Research

Strategies for Cost-Effective Big Data Management in the Cloud: Leveraging Auto-Scaling and Spot Instances

Ramesh Adhikari[1]

[1] Pokhara Engineering College, Department of Computer Science, Shivam Marga, Pokhara, Kaski, Nepal.

Abstract: This paper explores advanced strategies for managing large-scale datasets in cloud environments with a focus on minimizing operational costs while ensuring system responsiveness and reliability. Through extensive theoretical modeling and empirical reasoning, a framework for orchestrating big data workflows that leverages auto-scaling capabilities and spot instances is presented. The approach centers on dynamically allocating computational and storage resources in response to fluctuating workloads, guided by stochastic optimization methods. The objective is to address unpredictable spikes in demand that can arise in data analytics pipelines while capitalizing on reduced prices offered through transient, preemptible instances. Mathematical formulations detail the trade-offs between acquiring on-demand resources for guaranteed availability and utilizing cost-effective, potentially volatile spot instances. Performance metrics are derived to quantify the probability of service interruption, expected queuing delays, and overall system throughput. The technique draws on large-scale parallel processing paradigms, including distributed file systems and containerized execution engines, to accommodate multi-terabyte data streams. Results highlight the significance of accurate predictive models and real-time usage monitoring for resource scaling triggers. It is shown that the judicious blend of elastic resource provisioning and ephemeral infrastructure can significantly lower expenditures while maintaining robust service-level guarantees. The paper concludes with future directions for extending these models to more diverse big data scenarios and evolving cloud pricing mechanisms.

1. Introduction

The growing volume, velocity, and variety of data in contemporary computational landscapes has necessitated continual innovations in how large datasets are stored, processed, and analyzed in cloud environments [1]. The agility offered by virtualized infrastructures allows organizations to dynamically scale resources to accommodate fluctuating workloads, yet the cost ramifications of on-demand deployments can be non-trivial. Dynamic scaling orchestrations require comprehensive planning that balances the use of comparatively expensive, but guaranteed, on-demand instances with more affordable, but inherently transient, spot instances. This equilibrium becomes particularly relevant when data pipelines must handle unpredictable spikes in processing demands [2]. Transient cost structures, spot market volatility, and usage-based billing contribute to complex operational cost models. These realities call for rigorous mathematical frameworks that can quantify the trade-offs in availability, performance, and expense.

In the context of big data ecosystems, many analytical tasks involve batch processing of large volumes of data, real-time analytics on streaming data, and iterative machine learning workloads [3]. Effective management of these tasks in the cloud hinges on a scalable model for resource provisioning that is responsive to workload dynamics. Such a model must account for the potential preemptions associated with the use of spot instances, wherein a running instance may be terminated if market prices rise above a user's bid, or if the capacity is otherwise constrained. Moreover, the coupling of autoscaling triggers with

. . *Helex-science* **2023**, *8*, 1–15.

different layers of the big data processing architecture, including storage tiers and container orchestration systems, adds another dimension of complexity to operational strategies. [4]

A foundational component of this landscape is the interplay of cost structures in the computing layer, storage layer, and networking layer. While on-demand resources maintain fixed unit costs across time intervals, spot instances bear variable prices that often fluctuate depending on overall demand and available capacity in the cloud provider's pool. Consequently, a polynomial or piecewise function to represent cost can be employed to formulate resource allocation decisions mathematically [5]. Combining such cost functions with constraints on latency, throughput, and reliability can yield comprehensive optimization problems. These optimization goals often exist within high-dimensional parameter spaces—incorporating number of instances, their types, job priorities, concurrency levels, and target response times.

The concept of elasticity in the cloud underscores the need for real-time decision-making [6]. As data ingestion rates or analytics requests surge, additional instances should be automatically provisioned; conversely, resources should scale down to avoid idle capacity that accrues unnecessary expense. Advanced prediction models, such as those based on time series analysis or machine learning methods, can forecast incoming request loads and proactively adjust capacity. When combining such forecasts with spot instance pricing models, one obtains a dynamic scenario in which potential cost savings must be balanced against the probability of abrupt job termination [7]. Multi-layered caching and queueing systems may be introduced to mitigate adverse effects of instance revocations, but this design choice itself brings added complexity to capacity planning.

In striving for high-level concurrency and low-latency pipelines, big data engines rely on distributed storage solutions that replicate data across multiple nodes for fault tolerance. Such redundancy can influence spot instance selection because the system can tolerate partial failures if there is adequate replication [8]. The optimization framework, therefore, should incorporate a failure-tolerance coefficient that measures how sensitive the overall analytics pipeline is to sudden node removals or ephemeral resource constraints. Coupled with advanced scheduling algorithms in container orchestration frameworks, these considerations can help ensure that ephemeral and on-demand compute nodes form a cohesive, resilient infrastructure.

The subsequent sections delve into the theoretical underpinnings of cost-effective big data management, elaborate on architectural designs for auto-scaling, and investigate the strategies for effectively leveraging spot instances [9]. A rigorous mathematical formalism is introduced to minimize cost while preserving quality-of-service requirements, with an emphasis on controlling the risk of service interruptions. Through in-depth modeling, it is shown how a combination of advanced predictive engines, real-time analytics, and cloud-native orchestration capabilities can realize a cost-optimized yet performance-conscious platform for big data processing.

2. Theoretical Foundations of Cost-Effective Big Data Management

The cornerstone of big data management in cloud settings lies in constructing an analytical foundation that identifies the interplay of performance, reliability, and cost [10]. A natural way to represent the complexity of this interplay is through a stochastic framework that captures random fluctuations in the workload arrival rate and spot market prices. Consider a data pipeline that must process incoming tasks, each with a randomly distributed size, denoted by a random variable X. Similarly, the rate of arrival for these tasks can be modeled with a Poisson process of rate λ [11]. The cloud infrastructure has to ensure that adequate computational resources, denoted by a capacity function $C(t)$, are provisioned at any time t. The total incurred cost, K, over a time horizon $[0, T]$, can be approximated by an integral or summation over discrete intervals, depending on the specific pricing model.

A standard approach might set: [12]

$$K = \int_0^T \left[\alpha \cdot C_{\text{on\_demand}}(t) + \beta \cdot C_{\text{spot}}(t)\right] dt,$$

where α and β represent cost rates for on-demand and spot usage, respectively. This cost function can be enhanced to incorporate penalty terms that account for dropped tasks or extended latency when the available capacity is inadequate. Let L represent the average latency, measured as the expected time for a task to complete from the moment it enters the system [13]. A penalty term $\gamma \cdot \max(0, L - L_{\max})$ may be added to represent the cost of exceeding a latency threshold $L_{\max}$. The refined cost expression becomes:

$$K_{\text{ref}} = \int_0^T \left[\alpha \cdot C_{\text{on\_demand}}(t) + \beta \cdot C_{\text{spot}}(t)\right] dt + \gamma \cdot \max(0, L - L_{\max}).$$

This general representation allows for flexible adaptation to various cloud provider pricing structures and system performance constraints.

The random nature of spot pricing can be incorporated via time-varying $\beta(t)$, which captures fluctuations in market-based costs [14]. If the spot market is active and the bidding price for resources is $p(t)$, the actual cost of spot instances might be represented as $\beta(t) = f\big(p(t)\big)$, where f is a function mapping market conditions to actual costs paid. Furthermore, the probability of eviction for a spot instance, denoted $\epsilon(t)$, could be inversely related to the difference between the current market price and the user's bid. Should the market price exceed the bid or the provider's capacity, one or more spot instances might be revoked, introducing the risk of partial job failure or service interruption. [15,16]

Such a modeling approach opens the door to analyzing the stationary distribution of queue lengths, representing how tasks accumulate when capacity is insufficient. For instance, a continuous-time Markov chain can be used to capture the dynamics of system states, where each state corresponds to a certain number of active spot instances, on-demand instances, and queued tasks. The transition rates would be determined by job arrivals, completion rates, eviction events, and resource scaling triggers [17]. In analyzing the equilibrium or transient states of this Markov chain, one can derive expressions for average job waiting time, resource utilization, and cost. A key challenge is to incorporate the decision processes that determine when to scale resources up or down, given the knowledge of the system state and spot market conditions.

Another relevant aspect is the phenomenon of correlated failures [18]. In many situations, spot prices are not independent across multiple availability zones or instance types; this introduces a potential coupling effect that can lead to simultaneous revocations of numerous spot instances. Strategies for cost minimization should thus consider not only the number of spot instances used, but also their diversity across geographically dispersed regions or across multiple instance families. One line of analysis addresses the problem of diversification by formulating a multi-dimensional optimization problem whose solution indicates how to distribute workloads among different spot instance groups in order to reduce the probability of catastrophic simultaneous revocations. [19]

In addition to computational costs, data storage and movement charges can play a critical role in the total expenditure. When large volumes of data need to be transferred from on-premise systems to the cloud or across regions, egress fees and cross-zone data transfer costs might overshadow computational savings. A more generalized cost function may thus include terms for data storage (reflecting the per-gigabyte or per-hour rates) and data transfer [20]. This expanded perspective compels a holistic approach that balances the savings from low-cost computational resources with the additional overheads that might arise from re-locating data to zones with cheaper spot prices.

These theoretical underpinnings form the basis for an analytical approach to cost-effective big data management. By embedding resource allocation decisions in a stochastic framework, the interplay of performance metrics, financial outlays, and reliability constraints can be systematically explored. The subsequent section addresses architectural considerations needed to implement autoscaling strategies in practice, leveraging this theoretical framework as a guideline for design. [21]

3. Architecture for Auto-Scaling

Effective auto-scaling rests on a well-orchestrated architecture that can handle both real-time triggers and predictive resource allocation. A typical layered design includes user-facing applications, a cluster management layer, a container or virtual machine orchestration subsystem, and resource monitoring agents. The user-facing tier interfaces with incoming tasks or data streams and maintains service-level objectives for performance [22]. The cluster management layer coordinates virtualized computing instances, some of which may be on-demand while others are spot, to ensure that capacity aligns with the target throughput or latency constraints. Container orchestration platforms usually facilitate the deployment of microservices or batch jobs across these resources.

A monitoring and metrics layer is essential to detect changes in load or in cloud market conditions, updating the cluster management with near-real-time data about resource utilization [23], queue lengths, and spot prices [24]. Such metrics might include CPU usage, memory usage, network throughput, as well as job wait times and completion times for big data tasks. A specialized agent can poll the cloud provider's spot market feed to track the latest pricing and eviction probabilities, feeding this information into a policy engine that decides how many spot and on-demand instances to acquire.

In this auto-scaling architecture, a significant design consideration is the coupling between the underlying storage subsystem and the compute layer [25]. The data storage component may span object stores for raw data, distributed file systems for processing, and specialized databases for structured data queries. To accommodate ephemeral nodes that can be terminated unexpectedly, the architecture often includes data replication or snapshot mechanisms that ensure minimal data loss in the event of instance revocation. One approach uses an in-memory data grid that replicates hot data among a pool of instances, mitigating the impact of losing any single node [26]. The overhead of replication, however, becomes part of the cost equation in a full system analysis.

Another subtlety arises from the need to segment workloads based on priority or importance. A job that runs continuously in a production environment, serving user-facing needs, might require guaranteed capacity on on-demand instances [27]. Conversely, batch analytics jobs with flexible deadlines might tolerate multiple interruptions, making them excellent candidates for spot resources. The architecture may implement job categorization logic that classifies incoming tasks into tiers and routes them to the appropriate provisioning strategy. A well-designed auto-scaler can even subdivide large jobs into smaller tasks that can be distributed between on-demand and spot capacity [28]. If partial revocation occurs, only the spot-allocated portions of the job need to be re-run, thus limiting the impact on costs and timelines.

A critical aspect of such an architecture is the autoscaling policy engine, which can be realized as a finite state machine or as a reinforcement learning agent that observes system states and chooses scaling actions [29]. In simpler cases, threshold-based policies might compare current CPU utilization, queue lengths, or job wait times to predetermined setpoints, triggering scale-up or scale-down events [30]. More sophisticated approaches employ model predictive control, in which the system predicts future load, future spot market conditions, and the likely cost of resource usage. The policy then solves an online optimization problem aiming to minimize cumulative cost while satisfying performance constraints. Mathematically, this can be expressed as: [31,32]

$$\min_{u_1,\dots,u_H} \sum_{t=1}^{H} \Big(G(x_t, u_t) + P(x_t, u_t) \Big),$$

where x_t is the system state at time t, u_t is the control action (such as adding or removing instances), $G(\cdot)$ is a cost function capturing resource usage, and $P(\cdot)$ is a penalty function enforcing performance constraints. The horizon length H determines how far into the future the controller looks, with each new time step updating the prediction to reflect new system measurements.

To facilitate reactive decisions, the architecture can include checkpointing mechanisms in big data frameworks [33]. If a job is partially completed on spot instances and is abruptly interrupted, the checkpoint can be used to resume processing on new instances when they become available again. This design approach significantly reduces wasted computation time but introduces overhead for storing and managing checkpoints. This overhead might be offset by the substantial cost reductions that spot instances can provide, as well as by the capacity to handle bursts of activity that might otherwise exceed budgetary constraints if only on-demand instances were used. [34]

Bridging these architectural details with the earlier theoretical frameworks provides a blueprint for implementing autoscaling in practice. While the abstract modeling techniques determine the optimal combination of on-demand and spot resources from a mathematical viewpoint, the actual system must contend with the complexities of distributed data storage, container orchestration, and real-time spot market feeds. The resulting design is necessarily multifaceted and calls for specialized monitoring, robust fault tolerance measures, and advanced policy engines to execute the scaling logic. [35]

4. Utilization of Spot Instances

Spot instances can significantly reduce operational costs for big data tasks, particularly when large parallel jobs can be flexibly scheduled. Nevertheless, their ephemeral nature introduces uncertainties that must be carefully managed. The crux of such utilization strategies lies in establishing the optimal number of spot instances to request and the price bid for these instances, within the broader context of overall job scheduling and resource allocation. [36]

One fundamental approach is to treat spot instance utilization as a problem of maximizing expected utility subject to a set of performance constraints. If one denotes by u the utility of completing a task within a certain timeframe, and by c the associated cost, the expected utility of employing spot instances can be written as:

$$\mathbb{E}[\text{Utility}] = (1-\epsilon)\cdot u - \epsilon \cdot c_{\text{retry}},$$

where ϵ is the probability of eviction before the task concludes, and c_{retry} captures the cost of restarting the task or the penalty incurred if the task fails. A risk-averse strategy seeks to limit ϵ by adjusting the bid price upwards, thereby increasing the odds of retaining the instance but reducing potential savings compared to lower bids [37]. A risk-neutral strategy might bid close to the on-demand price to minimize cost while accepting a higher chance of eviction.

In the context of big data frameworks such as distributed map-reduce systems or container-based batch schedulers, tasks typically get split into smaller chunks of data [38]. Each chunk runs independently, which can mitigate the risk from spot terminations [39]. The partial completion approach ensures that only the chunk in progress is lost if an instance is revoked, and others continue uninterrupted. This chunk-level fault tolerance can be mathematically expressed by factoring in partial completions into the overall job completion time distribution. If n tasks run in parallel on m spot instances, the job completion time T_{job} can be represented as:

$$T_{\text{job}} = \max\{T_1, T_2, \ldots, T_n\},$$

where T_i is the time to complete task i [40]. Each T_i depends on the probability of intermittent revocations, leading to multiple restarts before final completion. Consequently, the distribution of T_{job} is shaped by the interplay between concurrency (the number of tasks running simultaneously) and the hazard rate of spot revocations.

Some implementations include checkpointing or state-saving approaches that allow a partially completed map or reduce function to be resumed on a new instance. Mathematically, one can define a state variable s_i representing how far task i has progressed [41]. If an instance is lost, a new instance can pick up from s_i rather than starting from zero. This scenario can be modeled with a piecewise function describing the time to completion after a revocation event. The cost trade-offs of such measures revolve around the overhead of writing state information to a durable store versus the cost saved by reducing wasted computation on restarts. [42]

Another important aspect is the heterogeneity of spot prices across different instance types and regions. If a big data pipeline can run equally well on multiple instance families, then optimization can be viewed as a resource selection problem with distinct cost distributions and revocation probabilities for each candidate resource type. One might construct a multi-dimensional linear or nonlinear optimization that decides the fraction of tasks allocated to each instance type to minimize overall cost while satisfying a deadline constraint. An illustrative objective function could be: [43]

$$\min_{\{x_j\}} \sum_{j \in \mathcal{I}} \beta_j x_j, \quad \text{subject to} \quad \sum_{j \in \mathcal{I}} \mu_j x_j \geq D,$$

where $\mathcal{I}$ is the set of instance types, β_j is the expected cost per unit of work on type j, μ_j is the throughput contribution of type j, x_j is the allocation variable indicating how much of the job is assigned to type j, and D is the total amount of work to be completed by the deadline. This formulation can be further extended to account for spot revocation probabilities, which reduce effective throughput.

When integrated into a broader auto-scaling framework, spot utilization strategies may shift over time as spot market conditions evolve. The system can continuously monitor bidding prices, instance availability, and the current progress of the workload [44]. Based on these observations, the system may rebalance tasks across different instance pools or gradually replace spot instances with on-demand ones if the financial advantage of spot pricing wanes.

In sum, the utilization of spot instances presents a dynamic challenge that integrates cost modeling, performance constraints, and risk management. By decomposing workloads into smaller tasks, enabling checkpointing, and exploiting heterogeneity across instance types, one can capitalize on transient low prices [45]. The theoretical and architectural underpinnings laid out in earlier sections thus converge to create a robust and flexible approach for cost-effective big data processing in the cloud.

5. Data Flow and Performance Metrics

Effective management of large-scale data hinges on orchestrating both data flow and computational tasks across an elastic infrastructure. Data pipelines typically ingest information from various sources—such as streaming sensors, transactional logs, or external databases—transform and aggregate the information, then feed it into analytical engines or long-term storage [46]. Performance metrics for this pipeline may range from end-to-end latency, throughput, and data freshness to resource utilization ratios and monetary costs over time.

An initial step in managing data flow is constructing ingestion layers that can buffer surges in input without overwhelming downstream analytical components. The ingestion layer might employ a distributed queue or a messaging system that decouples producers and consumers [47]. The concurrency level at which messages are consumed is tied to the number of provisioned compute instances. If an auto-scaling policy detects an accumulation of backlogged messages beyond a set threshold, it can trigger the provisioning of additional spot or on-demand instances to accelerate consumption.

The latency imposed by data movement between compute nodes and storage systems can become a bottleneck, particularly when data sets are extremely large [48]. The cost function introduced earlier can be modified to incorporate the cost of transferring data across regions if spot instances are predominantly available in certain geographic zones. Let C_{transfer} denote the per-unit data transfer cost, and let V be the volume of data being transferred. The total data transfer cost would be $C_{\text{transfer}} \times V$. If the system needs to replicate data for fault tolerance, or move partial results between nodes, these costs can become substantial.

A comprehensive performance analysis must integrate queueing theory and scheduling strategies [49]. Suppose the ingestion rate follows a Poisson process with parameter λ, and each message or data chunk requires a service time distributed with mean $\frac{1}{\mu}$. If the system scales to m instances, the effective service rate is $m\mu$. A fundamental metric is the average queue length, given by:

$$L = \frac{\rho}{1-\rho},$$

where $\rho = \frac{\lambda}{m\mu}$. When ρ approaches 1, the system experiences increasing waiting times and potential timeouts [50]. In practice, ρ should be maintained well below 1 to ensure acceptable response times. Auto-scaling policies often strive to keep the ratio of arrival rate to total service rate below a certain threshold by dynamically adjusting m. However, because μ may be dynamic in big data contexts (particularly when tasks become more complex or when partial caching speeds up repeated queries), advanced predictive approaches are required to avoid miscalculating resource demands. [51]

A further layer of complexity emerges if data flows are subject to service-level agreements specifying tail latency constraints. Such constraints might require that 99.9% of messages be processed within a certain timeframe. The tail latency is often governed by rare events, such as stragglers in distributed data processing or temporary spikes in load [52]. Some analyses use extreme value theory to estimate the probability of excessively long service times under heavy-tailed distributions. If the system fails to meet tail-latency targets, there may be financial penalties or user dissatisfaction that indirectly increases cost. These high-percentile latency constraints can be included as penalty terms in the cost function. [53]

Another performance dimension includes the data freshness in streaming analytics pipelines. If the system processes streams in mini-batches, there is an inherent delay before insights become available. The length of the mini-batch affects both the latency of processed data and the computational efficiency [54]. Smaller batch sizes yield more timely insights but increase overhead. Larger batch sizes improve throughput but might violate freshness requirements. Autoscaling can balance these trade-offs by adapting the concurrency level according to the perceived importance of real-time insights [55]. Mathematically, one might define a freshness cost function $F(\Delta)$ that grows with the delay Δ between data arrival and the production of analytical results. Integrating $F(\Delta)$ into the overall cost function guides the system to maintain an optimal batch size and concurrency setting.

In practice, performance metrics cannot be treated independently [56]. Minimizing cost may conflict with ensuring extremely low latency or extremely high throughput. The multi-objective nature of these trade-offs calls for solutions that produce Pareto-optimal points along a cost-performance frontier. This frontier can be computed by solving repeated optimizations under different weighting schemes, or by employing multi-objective evolutionary algorithms that search for solutions that balance the cost and performance criteria [57]. The next section builds upon these metrics and the architectural considerations to propose predictive cost models that synthesize the dynamic conditions of both incoming data rates and spot market fluctuations.

6. Predictive Cost Modeling

Predictive cost modeling aims to anticipate future workload patterns, spot market fluctuations, and system states in order to optimize resource allocation before the environment changes. Central to this endeavor is forecasting [58,59]. By examining historical data—such as past job arrival rates, seasonal patterns, or known cyclical events—an auto-scaling policy can proactively provision capacity just as demand is expected to rise. Forecasting spot prices is more challenging due to market volatility, but statistical and machine learning methods can extrapolate from prior bidding and revocation events.

A common technique is time-series modeling using autoregressive integrated moving average methods or more advanced neural architectures that capture nonlinear patterns [60]. Suppose $p(t)$ is the spot price at time t. A model might predict $\hat{p}(t+\Delta)$ based on historical observations $\{p(t), p(t-1), p(t-2), \dots\}$. By associating each predicted price with a confidence interval, the system can weigh the risk that the actual price surpasses a bid threshold. If the predicted distribution of $p(t+\Delta)$ suggests a high risk of eviction, the autoscaler can reduce reliance on spot instances during that interval. [61]

Similarly, a workload model might predict $\hat{\lambda}(t+\Delta)$, the arrival rate of tasks in a data processing pipeline. Combining these forecasts, one can predict the future cost of running a set of on-demand and spot instances:

$$\hat{K}(t,\Delta) = \int_{t}^{t+\Delta} \left[\alpha \cdot C_{\text{on\_demand}}(\tau) + \beta(\tau) \cdot C_{\text{spot}}(\tau)\right] d\tau,$$

where $\beta(\tau)$ is taken from the predicted spot price or cost function, and $C_{\text{on\_demand}}(\tau)$ and $C_{\text{spot}}(\tau)$ follow from predicted usage levels. The autoscaling policy might solve a discrete-time approximation of this integral, updating resource allocation decisions every interval of length Δ. The policy aims to minimize $\hat{K}(t,\Delta)$ while ensuring performance constraints are met for the predicted workload.

One approach to implementing this decision process is receding-horizon control [62]. At each time step t, the policy obtains forecasts of future workload and spot prices over a horizon $[t, t+H]$. It then solves an optimization problem to find the sequence of provisioning actions $u_t, u_{t+1}, \dots, u_{t+H-1}$ that minimizes total expected cost subject to constraints on performance. However, only the first action u_t is implemented. At the next time step, new forecasts are generated for $[t+1, t+H]$, and the procedure repeats. This provides a balance between long-term planning and responsiveness to forecasting errors. [63]

A system can also incorporate Bayesian updating if it seeks to adapt forecasting models online. The model maintains a posterior distribution over parameters or states, refining these distributions as new data arrives. This approach might yield better accuracy under non-stationary conditions, such as abrupt changes in spot prices or a sudden shift in workload patterns [64]. The Bayesian framework can provide not only point estimates for future demand and prices, but also confidence intervals that guide risk-averse or risk-neutral strategies.

Predictive cost modeling extends beyond compute resource decisions. It can inform the location of data storage if data egress fees or cross-region latencies are expected to change [65]. While such changes in network costs are typically less frequent, they may arise from certain promotions, revised billing policies, or global capacity constraints in specific regions. Furthermore, the model can incorporate elasticity constraints or warm-up times for certain instance types, capturing the fact that launching specialized hardware accelerators or large memory instances may require significant lead time.

The interplay of forecasting accuracy, system reactivity, and spot market volatility shapes the success of predictive cost modeling [66]. Under stable conditions with moderate variability, even simple threshold-based rules can approximate optimal resource allocations. Under highly volatile conditions, advanced predictive techniques that unify time-series analysis, queueing theory, and real-time optimization will outperform naive approaches. The system thus becomes an integrated control loop where data is continuously gathered, predictions are generated, optimization is performed, and scaling actions are executed in near real-time. [67,68]

7. Performance Analysis and Overheads

Analyzing the performance and overheads of a cost-optimized big data system requires scrutinizing multiple interconnected factors. These factors include queueing delays, partial failures due to spot revocations, overhead incurred by checkpointing or replication, forecasting inaccuracies, and the computational complexity of performing real-time scheduling decisions. Each of these elements can introduce latency or financial penalties that must be weighed against any cost savings realized. [69]

A formal approach to performance analysis can employ layered queueing models. In such a scheme, the system is decomposed into tiers: a load balancing tier, a resource allocation tier, and a data processing tier. Each tier can be modeled as a queueing node with distinct service processes [70]. Load balancing might route tasks based on real-time resource availability, resource allocation might represent the ramp-up or termination delay for instances, and data processing stands as the core stage that handles the actual computations on allocated machines. The total system response time is the sum of the queuing times and service times across these tiers.

When considering the overhead of using spot instances, a key parameter is the eviction rate [71]. Suppose $\epsilon(t)$ is the instantaneous eviction probability, which may depend on the spot market price and the chosen bid. The overhead from eviction includes lost work plus additional re-queuing time until a replacement instance is spun up. If checkpointing is employed, then the overhead is reduced but not eliminated [72]. A simplified expression for average overhead can be written as:

$$O_{\text{spot}} = \epsilon \cdot (W_{\text{lost}} + T_{\text{requeue}}) + (1 - \epsilon) \cdot T_{\text{checkpoint}},$$

where W_{lost} is the average amount of discarded work when an eviction occurs, T_{requeue} is the delay before the job resumes on a new instance, and $T_{\text{checkpoint}}$ is the checkpoint overhead that may be paid regardless of eviction. This overhead influences the effective throughput of the system and may degrade the timeliness of analytical results.

Another significant aspect is the computational overhead of real-time decisions [73]. If the autoscaler relies on complex optimization or machine learning inference at short intervals, the system must devote some fraction of its resources to these tasks. In heavily loaded systems, this overhead is offset by the cost savings that come from accurate forecasting and resource allocation. Nonetheless, one must ensure that the gains from sophisticated scheduling outweigh the computational expense incurred in running these algorithms [74]. This trade-off can be analyzed by constructing a model of control overhead:

$$O_{\text{control}} = n_{\text{decisions}} \times C_{\text{alg}},$$

where $n_{\text{decisions}}$ is the frequency of scaling decisions within a certain timeframe, and C_{alg} is the cost of running the optimization or forecasting algorithm at each decision point.

System reliability also plays a crucial role in performance analysis. In large clusters, the mean time between failures for on-demand instances is typically higher than that for spot instances [75]. Nonetheless, even on-demand nodes can fail due to hardware or network issues. Statistical reliability models might approximate the system's overall availability by factoring in both planned variability (spot revocations) and unplanned hardware failures. The system can be represented by a reliability block diagram or by a Markov chain capturing the transitions between states of partial and full functionality [76]. These models help quantify the overall probability of partial or complete service outage, potentially feeding back into cost models that penalize downtime.

Overall, performance analysis in this domain demands a multi-faceted approach that integrates queueing theory, reliability engineering, cost modeling, and control overhead estimations. The complexity arises from the heterogeneous mix of on-demand and spot instances, their time-varying costs, and the distributed nature of data pipelines [77,78]. The concluding section synthesizes these findings, highlighting the synergy between theoretical models, architectural design, and predictive strategies that enable cost-effective big data management in the cloud.

8. Conclusion

The pursuit of cost-effectiveness in big data management within cloud environments has catalyzed a blend of sophisticated theoretical modeling, system architecture innovations, and predictive scaling strategies. This paper has examined the core elements that enable organizations to handle massive, dynamic workloads at reduced costs, highlighting the trade-offs between guaranteed availability on on-demand instances and the volatile yet potentially far cheaper spot market [79]. Through the lens of stochastic optimization, queueing models, and continuous-time Markov chain analyses, it is shown how one can systematically balance performance objectives like latency, throughput, and reliability with budgetary constraints.

A key insight arises from recognizing that different classes of big data workloads exhibit diverse tolerance levels for interruptions and varying performance requirements. By carefully selecting which tasks to allocate to spot instances, employing checkpointing or replication, and dynamically adjusting bidding strategies, the probability of catastrophic losses can be minimized [80]. In addition, architectural elements such as container orchestration, distributed storage systems, and real-time monitoring collectively facilitate resilient designs that can exploit elasticity while mitigating risks.

Another fundamental contribution lies in predictive modeling, which allows autoscaling policies to forecast demand surges, spot price fluctuations, and system states. By applying advanced time-series and machine learning techniques, the system can shift from reactive strategies to proactive adjustments [81]. These predictive frameworks often work in tandem with model predictive control, in which the decision process is recast as an online optimization problem, balancing real-time data against the cost of acquiring and maintaining computational resources.

Performance analyses reveal the importance of overhead considerations. The complexity of real-time optimization must be justified by its net benefits in cost savings, while partial failures from spot revocations must be mitigated through checkpointing or diversity in instance selection [82]. The multi-objective nature of these decisions means that simple rule-based strategies may not suffice under highly variable conditions, prompting the incorporation of risk metrics, tail latency constraints, and multi-cloud or multi-regional approaches where beneficial.

The architecture discussed here will likely evolve as cloud providers introduce new pricing models, novel instance types, and refined mechanisms for handling transient capacity. Further research into advanced queueing models, improved forecasting under highly non-stationary conditions, and more integrated data management pipelines can continue to reduce costs while maintaining or enhancing service quality. This vision underscores the continued interplay between rigorous mathematical frameworks, robust systems engineering, and adaptive real-time decision-making, culminating in an agile platform well-suited for the rapidly expanding realm of big data in the cloud. [83]

References

1. Hinkson, I.V.; Davidsen, T.M.; Klemm, J.; Chandramouliswaran, I.; Kerlavage, A.R.; Kibbe, W.A. A Comprehensive Infrastructure for Big Data in Cancer Research: Accelerating Cancer Research and Precision Medicine. *Frontiers in cell and developmental biology* **2017**, *5*, 108–108. https://doi.org/10.3389/fcell.2017.00108;10.3389/fcell.2017.00083.
2. Doyen, S.; Dadario, N.B. 12 Plagues of AI in Healthcare: A Practical Guide to Current Issues With Using Machine Learning in a Medical Context. *Frontiers in digital health* **2022**, *4*, 765406–. https://doi.org/10.3389/fdgth.2022.765406.
3. Xiang, Z.; Du, Q.; Ma, Y.; Fan, W. Assessing Reliability of Social Media Data: Lessons from Mining TripAdvisor Hotel Reviews. *Information Technology & Tourism* **2017**, *18*, 43–59. https://doi.org/10.1007/s40558-017-0098-z.

4. Nguyen, T.L. AI Deep Learning with Convolutional Neural Networks on Google Cloud Platform. *Journal of Strategic Innovation and Sustainability* **2019**, *14*. https://doi.org/10.33423/jsis.v14i4.2169.
5. Chen, W.; Kirkby, L.; Kotzev, M.; Song, P.; Gilron, R.; Pepin, B. The Role of Large-Scale Data Infrastructure in Developing Next-Generation Deep Brain Stimulation Therapies. *Frontiers in human neuroscience* **2021**, *15*, 717401–. https://doi.org/10.3389/fnhum.2021.717401.
6. Costa, C.; Konstantinidis, A.; Charalampous, A.; Zeinalipour-Yazti, D.; Mokbel, M.F. Continuous decaying of telco big data with data postdiction. *GeoInformatica* **2019**, *23*, 533–557. https://doi.org/10.1007/s10707-019-00364-z.
7. Manavalan, M. Intersection of Artificial Intelligence, Machine Learning, and Internet of Things – An Economic Overview. *Global Disclosure of Economics and Business* **2020**, *9*, 119–128. https://doi.org/10.18034/gdeb.v9i2.584.
8. Wen, Z.; Lin, T.; Yang, R.; Ji, S.; Ranjan, R.; Romanovsky, A.; Lin, C.; Xu, J. GA-Par: Dependable Microservice Orchestration Framework for Geo-Distributed Clouds. *IEEE Transactions on Parallel and Distributed Systems* **2020**, *31*, 129–143. https://doi.org/10.1109/tpds.2019.2929389.
9. Khan, I.; Naqvi, S.K.; Alam, M.; Rizvi, S.N.A. An efficient framework for real-time tweet classification. *International Journal of Information Technology* **2017**, *9*, 215–221. https://doi.org/10.1007/s41870-017-0015-x.
10. Zhang, Y.; Niu, K.; Wu, W.; Li, K.; Zhou, Y. Speeding Up VM Startup by Cooperative VM Image Caching. *IEEE Transactions on Cloud Computing* **2021**, *9*, 360–371. https://doi.org/10.1109/tcc.2018.2791509.
11. Yin, L.; Zhang, Y.; Zhang, Z.; Peng, Y.; Zhao, P. ParaX: boosting deep learning for big data analytics on many-core CPUs. *Proceedings of the VLDB Endowment* **2021**, *14*, 864–877. https://doi.org/10.14778/3447689.3447692.
12. Li, M.; Tan, J.; Wang, Y.; Zhang, L.; Salapura, V. SparkBench: a spark benchmarking suite characterizing large-scale in-memory data analytics. *Cluster Computing* **2017**, *20*, 2575–2589. https://doi.org/10.1007/s10586-016-0723-1.
13. Jennings, D.G.; Nordo, A.H.; Vattikola, A.; Kjaer, J. Technology Considerations for Enabling eSource in Clinical Research: Industry Perspective. *Therapeutic innovation & regulatory science* **2020**, *54*, 1166–1174. https://doi.org/10.1007/s43441-020-00132-4.
14. Hendawi, A.M.; Gupta, J.; Liu, J.; Teredesai, A.; Ramakrishnan, N.; Shah, M.; El-Sappagh, S.; Kwak, K.S.; Ali, M. Benchmarking large-scale data management for Internet of Things. *The Journal of Supercomputing* **2019**, *75*, 8207–8230. https://doi.org/10.1007/s11227-019-02984-6.
15. Ahmed, A.; Heldenbrand, J.R.; Asmann, Y.W.; Fadlelmola, F.M.; Katz, D.S.; Kendig, K.I.; Kendzior, M.; Li, T.W.; Ren, Y.; Rodriguez, E.; et al. Managing genomic variant calling workflows with Swift/T. *PloS one* **2019**, *14*, e0211608–. https://doi.org/10.1371/journal.pone.0211608.
16. Kansara, M. Cloud Migration Strategies and Challenges in Highly Regulated and Data-Intensive Industries: A Technical Perspective. *International Journal of Applied Machine Learning and Computational Intelligence* **2021**, *11*, 78–121.
17. Ullah, A.; Li, J.; Shen, Y.; Hussain, A. A control theoretical view of cloud elasticity: taxonomy, survey and challenges. *Cluster Computing* **2018**, *21*, 1735–1764. https://doi.org/10.1007/s10586-018-2807-6.
18. Faria, R.; Triant, D.A.; Perdomo-Sabogal, A.; Overduin, B.; Bleidorn, C.; Santana, C.I.B.; Langenberger, D.; Dall'Olio, G.M.; Indrischek, H.; Aerts, J.; et al. Introducing evolutionary biologists to the analysis of big data: guidelines to organize extended bioinformatics training courses. *Evolution: Education and Outreach* **2018**, *11*, 1–10. https://doi.org/10.1186/s12052-018-0080-z.
19. Huang, W.; Wang, H.; Zhang, Y.; Zhang, S. A novel cluster computing technique based on signal clustering and analytic hierarchy model using hadoop. *Cluster Computing* **2017**, *22*, 13077–13084. https://doi.org/10.1007/s10586-017-1205-9.
20. Khan, F.A.; ur Rehman, M.; Khalid, A.; Ali, M.A.; Imran, M.; Nawaz, M.; ur Rahman, A. An Intelligent Data Service Framework for Heterogeneous Data Sources. *Journal of Grid Computing* **2018**, *17*, 577–589. https://doi.org/10.1007/s10723-018-9443-5.
21. Gad, R.; Pickartz, S.; Süß, T.; Nagel, L.; Lankes, S.; Monti, A.; Brinkmann, A. Zeroing memory deallocator to reduce checkpoint sizes in virtualized HPC environments. *The Journal of Supercomputing* **2018**, *74*, 6236–6257. https://doi.org/10.1007/s11227-018-2548-6.
22. Grace, C.; Ting, B.M.; Goi, S.W.; Ting, L.; Kong, C.; Goi, B.; Lee, K.; Lee, S.W.; Rahman, A. Robustness Security of Data Hiding for H.265/HEVC Video Streams. *International Journal of Recent Technology and Engineering* **2019**, *8*, 146–151. https://doi.org/10.35940/ijrte.c1026.1083s19.

23. Avula, R. Architectural Frameworks for Big Data Analytics in Patient-Centric Healthcare Systems: Opportunities, Challenges, and Limitations. *Emerging Trends in Machine Intelligence and Big Data* **2018**, *10*, 13–27.
24. Tien, J.M. The Sputnik of servgoods: Autonomous vehicles. *Journal of Systems Science and Systems Engineering* **2017**, *26*, 133–162. https://doi.org/10.1007/s11518-016-5325-1.
25. Hu, X.; Kesidis, G.; Heidarpour, B.; Dziong, Z. Media delivery competition with edge cloud, remote cloud and networking. *NETNOMICS: Economic Research and Electronic Networking* **2020**, *21*, 17–36. https://doi.org/10.1007/s11066-020-09139-3.
26. Shin, H.; Lee, K.; Kwon, H.Y. A comparative experimental study of distributed storage engines for big spatial data processing using GeoSpark. *The Journal of supercomputing* **2021**, *78*, 1–24. https://doi.org/10.1007/s11227-021-03946-7.
27. Cardoso, A.; Moreira, F.; Escudero, D.F. Information Technology Infrastructure Library and the migration to cloud computing. *Universal Access in the Information Society* **2017**, *17*, 503–515. https://doi.org/10.1007/s10209-017-0559-3.
28. Lee, I. Pricing schemes and profit-maximizing pricing for cloud services. *Journal of Revenue and Pricing Management* **2019**, *18*, 112–122. https://doi.org/10.1057/s41272-018-00179-x.
29. Sharma, A.; Forbus, K.D. Graph-based reasoning and reinforcement learning for improving Q/A performance in large knowledge-based systems. In Proceedings of the 2010 AAAI Fall Symposium Series, 2010.
30. Longa, M.E.; Tsourdos, A.; Inalhan, G. Human–Machine Network Through Bio-Inspired Decentralized Swarm Intelligence and Heterogeneous Teaming in SAR Operations. *Journal of Intelligent & Robotic Systems* **2022**, *105*. https://doi.org/10.1007/s10846-022-01690-5.
31. Choi, T.M.; Lambert, J.H. Advances in risk analysis with big data. *Risk analysis : an official publication of the Society for Risk Analysis* **2017**, *37*, 1435–1442. https://doi.org/10.1111/risa.12859.
32. Kansara, M. A Comparative Analysis of Security Algorithms and Mechanisms for Protecting Data, Applications, and Services During Cloud Migration. *International Journal of Information and Cybersecurity* **2022**, *6*, 164–197.
33. Kuenzi, B.M.; Ideker, T. A census of pathway maps in cancer systems biology. *Nature reviews. Cancer* **2020**, *20*, 233–246. https://doi.org/10.1038/s41568-020-0240-7.
34. Tucci, G.; Corongiu, M.; Flamigni, F.; Comparini, A.; Panighini, F.; Parisi, E.I.; Arcidiaco, L. The validation process of a 3D multisource/multiresolution model for railway infrastructures. *Applied Geomatics* **2019**, *12*, 69–84. https://doi.org/10.1007/s12518-019-00286-3.
35. Cho, E.; Jacobs, J.M.; Jia, X.; Kraatz, S. Identifying Subsurface Drainage using Satellite Big Data and Machine Learning via Google Earth Engine. *Water Resources Research* **2019**, *55*, 8028–8045. https://doi.org/10.1029/2019wr024892.
36. Alnafessah, A.; Casale, G. Artificial neural networks based techniques for anomaly detection in Apache Spark. *Cluster Computing* **2019**, *23*, 1345–1360. https://doi.org/10.1007/s10586-019-02998-y.
37. Matesanz, P.; Graen, T.; Fiege, A.; Nolting, M.; Nejdl, W. Demand-Driven Data Acquisition for Large Scale Fleets. *Sensors (Basel, Switzerland)* **2021**, *21*, 7190–. https://doi.org/10.3390/s21217190.
38. Shekhar, S. A Critical Examination of Cross-Industry Project Management Innovations and Their Transferability for Improving IT Project Deliverables. *Quarterly Journal of Emerging Technologies and Innovations* **2016**, *1*, 1–18.
39. Lee, S.; Jo, W.; Eo, S.; Shon, T. ExtSFR: scalable file recovery framework based on an Ext file system. *Multimedia Tools and Applications* **2019**, *79*, 16093–16111. https://doi.org/10.1007/s11042-019-7199-y.
40. Zhao, L.; Batta, I.; Matloff, W.; O'Driscoll, C.; Hobel, S.M.; Toga, A.W. Neuroimaging PheWAS (Phenome-Wide Association Study): A Free Cloud-Computing Platform for Big-Data, Brain-Wide Imaging Association Studies. *Neuroinformatics* **2020**, *19*, 285–303. https://doi.org/10.1007/s12021-020-09486-4.
41. Behrens, R.; Foutz, N.Z.; Franklin, M.; Funk, J.; Gutierrez-Navratil, F.; Hofmann, J.; Leibfried, U. Leveraging analytics to produce compelling and profitable film content. *Journal of Cultural Economics* **2020**, *45*, 171–211. https://doi.org/10.1007/s10824-019-09372-1.
42. Yang, H.; Kumara, S.R.T.; Bukkapatnam, S.T.S.; Tsung, F. The Internet of Things for Smart Manufacturing: A Review. *IISE Transactions* **2019**, *51*, 1190–1216. https://doi.org/10.1080/24725854.2018.1555383.
43. Ye, F.; Qian, Y.; Hu, R.Q. Background of the Smart Grid, 2018. https://doi.org/10.1002/9781119240136.ch1.

44. Agyekum, K.O.B.O.; Xia, Q.; Sifah, E.B.; Gao, J.; Xia, H.; Du, X.; Guizani, M. A Secured Proxy-Based Data Sharing Module in IoT Environments Using Blockchain. *Sensors (Basel, Switzerland)* **2019**, *19*, 1235–. https://doi.org/10.3390/s19051235.
45. Kwon, D.; Park, S.; Ryu, J.T. A Study on Big Data Thinking of the Internet of Things-Based Smart-Connected Car in Conjunction with Controller Area Network Bus and 4G-Long Term Evolution. *Symmetry* **2017**, *9*, 152–. https://doi.org/10.3390/sym9080152.
46. Diallo, S.Y.; Durak, U.; Mustafee, N.; Mittal, S. Special issue on modeling and simulation in the era of big data and cloud computing: theory, framework and tools:. *SIMULATION* **2017**, *93*, 271–272. https://doi.org/10.1177/0037549717696446.
47. Kirpich, A.; Ibarra, M.; Moskalenko, O.; Fear, J.M.; Gerken, J.; Mi, X.; Ashrafi, A.; Morse, A.M.; McIntyre, L.M. SECIMTools: a suite of metabolomics data analysis tools. *BMC bioinformatics* **2018**, *19*, 151–151. https://doi.org/10.1186/s12859-018-2134-1.
48. Bulkan, U.; Dagiuklas, T.; Iqbal, M. On the modelling of CDNaaS deployment. *Multimedia Tools and Applications* **2018**, *78*, 6805–6825. https://doi.org/10.1007/s11042-018-6441-3.
49. Jia, X.; He, D.; Kumar, N.; Choo, K.K.R. Authenticated key agreement scheme for fog-driven IoT healthcare system. *Wireless Networks* **2018**, *25*, 4737–4750. https://doi.org/10.1007/s11276-018-1759-3.
50. Qian, K.; Zhang, L.; Li, K.; Liu, J. Editorial: Machine Learning for Non/Less-Invasive Methods in Health Informatics. *Frontiers in digital health* **2021**, *3*, 763109–763109. https://doi.org/10.3389/fdgth.2021.763109.
51. Peisert, S.; Dart, E.; Barnett, W.K.; Balas, E.; Cuff, J.; Grossman, R.L.; Berman, A.E.; Shankar, A.; Tierney, B. The medical science DMZ: a network design pattern for data-intensive medical science. *Journal of the American Medical Informatics Association : JAMIA* **2017**, *25*, 267–274. https://doi.org/10.1093/jamia/ocx104.
52. Liu, J.; Zhou, S.; Cheng, E.; Chen, G.; Li, M. Reliability Evaluation of Bicube-Based Multiprocessor System under the g-Good-Neighbor Restriction. *Parallel Processing Letters* **2021**, *31*. https://doi.org/10.1142/s0129626421500183.
53. Hoang, T.; Fu, Y.; Mechitov, K.; Sánchez, F.G.; Kim, J.; Zhang, D.; Spencer, B.F. Autonomous end-to-end wireless monitoring system for railroad bridges. *Advances in Bridge Engineering* **2020**, *1*, 1–27. https://doi.org/10.1186/s43251-020-00014-7.
54. Zhang, Y.; He, D.; Choo, K.K.R. BaDS: Blockchain-Based Architecture for Data Sharing with ABS and CP-ABE in IoT. *Wireless Communications and Mobile Computing* **2018**, *2018*, 1–9. https://doi.org/10.1155/2018/2783658.
55. Prosperi, M.; Min, J.; Bian, J.G.; Modave, F. Big data hurdles in precision medicine and precision public health. *BMC medical informatics and decision making* **2018**, *18*, 139–139. https://doi.org/10.1186/s12911-018-0719-2.
56. Li, R.; Ruan, S.; Bao, J.; Li, Y.; Wu, Y.; Hong, L.; Zheng, Y. Efficient Path Query Processing Over Massive Trajectories on the Cloud. *IEEE Transactions on Big Data* **2020**, *6*, 66–79. https://doi.org/10.1109/tbdata.2018.2868936.
57. Mabry, P.L.; Yan, X.; Pentchev, V.; Rennes, R.V.; McGavin, S.H.; Wittenberg, J. CADRE: A Collaborative, Cloud-Based Solution for Big Bibliographic Data Research in Academic Libraries. *Frontiers in big data* **2020**, *3*, 556282–556282. https://doi.org/10.3389/fdata.2020.556282.
58. Cinemre, I.; Mahmoodi, T. Learning-Based Multi Attribute Network Selection in Heterogeneous Wireless Access. *Wireless Personal Communications* **2022**, *125*, 351–366. https://doi.org/10.1007/s11277-022-09553-w.
59. Avula, R. Optimizing Data Quality in Electronic Medical Records: Addressing Fragmentation, Inconsistencies, and Data Integrity Issues in Healthcare. *Journal of Big-Data Analytics and Cloud Computing* **2019**, *4*, 1–25.
60. Ehwerhemuepha, L.; Gasperino, G.; Bischoff, N.; Taraman, S.; Chang, A.C.; Feaster, W. HealtheDataLab – a cloud computing solution for data science and advanced analytics in healthcare with application to predicting multi-center pediatric readmissions. *BMC medical informatics and decision making* **2020**, *20*, 1–12. https://doi.org/10.1186/s12911-020-01153-7.
61. Martinelli, A.; Mina, A.; Moggi, M. The enabling technologies of industry 4.0: examining the seeds of the fourth industrial revolution. *Industrial and Corporate Change* **2021**, *30*, 161–188. https://doi.org/10.1093/icc/dtaa060.
62. Deka, R.K.; Bhattacharyya, D.K.; Kalita, J. DDoS Attacks: Tools, Mitigation Approaches, and Probable Impact on Private Cloud Environment, 2021. https://doi.org/10.1002/9781119740780.ch13.

63. Saha, O.; Dasgupta, P. A Comprehensive Survey of Recent Trends in Cloud Robotics Architectures and Applications. *Robotics* **2018**, *7*, 47–. https://doi.org/10.3390/robotics7030047.
64. Hu, Y.; Gao, S.; Newsam, S.; Lunga, D. GeoAI 2018 workshop report the 2nd ACM SIGSPATIAL international workshop on GeoAI: AI for geographic knowledge discovery seattle, WA, USA - November 6, 2018. *SIGSPATIAL Special* **2019**, *10*, 16–16. https://doi.org/10.1145/3307599.3307609.
65. Lee, G.Y.; Kim, M.; Quan, Y.J.; Kim, M.S.; Kim, T.J.Y.; Yoon, H.S.; Min, S.; Kim, D.H.; Mun, J.W.; Oh, J.W.; et al. Machine health management in smart factory: A review. *Journal of Mechanical Science and Technology* **2018**, *32*, 987–1009. https://doi.org/10.1007/s12206-018-0201-1.
66. Leevy, J.L.; Khoshgoftaar, T.M. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data* **2020**, *7*, 1–19. https://doi.org/10.1186/s40537-020-00382-x.
67. Dang, D.; Chen, C.; Li, H.; Yan, R.; Guo, Z.; Wang, X. Deep knowledge-aware framework for web service recommendation. *The Journal of Supercomputing* **2021**, *77*, 14280–14304. https://doi.org/10.1007/s11227-021-03832-2.
68. Kansara, M. A Structured Lifecycle Approach to Large-Scale Cloud Database Migration: Challenges and Strategies for an Optimal Transition. *Applied Research in Artificial Intelligence and Cloud Computing* **2022**, *5*, 237–261.
69. Fan, Y.; Jin, Z.; Shen, G.; Hu, D.; Shi, L.; Yuan, X. Three-stage Stackelberg game based edge computing resource management for mobile blockchain. *Peer-to-Peer Networking and Applications* **2021**, *14*, 1431–1445. https://doi.org/10.1007/s12083-020-01032-y.
70. Lee, I. An optimization approach to capacity evaluation and investment decision of hybrid cloud: a corporate customer's perspective. *Journal of Cloud Computing* **2019**, *8*, 1–13. https://doi.org/10.1186/s13677-019-0140-0.
71. Peña, A.; Bonet, I.; Lochmuller, C.; Tabares, M.S.; Piedrahita, C.C.; Sanchez, C.C.; Marín, L.M.G.; Gongora, M.; Chiclana, F. A fuzzy ELECTRE structure methodology to assess big data maturity in healthcare SMEs. *Soft Computing* **2018**, *23*, 10537–10550. https://doi.org/10.1007/s00500-018-3625-8.
72. Li, F.; Wang, Y.; Ju, H.; Yu, X.; Wang, Z.; Zhou, H. LacminCC: lightweight anonymous communication model in cloud computing. *EURASIP Journal on Wireless Communications and Networking* **2021**, *2021*, 1–16. https://doi.org/10.1186/s13638-021-01953-z.
73. Saabith, A.L.S.; Sundararajan, E.A.; Bakar, A.A. A Parallel Apriori-Transaction Reduction Algorithm Using Hadoop-Mapreduce in Cloud. *Asian Journal of Research in Computer Science* **2018**, pp. 1–24. https://doi.org/10.9734/ajrcos/2018/v1i124719.
74. Salehi, M.A.; Caldwell, T.W.; Fernandez, A.; Mickiewicz, E.; Rozier, E.W.D.; Zonouz, S.; Redberg, D.A. RESeED: A secure regular-expression search tool for storage clouds. *Software: Practice and Experience* **2017**, *47*, 1221–1241. https://doi.org/10.1002/spe.2473.
75. Eller, R.J.; Janga, S.C.; Walsh, S. Odyssey: a semi-automated pipeline for phasing, imputation, and analysis of genome-wide genetic data. *BMC bioinformatics* **2019**, *20*, 1–8. https://doi.org/10.1186/s12859-019-2964-5.
76. Zhang, X.; Wang, Y.; Lyu, H.; Zhang, Y.; Liu, Y.; Luo, J. The Influence of COVID-19 on the Well-Being of People: Big Data Methods for Capturing the Well-Being of Working Adults and Protective Factors Nationwide. *Frontiers in psychology* **2021**, *12*, 681091–681091. https://doi.org/10.3389/fpsyg.2021.681091.
77. Kuznetsov, V.; Giommi, L.; Bonacorsi, D. MLaaS4HEP: Machine Learning as a Service for HEP. *Computing and Software for Big Science* **2021**, *5*, 1–16. https://doi.org/10.1007/s41781-021-00061-3.
78. Avula, R. Overcoming Data Silos in Healthcare with Strategies for Enhancing Integration and Interoperability to Improve Clinical and Operational Efficiency. *Journal of Advanced Analytics in Healthcare Management* **2020**, *4*, 26–44.
79. Liu, H.; Li, G.; Lukman, J.F.; Li, J.; Lu, S.; Gunawi, H.S.; Tian, C. DCatch. *ACM SIGARCH Computer Architecture News* **2017**, *45*, 677–691. https://doi.org/10.1145/3093337.3037735.
80. Gravvanis, G.A.; Morrison, J.P.; Marinescu, D.C.; Filelis-Papadopoulos, C.K. Special section: towards high performance computing in the cloud. *The Journal of Supercomputing* **2018**, *74*, 527–529. https://doi.org/10.1007/s11227-018-2241-9.
81. Pham, Q.V.; Ruby, R.; Fang, F.; Nguyen, D.C.; Yang, Z.; Le, M.; Ding, Z.; Hwang, W.J. Aerial Computing: A New Computing Paradigm, Applications, and Challenges. *IEEE Internet of Things Journal* **2022**, *9*, 8339–8363. https://doi.org/10.1109/jiot.2022.3160691.

82. Yang, C.T.; Chen, S.T.; Liu, J.C.; Su, Y.W.; Puthal, D.; Ranjan, R. A predictive load balancing technique for software defined networked cloud services. *Computing* **2018**, *101*, 211–235. https://doi.org/10.1007/s00607-018-0665-y.
83. Sun, X.; Ansari, N. Adaptive Avatar Handoff in the Cloudlet Network. *IEEE Transactions on Cloud Computing* **2019**, *7*, 664–676. https://doi.org/10.1109/tcc.2017.2701794.