we propose a set of reference implementations of the PEAK architecture that will be benchmarked against a set of traditional systems. We will also propose a set of metrics that will be used to compare the performance of the systems. With the aim of showing that the PEAK architecture not only is viable but matches or exceeds the performance of traditional systems. We hope that the data driven approach of this paper will help to further the adoption of the PEAK architecture.

1.2 Background

Since the start of this century, the internet has grown from a small network of universities to a global network that connects billions of devices. Due to our limited address space provided by IPv4 [8] we have not been able to give every device a unique address. This has led to the development of two technologies. Network address translation (NAT) [7] and proxies.

NAT allows for the creation of subnets that can be used to connect multiple devices to the internet through a single shared IP address. This is a great solution for small networks but leads to an increase in complexity as the network grows. In a subnet consisting of servers none of these servers can be reached from the internet without the use of a proxy. The proxy simply

redirects traffic to the correct server in the subnet based on arbitrary rules.

While proxies works fine they introduce a single point of failure in distributed systems. If the proxy stops working, all the servers within the subnet become unreachable.

2 Part 2

2.1 Relevant Work

As mentioned in the PEAK article [5] distributed systems with DNS for load balancing and routing is not a new concept. However, it has proven to be a challenge to find recent research on the topic.

Table 1 shows the search terms we used and the results from these searches. As can be seen there is a lack of research on the topic. Our exclusion parameters were set to exclude any research older than 2020. We also excluded any research that did not mention DNS, load balancing and distributed systems in the abstract.

DB	**Keywords**	**Date**	**Res.**	**Rel.**
IEEE	Dist. Sys. AND DNS	17-Feb-25	117	0
IEEE	Load bal. AND DNS	17-Feb-25	33	0
IEEE	Dist. Sys. AND Load bal. AND DNS	17-Feb-25	13	0
Google Scholar*	CERN AND Load bal. AND DNS	17-Feb-25	40	2
ACM	Dist. Sys. AND DNS	18-Feb-25	388	0
ACM	Load bal. AND DNS	18-Feb-25	161	0
ACM	Dist. Sys. AND Load bal. AND DNS	18-Feb-25	154	0

Table 1: Search results **Google Scholar is a meta-database, The results are from the first four pages.*

Due to the lack of search results we quickly realized that it would be beneficial to use a meta-database such as Google Scholar. This gave us two relevant articles from CERN.

- **Title:** Achieving metric oriented load balancing [3]
- **Authors:** P. Canilho, I. Reguero, P. Saiz
- **Date:** 2020
- **Data:** "This article presented the objectives, advantages and usage of a metric-based load balancing solution that is actively maintained at CERN." [3]
- **Methods:** Theoretical Paper
- **Main Findings*:** Implemented metric-based load balancing through SNMP and their own lbclient. Does not mention how this can be applied to the larger IPv6 space.

- **Title:** A Modernized Architecture for the Post Mortem System at CERN [2]
- **Authors:** J. Barth, F. Bogyai, J. Garnier, M. Majewski, T. Martins Riberio, A. Mnich, M. Pocwierz, R. Selvek, R. Simpsons, A. Stanisz, D. Wollmann, M. Zerlauth
- **Date:** 2022
- **Data:** Redesign of the post mortem system at the large hadron collider [2].
- **Methods:** Theoretical Paper
- **Main Findings*:** Implemented load balancing trough DNS to make the post mortem system of the large hadron collider horizontally scalable.

***Main findings in relation to the PEAK architecture [5].**

2.2 Initial Analysis

We were able to find two articles that address routing trough DNS and metric based load balancing [5, 3]. However, none of these articles mention the use of IPv6 to create a public distributed system that is not reliant on proxies. Neither of the articles perform any form of benchmarking to compare the performance of their systems to more traditional systems. Due to the lack of research on this topic we believe that we have found a gap in research on the topic of modern distributed systems. While it is somewhat disconcerting that we were not able to find more research on the topic, we are also presented with an exciting opportunity to further the field of distributed systems. This further demonstrates the need for a quantitative study of not only the PEAK architecture but also more conventional systems.

As a system grows in complexity qualitative studies [4] become more and more disconnected from the reality of the system. This is why we believe that a quantitative study is needed to further the field of distributed systems.

3 Part 3

3.1 Research Questions

We aim to answer the following questions:

1. How does the PEAK architecture perform compared to traditional systems?

2. What metrics can be used to compare the performance of the PEAK architecture to traditional systems?

3. How does the PEAK architecture scale compared to traditional systems?

To achieve this goal this paper will propose a set of reference implementations of the PEAK architecture and a set of traditional counterparts. We will also propose a set of metrics that can be used to compare the performance of the systems.

3.2 Reference Implementations

We propose the following two reference implementations and their traditional counterparts:

3.2.1 Reference Implementation 1

This reference implementation consists of a set of servers where routing and load balancing is handled through the cluster level DNS server. This can be seen in *Figure 1*.

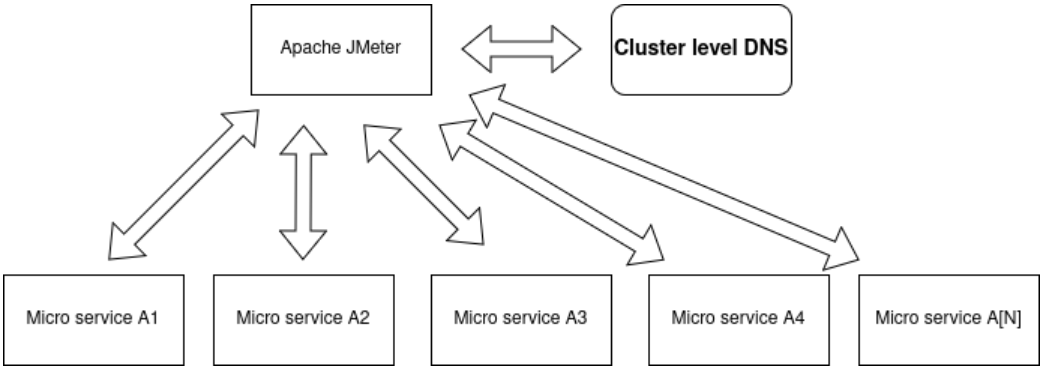

Figure 1: PEAK reference implementation 1

A counterpart to this reference implementation consists of a proxy server that routes traffic to the servers. This can be seen in *Figure 2*.

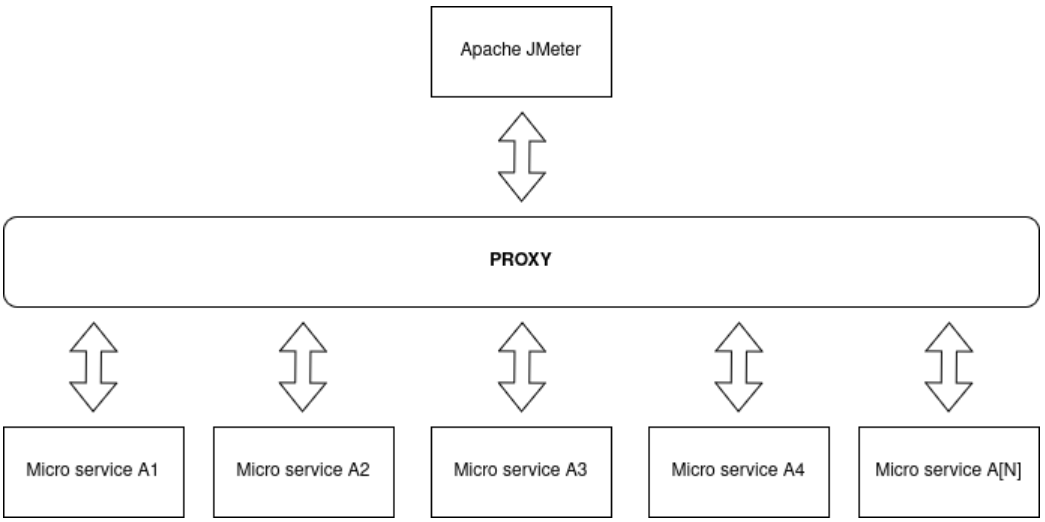

Figure 2: Reference implementation 1 traditional counterpart

3.2.2 Reference Implementation 2

This reference implementation is designed to test the client-mediated communication suggested in the PEAK architecture [5]. This can be seen in *Figure 3*.

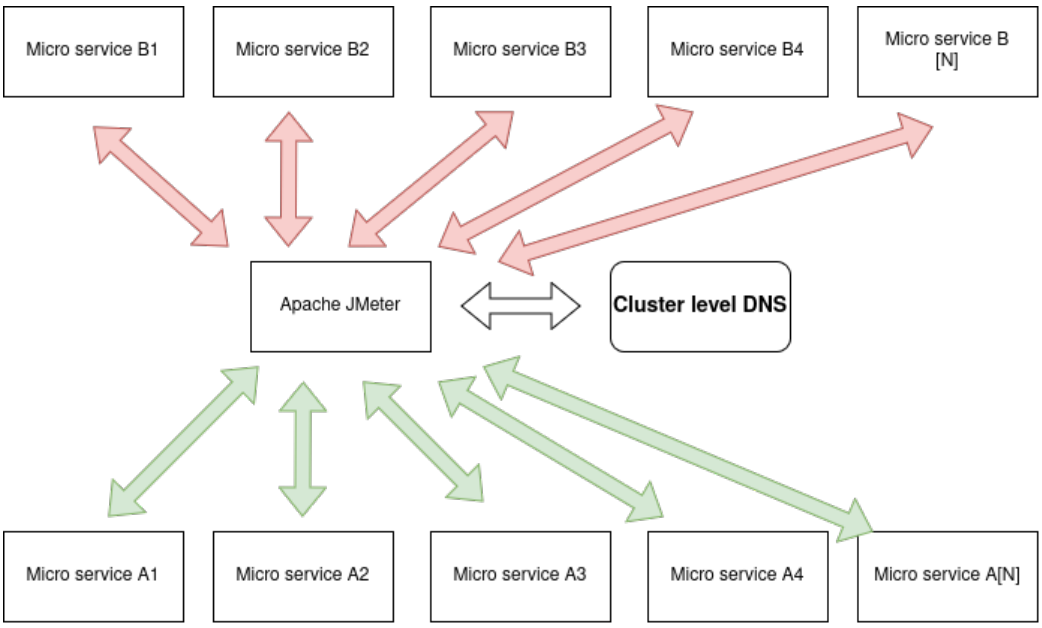

Figure 3: PEAK reference implementation 2. The green arrows represent request 1 and the red arrows represent request 2. The A set of servers signs a token that must be presented to the B set of servers to be able to communicate.

A counterpart to this reference implementation that replaces the client mediated communication with a service mesh can be seen in *Figure 4*.

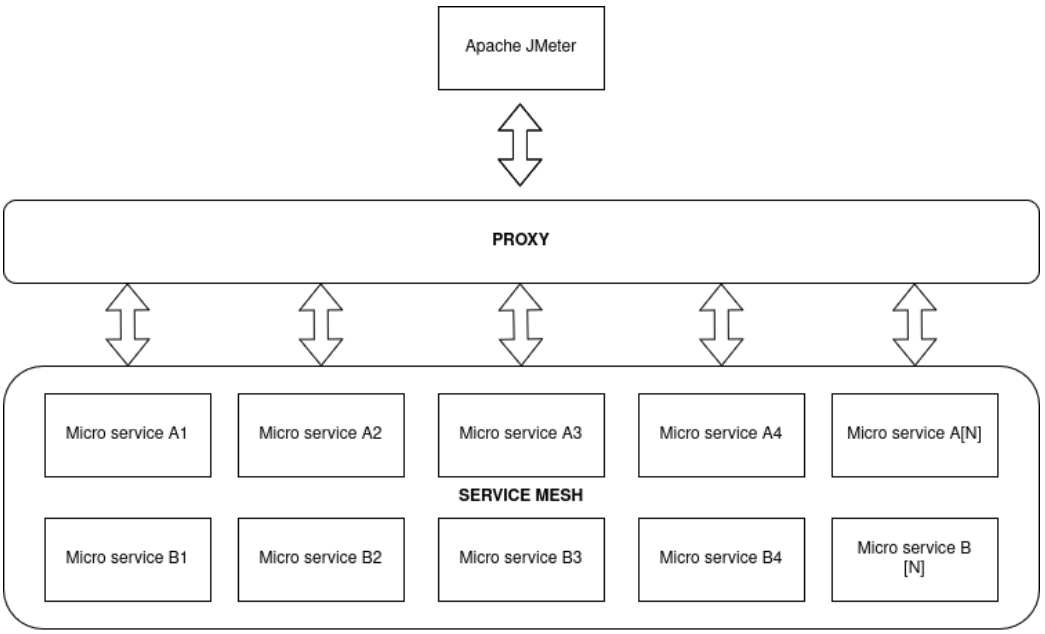

Figure 4: Reference implementation 2 traditional counterpart

3.3 Metrics

There are six metrics that we can use to compare the performance of the PEAK architecture to traditional systems:

1. **Latency:** The time it takes for a request to be processed.
2. **Throughput:** The number of requests that can be processed in a given time frame.
3. **Scalability:** How well the system scales as the number of servers increases.
4. **Balance:** How well the system balances the load between the servers.
5. **Inner service communication:** How well the system handles communication between servers.
6. **Fault tolerance:** How well the system handles server failures.

There are also several combinations of these metrics that can be used to compare the performance of the systems. Like the ratio between throughput and Scalability. In a perfect system we expect the throughput to increase linearly as the number of servers increases. Latency and Scalability can also be combined to see how the latency increases as the number of servers increases.

3.4 Methodology

The reference implementations will be implemented in a Kubernetes cluster. The cluster will contain a single cluster level DNS server [5] or a single proxy server. We will ramp up Apache JMeter [1] until the system fails. After the system has failed we will increase the number of servers and try again. This will be done for both the PEAK architecture and the traditional systems. A flow diagram of the testing process can be seen in *Figure 5*. The results will be compared using the metrics proposed in the previous section through a comparative analysis.

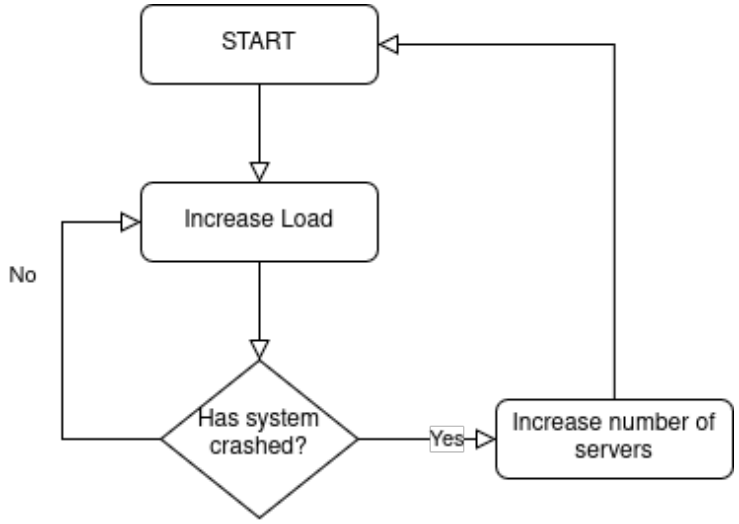

Figure 5: A flow chart of the testing process. This test runs until the desired number of servers has been reached.

4 Part 4

4.1 Discussion

While our approach to quantitative testing [4] seems simple enough on paper. It is worth remembering that there are hundreds of configurations not only for the PEAK architecture but also for the traditional systems that can be tested. It is essential that both of the systems are created in an as efficient and lean way as possible. This helps to mitigate the risk of a system underperforming due to a misconfiguration. A potential solution to this problem is outsourcing the creation of the traditional systems to multiple parties. Multiple implementations of the traditional system created by different parties will strengthen the validity of the results by guaranteeing that the traditional systems are created in an impartial manner.

Providing that the proposed quantitative study is performed correctly, we believe that this work has the potential to disrupt the status quo of distributed systems. Resulting in less bloated systems that are more fault-tolerant and scalable. We also hope that this work will inspire others to further the field of distributed systems with original ideas and architectures.

References

[1] *Apache JMeter - Apache JMeter™*. https://jmeter.apache.org/. (Visited on 02/20/2025).

[2] Jonas Barth et al. "A Modernized Architecture for the Post Mortem System at CERN". in: (2022), 4 pages, 0.219 MB. ISSN: 2673-5490. DOI: 10.18429/JACOW-IPAC2022-TUPOMS055. (Visited on 02/19/2025).

[3] Paulo Canilho, Ignacio Reguero, and Pablo Saiz. "Achieving Metric Oriented Load Balancing". In: *EPJ Web of Conferences* 245 (2020). Ed. by C. Doglioni et al., p. 07004. ISSN: 2100-014X. DOI: 10.1051/epjconf/202024507004. (Visited on 02/19/2025).

[4] John Creswell and David Creswell. "Research Design: Qualitative, Quantitative, and Mixed Methods Approaches". In: *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Chapter One: The Selection of a Research Approach. (Visited on 02/20/2025).

[5] Joar Heimonen. *Preprint PEAK Proxy Eliminating Architecture Using Kubernetes: Leveraging Proven Technologies to Create the Distributed System of the Future*. 2024. (Visited on 02/18/2025).

[6] Bob Hinden and Steve E. Deering. *Internet Protocol, Version 6 (IPv6) Specification*. Request for Comments RFC 2460. Internet Engineering Task Force, Dec. 1998. DOI: 10.17487/RFC2460. (Visited on 02/19/2025).

[7] Matt Holdrege and Pyda Srisuresh. *IP Network Address Translator (NAT) Terminology and Considerations*. Request for Comments RFC 2663. Internet Engineering Task Force, Aug. 1999. DOI: 10.17487/RFC2663. (Visited on 02/18/2025).

[8] *Internet Protocol*. Request for Comments RFC 791. Internet Engineering Task Force, Sept. 1981. DOI: 10.17487/RFC0791. (Visited on 02/18/2025).