IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

Energy-Efficient Design Patterns for Large-Scale Banking Applications Deployed on AWS Cloud

Vijaya Rama Raju Gottimukkala

Software Engineer

ORCID ID: 0009-0009-6758-6716

Abstract: Cloud computing substantially reduces the carbon footprint of IT services, with consumption of energy from renew- able resources being the key driver behind energy-efficient design in cloud data centres. The energy footprint of applications hosted on public clouds is nevertheless non-negligible and sustainability- aware design choices should be employed to minimize it. Large- scale banking applications represent an ideal case for such patterns as they are expected to operate round-the-clock while generally being subject to low resource utilization for significant periods of time. Additionally, besides energy concerns, they are also subjected to nearby real-time performance SLAs, high avail- ability requirements, and predictable traffic provisioning. Typical energy-aware design choices made for services hosted on the AWS public cloud include using energy-demand-aware variants of the services (e.g. EC2 Reserved Instances instead of EC2 On-Demand Instances), reducing the traffic across regions, and adapting the allocation of network and compute resources (e.g. RDS read replicas) to the system load. The recommendations summarized here relate to a full-scale case-study banking application—the piece of project work—deployed into the AWS public cloud. The AWS support for energy-aware design patterns is illustrated by the practical application of System Architecture and AWS deployment design guidelines proposed for making the systems energy-efficient while meeting the aforementioned constraints. The suggested design patterns include principles for sustainable software architecture, patterns that drive power-aware design at the service layer, and techniques for optimizing compute and network resources under the joint action of energy demand and service responsiveness.

Keywords: Energy-Efficient Cloud Architecture, Green Computing in Financial Services, Sustainable AWS Deployment Patterns, Serverless Energy Optimization, Cloud Cost and Carbon Efficiency,E lastic Scaling for Power Reduction, Workload Scheduling and Auto-Scaling, Sustainable Fin- Tech Infrastructure, Carbon-Aware Cloud Orchestration, Energy- Optimized Microservices Design.

I. INTRODUCTION AND SCOPE

The growing emphasis on energy-efficiency in IT, mandated by regulations and driven by public awareness, forces major corporations to integrate these criteria into software design. For companies running banking applications on AWS, the demand for low latency, high availability, and fault tolerance plays a major role in making the choice for conventional design patterns. These patterns, however, increase power con- sumption and carbon emissions. While the cost of transactions reflects requirements and quality of service, electricity is generally seen as free. A combination of cost and energy metrics is needed to optimize energy use without deterioration of performance. Length-of-life indicators (PUE, GWP, CO2e),

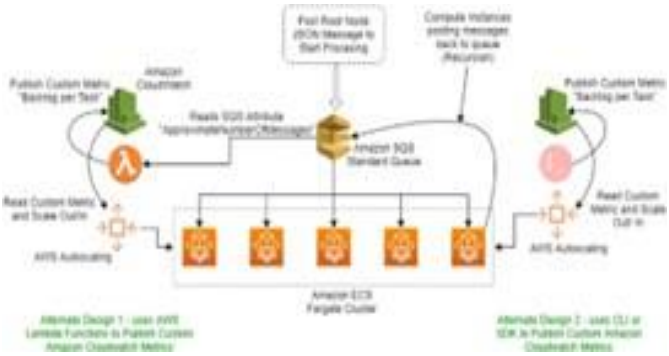

Fig. 1. Architecture: Cloud Design Patterns (AWS)

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 12, December 2020
DOI 10.17148/IJARCCE.2020.91225

power metrics (CPU/GPU utilization, request per second per watt), and cost-per-transaction curves provide guidance for application design, as do three energy-aware principles of cloud applications—adaptability, event-driven behavior, and finishable tasks. Power-aware design patterns indicate when these principles should be applied. Their consideration within the architecture ensures that application behavior stays within the expected boundaries. Power-aware design patterns indicate when principles of sustainable cloud applications—adaptive scaling, event-driven behavior, and tasks that can be fin- ished—should be applied. Application structure and business rules must support the direct integration of autoscaling into production environments. They should also minimize the acti- vation of cross-AZ and cross-region traffic. By reducing zone latency differences to the inside of common-values regions, energy- and cost-conscious consumption patterns can be built.

A. Context and Motivation

Building and operating large-scale online banking services is a challenging task, given the stringent regulatory requirements for latency, reliability, and data safety of such solutions. Besides these requirements, the sustainability of deployed systems is also becoming a major concern for all companies. Online banking systems hosted on AWS are already optimized for energy consumption, thanks to the adoption of Energy Star- compliant hardware, utilization of renewable energy sources, and innovative cooling solutions. Moreover, many AWS ser- vices (e.g., Amazon CloudFront, Amazon Aurora with Global Database, Amazon EC2 Auto Scaling) automatically provide energy optimizations. However, practical implementations are far from optimal. Identifying energy-aware design patterns for banking solutions deployed on AWS, together with formalizing a well-defined set of applicable AWS services, enables practitioners to minimize energy consumption while maintaining the specified performance levels. Energy Efficiency in Banking Applications. The electricity consumed when running a banking solution and the related operational expenses are the most important indicators of energy efficiency in online bank- ing applications. A well-defined set of metrics helps define the minimization process. It includes measurements such as Power Usage Effectiveness (PUE), Global Warming Potential (GWP), CO2e emissions, CPU/GPU utilization, requests per second per watt, and cost per transaction; the last increases the service cost and has to be kept as low as possible as well. These metrics are connected to the energy-efficiency design patterns discussed in the following sections.

B. Key Energy and Cost Metrics in Banking Applications

AWS can be assessed by key performance metrics important for both sustainable software architecture and cloud-network- data optimization. The potential energy use in data centers is indicated by the Power Usage Effectiveness (PUE) metric, while the Global Warming Potential (GWP) and CO2e values associated with energy consumption reflect the application's contribution to greenhouse effect and climate change. Request per second per watt is a measure for the efficient use of compute resources in executing business transactions. Low CPU/GPU utilization exposes a poor compute-resource uti- lization, while the cost per transaction quantifies the expense of providing the services. Together with the latency asked by users, all these key performance indicators (KPIs) are incorporated in a single graph representing banking application energy-efficiency properties on the AWS cloud; this graph highlights the best-cost-maintaining trade-off in Low Latency Demand. The analysis of architectural principles for energy efficiency, along with the identification of compute-resource optimization strategies with main focus on right-sizing big- data workloads, form the foundation for deploying large-scale global banking applications on AWS clouds in a sustainable, cost-efficient, and latency-driven way.

II. ARCHITECTURAL PRINCIPLES FOR ENERGY EFFICIENCY

Fig. 2. Instances over time

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

TABLE I
DAILY SUMMARY: FIXED VS AUTO SCALE

	IT energy (kWh)	Total energy @PUE=1.2 (kWh)
Fixed (peak)	36.03486	43.24183
Autoscale (60% u*)	17.58236	21.09883

construction choices aimed at energy efficiency. The fulfilment of these principles aids the natural pursuit of sustainability, even when no particular green objective is defined. Energy use in banking software is driven by the energy profile of the underlying cloud platform, along with the way in which the on-demand components—mostly databases and data stores—are used. A specialized database design and tiered caching systems, together with carefully thought-out data-access patterns, can minimize the energy consumed by these non-compute parts of the system.

Equation 01: Energy & carbon fundamentals

Power Usage Effectiveness (PUE)

$$PUE \equiv PITPfacility \quad (1)$$

$$\text{so facility} = \text{PUE} \cdot \text{IT} \Rightarrow P_{facility} = \text{PUE} \cdot P_{IT} \cdot P_{IT} \quad (2)$$

$$\textit{Overtime, with energy } E = \int P\, dt E_{facility} = PUE \cdot E_{IT} \quad (3)$$

A. Principles of Sustainable Software Architecture

Decoupled architecture enables applications to evolve in- dependently. Each component has a well-defined interface, facilitating debugging, fault isolation, technology updates, and risk exposure. Statelessness improves robustness by removing state dependencies that prolong recovery from VM or con- tainer failures; resilient external storage mitigates remaining state risks. Consistent observability simplifies understanding runtime behavior and resolving problems. Designing demand- driven systems ensures that execution depends on user requests or business events, avoiding additional work during inactivity. Such design improves fault tolerance and support for short- lived functions, leading to elastically sized, on-demand-priced operations consistent with carbon and electricity pricing. The following cloud-native patterns related to demand-driven be- havior foster power-aware software design: adaptive scaling based on monitored power usage or predicted demand, event- driven operations invoked through messaging queues, and task-based workflows to support scheduling of bursts with capacity interruptions. Demand-driven traffic patterns, especially with peaky usage, minimize costs in a carbon-aware infrastructure.

B. Cloud-Native Patterns for Power-Aware Design

Power-aware design patterns inherent in cloud-native ap- plications provide an essential foundation for energy-efficient deployments of banking systems on AWS. Functionally de- coupled, event-driven, and finishable-process architectures na- tively minimize energy waste and enable demand-sensitive resource provisioning, a property that can be further exploited with dedicated traffic-shaping techniques. When properly ad- justed, these patterns allow all components in the banking application and its back-end systems—including data access nodes, databases, and storage—to benefit from adaptive scal- ing, utilization-based right-sizing, or purely serverless execu- tion. The completeness of this set of design patterns ensures that further energy-related decisions for data, network, and I/O layers remain sound and focus primarily on performance optimization. Functionally decoupled subcomponents require only the resources needed for their own individual load levels. When incorporated into transportation system state machines, throughput per state enables provisioned-layer uti- lization and activity levels to be monitored. So long as the expected workload modes are understood, these patterns can be enhanced with external traffic-shaping techniques to reduce energy wasted in idle states. Emphasizing demand horizontal- ity and time-limited workload duration allows all submove build processes to be run as finishable tasks, exploiting the underlying vanishing-residuals principle to ensure that all tasks fit into available execution resources.

III. COMPUTE AND RESOURCE OPTIMIZATION

Balancing performance and energy use requires right-sizing resources and defining appropriate autoscaling strategies. Right-sizing is driven by business-critical performance re- quirements, and scaling strategies support dynamic

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

workloads with distinct seasonal trends or daily peaks during business hours. It defines thresholds and deltas that trigger scaling actions. Many AWS services can be set up to maximize en- ergy efficiency while still meeting performance requirements. Events can be leveraged to trigger processing in serverless and container solutions (AWS Fargate and a managed Kubernetes installation). A suitable design writes finishable tasks to event sources that AWS Lambda or Fargate consumes, using native integration whenever possible. Underlying processing can use a delayed broker (Amazon SQS with a short visibility time- out) or a fully fledged data-centric stream messaging service

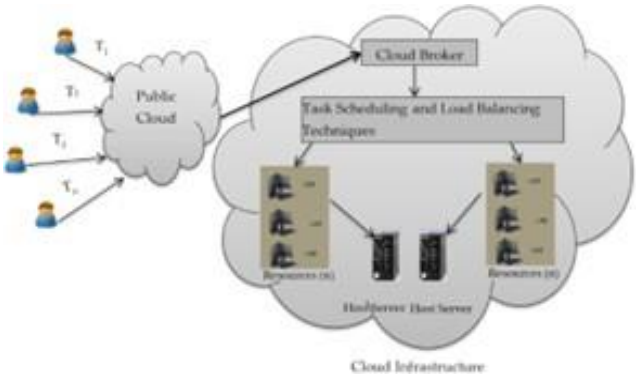

Fig. 3. An Optimized Framework for Energy-Resource Allocation in a Cloud Environment

(Amazon Kinesis). Each consumer group is provisioned for the processing throughput of the complete application, even if the load is not constant. However, sub-second delays are difficult to provide with a finishable-task design, since no native event sources are available. Resource consumption can be adjusted to match the load, which means the run time fluctuates with demand rather than a workload-characterizing metric—CPU use, for example. The ideal is a resource consumption set point, around 60% utilization, that reduces the time-based cost while allowing peak load to fit within non-overscaled resources. For EC2 and Amazon RDS workloads, utilization thresholds cause scaling-ins and scaling-outs, adjusting the number of running resources based on usage. For AWS Fargate containers, the active service size defines scaling, while for managed Kubernetes workloads, Karpenter can automatically provision instances in line with load.

A. Right-Sizing and Autoscaling Strategies

Key metrics for optimizing performance while minimizing power consumption have been defined. Appropriate thresh- olds of CPU and memory utilization that trigger scaling actions have been identified. Workload characteristics have been studied to derive the trade-offs, ramifications, and cri- teria for effective usage of horizontal, vertical, and network- aware scaling. Best practices span the spectrum, covering dynamic adjustment of instance size and type, deployment of adaptive application-layer scaling, integration of autoscaling groups with optimized scaling policies, and characterization of workloads to guide these optimizations. Monitoring CPU and memory utilization is common for horizontal and verti- cal scaling. Tail latency is often employed for vertical and network-aware scaling of managed services. Power-efficiency considerations can also supplement the usage of other met- rics. For extreme long-running and read-heavy workloads, network-endpoint location diversity should be exploited to decrease latency and thus power consumption. Additionally, adaptive scaling and autoscaling groups should operate on traffic patterns rather than on raw usage metrics, striking a balance between false positives and long-polling overheads.

Performance exacerbation has been a recurring problem for latent cloud services. PUE and power consumption per op- eration for virtualized sharing of compute and I/O systems have increased. For transient workloads on low-utilization servers, power consumption of the idle state has overwhelmed compute. Power-aware latent scaling of services, including multi-region architectures and pooling of client resources, can reverse this development.

B. Serverless and Container-Based Workloads

Although serverless architectures are popularly associated with cost savings—only paying when the function is exe- cuted—this is mostly not the case with AWS Lambda imple- mentations of banking applications. Besides, AWS Lambda comes with a major caveat, namely its cold start. When a function is not executed for a while, the executors are deactivated, and the next time the function is called, it takes more time to execute as a new executor needs to be created. Therefore, when mainly idle functions are present, paying for idle time is not optimal. So, when the majority of the execution is near idle time, AWS Lambda becomes expensive. However, with traffic shaping, this caveat can be mitigated to some extent by pacing the traffic. When the function is at hundreds of request per minute or higher, paying for idle time is beneficial compared to using EC2 instances that need to be running and therefore consuming power all the time. Therefore, during these periods of high demand, the function

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

based execution can lead to a higher overall request per second per watt indicator. Besides AWS Lambda, Lambda, the AWS Fargate service that runs containers and the elastic Kubernetes should also be considered similarly when none idle requests are present. Auto scaling is a key cost-energy optimization pattern, but data locality and data access patterns have a great merit on the overall cost-energy indicators. When using AWS services, its architecture is designed to favour the data locality for all services, meaning for example that for an EC2 instance the data will be closest to it in terms of hop count and latency. When using services that do not favour data locality, significant efforts must be put in understanding how to minimize the involved data transfer latency-energy. The ap- plication Cosmos, using a Microservices oriented architecture style, shows that on the APN multiple AWS services are used, including Lambda, EC2, and Fargate, creating a performance oriented implementation. While testing Cosmos in production environment, it became clear that the timing of executions in each of the independent services leads to a waste of resources on the APN, making it a good candidate for traffic shaping.

IV. DATA LAYER AND STORAGE EFFICIENCY

Energy consumption during computation is not the only concern. The energy cost of data movement is significant as well, affecting not just the energy consumption and carbon footprint but also the financial cost. Energy-aware patterns and best practices in the data layer therefore contribute to the overall energy efficiency of the application. Efficient energy- aware design requires careful consideration of where, how,

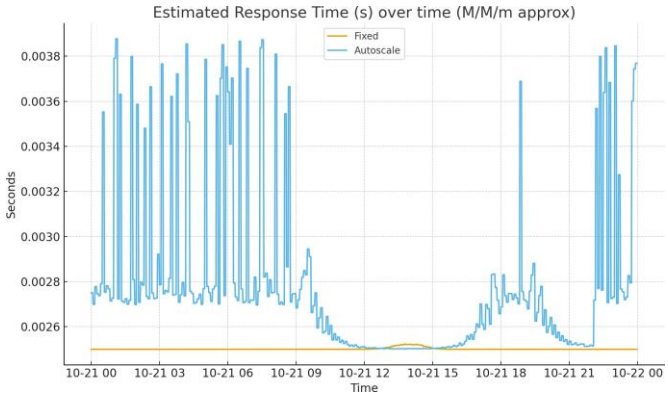

Fig. 4. Estimated Response Time (S) Over Time (M/M/M Approx)

TABLE II
TOTALS COMPARISON

	Total energy (kWh)	CO2e (kg)
Fixed (peak)	43.24183	19.45882
Autoscale (60% u*)	21.09883	9.494473

and when the data is stored. Not only the storage technology (SSD/HDD) but also the choice of indexing, partitioning, and replication affect energy consumption. Caches and tiered caches help reduce energy spent on access in I/O operations. Careful tuning of caching strategies, time-to-live settings, and hot-cold separation can reduce response times, make better use of caches, and minimize latency and associated energy during I/O-intensive compute operations, such as search queries. Additionally, data locality contributes to energy efficiency by limiting the distance data need to travel. When these points are considered, the entire architecture is designed for minimal energy and cost-consumption in data operations.

Equation 02: Cost & efficiency

Let total cost over the period be C and total transactions be
N

$$Cost/tx = NC \quad (4)$$

% In the sim, =instances+egressC=Cinstances+Cegress % Cegress (illustrative), and

$$N = \sum t\, RPS(t) \cdot \Delta t \quad (5)\ (6)$$

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 12, December 2020
DOI 10.17148/IJARCCE.2020.91225

A. Energy-Aware Database Design

Supporting cloud-based applications with energy-efficient databases requires a combination of replication, partitioning, indexing, data transfer optimizations, and judicious cost con- siderations. Energy-aware database operation must connect closely with application-level patterns. An energy-aware ap- plication profile guides decisions on load and burst distri- bution, helping to determine the number of active replicas and whether multi-AZ deployment is appropriate for latency and redundancy. Existence of energy-aware database cost analytics promotes energy-cost consideration when trading read, write, and transfer costs. Partitioning according to access patterns ensures that the energy cost of data transfers does not dominate query costs on partitions. Caution should be exercised during partition replication to minimize write traf- fic for seldom-accessed replicas, and cross-region replication must be matched with an evaluation of transfer energy in mul- tiregion deployments. The most powerful energy-motivated optimization is the use of application-traffic-aware partitioning together with strong-read-quorum, weak-write-quorum repli- cation for NoSQL databases. Carefully planned query struc- tures that minimize indexed-query-set size can reduce index- related energy, especially when combined with read-heavy traffic characteristics downstream of the partition function. With image data, a clear separation of the image-fetch oper- ation from the transcoding operation can allow highly loaded services to draw from a much smaller dataset than is used by the general application-case transcoding. Dominance of write costs in a replica's operation tempts consideration of traffic-pacing strategies that can trade cooling-energy economy against hardware-lifetime economy if the replication lag can be tolerated.

B. Catching, Tiering, and Data Locality

Minimizing the compute energy component is vital for overall energy efficiency. Substantial gains can be achieved by adopting 2.2 cloud-native design patterns. Caching is essential to limit redundant computation and data I/O. Data move- ment is typically the most energy-intensive action, so placing tasks and data in proximity to each other is crucial. Traffic shaping is necessary to alleviate the effect of network idling on energy consumption. Application logic patterns such as request shaping, bulk data movement, and the use of long-lived egress streams or connections can be exploited to minimize energy use during data transfer. Network and transport layer QoS mechanisms must be leveraged to reduce the need for excessive provisioning of network bandwidth. Idle resources should be powered down or scaled to lower configurations. Multi-tier cache hierarchies, ideally with separate TTL policies for different data classes, can significantly reduce redundant compute energy. The potential benefit is maximized when tasks are deployed in geographical proximity to cache tiers. Caching mechanisms, however, may incur additional compute costs when monitoring and maintaining data consistency. Care- ful strategy selection is required to maximize the net gain. Data properties and access patterns must be analyzed to determine the most useful set of caches. The optimization problem can be partly solved by spatially partitioning workloads based on user request locality prior to execution.

V. NETWORKING AND I/O EFFICIENCY

Energy-efficient banking applications should also optimize I/O operations and their networking topology. An isolated service can behave well under low latency requirements using

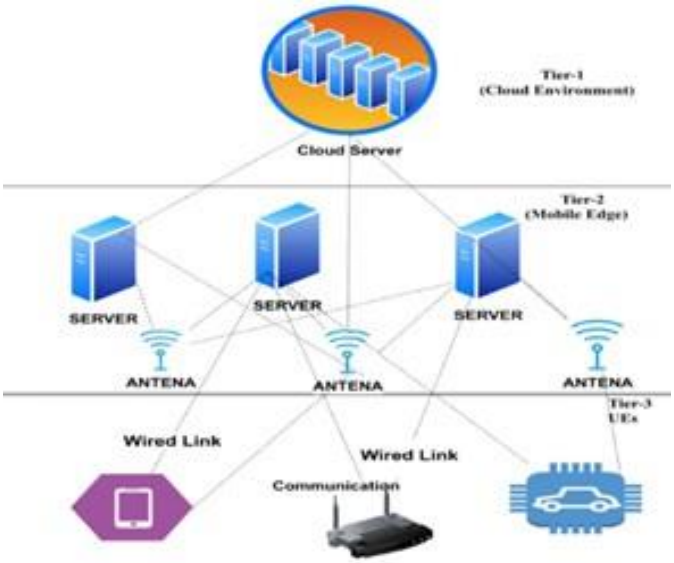

Fig. 5: Networking and I/O Efficiency

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

AWS services that cross regional boundaries. However, every such region has a cost involved not only for data transfer but also in terms of the amount of energy used for that data transfer. So, some users may prefer paying a little more for the entire service if that service shows energy efficiency. Seamless operation across AWS zones may provide HA at the application level in the event open/close requests are issued on the same service instance and run across separate AWS zones. Thus, these should be efficiently planned considering both HA and energy consumption. With wired networks in- creasingly more efficient than wireless, power consumption is often higher when the devices are idle rather than under heavy use, resulting in idle power levels close to the peak. Therefore, whenever the objectives include energy efficiency, ideally, these idle phases must be eliminated or reduced. Rainflow congestion, bandwidth throttling, and QoS level usage can help in reducing inter-service communication when using VB across services. Further, software optimization (of HTTP long polling, TCP connection, etc.), connection pooling, and connection reuse techniques would also provide energy savings.

1) Cross-Region and Multi-AZ Considerations for Energy Use: Among the diverse aspects relevant to energy and performance in banking applications, operating in multiple availability zones (AZs) and AWS regions can be significant. Of particular interest are the energy costs involved when data must be transferred between these locations due to replication or other reasons, or during a failover. Although moving resources across regions may have implications for latency and CO2 emissions, the energy introduced by the extra hops in the network should also be considered. Communication between regions normally travels over the public Internet. While all data packets are sent through multiple carriers and data centers of varying energy efficiency, these hops consume electric power without any useful computational work be- ing performed. Region pairs offering accelerated cross-region transfer using the AWS backbone tend to have considerable economic and ecological advantages—and in addition, can reduce the energy impact. The fact that these locations remain fixed over time makes them suitable for database architectures requiring low latency. Customers willing to pay a higher price may prefer private links, which may provide the fastest or the most energy-efficient solution.

A. Traffic Shaping and Connection Management

Effective traffic shaping and connection management can significantly reduce the idle power consumption of servers in a system while maintaining service quality. One approach is to characterize the demand for connection establishment and treat it as a discrete event, triggering connection creation and destruction. Doing so minimizes the amount of time the connection is idle. A different approach is to change the shape of the traffic being sent, smoothing and spacing it out to limit peak bandwidth requirements. Finally, the reuse of TCP connections when using HTTP can be leveraged; keeping HTTP connections open can prevent large delays at the cost of keeping the connection active. Excessive numbers of kept open connections must also be avoided, as they still consume resources on both the client and server side. For databases, a certain amount of idle persistent connections must be maintained to allow fast response to queries. However, it is better to optimize for high-buffer-cache efficiency and keep the connection over TCP open for a long time (e.g., hours), as the idle connection is lightweight. It is, nonetheless, still prudent to avoid excessive fragmentation and excessively small queries. Queued connections must be reused when tuning the server's socket settings and can be gained from placing a limit on the number of allowed connections.

VI. CONCLUSION

In sum, the principles outlined contribute to energy- efficient design patterns for large-scale banking applications deployed in the AWS public cloud. The balance between cost, performance, and energy use can be efficiently managed through an integrated approach whereby all layers of the application consider all dimensions. Control of these dimensions at the fastest time scale possible leads to limit- compliant energy-efficient banking applications that are not only environmentally friendly but also cost effective. The economic aspect is crucial as the sustainability and environmental damage aspects play only a minor role in such a solution. The various proposals should, however, undergo further practical validation. Finally, some of the tasted patterns are expected to become even more relevant in the near future. The interest in AI/ML is moving toward the scheduling domain. Scheduling workloads on a plethora of heterogeneous hardware resources with different levels of energy efficiency is a fascinating area of research, and power-aware scheduling of AI workloads for automatic resource provisioning should become commercially available soon. Hardware engineers are becoming more aware of the problem. The provision of a hardware monitoring tool through the AWS cloud that can estimate the carbon impact of the operation of EC2 instances should grow interest in the selection of the right instances, not only for cost but also for a lower carbon footprint. The migration of applications in the banking domain toward a microservice architecture, together with the possibility of routing DB calls to the least loaded regions, opens the door to a powerful mean of distribution of the load on the Networking I/O dimension across regions; a method that should be explored in the near future is the class-based routing for different types of services with different environmental impact when the load is reduced.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

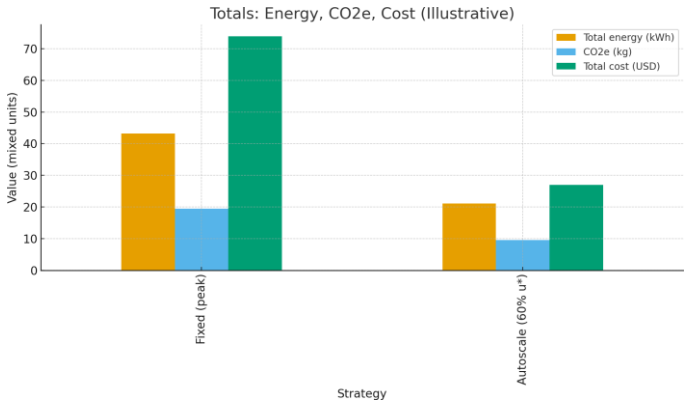

Fig. 6. Totals: Energy, CO2e, Cost (Illustrative)

Equation 03: Latency/SLA orientation (queueing view)

Arrival rate $\lambda = L(t)$ (RPS) Per-instance service rate $\mu = \kappa$ (RPS) Servers $m = n(t)$

Utilization per server $\rho = \lambda/(m\mu)$ (needs < 1)

Erlang-C wait probability PW and waiting time

$$W = \sum k = 0m - 1k!(m\rho)k + m!(1-\rho)(m\rho)mm!(1-\rho) \quad (7)$$

$$(m\rho)m \; W_q = m\mu - \lambda P_W, \; W = W_q + \mu 1 \quad (8)$$

A. Emerging Trends

Several emerging trends may influence future energy- efficient designs on the AWS cloud. AWS powers services and servers with sustainable energy such as hydro and wind, but mainframes for AI training still consume excessive resources. Power-efficient AI-ML deployment models integrated into AWS services or with third parties that follow energy-aware design patterns would extend such scheduling to service- oriented architecture. Reference [36] shows that real-time energy forecasting on IoT devices can inform scheduling, and AI-ML models can also enable cloud service providers to deliver AI-ML jobs with reduced energy consumption without compromising users' continual real-time requirements. Greener cloud- and hardware-side designs may improve sus- tainability by interacting with energy providers at larger power stations. Copying AWS products, future AWS and cloud ser- vice providers should integrate AI-ML prediction functions to choose energy sources with the lowest carbon index and real- time update devices to work with energy-polluting centers in non-peak hours. Sustainability benchmarking should influence such ecosystems and products. Incorporating carbon-aware routing functions into Layer 4 and 7 services would provide traffic routing based on carbon consumption without perfor- mance loss during peak hours. Moreover, OpenAI and Google are developing their own LLM models that require huge resources and energy. Incorporating power and carbon metrics into planned language models would reduce energy consump- tion in prediction and training and control I/O bandwidth and energy consumption in data exchange. This function can be in- tegrated into basic data exchange and training services of AWS and similar service providers. Power-aware design patterns that target energy reduction while maintaining integrity and reliability strongly resonate with banking applications. Energy consumption of such battery-powered devices but with lower performance demand may attract industrial attention.

REFERENCES

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., . . . Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33.

[2] Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. The Lancet Infectious Diseases, 20(5), 533–534.

[3] Gadi, A. L. (2020). Evaluating Cloud Adoption Models in Automotive Manufacturing and Global Distribution Networks. Global Research Development (GRD) ISSN: 2455-5703, 5(12), 171-190.

[4] Wu, Z., & McGoogan, J. M. (2020). Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China. JAMA, 323(13), 1239–1242.

[5] Zhou, F., Yu, T., Du, R., Fan, G., Liu, Y., Liu, Z., . . . Cao, B. (2020).Clinical course and risk factors for mortality of adult inpatients with COVID-19 in Wuhan, China. The Lancet, 395(10229), 1054–1062.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 12, December 2020

DOI 10.17148/IJARCCE.2020.91225

[6] Richardson, S., Hirsch, J. S., Narasimhan, M., Crawford, J. M., McGinn, T., Davidson, K. W., . . . the Northwell COVID-19 Research Consortium. (2020). Presenting characteristics, comorbidities, and outcomes among 5,700 patients hospitalized with COVID-19 in the New York City area. The New England Journal of Medicine, 382(24), 2302–2311.

[7] Pandiri, L. (2020). Predictive Modeling of Claims in Flood and Mobile Home Insurance using Machine Learning. Global Research Development (GRD) ISSN: 2455-5703, 5(12), 1-18.

[8] Li, Q., Guan, X., Wu, P., Wang, X., Zhou, L., Tong, Y., . . . Feng, Z.(2020). Early transmission dynamics in Wuhan, China, of novel coron- avirus–infected pneumonia. The New England Journal of Medicine, 382, 1199–1207.

[9] Cucinotta, D., & Vanelli, M. (2020). WHO declares COVID-19 a pandemic. Acta Biomedica, 91(1), 157–160.

[10] Rubin, G. J., & Wessely, S. (2020). The psychological effects of quarantines and how to reduce them. The Lancet, 395(10227), 912–920.

[11] Botlagunta, P. N., & Sheelam, G. K. (2020). Data-Driven Design and Validation Techniques in Advanced Chip Engineering. Global Research Development (GRD) ISSN: 2455-5703, 5(12), 243-260.

[12] Polack, F. P., Thomas, S. J., Kitchin, N., Absalon, J., Gurtman, A., Lockhart, S., . . . Gruber, W. C. (2020). Safety and efficacy of the BNT162b2 mRNA Covid-19 vaccine. The New England Journal of Medicine, 383(27), 2603–2615.

[13] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., . . . Houlsby, N. (2020). An image is worth 16×16 words: Transformers for image recognition at scale. arXiv Preprint arXiv:2010.11929.

[14] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9729–9738.

[15] Chakilam, C., Koppolu, H. K. R., Chava, K. C., & Suura, S. R. (2020). Integrating Big Data and AI in Cloud-Based Healthcare Systems for Enhanced Patient Care and Disease Management. Global Research Development (GRD) ISSN: 2455-5703, 5(12), 19-42.

[16] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. Proceedings of the 37th International Conference on Machine Learning (ICML), 1597–1607.

[17] Kolesnikov, A., Zhai, X., & Beyer, L. (2020). Big Transfer (BiT): Gen- eral visual representation learning. European Conference on Computer Vision (ECCV) 2020, 491–507.

[18] Le Que´re´, C., Jackson, R. B., Jones, M. W., Smith, A. J. P., Abernethy, S., Andrew, R. M., . . . Peters, G. P. (2020). Temporary reduction in daily global CO_2 emissions during the COVID-19 forced confinement. Nature Climate Change, 10(7), 647–653.

[19] Dwaraka Nath Kummari, Srinivasa Rao Challa, "Big Data and Machine Learning in Fraud Detection for Public Sector Financial Systems," Inter- national Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), DOI: 10.17148/IJARCCE.2020.91221

[20] Hsiang, S., Allen, D., Annan-Phan, S., Bell, K., Bolliger, I., Chong, T.,
. . . Walsh, S. (2020). The effect of large-scale anti-contagion policies on the COVID-19 pandemic. Nature, 584(7820), 262–267.

[21] Chetty, R., Friedman, J. N., Hendren, N., Stepner, M., & the Opportunity Insights Team. (2020). How did COVID-19 and stabilization policies affect spending and employment? NBER Working Paper No. 27431.

[22] World Health Organization. (2020). Mask use in the context of COVID- 19: Interim guidance (1 December 2020). WHO.

[23] Meda, R. (2020). Designing Self-Learning Agentic Systems for Dynamic Retail Supply Networks. Online Journal of Materials Science, 1(1), 1–20. Retrieved from https://www.scipublications.com/journal/index.php/materials/article/view/1336.

[24] United Nations. (2020). The Sustainable Development Goals Report 2020. United Nations.

[25] International Monetary Fund. (2020). World Economic Outlook, October 2020: A long and difficult ascent. IMF.

[26] OECD. (2020). Education at a Glance 2020: OECD indicators. OECD Publishing.

[27] Somu, B. (2020). Transforming Customer Experience in Digital Bank- ing Through Machine Learning Applications. International Journal Of Engineering And Computer Science, 9(12).

[28] Piketty, T. (2020). Capital and ideology (A. Goldhammer, Trans.). Harvard University Press.

[29] Case, A., & Deaton, A. (2020). Deaths of despair and the future of capitalism. Princeton University Press.

[30] Wilkerson, I. (2020). Caste: The origins of our discontents. Random House.

[31] Inala, R. (2020). Building Foundational Data Products for Financial Services: A MDM-Based Approach to Customer, and Product Data Integration. Universal Journal of Finance and Economics, 1(1), 1–18.