Distributed MAPE: Final submission

Bruno Figueroa

December 2025

Abstract

Self-adaptive systems must operate in uncertain and dynamic environments such as cloud, IoT, and edge computing infrastructures. The MAPE-K control loop: Monitor, Analyze, Plan, Execute, and Knowledge, provides a well-established framework for autonomous decision making and runtime adaptation. While numerous distributed MAPE-K architectures have been proposed in the literature, their practical performance strongly depends on system scale, workload characteristics, and deployment constraints.

This paper presents both a state-of-the-art review of distributed MAPE-K architectures and an experimental evaluation of three concrete implementations: a centralized MAPE, a functionally distributed MAPE, and a decentralized ring-based MAPE. Using a custom benchmark based on Apache Flink and Apache Kafka, the study evaluates latency, throughput, and adaptation behavior under sustained load. The results highlight the trade-offs between architectural simplicity and distribution overhead, showing that centralized approaches can outperform distributed designs in small-scale, resource-constrained environments.

1. Introduction

Modern distributed computing systems must operate under highly dynamic conditions, including fluctuating workloads, resource contention, partial failures, and changing network conditions. Manual management of such systems is increasingly impractical, motivating the adoption of self-adaptive mechanisms capable of autonomously adjusting system behavior at runtime.

The MAPE-K (Monitor, Analyze, Plan, Execute over Knowledge) control loop formalizes a feedback-driven approach to self-adaptation, enabling systems to observe their own state, reason about deviations from objectives, and enact corrective actions. Originally conceived for centralized management systems, MAPE-K has since been extended to support decentralized, hierarchical, and peer-to-peer control structures.

Most existing research emphasizes the advantages of distributed MAPE-K architectures in terms of scalability, fault tolerance, and resilience, particularly in large-scale cloud, IoT, and edge computing environments. However, fewer studies empirically evaluate the performance cost introduced by distribution itself, especially under constrained hardware conditions or limited deployment scales.

This work addresses this gap by combining a structured review of distributed MAPE-K architectures with a hands-on experimental evaluation of three representative models. A custom benchmark based on Apache Flink and Kafka is used to compare centralized, functionally distributed, and ring-based MAPE-K implementations under identical workloads. The goal is not only to assess raw performance metrics, but also to understand how architectural complexity impacts adaptation efficiency in practice.

2. The MAPE-K Model

The MAPE-K loop decomposes the adaptation process into five functional elements:

- **Monitor:** gathers metrics from the system and its environment.
- **Analyze:** interprets monitored data to detect trends, anomalies, or violations of adaptation goals.
- **Plan:** determines appropriate adaptation actions or reconfiguration strategies.
- **Execute:** enacts the selected actions through effectors or control interfaces.
- **Knowledge:** stores policies, historical data, models, and runtime information shared across the loop.

In distributed systems, MAPE-K loops may

be replicated, decomposed, or hierarchically composed across nodes. In such settings, the Knowledge component plays a critical role in maintaining consistency and coordination among multiple adaptation loops.

3. Architectural Models and State of the Art

Distributed MAPE-K systems adopt several architectural paradigms, each involving different trade-offs between scalability, fault tolerance, communication overhead, and coordination complexity.

Centralized MAPE: All monitoring, analysis, planning, and execution activities are performed by a single control entity. This architecture is simple to implement and presents minimal coordination overhead, making it suitable for small-scale or resource-constrained environments. However, it introduces a single point of failure and can become a performance bottleneck as system scale increases.

Functionally Distributed / Hybrid MAPE: The MAPE loop is decomposed into separate functional components that may be deployed independently. This improves modularity and extensibility, allowing individual components to scale or evolve separately. The main drawback is the additional communication overhead and increased system complexity.

Hierarchical MAPE: Local MAPE controllers manage subsets of the system and report to higher-level controllers responsible for global coordination. This model balances local autonomy with global optimization but may introduce congestion or failure points at higher levels.

Decentralized / Peer-to-Peer MAPE: Each node runs a complete MAPE loop and coordinates with others using peer-to-peer or ring-based communication. This architecture maximizes fault tolerance and adaptability but requires sophisticated protocols to maintain coherent knowledge and avoid conflicting decisions.

4. Applications and Case Studies

Centralized and hierarchical MAPE-K frameworks are widely used in cloud environments to support elastic resource management. In contrast, edge computing and IoT systems often rely on decentralized or semi-decentralized adaptation to reduce latency and energy consumption.

Several studies report the use of energy-aware MAPE-K strategies in serverless edge platforms, smart homes, and sensor networks. Multi-agent systems further demonstrate peer-to-peer MAPE coordination, particularly in experimental robotic and cyber-physical applications.

5. Common Evaluation Metrics

The evaluation of MAPE-K architectures typically considers the following metrics:

- Adaptation latency and response time.
- Throughput and system productivity.
- Scalability with increasing numbers of nodes.
- Robustness and fault tolerance.
- Communication overhead and bandwidth usage.

6. Experimental Evaluation

This section presents the experimental design, the evaluated architectures, and the results obtained from the benchmark developed to compare different MAPE-K models. The goal of the experiment is to analyze the impact of centralization and functional distribution of the MAPE loop in terms of latency,

throughput, and adaptation capability under sustained load.

6.1 Experimental Environment

The benchmark was implemented using *Apache Flink* as the stream processing engine and *Apache Kafka* as the messaging system and metric source. The workload consists of computing the n-th prime number, a computationally intensive operation that rapidly saturates CPU resources and allows the observation of scaling and adaptation behavior.

Initially, the *Nexmark* benchmark was considered, as it is widely used in the literature for evaluating stream processing systems. However, it was discarded due to its limited flexibility and high hardware requirements. Nexmark assumes infrastructures composed of multiple high-capacity nodes (e.g., three machines with 8 CPU cores and 32 GB of RAM each as minimum requirements), which significantly exceed the resources available for this study.

Instead, a controlled synthetic benchmark was adopted, based on progressively saturating the system through a high volume of Kafka messages and computationally expensive operations. Although this approach is less standardized, it provides fine-grained control over the workload and is suitable for comparing adaptation strategies in resource-constrained environments.

All experiments were executed on a single machine. As a result, communication costs between components are relatively low (on the order of milliseconds), while scaling is strongly limited by the available CPU capacity.

6.2 Evaluated MAPE Models

Three different MAPE-K architectures were implemented and evaluated.

6.2.1 Centralized MAPE

The centralized model implements a single MAPE loop that:

- Collects latency and throughput metrics directly from Kafka.
- Analyzes these metrics and decides whether to scale the Flink job.
- Performs scaling up to a maximum of 8 parallel instances.

This model prioritizes simplicity, minimizing communication and coordination overhead among components.

6.2.2 Functionally Distributed MAPE

In this architecture, the MAPE loop is decomposed into four independent containers, each responsible for a specific function:

- **Monitor (M):** consumes metrics from Kafka topics and forwards them to the analysis module.
- **Analyze (A):** processes incoming messages, computes averages using a 1-second sliding window, and sends aggregated metrics to the planner.
- **Plan (P):** decides whether to scale up, scale down, or take no action based on the received metrics.
- **Execute (E):** applies scaling decisions by removing and redeploying the Flink job with a new parallelism level, enforcing minimum and maximum limits and introducing a 30-second cooldown period to prevent overly frequent reconfigurations.

This model improves modularity and extensibility at the cost of increased communication overhead.

6.2.3 Ring-Based MAPE

The ring-based model consists of multiple local centralized MAPE instances, each managing local jobs. These instances:

- Communicate with each other using HTTP messages or Kafka.
- Exchange information to agree on global parameters, such as parallelism limits or waiting times.
- Make independent scaling decisions based on local load measurements.

This approach aims to provide high adaptability and eliminate single points of failure, but introduces higher complexity and coordination costs.

6.3 Evaluation Metrics

The primary metrics used for comparison were:

- Average system latency.
- Total throughput (number of completed jobs).
- Scaling behavior under prolonged saturation.

6.4 Experimental Results

6.4.1 Latency

Average system latency is the most relevant metric in this experiment, as it directly reflects the adaptation capability of each MAPE architecture under sustained load. Figure 1 shows the evolution of latency for the three evaluated models.

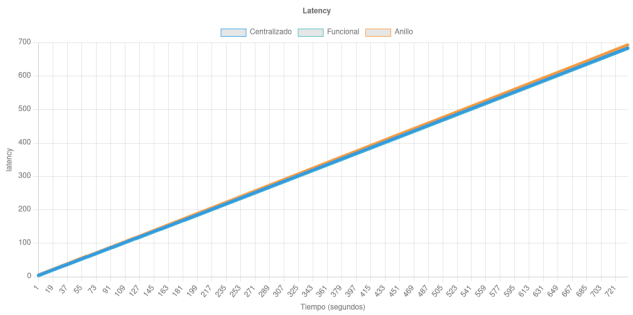

Figure 1: Evolution of average latency for centralized, functionally distributed, and ring-based MAPE models.

As observed, all models exhibit progressively increasing latency, indicating limited scaling capacity under the imposed workload. The final latency values were:

1. Centralized MAPE: 684.28 ms
2. Functionally distributed MAPE: 682.83 ms
3. Ring-based MAPE: 693.51 ms

Although the differences are relatively small, the centralized model achieved the best overall latency, followed closely by the functionally distributed model, while the ring-based approach showed the highest final latency.

6.4.2 Throughput

Total throughput, measured as the number of completed jobs during the experiment, is shown in Figure 2. This metric captures the cumulative impact of latency on system productivity.

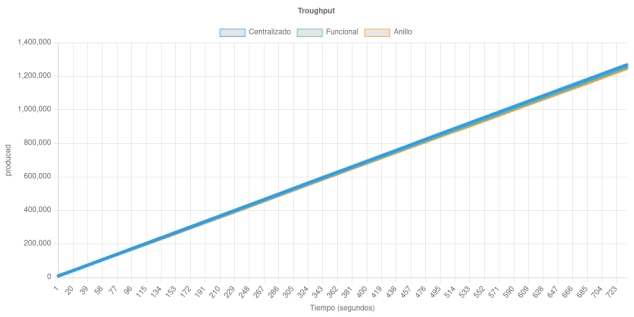

Figure 2: Total number of completed jobs for each MAPE architecture.

The observed final results were:

1. Centralized MAPE: 1,269,023
2. Functionally distributed MAPE: 1,258,092
3. Ring-based MAPE: 1,247,160

The centralized model outperformed the functionally distributed model by 10,931 jobs and the ring-based model by 21,863 jobs. These differences, although moderate, are consistent with the latency measurements: small variations in accumulated latency result in significant throughput differences over time.

6.5 Analysis of results

The results indicate that, in this experimental setting, the simplicity of the centralized MAPE model leads to superior performance, leading by a modest but consistent margin. The performance losses observed in distributed models can be attributed to:

- Communication overhead between components in the functionally distributed model.
- Increased operational complexity and coordination costs in the ring-based model.
- Additional overhead introduced by Kafka and Flink when handling multiple concurrent message streams.

It is important to emphasize that these results are neither definitive nor universally generalizable. Since the experiments were conducted on a single machine, communication costs are artificially low and scaling is dominated by CPU limitations. In real distributed environments, such as edge architectures or multi-region deployments, results will probably differ significantly.

6.6 Implications Across System Scales

Based on the experimental results and the state of the art analysis, different MAPE-K architectures can be considered appropriate depending on system scale:

- **Small-scale environments** benefit from simple, low-overhead architectures such as centralized MAPE.
- **Medium-scale systems**, consisting of several dozen nodes, can leverage functionally distributed MAPE models that balance modularity and robustness.
- **Large-scale distributed infrastructures**, spanning hundreds of nodes across multiple regions, require hierarchical or strongly decentralized approaches, particularly in edge computing scenarios where minimizing end-to-end latency is critical.

In this context, the ring-based MAPE model showed limited effectiveness at small scale, where coordination overhead consumes a significant portion of available resources. However, its adaptability and flexibility suggest potential benefits in large-scale systems, where communication costs are amortized and resilience becomes a primary concern. Validating this hypothesis requires further experimentation at medium and large scales, which lies beyond the scope of this work.

7. Conclusion

This work presented a combined literature review and experimental evaluation of centralized and distributed MAPE-K architectures. While existing research often emphasizes the benefits of decentralization, the experimental results demonstrate that architectural simplicity remains a decisive factor in small-scale and resource-constrained environments.

The benchmark results show that the centralized MAPE implementation achieved the lowest latency and highest throughput, despite lacking the resilience and scalability properties typically associated with distributed approaches. The functionally distributed MAPE exhibited comparable performance, with modest overhead introduced by inter-component communication. In contrast, the ring-based MAPE incurred higher coordination costs, resulting in reduced efficiency under the evaluated conditions.

These findings do not contradict the relevance of distributed MAPE-K architectures, but rather highlight the importance of context when selecting an adaptation model. Distributed and decentralized approaches are likely to outperform centralized designs in large-scale or geographically distributed systems, where communication costs are amortized and fault tolerance becomes critical.

However, in small deployments, the additional complexity of distributed control can outweigh its theoretical benefits.

Possible future work includes extending the experimental evaluation to multi-node and geographically distributed environments, exploring hierarchical MAPE-K configurations, and incorporating learning-based adaptation strategies. Such studies are necessary to fully characterize the conditions under which different distributed MAPE-K architectures provide optimal performance and practical value.

The implementation and experimental prototype developed for this work is available at https://github.com/BrunoFigueroa/practica_electiva.

References

- Sharmin Jahan et al. (2020) – MAPE-K/MAPE-SAC: An Interaction Framework for Adaptive Systems with Security Assurance Cases. https://www.sciencedirect.com/science/article/pii/S0167739X19320527
- Weyns et al. (2009) – On Patterns for Decentralized Control in Self-Adaptive Systems. https://seal.ics.uci.edu/publications/2012aSefSAS.pdf
- Quin, Weyns, Gheibi (2021) – Decentralized Self-Adaptive Systems: A Mapping Study. https://people.cs.kuleuven.be/~danny.weyns/papers/2021SEAMSb.pdf
- Zeadally et al. (2019) – Self-Adaptation Techniques in Cyber-Physical Systems. https://scispace.com/pdf/self-adaptation-techniques-in-cyber-physical-systems-cpss-qsry4k8wsn.pdf
- Villegas et al. (2012) – Self-Adaptive Cloud Applications Using MAPE-K. https://www.researchgate.net/publication/266618283_Self-Adaptive_Applications_On_The_Development_Of_Personalized_Web-Tasking_Systems
- Ghorbian, Ghobaei-Arani, Esmaeili (2025) – An Energy-Conscious Scheduling Framework for Serverless Edge Computing in IoT. https://link.springer.com/article/10.1186/s13677-025-00780-7
- Giussani (2022) – A Semi-Decentralized Self-Adaptive IoT Architecture for Energy Efficiency in Smart Households. https://ceur-ws.org/Vol-3620/mess23_paper03.pdf