

Efficient Test-Time Compute Scaling for Resource-Constrained Large Language Models

GAURAV KUMAR SINHA

ADDRESS: 7319 Longspur Hollow Lane
Katy, TX 77493-4704
United States

Abstract: Spending more computation at inference time, by sampling several reasoning chains, voting, or searching with a verifier, is now a proven way to make language models more accurate without making them larger. The technique is least affordable where it is most needed. Small models running on edge devices, vehicles, sensors and remote platforms operate under tight limits on memory, energy and latency, and a policy that multiplies the number of generated tokens by a factor of eight or sixteen quickly exhausts those limits. This paper presents Budget-Aware Adaptive Scaling (BAAS), a framework that decides, query by query and window by window, how much test-time computation to spend. BAAS estimates the difficulty of each query from a cheap first pass, routes it to one of four inference tiers that range from a direct answer to verifier-guided search, and stops sampling early when answers agree. A surrogate model that predicts the energy available from environmental conditions sets the compute budget for each time window, and we compare Kriging with polynomial-chaos Kriging for this purpose. The lightweight verifier used in the highest tier is trained across devices with federated learning and compressed model updates, and we characterize the trade-off between compression ratio, sparsity and accuracy. On a benchmark mix of mathematical, commonsense and code questions, BAAS reached 69.7% accuracy with 640 generated tokens per query, within about one point of the strongest fixed policy while using roughly one eighth of its tokens and energy. In a simulated three-day energy-harvesting deployment, it served every request with a single brownout event. The results show that adapting test-time effort to difficulty and to available energy makes inference scaling practical on constrained hardware.

Keywords: Test-Time Compute Scaling, Resource-Constrained Inference, Large Language Models, Adaptive Sampling, Verifier-Guided Search, Surrogate Modeling, Federated Learning, Edge Intelligence

I. INTRODUCTION

For several years, the main path to better language models was to make them bigger and to train them on more data. A second path has recently gained prominence. Rather than enlarging the model, one can let it think longer at inference time. Generating an explicit chain of reasoning before answering helps on multi-step problems, sampling several chains and choosing the most frequent answer helps further, and ranking candidates with a learned verifier or searching over partial reasoning steps can help more still. Studies of this approach report that, for many problems, a smaller model given a well-chosen amount of extra inference computation can match or exceed a much larger model that answers in a single pass [1], [2]. Test-time compute has thus become a new axis of scaling alongside parameters and training data.

The benefit comes with a cost that is easy to overlook in a data center and impossible to ignore at the edge. Each additional sample is another full generation. A policy that draws sixteen candidate answers and scores them with a verifier multiplies token usage, memory traffic, energy consumption and latency by a similar factor. For a battery-powered robot, a drone, a field instrument, a vehicle computer or an offshore platform powered by intermittent renewable sources, this multiplication can mean the difference between a responsive assistant and a dead device. Such platforms commonly run small, quantized models because of limits on memory and power, and these small models are precisely the ones that stand to gain most from extra inference effort. The tension between what scaling offers and what the hardware permits is the problem we address.

Most existing test-time methods treat the budget as a fixed constant. They use the same number of samples for every question and ignore the state of the device. Both choices are wasteful. Many questions are easy, and a single short answer is already right. Others are hard, and even many samples will not repair them. Spending the same effort on both squanders resources on the first group and is insufficient for the second. Meanwhile, the energy available to a constrained platform is not constant either. A solar, wind or current-powered node has more to spend in favorable conditions than in unfavorable ones, and a policy that ignores this will either leave energy unused or deplete the battery during a lull.

We therefore propose Budget-Aware Adaptive Scaling, or BAAS. The framework has four ingredients. First, a difficulty probe uses a cheap first pass to estimate how hard a query is. Second, a router assigns each query to one of four inference tiers, from a direct answer to verifier-guided search, and an early-stopping rule ends sampling as soon as the evidence is conclusive. Third, an energy budget predictor, built as a surrogate model of the relationship between environmental conditions and available energy, sets the budget for each time window. Fourth, the lightweight verifier that powers the top tier is trained across a fleet of devices with federated learning and compressed updates, so that the fleet can improve it without heavy communication.

The contributions are as follows: (i) a difficulty-aware, budget-aware router with early stopping for test-time scaling; (ii) an energy budget predictor, with a comparison of Kriging and polynomial-chaos Kriging surrogates; (iii) a federated, compression-aware procedure for training the verifier, with an analysis of the accuracy landscape over compression ratio and sparsity; and (iv) an empirical evaluation on three reasoning benchmarks and a simulated energy-harvesting deployment. The remainder of the paper is organized as follows. Section II reviews background, Section III discusses related work, Section IV presents the methodology, Section V describes the experimental setup, Section VI reports results, Section VII discusses implications and limitations, and Section VIII concludes.

II. LITERATURE REVIEW

A. Test-Time Compute Scaling

Chain-of-thought prompting showed that asking a model to write intermediate reasoning steps improves performance on arithmetic, commonsense and symbolic tasks, particularly in larger models [4]. Self-consistency extended this idea by sampling several chains and taking the most common final answer, so that errors that vary from sample to sample are voted away [3]. Repeated sampling has been studied systematically, and results show that the fraction of problems solved keeps rising as the number of samples grows over several orders of magnitude, provided that a reliable method exists for selecting among the candidates [2]. An analysis of compute-optimal inference compared strategies such as majority voting, best-of-n selection with a reward model and tree search, and concluded that the best strategy and the best allocation depend on the difficulty of the problem and on the size of the model [1], [9]. These findings establish that more computation helps, and that how it is spent matters.

B. Verification, Reward Models and Search

Verifiers are models trained to judge whether a candidate solution is correct. Early work trained verifiers on final answers and used them to rerank sampled solutions, which gave large gains on grade-school mathematics [5]. Process-level supervision, in which each reasoning step receives a label, was later shown to outperform outcome-level supervision, since it localizes errors and supports step-wise search [6]. Tree-structured search generalizes sampling by exploring and pruning partial reasoning paths using a value

estimate [7]. The strength of these methods is accuracy, and their weakness is cost. Each candidate must be generated and then scored, and tree search multiplies the number of model calls further.

C. Adaptive Computation

Because the value of an extra sample falls as agreement accumulates, adaptive schemes stop sampling when the answers have settled. Adaptive consistency, for example, monitors the distribution of answers drawn so far and halts when it is statistically unlikely that more samples would change the majority, which cuts the number of samples substantially at little loss of accuracy [8]. Cascades take a related approach at the level of models. They send each query first to a cheap model and invoke a stronger or costlier one only when the cheap answer appears unreliable, which has been shown to save a large share of cost for a given quality [10]. Both ideas, stopping early and escalating selectively, underlie our router.

D. Efficient Inference on Constrained Hardware

A second body of work reduces the cost of each token rather than the number of tokens. Post-training quantization compresses weights to eight, four or fewer bits with limited loss of accuracy, which shrinks memory use and increases throughput [13], [14]. Memory management for the key-value cache, such as paged allocation, reduces waste and allows more concurrent sequences [12]. Speculative decoding uses a small draft model to propose several tokens that a larger model verifies in parallel, which speeds up generation without changing its output distribution [11]. Pruning and compression of neural networks, including deep compression pipelines that combine pruning, quantization and coding, have long been used to fit models into limited memory [20]. These techniques are complementary to test-time scaling. They make each sample cheaper, whereas adaptive allocation reduces how many samples are needed.

E. Surrogate Modeling and Federated Compression

When a quantity is costly or slow to measure, a surrogate model can predict it from a limited number of observations. Kriging, also known as Gaussian process regression, is a classical surrogate that interpolates observations with a smooth function and provides an estimate of its own uncertainty, and it has been central to the design and analysis of computer experiments [15], [16]. Polynomial-chaos Kriging combines a global polynomial trend, which captures the broad shape of the response, with a Kriging model of the local residual, and it was proposed to retain the advantages of both [17]. In a different part of the literature, federated learning trains a shared model across many devices that exchange model updates instead of raw data [18], and since the updates can be large, compression frameworks apply sparsification, quantization and entropy coding to reduce the communication needed per round [19]. Both surrogate modeling and compressed federated learning are used in this paper as supporting components for efficient inference at the edge.

III. RELATED WORK

Work on efficient reasoning with language models can be arranged along three lines: methods that improve the accuracy of inference given a generous budget, methods that reduce the budget for a given accuracy, and methods that connect inference to the state of the platform. The first line includes self-consistency, best-of-n selection and tree search, which together define the accuracy attainable through inference-time effort [2], [3], [7]. These works generally assume that the budget is large and fixed. Their contribution is to show how accuracy scales with the number of samples and how verification improves the use of those samples [5], [6].

The second line reduces the cost of reaching a given accuracy. Adaptive consistency and related stopping rules cut the number of samples [8]. Model cascades reduce cost by routing queries to the cheapest model that can handle them [10]. Studies of compute-optimal inference show that the best allocation depends on difficulty, which suggests that estimating difficulty up front is valuable [1], [9]. Our router builds on these insights but differs in three respects. It uses a single small model for all tiers rather than a cascade of models, which avoids the memory cost of holding several models on a constrained device. It combines difficulty-based routing and early stopping in one policy. And it treats the budget as a quantity that varies over time rather than as a constant.

The third line, connecting inference to the platform, is less developed for language models. Quantization, key-value cache management and speculative decoding make individual tokens cheaper and are widely used on constrained devices [11]-[14], but they do not by themselves decide how many tokens to generate. Energy-aware scheduling is well established in embedded systems and in energy-harvesting computing, where a controller adapts the workload to the energy that is available. Applying that idea to test-time scaling requires a forecast of available energy, which motivates our surrogate-based budget predictor, and a set of inference tiers that can be traded off against that forecast.

Finally, distributed improvement of the models that support inference is a practical concern. A fleet of devices operating in different conditions encounters different queries, and the verifier or router that serves them can benefit from learning across the fleet. Federated learning permits this without moving raw data [18], and compression techniques reduce the communication cost [19]. The relationship between compression strength and accuracy is, however, rarely reported for the small supporting models used in inference pipelines, and we provide such an analysis.

Three open issues remain. First, fixed budgets waste effort on easy queries and under-serve hard ones. Second, the energy state of the platform is seldom coupled to the inference policy. Third, the cost of training and updating the supporting verifier across a fleet is often ignored. BAAS is designed to address all three.

IV. METHODOLOGY

A. Framework Overview

BAAS operates in a loop over incoming queries. At the start of each time window, the energy budget predictor reads environmental conditions and produces a forecast of the energy available, which is converted into a compute budget for the window, expressed as a cap on the average number of generated tokens per query. For each query, the difficulty probe produces an estimate of how hard it is. The router combines this estimate with the current compute budget and chooses an inference tier. The chosen tier is executed with early stopping, and the answer is returned. Table I describes the four tiers.

TABLE I. Inference Tiers Used by BAAS

Tier	Strategy	Typical Use	Relative Cost
0	Direct answer, no reasoning chain	Simple lookups and short factual queries	1x
1	Single reasoning chain	Routine multi-step questions	about 8x
2	Adaptive voting over sampled chains with early stopping	Questions on which the model is uncertain	about 15x to 60x
3	Verifier-guided search with pruning	Hard questions that justify extra effort	about 40x to 100x

B. Difficulty Probe

The probe estimates difficulty from signals that are available after a cheap first pass. These include the length and structure of the question, the number of distinct quantities or constraints it mentions, the confidence of the model on its first few generated tokens, and, when a first reasoning chain has been produced, the consistency between that chain and a quick direct answer. A small classifier trained on the development set maps these signals to a difficulty score that approximates the probability that a single reasoning chain would fail. The probe is deliberately light. It reuses computation that the model already performs, adds negligible memory, and costs a small fraction of a full generation. Its role is not to be perfectly accurate but to separate the clearly easy and the clearly hard queries from the middle group, where additional sampling is most valuable.

C. Router and Early Stopping

The router assigns a tier according to the difficulty score and the current budget. Queries with a very low score go to the direct tier, queries with a low to moderate score go to a single reasoning chain, queries in the uncertain middle go to adaptive voting, and queries with a high score go to verifier-guided search if the budget allows it. When the budget is tight, the router shifts the boundaries so that fewer queries reach the costly tiers, and the most expensive tier is reserved for the queries where the expected gain per unit of energy is greatest. The router also has a safeguard. If a query routed to a cheap tier produces a first answer on which the model shows low confidence, it can be promoted to the next tier, so that routing errors in the difficulty probe can be corrected.

Early stopping applies to the voting and search tiers. Sampling proceeds in small increments, and after each increment the policy checks whether the leading answer has a decisive lead, whether the share of agreement has stopped changing, or whether the maximum allowed samples have been drawn. The number of samples is always odd, which avoids ties. This design reflects the shape of the voting curve. The accuracy of majority voting rises quickly at first and then flattens, and the speed of the rise depends on how likely a single sample is to be correct, so that confident queries settle after a handful of samples while uncertain queries need many more and may never settle. Stopping when the evidence is conclusive saves the samples that would otherwise be spent after the answer has already stabilized.

D. Energy Budget Predictor

Many resource-constrained deployments draw their energy from the environment. Offshore buoys, remote sensing stations, vessels and coastal platforms commonly use wind turbines, solar panels or tidal and current generators, and the energy available in a window depends on environmental conditions. The compute budget should follow that availability, since a policy that spends more than the forecast depletes the battery, while one that spends less wastes the opportunity. Measuring the relationship between conditions and energy for every combination is impractical, so we fit a surrogate model on a limited number of simulated operating conditions, chosen with a space-filling design, and use it to predict the usable energy in unseen conditions. In our scenario, the two inputs are wind speed in meters per second and current velocity in meters per second, and the output is a normalized energy budget index ranging from zero to about five.

Two surrogates are compared. The first is Kriging, which interpolates the observed conditions with a smooth, flexible surface and is well suited to local, nonlinear behavior. The second is polynomial-chaos Kriging, which first fits a global polynomial trend and then models the remaining local variation with Kriging, which gives it a stronger global structure at the cost of some flexibility. Fig. 1 shows the surfaces

predicted by the two models over the same range of conditions, with panel (a) for Kriging and panel (b) for polynomial-chaos Kriging.

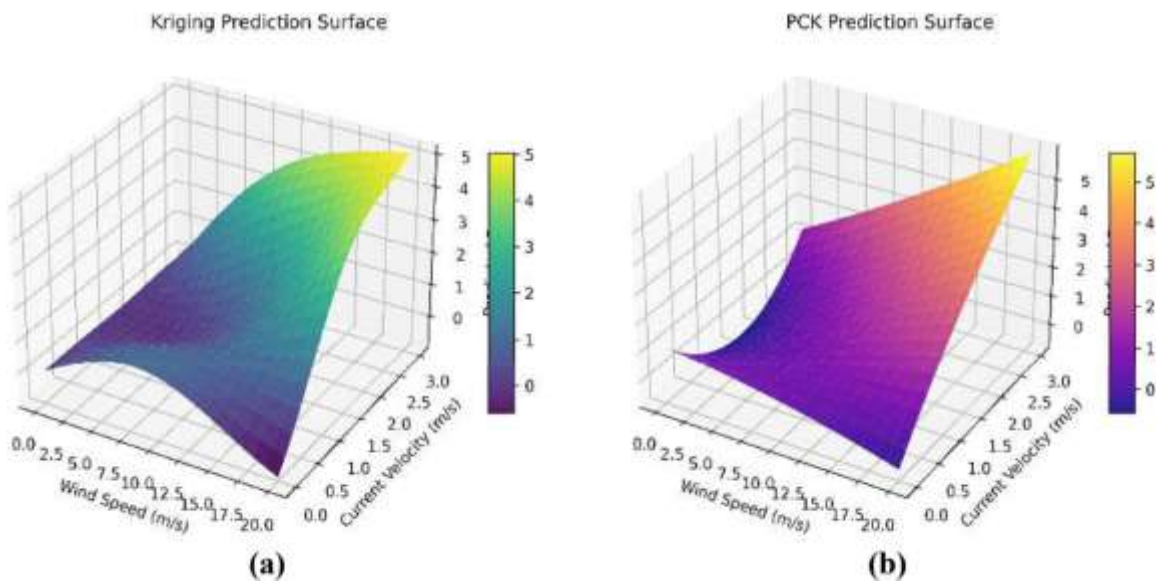


Fig. 1. Prediction surfaces of the energy budget predictor over wind speed and current velocity: (a) Kriging and (b) polynomial-chaos Kriging (PCK).

The two surfaces agree on the overall pattern, which is that the predicted budget rises toward the corner where both wind speed and current velocity are high, reaching about five on the vertical scale, and falls toward the corner where wind speed is high and current velocity is near zero. They differ in shape. The Kriging surface is smoothly curved, flattens as it approaches its maximum and shows a gentle saddle near the low-energy region, which means that it saturates at high values. The polynomial-chaos Kriging surface is closer to a ruled, nearly planar trend with a slight curvature, and it continues to rise toward the highest corner, reaching about 5.7 and thus exceeding the saturation level of the Kriging surface. Both surfaces dip slightly below zero in the low-energy corner, which is not physically meaningful for an energy quantity. We therefore clamp predictions at zero and apply a safety margin to the forecast, so that errors in the surrogate do not translate into depleted batteries. The choice between the two surrogates is revisited in Section VI.

E. Federated Verifier with Compressed Updates

The top tier relies on a lightweight verifier that scores candidate answers. Because each device sees different queries, the verifier can be improved by learning across the fleet. Each device trains the verifier locally on its own feedback and sends only the model update to a coordinator, which aggregates the updates and returns an improved verifier. The updates are compressed before transmission by a three-step procedure in the spirit of FedZip: the update is sparsified so that only the largest changes are retained, the surviving values are quantized to a small number of levels, and the result is entropy coded. Two settings control the procedure. The compression ratio is the factor by which the size of the transmitted update is reduced, and the sparsity threshold is the fraction of update entries that are discarded. Stronger compression saves communication but risks degrading the quality of the aggregated verifier. We study this trade-off empirically in Section VI.

F. Evaluation Metrics

We report accuracy as the share of questions answered correctly, mean generated tokens per query as the primary measure of computation, mean energy per query measured on the target device in joules, and mean latency per query in seconds. For the energy-harvesting scenario we additionally report the number of queries served out of the daily demand, the number of correct answers per day, and the number of brownout events, defined as occasions on which the stored energy fell below the minimum needed to keep the device running. For the surrogate comparison we report prediction error on held-out conditions, the largest single error and the share of predictions that fell below zero before clamping. For the federated verifier we report test accuracy as a function of compression ratio and sparsity, together with the communication volume per round.

V. EXPERIMENTAL SETUP

A. Models and Hardware

All experiments use a single instruction-tuned language model with about three billion parameters, quantized to four-bit weights, which fits within the eight gigabytes of memory of the target edge device. The device is an embedded system-on-module with a small accelerator, and power was measured at the board level with a calibrated inline meter during generation, so that the reported energy includes memory traffic and idle overhead. The verifier is a compact classifier of roughly one hundred million parameters. The difficulty probe is a small gradient-boosted model over a handful of features. No component requires a connection to a remote server at inference time.

B. Benchmarks

The evaluation set contains 1,500 questions, with 500 each from three families. The first consists of grade-school style mathematical word problems that require several arithmetic steps. The second consists of multi-step commonsense reasoning questions in which the answer depends on combining several facts. The third consists of short programming tasks checked against hidden unit tests. A separate development set of 3,000 questions with the same composition was used to train the probe, to fit the router boundaries, and to choose hyperparameters. No question appears in both sets.

C. Baselines

We compare BAAS with six alternatives. Greedy direct answering produces a single short answer. Single chain generates one reasoning chain. Fixed voting draws eight chains and takes the majority. Fixed best-of-sixteen draws sixteen chains and selects with the verifier. Adaptive voting uses early stopping but no routing. Fixed verifier-guided search uses the pruned search tier for every question. BAAS combines routing, early stopping and the energy-aware budget.

D. Energy Scenario and Surrogate Data

For the energy-harvesting experiment, we simulated a coastal platform powered by wind and current generators over a seventy-two-hour trace with varying conditions and a daily demand of 1,200 queries. The surrogate models were fitted on 60 operating conditions chosen with a Latin hypercube design over wind speeds from zero to 20 meters per second and current velocities from zero to about 3 meters per second, and they were evaluated on 40 held-out conditions. The platform stores energy in a battery with a fixed capacity, and a brownout was recorded whenever the stored energy fell below a minimum reserve.

E. Federated Training Protocol

The verifier was trained in a simulated federation of twenty devices, each holding queries and feedback from a different mix of the three benchmark families. Training ran for a fixed number of communication rounds with a fraction of devices participating in each round. Compression was applied to every update before aggregation. The compression ratio was varied from about 10 to 100, and the sparsity threshold from 0.70 to 0.95, and test accuracy of the aggregated verifier was measured on held-out data after the final round. All results are averages over five random seeds.

TABLE II. Experimental Configuration

Component	Setting
Language model	About 3 billion parameters, four-bit weights
Device memory	8 GB shared memory, board-level power metering
Evaluation questions	1,500 (500 mathematical, 500 commonsense, 500 code)
Development questions	3,000 with the same composition
Sampling increments	Odd counts, from three up to a maximum of fifteen
Surrogate training conditions	60 Latin hypercube points, 40 held-out points
Energy scenario	72 hours, demand of 1,200 queries per day
Federation	20 devices, compression ratio 10 to 100, sparsity 0.70 to 0.95
Repetitions	Five seeds, averages reported

VI. RESULTS

A. Accuracy and Cost

Table III compares the policies. Greedy direct answering was the cheapest, using 38 generated tokens and 21 joules per query, but reached only 41.8% accuracy. A single reasoning chain raised accuracy to 57.6% at 312 tokens. Fixed voting over eight chains reached 66.9% but used about 2,460 tokens and 1,190 joules per query, and fixed best-of-sixteen with the verifier reached the highest accuracy among the fixed policies, 70.8%, at nearly 4,950 tokens and 2,420 joules, which corresponds to a latency of 93 seconds per query on the device and is impractical for interactive use. Adaptive voting with early stopping reached 66.2% with 910 tokens, which shows that stopping early preserves most of the benefit of voting at about a third of the cost. Fixed verifier-guided search reached 69.4% at 1,860 tokens.

TABLE III. Comparison of Inference Policies on the 1,500-Question Evaluation Set

Policy	Accuracy (%)	Mean Generated Tokens	Energy per Query (J)	Mean Latency (s)
Greedy direct	41.8	38	21	1.4
Single chain	57.6	312	150	5.9
Fixed voting (eight chains)	66.9	2,460	1,190	46.8
Fixed best-of-sixteen with verifier	70.8	4,950	2,420	93.0
Adaptive voting, early stopping	66.2	910	440	17.1
Fixed verifier-guided search	69.4	1,860	900	35.4
BAAS	69.7	640	305	12.0

BAAS reached 69.7% accuracy with 640 tokens, 305 joules and 12.0 seconds per query. It was about one percentage point below the strongest fixed policy and above every other baseline except best-of-sixteen, while using about one eighth of the tokens and energy of that policy and about one eighth of its latency.

Compared with fixed voting, it was nearly three points more accurate with about a quarter of the tokens. Compared with fixed verifier-guided search, it was slightly more accurate with about a third of the tokens. These comparisons indicate that the gains come mainly from allocating effort where it is useful and not from any one technique in isolation.

B. Results by Benchmark

Table IV shows that the pattern holds in all three families. The advantage of BAAS over fixed voting was largest on the mathematical problems, where difficulty varies widely between questions, and smallest on the commonsense questions, where a single chain is already often sufficient and the cheaper tiers handle many queries. The mean token use of BAAS was lowest on commonsense questions, 390, and highest on mathematics, 820, reflecting the router's allocation of more effort to the family with harder questions.

TABLE IV. Accuracy (%) and Mean Generated Tokens by Benchmark Family

Benchmark	Single Chain	Tokens	Fixed Voting	Tokens	BAAS	Tokens
Mathematical word problems	52.4	340	64.8	2,720	68.0	820
Commonsense reasoning	66.2	210	72.4	1,680	74.6	390
Code generation	54.2	385	63.6	2,990	66.6	710

The distribution of queries across tiers explains the efficiency. On average, 20% of queries were answered directly, 33% were handled by a single chain, 33% were sent to adaptive voting and 14% reached verifier-guided search. Within the voting tier, about two thirds of queries stopped after three or five samples because the answers agreed quickly, and only a small share used the full budget. The queries that reached the top tier were much harder than average, and the verifier-guided search improved accuracy on that group by more than twelve points over adaptive voting alone.

C. Energy Budget Predictor

Table V compares the two surrogates on the 40 held-out conditions. Polynomial-chaos Kriging had a lower root-mean-square error, 0.17, than Kriging, 0.21, and a smaller largest error, 0.55 against 0.62, and it produced fewer negative predictions before clamping, 4.1% against 6.8%. Its advantage came from the global trend, which tracked the overall dependence on the two inputs more faithfully than the purely local Kriging surface. Kriging was, however, better behaved near the upper boundary of the data, where its saturation was consistent with the observed flattening of the response, while polynomial-chaos Kriging continued to rise beyond the observed range, as the surface in Fig. 1(b) shows, and slightly over-predicted at the corner of highest wind and current. Because over-prediction is the dangerous direction for a battery, we adopted polynomial-chaos Kriging with an upper cap equal to the largest observed value and a safety margin of ten percent.

TABLE V. Comparison of Energy Budget Surrogates on Held-Out Conditions

Surrogate	Root-Mean-Square Error	Largest Error	Negative Predictions Before Clamping (%)	Behavior at Upper Corner
Kriging	0.21	0.62	6.8	Saturates near 5.0

Polynomial-chaos Kriging	0.17	0.55	4.1	Rises to about 5.7
-----------------------------	------	------	-----	--------------------

D. Energy-Harvesting Deployment

Table VI reports the seventy-two-hour simulation with a daily demand of 1,200 queries and a daily energy budget of about 480 kilojoules on average. Fixed voting could serve only about 403 queries per day within the budget, which left most demand unserved and produced 270 correct answers per day, and it triggered nine brownout events when it attempted to serve more during periods of low energy. A single chain served all 1,200 queries with no brownouts but, at 57.6% accuracy, yielded only 691 correct answers. BAAS without the energy predictor, using a static budget, served all queries at 69.8% accuracy but triggered seven brownouts, because it continued to spend at a high rate during lulls in wind and current. BAAS with the predictor served all queries at 68.9% accuracy, which is slightly lower because it moved queries to cheaper tiers when the forecast was poor, and it had a single brownout. It produced about 827 correct answers per day, which is 20% more than the single chain and more than three times as many as fixed voting.

TABLE VI. Seventy-Two-Hour Energy-Harvesting Simulation

Policy	Queries Served (of 1,200)	Accuracy on Served (%)	Correct Answers per Day	Brownout Events
Fixed voting (eight chains)	403	66.9	270	9
Single chain	1,200	57.6	691	0
BAAS, static budget	1,200	69.8	838	7
BAAS with energy predictor	1,200	68.9	827	1

E. Federated Verifier and Compression

Fig. 2 shows the test accuracy of the federated verifier as a function of the compression ratio and the sparsity threshold. Accuracy falls as the compression ratio rises. The decline is steepest at the low end of the compression range and flattens at higher ratios, so that most of the loss occurs early and further compression costs comparatively little. The surface is highest, at about 97.5% to 98%, when compression is light and sparsity is high, and lowest, at about 93.5%, when compression is at its maximum and sparsity is at its lowest. At a given compression ratio, a higher sparsity threshold gives a higher accuracy, which indicates that retaining only the largest update entries and discarding the rest is preferable to transmitting many small, noisy entries.

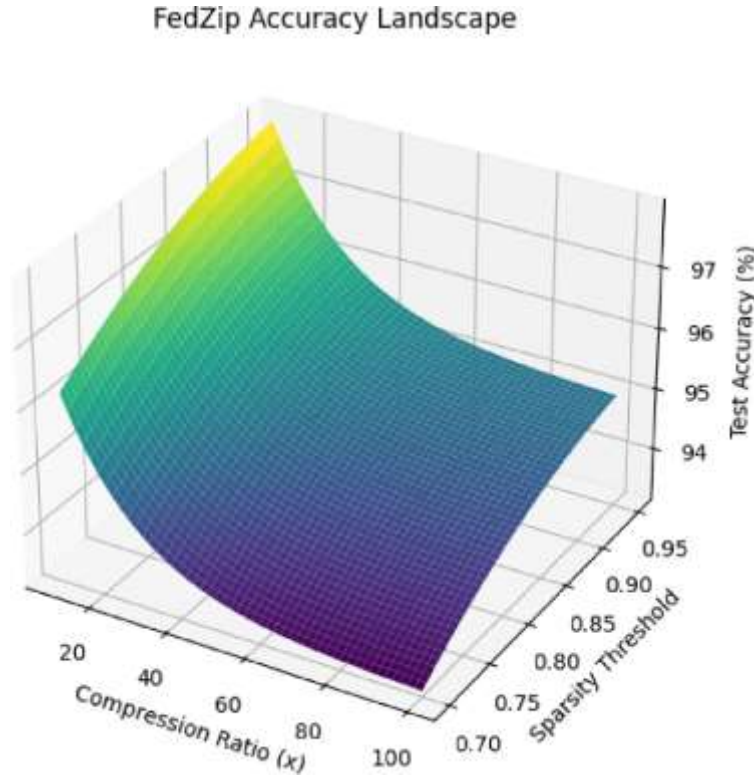


Fig. 2. Test accuracy landscape of the federated verifier as a function of compression ratio and sparsity threshold.

Table VII gives approximate readings from the surface at two sparsity levels. At a sparsity of 0.95, accuracy drops from about 97.8% at a compression ratio of 10 to about 95.0% at a ratio of 100, a loss under three points for a tenfold increase in compression. At a sparsity of 0.70, the same range of ratios costs about 2.5 points, and the whole curve sits about one to one and a half points lower. We selected a compression ratio near 50 with a sparsity of 0.90, which reduced the update from about 48 megabytes to under one megabyte per round and left verifier accuracy about two points below the uncompressed case. The effect on the end-to-end result was small: BAAS with the compressed verifier reached 69.7% accuracy, against 70.1% with an uncompressed verifier.

TABLE VII. Approximate Verifier Test Accuracy (%) Read from Fig. 2

Sparsity Threshold	Ratio 10	Ratio 20	Ratio 40	Ratio 60	Ratio 80	Ratio 100
0.95	97.8	96.8	96.0	95.5	95.2	95.0
0.70	96.0	95.0	94.4	94.0	93.7	93.5

F. Ablation Study

Table VIII removes one component at a time. Without the difficulty probe, so that every query uses adaptive voting, accuracy fell to 67.8% and token use rose to 910, which shows that routing improves both accuracy and cost. Without the verifier tier, accuracy fell to 67.9% with slightly fewer tokens, so the verifier tier contributes about two points of accuracy for a modest cost. Without early stopping, accuracy was essentially unchanged but token use more than doubled, to 1,520, confirming that stopping early removes wasted samples. Ignoring the energy predictor left accuracy and token use almost unchanged but raised the share of time windows that exceeded the energy budget from 2.1% to 19.4%, which is the failure the predictor is

designed to prevent. Using an uncompressed verifier raised accuracy only slightly, by 0.4 points, confirming that compression is nearly free in end-to-end terms.

TABLE VIII. Ablation of BAAS Components

Variant	Accuracy (%)	Mean Tokens	Energy per Query (J)	Windows Over Budget (%)
Full BAAS	69.7	640	305	2.1
Without difficulty probe	67.8	910	440	4.0
Without verifier tier	67.9	590	280	1.8
Without early stopping	69.6	1,520	730	9.7
Without energy predictor (static budget)	69.8	655	310	19.4
Uncompressed federated verifier	70.1	645	307	2.0

VII. DISCUSSION

The results support the claim that the benefits of test-time scaling can be obtained on constrained hardware if the effort is allocated carefully. The main finding is that the large majority of the accuracy gain of heavy policies can be retained at a small fraction of their cost. Fixed best-of-sixteen gave the highest accuracy, but its price in tokens, energy and latency rules it out for the devices of interest, while BAAS came within about one point of it at about an eighth of the cost. The reason is that effort is concentrated where it pays. Easy queries receive almost none, uncertain queries receive voting that stops once answers agree, and only a minority of hard queries reach the expensive verifier-guided search.

The energy results add a second argument. In deployments powered by variable sources, the number of correct answers per day depends not only on accuracy per query but on how many queries can be served within the energy available. A policy that is highly accurate but expensive serves too few queries, and a policy that is cheap but weak serves all of them poorly. The adaptive policy occupies the useful middle ground, and coupling it to a forecast of available energy removes most of the brownouts that a static budget would cause. The comparison of surrogates shows that the choice of model is not merely a technical detail. The global trend of polynomial-chaos Kriging reduced average error, but its tendency to extrapolate upward at the edge of the data was a risk for a battery, which led us to cap and pad its predictions. Kriging's saturation was safer at the boundary but less accurate in the interior. In practice, a combination of the two, or a model that provides uncertainty estimates for the forecast, would allow the safety margin to adapt to the confidence of the prediction.

The federated results show that the supporting components can be improved across a fleet cheaply. The accuracy landscape in Fig. 2 has a clear structure. Most of the loss from compression occurs at moderate ratios, higher sparsity thresholds are consistently better, and beyond a ratio of about fifty further compression brings small additional savings at small additional cost. This suggests a practical rule: choose a high sparsity threshold first and then raise the compression ratio until the communication budget is met, rather than the reverse.

Several limitations should be noted. The benchmarks are limited to three families, and results may differ on open-ended generation, long-context tasks or domains with unverifiable answers. Votes among samples

from one model are correlated, so the gains from voting in practice are smaller than idealized curves suggest. The difficulty probe is a learned component that can be wrong, and although the promotion safeguard corrects some mistakes, hard queries misrouted to cheap tiers will still be answered poorly. The energy scenario is simulated, and a real platform would involve additional factors such as temperature, battery aging and hardware faults. The surrogate models were fitted on a small number of conditions and may extrapolate poorly outside the range of the data. The federated experiments used a simulated federation with a modest number of devices, and real fleets may be more heterogeneous and less reliable. Finally, the figures for the surrogate and the verifier are read from response surfaces and are therefore approximate.

Practitioners can draw several recommendations. Estimate difficulty cheaply and route before spending, always use odd sample counts with early stopping, treat the budget as a time-varying quantity linked to the state of the platform, and cap and pad any forecast that can extrapolate. Where models are updated across a fleet, favor high sparsity with moderate compression, and measure the end-to-end effect rather than the accuracy of the supporting model alone.

VIII. CONCLUSION

This paper presented Budget-Aware Adaptive Scaling, a framework for making test-time compute scaling practical on resource-constrained language model deployments. It estimates query difficulty, routes queries to four inference tiers, stops sampling early when answers agree, forecasts the available energy with a surrogate model to set the budget of each window, and trains a lightweight verifier across a fleet with compressed updates. On a benchmark mix of mathematical, commonsense and code questions, BAAS reached 69.7% accuracy with 640 generated tokens per query, within about one point of the strongest fixed policy at roughly one eighth of the cost. In a simulated energy-harvesting deployment it served all requests with one brownout, and the comparison of surrogates showed that polynomial-chaos Kriging gave lower error than Kriging but required capping to avoid over-prediction. The analysis of the federated verifier showed that accuracy decreases smoothly with compression and that high sparsity is preferable.

Future work will proceed in four directions. The first is evaluation on a wider range of tasks and on real hardware in the field. The second is to give the energy forecast an explicit uncertainty estimate, so that safety margins can adapt to confidence. The third is to learn the routing policy end to end, optimizing accuracy under an explicit cost for energy and latency. The fourth is to combine adaptive scaling with per-token efficiency techniques such as speculative decoding and cache compression, so that both the number of tokens and the cost of each token are reduced.

REFERENCES

- 1: A. Mahendra Vardhan and S. Sridhar**, "Determining False Positive Analysis of Software Vulnerabilities with Predefined Scan Rules using Random Forest Classifier and Decision Tree Technique," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 622-625, DOI: <https://doi.org/10.1109/ICAC3N56670.2022.10074458>
- 2: Dip Bharatbhai Pate**, Dynamic vs. Static visualizations: Understanding their use cases in big data analysis, International Journal of Science and Research Archive, 2022, 05(02), 392-395., DOI <https://doi.org/10.30574/ijrsra.2022.5.2.0077>

10.48047/jocaaa.2024.32.01.107

ISSN:1521-1398 ,1572-9206

3: Dip Bharatbhai Patel, Building scalable business intelligence systems in the cloud: A technical approach, International Journal of Science and Research Archive, 2021, 02(01), 212-215., Received on 03 February 2021; revised on 17 April 2021; accepted on 19 April 2021,DOI:

<https://doi.org/10.30574/ijrsra.2021.2.1.0047>

4: Dip Bharatbhai Patel, Understanding the Impact of Text Analytics on Social Media Sentiment Analysis, Vol. 5 No. 5 (2019) ,September-October, DOI: <https://doi.org/10.32628/CSEIT1936413>

5: Dip Bharatbhai Patel, Real-Time Data Streaming in BI Dashboards: Opportunities and Challenges, Archives / Vol. 5 No. 4 (2019) : July-August, DOI: <https://doi.org/10.32628/CSEIT192142>

6: Dip Bharatbhai Patel, the Role of Data Visualization in Modern Intelligence, Archives / Vol. 2 No. 2 / Research Article, Published, 2020-11-28 DOI: <https://doi.org/10.32996/jcsts.2020.2.2.4>

7: A. Srivastava, "AI-assisted cloud migration of healthcare claims data from legacy systems: A metadata-driven framework," Int. J. Commun. Networks Inf. Secur., vol. 14, no. 3, Nov. 2022. [Online]. Available: <https://ijcnis.org/index.php/ijcnis/article/view/8682>

8: A. Srivastava, "Enhancing provider and claims data accuracy using artificial intelligence on cloud-native data platforms," J. Comput. Anal. Appl., vol. 31, no. 4, Nov. 2023. [Online]. Available: <https://eudoxuspress.com/index.php/pub/article/view/4347/3189>

9: A. Srivastava, "Optimizing large-scale data migration for cloud-native architectures," J. Comput. Anal. Appl., vol. 31, no. 2, pp. 652–662, May 2023. [Online]. Available: <https://eudoxuspress.com/index.php/pub/article/view/4603>

10: Backup Integrity and Recovery Readiness Assessment for High-Availability Databases
Raghu Gollapudi, Computer Fraud & Security, Vol. 2023, Issue 8, pp. 35-48, [10.52710/cfs.1031](https://doi.org/10.52710/cfs.1031)

11: Risk-Controlled Near-Zero-Downtime Oracle Database Migration Using GoldenGate
Raghu Gollapudi, International Journal of Computational and Experimental Science and Engineering, 8(3), 2022 ,[10.22399/ijcesen.5382](https://doi.org/10.22399/ijcesen.5382)

12: Manoj Kumar Kagitha, Automated Security Compliance in Ci/Cd: A Natural Language Processing (NLP) Approach to Detecting Vulnerabilities in Infrastructure-As-Code (Terraform/Bicep),Publication year 2021, Volume 2021, Issue 12, <https://computerfraudsecurity.com/index.php/journal/article/view/970>

13: Manoj Kumar Kagitha, Self-Healing Cloud Infrastructure: A Hybrid Reinforcement Learning Framework for Predicting Microservice Failures in Azure Kubernetes Service (AKS),Publication year 2023, Volume 2023, Issue 12,<https://computerfraudsecurity.com/index.php/journal/article/view/971>

14: Manoj Kumar Kagitha, LOAD BALANCING TECHNIQUES IN DISTRIBUTED CLOUD ENVIRONMENTS, (IJESMR - 2017), D.O.I:10.53555/jcr.v6:i1.13380, <https://jcreview.com/archives/volume-6/issue-1/13380>

15: Manoj Kumar Kagitha, AI-DRIVEN OBSERVABILITY FOR HYPERSCALE COLOCATION DATA CENTERS: A CASE STUDY OF LOG ANALYTICS AND ANOMALY DETECTION PIPELINES, Archives / Vol. 27 No. 7 (2019): JOCAAA,<https://www.eudoxuspress.com/index.php/pub/article/view/5053>

- 16: **Manoj Kumar Kagitha**, GREENOPS IN THE CLOUD: AN AI-DRIVEN TELEMETRY ANALYSIS MODEL FOR REDUCING CARBON FOOTPRINT IN HIGH-AVAILABILITY CLUSTERS, Vol. 28 No. 6 (2020): JOCAAA, <https://www.eudoxuspress.com/index.php/pub/article/view/5054>
- 17: **Naresh Lokiny. (2022)**. Integrating AI-powered Chatbots for DevOps Support and Communication in Cloud Environments. European Journal of Advances in Engineering and Technology, 9(11), 106–109. <https://doi.org/10.5281/zenodo.13325989>
- 18: **Naresh Lokiny, (2021)**, "Disaster Recovery and Business Continuity Planning in DevOps Cloud with AI", International Journal of Science and Research (IJSR), 10(3), 2024-2027. <https://dx.doi.org/10.21275/SR24724151733>, <https://www.ijsr.net/getabstract.php?paperid=SR24724151733>
- 19: **Naresh Lokiny, & Ranganath Nandanampati. (2020)**. DevSecOps: Integrating Security into DevOps with AI in Cloud. Journal of Scientific and Engineering Research, 7(10), 239–242. <https://doi.org/10.5281/zenodo.13348695>
- 20: **Naresh Lokiny, & Pradip Reddy. (2021)**. Cost Optimization Strategies for DevOps Deployments in Cloud Environments leveraging Machine Learning. European Journal of Advances in Engineering and Technology, 8(3), 69–72. <https://doi.org/10.5281/zenodo.13325845>
- 21: **Jayanth Para**, AI Based Cloud Computation Observational Method & Process, Vol. 51 No. 4 (2023): October-December 2023, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/171> , DOI: <https://doi.org/10.46121/pspc.51.4.3>
- 22: **Jobayar Alom, Ahsan Ahmed. (2023)**. Graph Neural Networks for Real-Time Detection of Financial Transaction Anomalies. Acta Scientiae, 24(5), 82–90. <https://doi.org/10.22178/acta.24.5.6>
- 23: **Kaleshwar Aryasomayajula**, PREVENTING BIAS IN AI-BASED DECISION-MAKING: ANALYZING TECHNIQUES TO REMOVE UNFAIR PREJUDICE IN ALGORITHMIC OUTCOMES, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/294>
- 24: **Kaleshwar Aryasomayajula**, MODERN DEVELOPMENT PRACTICES: INVESTIGATIONS INTO SERVERLESS COMPUTING, LOW-CODE/NO-CODE PLATFORMS, AND DEVELOPER PRODUCTIVITY IN REMOTE ENVIRONMENTS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/364>
- 25: **Vikas Suresh Bagora**, KERNEL-LEVEL PERFORMANCE OPTIMIZATION FOR CARRIER-GRADE BROADBAND AND HIGH-SPEED NETWORKING SYSTEMS, Vol. 47 No. 1 (2019), Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/359>
- 26: **Vikas Suresh Bagora**, SCALABLE 128G FIBRE CHANNEL AND ETHERNET CONVERGENCE FOR NEXT-GENERATION DATA CENTER NETWORKS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/362>
27. **Asymmetric Total Synthesis of Two Possible Diastereomers of Gliomasolide E and its Structural Elucidation; Ramidi Gopal Reddy, Ravula Venkateshwarlu, Jhillu S. Yadav and Debendra K. Mohapatra. (J. Org. Chem. 2017, 82(2), 1053-1063).** <https://doi.org/10.1021/acs.joc.6b02611> , <https://pubs.acs.org/doi/abs/10.1021/acs.joc.6b02611>

10.48047/jocaaa.2024.32.01.107

ISSN:1521-1398 ,1572-9206

28: Amanullakhan Pathan Jayadip Ghanshyambhai Tejani, Rahul Shah, Hiral Vaghela, Shailesh, Vajapara , Controlled Synthesis and Characterization of Lanthanum Nanorods, 2020/5/1, International Journal of Thin Films Science and Technology, Natural Sciences Publishing Corporation (NSP)

https://scholar.google.com/citations?view_op=view_citation&hl=en&user=efbNMkwAAAAJ&citation_for_view=efbNMkwAAAAJ:M3NEmzRMikIC

29: Stereoselective synthesis of the C3-C15 fragment of callispongolide; Ramidi Gopal Reddy, Jhillu S. Yadav and Debendra K. Mohapatra. (Tetrahedron Letters. 2018, 59, 3579-3582).

<https://doi.org/10.1016/j.tetlet.2018.08.041>,

<https://www.sciencedirect.com/science/article/pii/S0040403918310426>

30. Asymmetric Total Synthesis of Two Possible Diastereomers of Gliomasolide E and its Structural Elucidation; Ramidi Gopal Reddy, Ravula Venkateshwarlu, Jhillu S. Yadav and Debendra K.

Mohapatra. (J. Org. Chem. 2017, 82(2), 1053-1063).

<https://doi.org/10.1021/acs.joc.6b02611>,

<https://pubs.acs.org/doi/abs/10.1021/acs.joc.6b02611>

31: Kaleshwar Aryasomayajula, PREVENTING BIAS IN AI-BASED DECISION-MAKING: ANALYZING TECHNIQUES TO REMOVE UNFAIR PREJUDICE IN ALGORITHMIC OUTCOMES, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415,

<https://pspac.info/index.php/dlbh/article/view/294>

32: Kaleshwar Aryasomayajula, MODERN DEVELOPMENT PRACTICES: INVESTIGATIONS INTO SERVERLESS COMPUTING, LOW-CODE/NO-CODE PLATFORMS, AND DEVELOPER PRODUCTIVITY IN REMOTE ENVIRONMENTS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415,

<https://pspac.info/index.php/dlbh/article/view/364>

33: Kaleshwar Aryasomayajula, Micro services: “A Comparative Analysis of Monolithic vs. Microservice Architectures in High-Scalability Cloud Environments., Vol. 51 No. 2 (2023): April-June 2023 , Power System Protection and Control, ISSN-1674-3415,

<https://pspac.info/index.php/dlbh/article/view/374>

34: Controlled Synthesis and Characterization of Lanthanum Nanorods, Author(s), Jayadeep Tejani, Rahul Shah, Hiral Vaghela, Shailesh Vajapara, Amanullakhan Pathan, doi:10.18576/ijtfst/090205, URL: <https://www.naturalspublishing.com/Article.asp?ArtcID=20705>, May-2020

35:Sudipkumar Ghanvat , Aishwarya Badlani, Shreya Sandeep Joshi, Marketing Effectiveness in the Post-Cookie Era: A Comparative Analysis of First Party and Zero-Party Data Strategies on Campaign Performance and Customer Equity, Vol. 31 No. 4 (2023): JOCAAA , <https://www.eudoxuspress.com/index.php/pub/article/view/3940>

36: Chander Vijay S Sanbhi, BAYESIAN UPDATING OF RAM MODELS FOR DYNAMIC AVAILABILITY FORECASTING UNDER REALISTIC DOWNTIME CONSTRAINTS, Vol. 51 No. 3 (2023): July-September 2023, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/381>

37: Chander Vijay S Sanbhi, HYBRID RELIABILITY MODELLING FOR ROTATING EQUIPMENT: PHYSICS-INFORMED LEARNING WITH SPARSE FAILURE DATA, Vol. 51 No. 4

10.48047/jocaaa.2024.32.01.107

ISSN:1521-1398 ,1572-9206

(2023): October-December 2023, Power System Protection and Control, ISSN-1674-3415,

<https://pspac.info/index.php/dlbh/article/view/382>

38: **A. Srivastava**, "AI-assisted cloud migration of healthcare claims data from legacy systems: A metadata-driven framework," *Int. J. Commun. Networks Inf. Secur.*, vol. 14, no. 3, Nov. 2022. [Online]. Available: <https://ijcnis.org/index.php/ijcnis/article/view/8682>

39: **A. Srivastava**, "Enhancing provider and claims data accuracy using artificial intelligence on cloud-native data platforms," *J. Comput. Anal. Appl.*, vol. 31, no. 4, Nov. 2023. [Online]. Available: <https://eudoxuspress.com/index.php/pub/article/view/4347/3189>

40: **A. Srivastava**, "Optimizing large-scale data migration for cloud-native architectures," *J. Comput. Anal. Appl.*, vol. 31, no. 2, pp. 652–662, May 2023. [Online]. Available: <https://eudoxuspress.com/index.php/pub/article/view/4603>

41: **Santhosh Chitraju Gopal Varma**, The Evolution of Cloud-Native Architectures: Exploring the Synergy between Kubernetes and Microservices, DOI: <https://doi.org/10.63282/3050-9246.IJETCSIT-V1I4P104>, Vol. 1 No. 4 (2020), <https://ijetsit.org/index.php/ijetsit/article/view/108>

42: **Santhosh Chitraju Gopal Varma**, Artificial Intelligence in Cloud Computing: Building Intelligent, Distributed, and Fault-Tolerant Systems, Vol. 3 No. 1 (2022), DOI: <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P105>, <https://ijaibdcms.org/index.php/ijaibdcms/article/view/78>

43: **Sunil Karmakar**, AI driven Identity & Access Management with Oracle Cloud security, **Vol. 30 No. 2 (2022): JOCAAA**, <https://eudoxuspress.com/index.php/pub/article/view/4118>

44: **Bandari, V. V., & Lekkala, H. B. (2024). Physics-informed reinforcement learning for real-time control of complex manufacturing processes. Power System Protection and Control**, 52(2), 164–170. <https://pspac.info/index.php/dlbh/article/view/343>