IJIRSS
International Journal of Innovative Research & Scientific Studies

ISSN: 2617-6548

URL: www.ijirss.com

The Billion Dollar cloud: Architectural patterns for hyperscale cost optimization

Hardik Mahant[1*], Lokesh Karanam[2]

[1]*San Jose, California, USA.*
[2]*Austin, Texas, USA.*

Corresponding author: Hardik Mahant (*Email: hardik.s.mahant@gmail.com*)

Abstract

In this paper, we will explore the patterns that are crucial when designing data center infrastructure at hyperscale to minimize costs and enhance effectiveness in terms of latency, consistency, availability, and reliability of the deployed infrastructure. Findings indicate that up to 90% of network communication costs could be saved by dynamically provisioning on-demand compute resources. Inspired by fluid dynamics principles, server temperatures were observed to be up to 30% lower, resulting in higher power savings. The patterns to maximize the number of virtual machines on a physical server optimize performance by an average of 34.41%. The dynamic utilization of old and new hardware configurations results in a significant 40% reduction in hardware replacement costs and improves reliability. Predictive state monitoring and compute resource provisioning lead to over 20% reduction in power consumption. Operating at internet scale, this hybrid setup requires billions of dollars in operational costs for maintaining cloud and physical infrastructure. This paper describes architectural patterns that contribute to overall cost reduction when applied to different layers in cloud or on-premises hyperscale infrastructure.

Keywords: Data center architectural patterns, Heterogeneous infrastructure, Hybrid multi-cloud, Hyperscale cost optimization.

DOI: 10.53894/ijirss.v8i6.9600
Funding: This study received no specific financial support.
History: Received: 11 July 2025 / **Revised:** 13 August 2025 / **Accepted:** 15 August 2025 / **Published:** 1 September 2025

Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Transparency: The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.
Publisher: Innovative Research Publishing

1. Introduction

This paper focuses on optimizing costs for hyperscale infrastructure, which involves multiple dimensions, including the number of concurrent users, infrastructure scale, latency, throughput, operational costs, and energy consumption. As demand for compute resources grows, optimizing at various layers of hyperscale infrastructure becomes increasingly critical. Previous research has identified challenges at individual layers of the infrastructure, but a comprehensive approach is necessary. From a business perspective, it is essential to consider all layers of the stack and implement strategies that balance performance with cost-efficiency. Costs at each layer of hyperscale infrastructure can reach billions of dollars. As workloads change, proven optimization patterns must be adopted to maintain service quality and cost efficiency. This paper discusses patterns across five layers of hyperscale architecture. When applied together or in a hybrid manner, these patterns can significantly reduce costs and increase performance. A typical hyperscale data center setup includes millions of servers distributed globally to provide internet-scale services. The initial deployment costs can reach billions of dollars, and maintenance costs remain high. While offered as a service, infrastructure must follow patterns that help users plan for cost, energy, and efficiency. By refining the architecture, users can optimize both their custom setup and overall operational costs, benefiting economically and environmentally. With the increasing adoption of Large Language Models (LLMs) and AI workloads, power and resource consumption are growing. To accommodate this, optimized resource usage and power consumption must be carefully addressed by revising the architecture to meet changing needs. Existing studies have been performed to address cost optimizations at a specific layer or in a subdomain of the overall infrastructure. While beneficial for that specific area, the other unoptimized layers would still incur higher costs and lower performance. We propose adopting an approach to reduce costs across all layers of hyperscale infrastructure by implementing proven patterns.

1.1. Architectural Patterns

While considering cost-effectiveness in a hyperscale environment, we must think in layers. In this section, we delve deeper into the key layers of hyperscale infrastructure and the cost-optimization architecture patterns for each layer. These patterns are divided into the areas of network, resources, system, hardware, and power consumption. For each of these areas, the costs could amount to billions if not optimized for the highest level of efficiency.

1.2. Network Layer

The network layer architectural patterns for hyperscale architecture consider data center location, footprint allocation, and resource provisioning in terms of network latency and bandwidth for the proposed workload to optimize cost and improve Quality of Service (QoS). The research indicates that the Lagrangian relaxation method, which combines location, allocation, and resource provisioning, produces more cost- and energy-efficient results compared to a more hierarchical approach [1]. For the hyperscale network architecture, Integer Linear Programming (ILP) techniques such as Branch, Cut, and Price are used to minimize virtualization costs and processing power while achieving maximum network throughput. A particular scheme, as per this research, EWOS, appears to be effective across various combinations of frameworks like "Branch and Price," "Branch and Cut," and "Branch and Bound" [2].

1.3. Resource Layer

Depending on how the resources are utilized, the overall efficiency varies. Under resources, there are three primary resource classes provisioned for running various application workloads. These classes are compute, storage, and network resources. A combination of techniques is used to optimize resource allocation, consumption, and maintenance.

Some of these techniques include dynamic resource allocation, virtualization, virtual machine (VM) migration, and temperature-aware workload deployment [3]. Whenever there is a request for workload allocation, a centralized queue determines the state of the available resources in the closest data center. The drawback here is that communication with the centralized queue contributes to added latency and increased compute power, especially when millions of requests need to be processed. A better methodology to optimize this issue is to maintain a pool of available servers, allocating the next available pool of server to each request in the queue. Servers that remain idle for a long period would transition into sleep mode and become available as needed. This approach saves energy and makes power usage more cost-effective. The difference between this approach and Join Idle Queue (JIQ) is that, in JIQ, resources stay in an idle state and continue consuming power, whereas in this method, idle resources do not consume power, resulting in significant cost savings. Another important pattern in resource management is Auto Scaling[4], which predicts traffic patterns and handles bursts of ad-hoc resource demands. When such phenomena occur, the infrastructure scales up to meet the demand and scales down based on the initial configuration once the demand has been fulfilled [4]. In hyperscale deployments such as data centers or public clouds, it is important to map VMs in demand to the physical servers where they can be hosted. Various techniques, such as First Fit Decreasing (FFD) and Particle Swarm Optimization (PSO), can be used for this mapping. Studies indicate that a hybrid approach combining these techniques could achieve superior results [5]. Another technique is to employ heterogeneous VM clusters depending on the workload requirement. In a traditional setup, all VMs are of the same configuration; this poses a significant limitation on varying patterns of the workload. Having heterogeneous configuration-based VMs allows the workload to be served more efficiently, optimizing power and application throughput [6]. This approach applies to physical servers as well. Data centers would identify and allocate workloads to machines with different configurations to address various types of workloads. This can be achieved with the Complete Sets pattern in combination with machine learning [7]. There are certain patterns, like caching, that need to be considered for efficient deployment of overall architecture in a cost-effective way. The research indicates that, for read-heavy applications, the right caching strategy is essential. Not having or poorly implementing caching may cause significant power and resource consumption, which may lead to higher costs for serving during normal and peak load situations [8]. Table 1 presents the comparison of Central Processing Unit (CPU) and VM consumption in an application. Based on Google’s Cost Calculator, we would require about 50,000 web VMs, 60,000 VMs, 5,000 Database VMs, 1,000 Caching VMs, and varying API Gateway, data storage bandwidth, and the costs associated with auxiliary services to monitor, secure, and scale the architecture. The estimate for this setup would cost about 20 million dollars per month. Note that the estimate could vary depending on workload changes and resource class.

Table 1.
Number of VMs and power consumed to support various workloads in cached and uncached environments.

Operation Type (1 million / Sec)	Cached	Number of VMs Required	Power Consumption
Read	YES	110,500	0.314 megawatts
Write	YES	175,400	0.497 megawatts
Read-Write	YES	190,000	0.538 megawatts
Read	NO	427,000	1.209 megawatts
Write	NO	500,000	1.416 megawatts
Read-Write	NO	600,0000	1.687 megawatts

We can derive from the observed data in Table 1 that there is more compute power required in non-cached environments compared to cached environments, resulting in increased power consumption.

In modern data center deployments with a hybrid approach, where workloads run on premises and in the public cloud, “Spot Instances” are used for cheaper compute. This pattern may cost more if the instances are terminated before the job is completed. Although there is a prediction mechanism, and the user has the option to promote the instance to a paid instance, migrate the work to a new instance, or terminate the instance entirely, this can increase costs and power consumption by discarding the work that has already been performed [9]. Green Cloud Optimized Data Center strategy uses resource consolidation and a combination of hardware and software with the right set of power sources, which shows a reduction in energy consumption and carbon emissions [10] without compromising the service quality of the infrastructure.

1.4. System Layer

We will identify the system layer as a combination of telemetry, analytics, self-healing, operating system version compliance, pathing, and ongoing maintenance functions. Telemetry provides detailed insights into various metrics such as load average, memory utilization, temperature, requests per second, long-running processes, disk utilization, bandwidth utilization, network time protocol mismatch, system error count, and server availability metrics. Table 2 lists these metrics and a configurable threshold type. These metrics are configured in systems like Prometheus [11]. Prometheus provides us with a JSON-based schema to define the rules for different types of checks. The Prometheus agent is deployed on each server that is being monitored. The agent periodically sends signals for each of the configured checks and notifies a system that can handle critical notifications from Prometheus. We call this alert receiver the Alert Manager.

Table 2.
Common system level checks configured in telemetry systems.

Check Name	Measured Value Type
Memory Utilization	Threshold
Process Count	Threshold
File System	Threshold
Disk Utilization	Threshold
CPU Utilization	Threshold
Temperature	Threshold
Server Unreachable	Boolean
Server Unresponsive	Boolean
Requests Per Second	Threshold
Vulnerabilities	String
Hardware Age	Threshold
Package Versions	Number
Uptime	Threshold
OS Version	Number

This system receives critical payload which has the information about the node which has alert threshold above the defined values. Appendix A describes a detailed view of the configured thresholds for the checks mentioned in Table 2. The thresholds are configured at increasing levels of values, and different alerting mechanisms such as emails, pagers, incident, or on-call notifications are configured to address critical events as they occur across the infrastructure. Based on the organization’s requirements, this system will communicate further with systems like PagerDuty, ServiceNow, or any other system that can alert the on-call support teams. Once the support team receives a ticket, it will be reviewed and addressed based on the severity of the ticket. Figure 1 showcases the architecture of a monitoring system that is widely adopted for on-premises infrastructure monitoring of all kinds of resources.

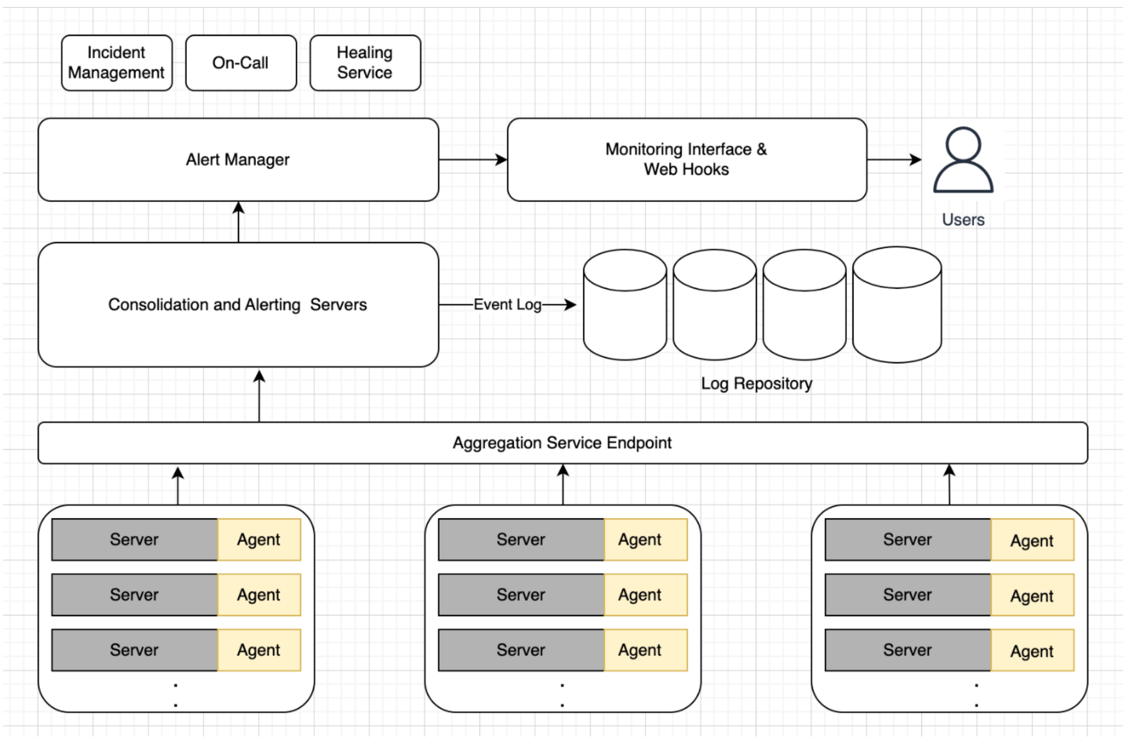
Incident Management
On-Call
Healing Service
Alert Manager
Monitoring Interface & Web Hooks
Users
Consolidation and Alerting Servers
Event Log
Log Repository
Aggregation Service Endpoint
Server
Agent

Figure 1.
The architecture of the monitoring system.

The checks mentioned in Table 2 do not represent a comprehensive list of available checks. These checks can be utilized from Prometheus or an equivalent monitoring agent out of the box, or organizations can define their custom metrics depending on their monitoring needs.

Given this context, analytics play a critical role in collecting, aggregating, interpreting, and reporting these critical events. Grafana [12] is an open-source visualization tool widely used for reporting analytics and creating sophisticated dashboards with a wide variety of charts. This is a general setup for on-premises infrastructure. While working with multi-cloud providers, the provider may have their own methods of monitoring the infrastructure and reporting it to the users. Amazon provides CloudWatch [13] and the equivalent to that with Google Cloud is CloudMonitoring.

Self-healing is a process that constantly listens to incoming events and knows how to handle each event. The event here represents a hardware issue observed by the monitoring system. The server may not be fully functional due to this anomaly. Self-healing orchestration saves thousands of man-hours in maintaining hybrid multi-cloud infrastructure [14].

In Figure 2, we represent the architecture of representing self-healing workflow in hybrid and multi-cloud deployment.

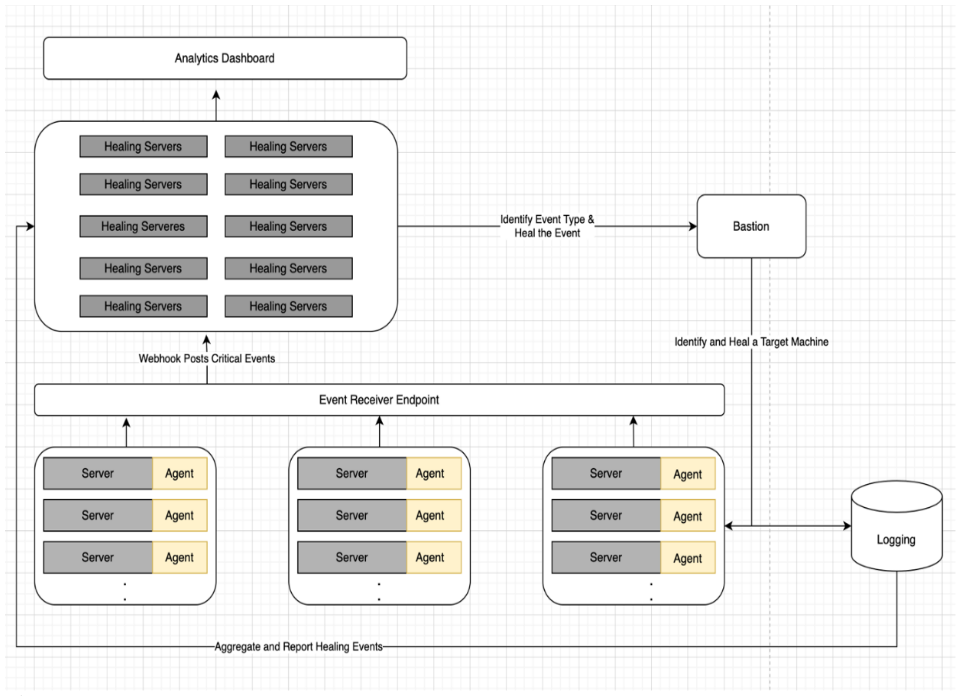
Analytics Dashboard
Healing Servers
Healing Serveres
Identify Event Type & Heal the Event
Bastion
Identify and Heal a Target Machine
Webhook Posts Critical Events
Event Receiver Endpoint
Server
Agent
Logging
Aggregate and Report Healing Events

Figure 2.
Self-healing architecture for server health remediation.

This approach allows organizations that run internet-scale businesses to maintain a small team of individuals who are responsible for maintaining self-healing workflows and keeping the infrastructure up and running.

1.5. Hardware Layer

Older hardware, in cases of low loads, can emit lower levels of carbon without impacting performance. A carbon-aware scheduler is deployed to decide the hardware to run the workload on. This also mitigates the need to constantly refresh the hardware. In organizations where there are multiple data center sites across different regions for high availability and internet-scale demands, tracking the hardware is often performed regularly. Replacing older hardware by introducing a tech-refresh program is often implemented. In our study, we conducted an analysis on 400,000 physical servers that were 3 to 5 years older. While deployed in a production environment, these servers' configurations remained stable while serving regular traffic loads, with consistently lower outage incidents reported. We learned that tech-refresh and new hardware may bring higher compute power, but there could be unknowns when onboarding heavy workloads for applications. This may lead to surprises and cause outages that can have severe business impacts. It is recommended to deploy a hybrid approach for incremental workload onboarding on new servers and fall back on old servers until they reach the end of life (EOL), when upgrades and patching are no longer permitted. This strategy results in higher cost savings and increased reliability for hyperscale infrastructures serving an internet-scale organization.

1.6. Power Layer

For a data center, the demand for high power increases as they are developed and grow. It becomes extremely critical to optimize energy consumption by considering environmental impacts. Studies show that data centers consume a little over 2 percent of the total energy used in the US [15] hence, it brings a lot of value in optimizing higher layers of the architecture, which can, in turn, contribute towards power consumption. Mixed Integer Linear Programming, a powerful optimization technique, dynamically manages data center workloads, battery banks, diesel generators, and electricity trading in real time and for future demands that may arise. This is done in conjunction with forecasting algorithms that provide accurate predictions, enabling the system to plan for required power consumption. Mixed Integer Linear Programming can be combined with Service Level Agreement (SLA) for various services, which can generate an optimized power generation and consumption plan [6]. Research shows that a stochastic model-based Swarm Algorithm maintains a pool of hot, warm, and cold Physical Machines (PMs) and keeps a balanced count of servers in each pool using a queuing mechanism. The research, however, was restricted to a homogeneous configuration of PMs, but an improvement in power consumption was noted, with a note to research further to enhance and experiment with the model for heterogeneous hardware configurations [16].

2. Discussion

We have explored many architectural patterns for cost optimization and reducing energy consumption in this paper; many of them have already been tested in real-world scenarios and proven in their respective areas, while others have more theoretical evaluation and simulated outcomes. Future research may include areas such as dynamic resource allocation, dynamic routing, and low-power consumption techniques that may work in heterogeneous environments of hardware configurations and on-premises and public cloud-based hyperscale data centers. It often becomes challenging to test workloads in near real-time conditions due to the costs associated with deploying the architecture, the time required to configure the entire setup, and making it testing-ready [17]. The future research can be conducted to gather data from different layers of the architecture and to provide a high-level, parameterized input for user requirements such as scale, operations per second, users per second, latency, infrastructure availability, and peak operations. Given these parameters, if there was a suggested way that showed the layout for the entire deployment stack, it would be economically and environmentally beneficial. Practically, measuring embodied and emitted levels of carbon at present is difficult; this is an area for future research in carbon footprint analysis [18]. Optimizing traffic patterns while moving VMs can save up to 90% of communication costs in non-dynamic environments. The idea is to find suboptimal solutions for the surrounding regions of a VM to be migrated [19]. When managing billions of dollars of cloud resources, organizations often deploy the strategy of hybrid and multi-cloud architecture. Hybrid cloud maintains on-premises infrastructure combined with public cloud providers. Multi-cloud setup combines multiple cloud providers to be utilized. A well-defined strategy to identify the workloads for peak traffic patterns, periodic bursts of requests, and special events like holidays, music launches, and global public events can cause increased demands. In handling these situations, it's best to deploy hybrid and multi-cloud infrastructure to optimize cost and to provide the best user experience in terms of reliability, availability, consistency, and speed. On-premises infrastructure, combined with multi-cloud, prevents organizations from vendor lock-in and puts them in a better position to explore the best available infrastructure for their changing needs. The cost to maintain hybrid and multi-cloud infrastructure often outweighs the cost to manage single cloud or on-premises deployment in many ways. With ever-evolving hardware configurations and workloads, it becomes difficult to measure the performance of a system in pace without having the key performance indicators that can provide insights across all layers of the stack. There seems to be a significant gap in sophisticated tooling and patterns that can predict near-scale values for cost, execution time, and energy consumption on any given target architecture. This will provide a very powerful estimate for organizations that are constantly deploying and maintaining infrastructure for internet-scale workloads [20].

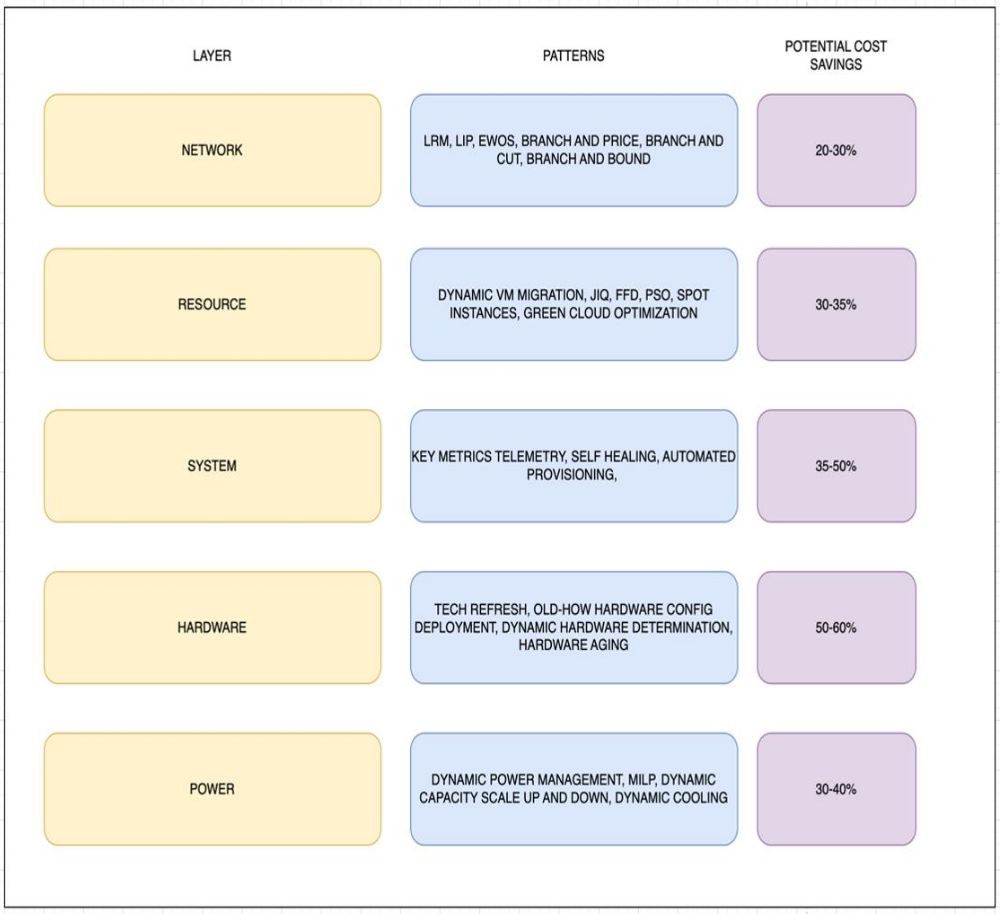
LAYER
PATTERNS
POTENTIAL COST SAVINGS
NETWORK
LRM, LIP, EWOS, BRANCH AND PRICE, BRANCH AND CUT, BRANCH AND BOUND
20-30%
RESOURCE
DYNAMIC VM MIGRATION, JIQ, FFD, PSO, SPOT INSTANCES, GREEN CLOUD OPTIMIZATION
30-35%
SYSTEM
KEY METRICS TELEMETRY, SELF HEALING, AUTOMATED PROVISIONING,
35-50%
HARDWARE
TECH REFRESH, OLD-HOW HARDWARE CONFIG DEPLOYMENT, DYNAMIC HARDWARE DETERMINATION, HARDWARE AGING
50-60%
POWER
DYNAMIC POWER MANAGEMENT, MILP, DYNAMIC CAPACITY SCALE UP AND DOWN, DYNAMIC COOLING
30-40%

Figure 3.
Patterns and savings across different infrastructure layers.

3. Conclusion

In this paper, we performed a comprehensive analysis of the patterns involved in designing hyperscale infrastructures. Hyperscale infrastructure incurs significant costs for deployment and management. To optimize these costs, a holistic strategy across different layers, such as Network, Resource, System, Hardware, and Power, is essential. Each of these layers can result in multi-million or even billion-dollar expenses if the architecture is not designed considering workload and scale. Environmental and economic implications are equally important factors to consider when designing hyperscale architectures. Low power consumption, reusable materials, low carbon emissions, and optimal energy use during operation are critical areas in modern implementations. As the demand for hyperscale infrastructure evolves, the principles and patterns identified in this study will serve as a foundation for sustainable and cost-effective infrastructure development. The network, resource, and hardware layers typically account for costs in the billions of dollars in procurement and provisioning. In Figure 3, we summarize how optimization across these layers can achieve up to 35% cost reduction on average, representing a significant decrease in overall infrastructure and operational costs.

References

[1] Y. Liang, M. Lu, Z. J. M. Shen, and R. Tang, "Data center network design for internet-related services and cloud computing," *Production and Operations Management*, vol. 30, no. 7, pp. 2077-2101, 2021. https://doi.org/10.1111/POMS.13355

[2] N. Salhab, R. Rahim, and R. Langar, "Optimization of virtualization cost, processing power and network load of 5G software-defined data centers," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1542-1553, 2020. https://doi.org/10.1109/TNSM.2020.2990664

[3] A. Osman, A. Sagahyroon, R. Aburukba, and F. Aloul, "Optimization of energy consumption in cloud computing datacenters," *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 11, no. 1, pp. 686–698, 2021. https://doi.org/10.11591/IJECE.V11I1.PP686-698

[4] D. Mukherjee, S. Dhara, S. C. Borst, and J. S. van Leeuwaarden, "Optimal service elasticity in large-scale distributed systems," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 1-28, 2017. https://doi.org/10.1145/3078505.3078532

[5] R. B. Madhumala, H. Tiwari, and D. Verma, "Hybrid model for virtual machine optimization in cloud data center," presented at the International Conference Intelligent Computing and Control Systems, 2021.

[6] M. Jawad *et al.*, "A robust optimization technique for energy cost minimization of cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 447-460, 2018. https://doi.org/10.1109/TCC.2018.2879948

[7] A. Mahgoub *et al.*, "OPTIMUSCLOUD: Heterogeneous configuration optimization for distributed databases in the cloud," presented at the USENIX Annual Technical Conference, 2020.

[8] H. Xiong, F. Fowley, C. Pahl, and N. Moran, *Scalable architectures for platform-as-a-service clouds: Performance and cost analysis*. Cham: Springer, 2014.
[9] C. Thorpe, W. Josephson, J. Moorthi, and S. R. Willis, "Cost optimization of cloud computing resources," 2016. https://patents.google.com/patent/US20160321115A1/en
[10] K. MuthuPandi and K. Somasundaram, "Design of energy efficient datacenter with optimized virtual infrastructure," in *AIP Conference Proceedings*, 2020. https://doi.org/10.1063/5.0024789
[11] What is Prometheus, "What is Prometheus?," 2016. https://prometheus.io/docs/introduction/overview/
[12] Our core stack, "Our core stack," 2025. https://grafana.com/docs/
[13] Amazon CloudWatch Documentation, "Amazon cloudwatch documentation," 2025. https://docs.aws.amazon.com/cloudwatch/
[14] Cloud Monitoring overview, "Cloud monitoring overview," 2025. https://cloud.google.com/monitoring/docs/monitoring-overview
[15] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732-794, 2015. https://doi.org/10.1109/COMST.2015.2481183
[16] Y. Cui, S. Jin, W. Yue, and Y. Takahashi, "Performance optimization of cloud data centers with a dynamic energy-efficient resource management scheme," *Complexity*, vol. 2021, no. 1, p. 6646881, 2021. https://doi.org/10.1155/2021/6646881
[17] Energy Optimization, "Energy optimization in data centers for cloud computing," *Soft Computing Research Society eBooks*, pp. 127–136, 2022. https://doi.org/10.52458/978-81-955020-5-9-12
[18] J. Wang, U. Gupta, and A. Sriraman, "Peeling back the carbon curtain: Carbon optimization challenges in cloud computing," in *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, 2023. https://doi.org/10.1145/3604930.3605718
[19] F. P. Tso, K. Oikonomou, E. Kavvadia, G. Hamilton, and D. P. Pezaros, "S-CORE: Scalable communication cost reduction in data center environments," in "School of Computing Science, University of Glasgow, Tech. Rep. TR-2013-338," 2013. http://eprints.gla.ac.uk/97360/
[20] N. Prajapati, "Analytical cost metrics: Days of future past," Doctoral dissertation, Colorado State University, 2019.

Appendix A.
Configuration for Monitoring Metrics.

```
{
 "groups": [
  {
   "name": "server-monitoring",
   "rules": [
    {
     "alert": "HighMemoryUtilization",
     "expr": "node_memory_Active_bytes / node_memory_MemTotal_bytes > 0.9",
     "for": "5m",
     "labels": { "severity": "warning" },
     "annotations": {
      "summary": "Memory utilization is over 90%",
      "description": "Active memory usage is above threshold on {{ $labels.instance }}"
     }
    },


          {
     "alert": "HighProcessCount",
     "expr": "node_procs_running > 300",
     "for": "5m",
     "labels": { "severity": "warning" },
     "annotations": {
      "summary": "Process count high",
      "description": "More than 300 processes running on  {{ $labels.instance }}"
     }
    },
    {
     "alert": "FileSystemFull",
     "expr": "node_filesystem_usage_bytes / node_filesystem_size_bytes > 0.9",
     "for": "5m",
     "labels": { "severity": "critical" },
     "annotations": {
      "summary": "File system usage above 90%",
      "description": "File system nearly full on {{ $labels.instance }}"
     }
    },
    {
     "alert": "HighDiskUtilization",
     "expr": "rate(node_disk_io_time_seconds_total[5m]) > 0.9",
     "for": "5m",
     "labels": { "severity": "warning" },
     "annotations": {
      "summary": "Disk utilization high",
      "description": "High disk I/O usage on {{ $labels.instance }}"
     }
    },
    {
     "alert": "HighCPUUtilization",
     "expr": "100 - (avg by(instance) (rate(node_cpu_seconds_total{mode=\"idle\"}[5m])) * 100) > 90",
     "for": "5m",
     "labels": { "severity": "warning" }ccc,
     "annotations": {
      "summary": "CPU usage over 90%",
      "description": "CPU usage is high on {{ $labels.instance }}"
     }
    },
    {
     "alert": "HighTemperature",
     "expr": "node_hwmon_temp_celsius > 80",
     "for": "2m",
     "labels": { "severity": "critical" },
     "annotations": {
      "summary": "Temperature above 80°C",
      "description": "Server temperature is high on {{ $labels.instance }}"
     }
    },
    {
     "alert": "ServerUnreachable",
     "expr": "up == 0",
     "for": "1m",
     "labels": { "severity": "critical" },
     "annotations": {
      "summary": "Server is unreachable",
      "description": "Prometheus cannot reach {{ $labels.instance }}"
     }
    },
    {
     "alert": "ServerUnresponsive",
     "expr": "probe_success == 0",
     "for": "1m",
     "labels": { "severity": "critical" },
     "annotations": {
      "summary": "Server not responding",
      "description": "Health check failed for {{ $labels.instance }}"
     }
    },
    {
     "alert": "HighRequestRate",
     "expr": "rate(http_requests_total[1m]) > 1000",
     "for": "2m",
     "labels": { "severity": "warning" },
     "annotations": {
      "summary": "High request rate",
      "description": "Over 1000 requests/sec on {{ $labels.instance }}"
     }
    },
    {
     "alert": "VulnerabilitiesDetected",
     "expr": "vuln_scan_vulnerabilities_total > 0",
     "for": "1m",
     "labels": { "severity": "critical" },
     "annotations": {
         "summary": "Security vulnerabilities detected",
         "description": "One or more vulnerabilities found on {{ $labels.instance }}"
      }
     },
     {
      "alert": "OldHardwareDetected",
      "expr": "node_hw_age_days > 1825",
      "for": "1m",
      "labels": { "severity": "info" },
      "annotations": {
         "summary": "Hardware is over 5 years old",
         "description": "Consider replacing old hardware on {{ $labels.instance }}"
      }
     },
     {
      "alert": "OutdatedPackages",
      "expr": "package_outdated_total > 0",
      "for": "1m",
      "labels": { "severity": "warning" },
      "annotations": {
         "summary": "Outdated packages detected",
         "description": "{{ $value }} outdated packages on {{ $labels.instance }}"
      }
     },
     {
      "alert": "LowUptime",
      "expr": "node_time_seconds - node_boot_time_seconds < 300",
      "for": "1m",
      "labels": { "severity": "info" },
      "annotations": {
         "summary": "Server rebooted recently",
         "description": "Uptime is less than 5 minutes on {{ $labels.instance }}"
      }
     },
     {
      "alert": "OSVersionMismatch",
      "expr": "os_version != 'expected_version_string'",
      "for": "1m",
      "labels": { "severity": "info" },
      "annotations": {
         "summary": "Unexpected OS version",
         "description": "Running unsupported OS version on {{ $labels.instance }}"
      }
     }
    ]
  }
 ]
}
```