Drone & Ksurf: Attention Kalman Filter for Resource Allocation in the Cloud with Drone

Michael Dang'ana
University of Toronto
Toronto, Ontario, Canada
michael.dangana@mail.utoronto.ca

Yuqiu Zhang
University of Toronto
Toronto, Canada
yuqiu.zhang@mail.utoronto.ca

Hans-Arno Jacobsen
University of Toronto
Toronto, Canada
jacobsen@eecg.toronto.edu

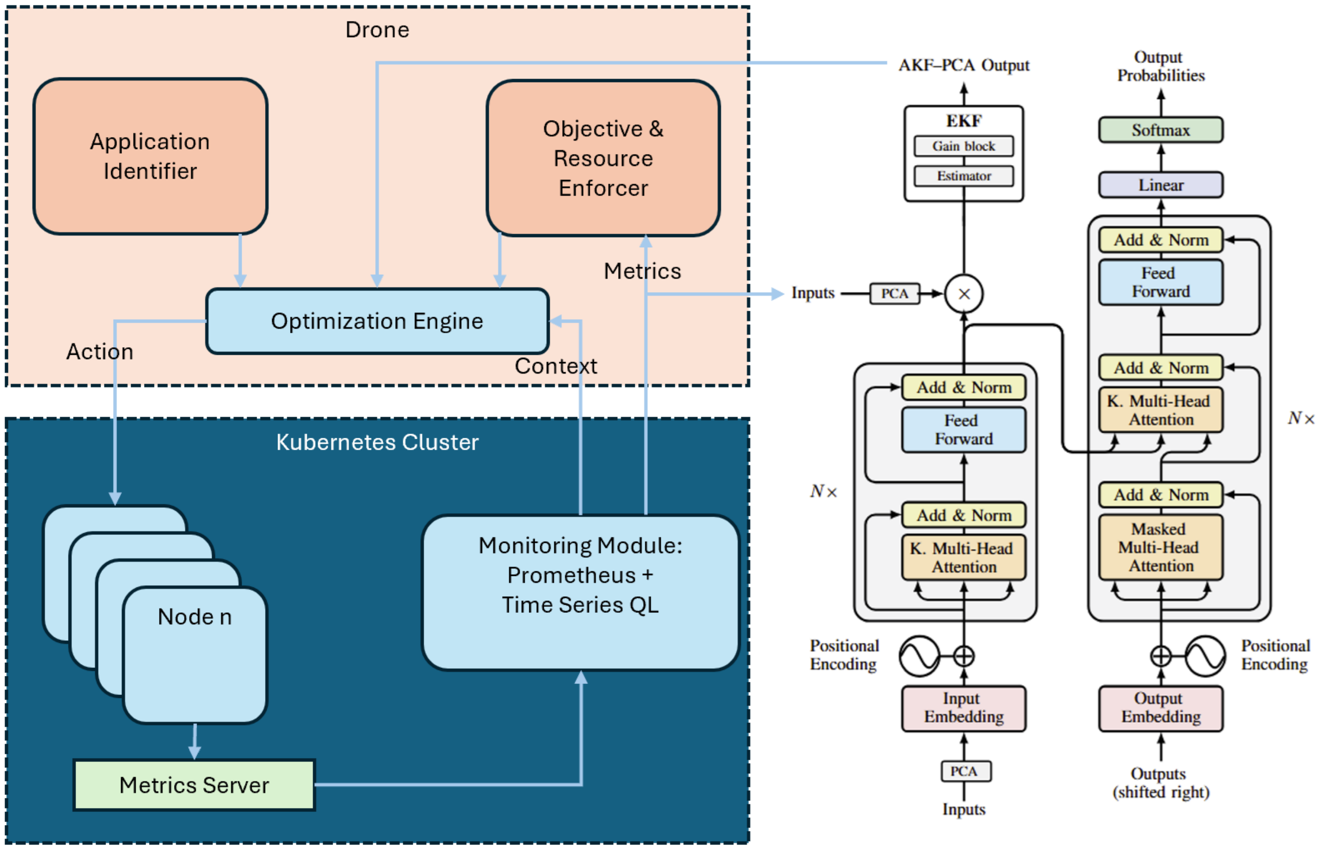

Figure 1: Drone & Ksurf Optimization Architecture

Abstract

Resource orchestration and configuration parameter search are key concerns for container-based infrastructure in cloud data centers. Large configuration search space and cloud uncertainties are often mitigated using contextual bandit techniques for resource orchestration including the state-of-the-art Drone. Complexity in the cloud provider environment due to varying numbers of virtual machines introduces variability in workloads and resource metrics, making orchestration decisions less accurate due to increased nonlinearity and noise. Ksurf is a state-of-the-art variance-minimizing estimator method ideal for highly variable cloud data which provides potential for optimized estimation-based resource orchestration under conditions of high cloud variability.

This work evaluates Ksurf performance on estimation-based resource orchestration under highly variable workloads employed as a contextual multi-armed bandit objective function model for private cloud scenarios. Ksurf demonstrates a 4% reduction in CPU usage and a 7 MB reduction in master node memory usage in the Kubernetes cluster resulting in a 7% cost savings in average

Under Review, NA
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

worker pod count on *VarBench* Kubernetes benchmark for the Drone contextual multi-armed bandit algorithm.

CCS Concepts

• **Computing methodologies → Online learning settings**; • **Computer systems organization → Cloud computing**.

Keywords

Kalman Filter, Contextual Bandits, Gaussian Process Regression, Microservices, Online Learning, Flash Crowds, Resource Prediction, Cloud Computing, Attention Mechanism

ACM Reference Format:

Michael Dang'ana, Yuqiu Zhang, and Hans-Arno Jacobsen. 2018. Drone & Ksurf: Attention Kalman Filter for Resource Allocation in the Cloud with Drone. In *Proceedings of Submitted for ACM Publication (Under Review).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Containers have become the prevalent technology for deployment of applications and compute jobs in the cloud at scale due to their support for fine-grained controllability, fluid orchestration and scalability when compared to virtual machines [39] [12] [23]. Wide adoption of containers has introduced *cloud* and *workload uncertainties* leading to increased risks of under- and over-provisioning cloud infrastructure [15] [10] [24]. Unstructured and time-variant resource performance and cost relationships are prevalent in public cloud systems exasperating uncertainties in cloud resource usage and inter-container traffic patterns [2] [6] [30].

Endogenous variations in cloud infrastructure including varying numbers of virtual machines, resource availability fluctuations and growth in user numbers introduce *cloud* and *resource variability* increasing uncertainty in orchestration decisions [16] [22]. These make it difficult to minimize costs, optimize resource usage, and risk Service Level Agreement (SLA) violations [35].

Drone, an online resource orchestration framework was introduced as the state-of-the-art solution to *cloud* and *resource uncertainty* mitigation for containerized clouds using Contextual Bandit techniques [39]. Drone requires minimal configuration and is autonomous over the orchestration lifespan eschewing the need for costly workload profiling employing Guassian process-based contextual bandits to model uncertainty, demonstrating sublinear growth of convergence time in both unlimited resource conditions of public clouds and resource-constrained private clouds [36]. Gaussian process models are susceptible to nonlinearity in data which can be effectively mitigated using Extended Kalman Filter-based techniques including Ksurf which has demonstrated state-of-the-art performance in resource estimation under conditions of high variability [19] [38] [9].

This work evaluates Drone under high *workload and resource variability* conditions using *VarBench* Kubernetes benchmark and examines the performance gains from incorporating Ksurf for uncertainty modelling [9]. The key contribution involves a comparison of Ksurf to Gaussian process modelling for variability mitigation in orchestration decision making scenarios when combined to Drone, and an evaluation of the performance of Drone in comparison to the Kubernetes Horizontal Pod Autoscaler in non-recurring job scenarios for which bandit-based approaches are sub-optimal [1] [16].

This work advances Drone by making the following contributions:

- Proposes a novel Drone+Ksurf algorithm for cloud resource orchestration under conditions of environment *uncertainty* and *variability*
- Incorporates Drone+Ksurf into a Kubernetes cluster benchmark
- Comparison of Drone+Ksurf and Drone on container orchestration tasks under high *resource* and *workload variability*

2 Background

Heuristics, model analytics, and predictive methods represent the main categories of cloud resource orchestration techniques [39]. Contextual bandit algorithms are a new technique that avoids dependency on model pre-training and human intervention inherent in the main cloud resource orchestration categories [36]. Contextual bandit is a variant of the Multi-Armed Bandit (MAB) problem using contextual information about environment variables in the form of uncertainties in the cloud, which is a discrete form of Bayesian Optimization that seeks to progressively build up an objective function model of an unknown system [4].

The need to model trade-offs between multiple objectives such as the need to minimize CPU usage, memory and latency simultaneously demands the use of multi-objective multi-armed bandits where the effects of decisions are modeled as a vector of objectives, over which no assumptions are made about utility function values [29]. Whereas the strength of Gaussian process algorithms is their ability to learn arbitrary utility functions, constraints are useful to enable context awareness about the environment into the utility functions and enable action selection at each iteration, hence the development of contextual bandit algorithms [36].

Gaussian process regression (GP) is widely used in contextual bandit algorithms for non-parametric Bayesian modeling of unknown systems [36]. In Drone, GP is used to select actions for exploration and is updated after exploitation in private and public cloud scenarios. GP regression is a batch processing technique with appreciable execution cost $O(n^3)$ while Kalman filter methods are online, low cost, ideal for nonlinear data where system model and Jacobians are well defined $O(n)$ [14]. Ksurf (AKF-PCA) provides further benefits over GP for Drone by employing noise reduction using principal component analysis and modeling of temporal relationships in data through the attention mechanism adopted from state-of-the-art Transformer neural network architecture, and where GP struggles with abrupt transitions and non-stationarities, data-driven Kalman filters can handle adapt quickly to regime shifts [11] [9] [28].

3 Related Work

Container cloud orchestration is becoming more relevant due to the prevalence of high workload variability and the need to ensure SLA and minimize costs in cloud computing. Techniques employed to address these issues often combine Bayesian optimization, and multi-objective decision-making.

State-of-the-art bandit algorithms for systems research include Cherrypick which lacks convergence guarantees for its acquisition function and Accordia, which study the virtual machine configuration selection problem and recurring analytical jobs with predictable workloads [1] [21]. The container configuration selection problem is qualitatively unique requiring fine-grained continuous control well suited to Drone, which uniquely generalizes to workload variations and variable cloud settings due to lack of need for pre-training, a key step for Cherrypick and Accordia, and a feature shared with RAMBO which uses multi-objective BO to solve the performance-cost trade-off between Service Level Agreements (SLA) and Total Cost of Ownership (TCO) for the customer in multi-microservice scenarios, achieving pareto-optimal performance guarantee [20].

High approximation accuracy and analytic uncertainty characterization are key advantages of contextual decision-making algorithms leveraging neural network-accompanied Gaussian processes (NN-AGP) [37]. When decision and reward functions are derived from continuous sets, estimating the reward function while taking uncertainty and accuracy into account is a significant challenge often tackled using GP, which remains susceptible to time-varying reward variables and graph-structured contextual variables [25]. Similar to neural network-based model completion schemes like KalmanNet, NN-AGP performs model completion by approximating the unknown reward function and allowing GP to compute the decision variable [37] [28]. This leads to state-of-the-art approximation accuracy and explicit GP uncertainty quantification.

Ksurf improves upon these approaches by incorporating Kalman filter-based online learning which provides robustness under high variability while avoiding configuration selection and the complexity cost of offline batch methods, well adapted to time-varying variable estimation [9]. Drone with Ksurf combines these approaches into a dynamic contextual bandit optimization module, outperforming GP-based optimization regarding resource variability and convergence rate, and offers more scalable complexity for resource-constrained scenarios. Kalman filters have demonstrated the capability for inference of graph-structured models offering the potential to tackle a key limitation of GP [5].

4 Drone+Ksurf

This work involves incorporating Ksurf as an optimization function for the Drone contextual bandit to benefit from robust noise reduction and modeling of temporal relationships [9]. The Drone optimization module is made extensible to support configurable optimization functions, and a function selector parameter is added to the configuration file as shown in Figure 2. To initialize the Ksurf attention mechanism, a sample of time series data is incorporated into the system configuration, and a training phase is executed when installing Drone software.

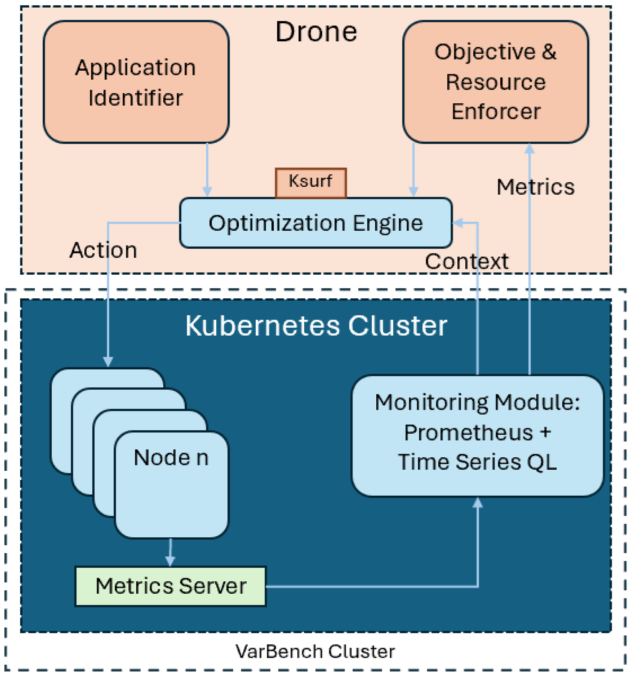

Figure 2: Drone *VarBench* Kubernetes Cluster with Prometheus

Drone consists of a Monitoring Module responsible for performance metric including CPU, RAM, network bandwidth, and cloud context collection using Prometheus [26]. The Application Identifier determines the application type, which can be user-specified, for targeted orchestration decisions [39]. The Objective and Resource Enforcer sets the objective for the optimization engine, where users can specify model parameters, performance-cost coefficients and resource limits. The Optimization Engine executes the optimization process, receiving and processing performance and contextual metrics and exploring resource orchestration actions iteratively, which are exploited through the Kubernetes API [27]. Ksurf is executed as the objective function model by the Optimization Engine in the private cloud algorithm during the exploration action selection and exploitation posterior-update steps.

The complexity of GP is $O(n^3)$ while Ksurf has a complexity of $O(m^3+LN^2d)$ where m is the most recent measurement size, L, N, d are attention network layer, token and query size, and n the data set size is the largest parameter $n > N \gg m$ [.] Due to the need to perform regression over the entire data set at each iteration, GP is inherently more computationally intensive than Ksurf, however, this is apparent on very large data sets and grows as n increases [13] [33].

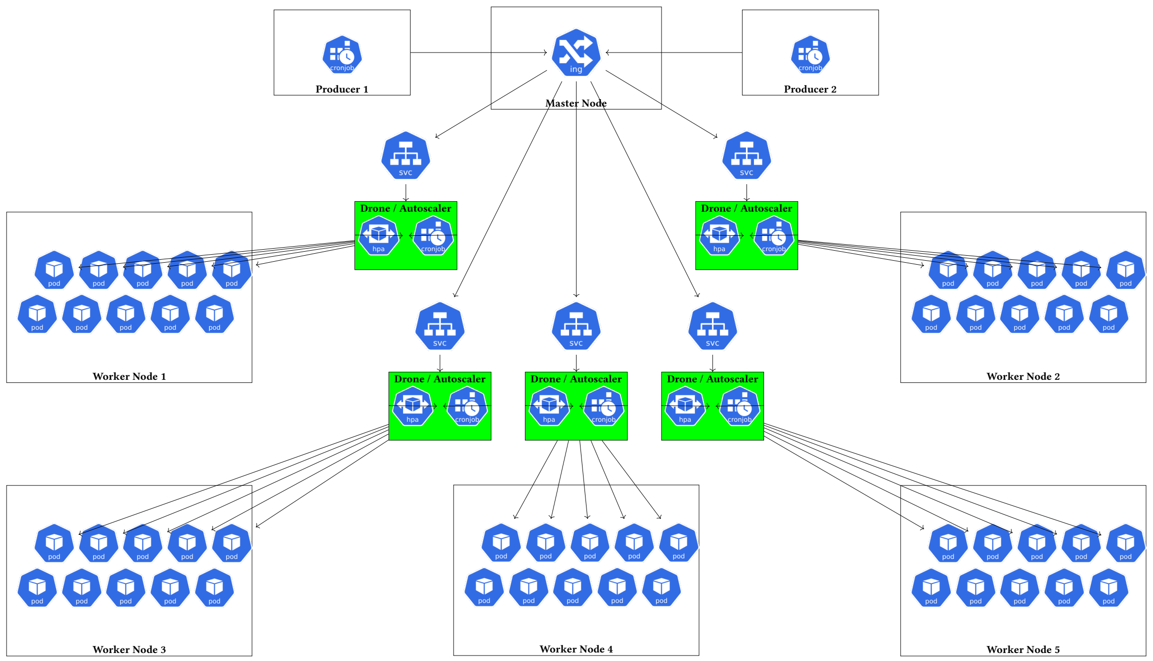

Figure 3: VarBench Kubernetes Cluster With 5 HPA & Auto-scaler Nodes, 50 Worker Pods and 2 Producer Nodes

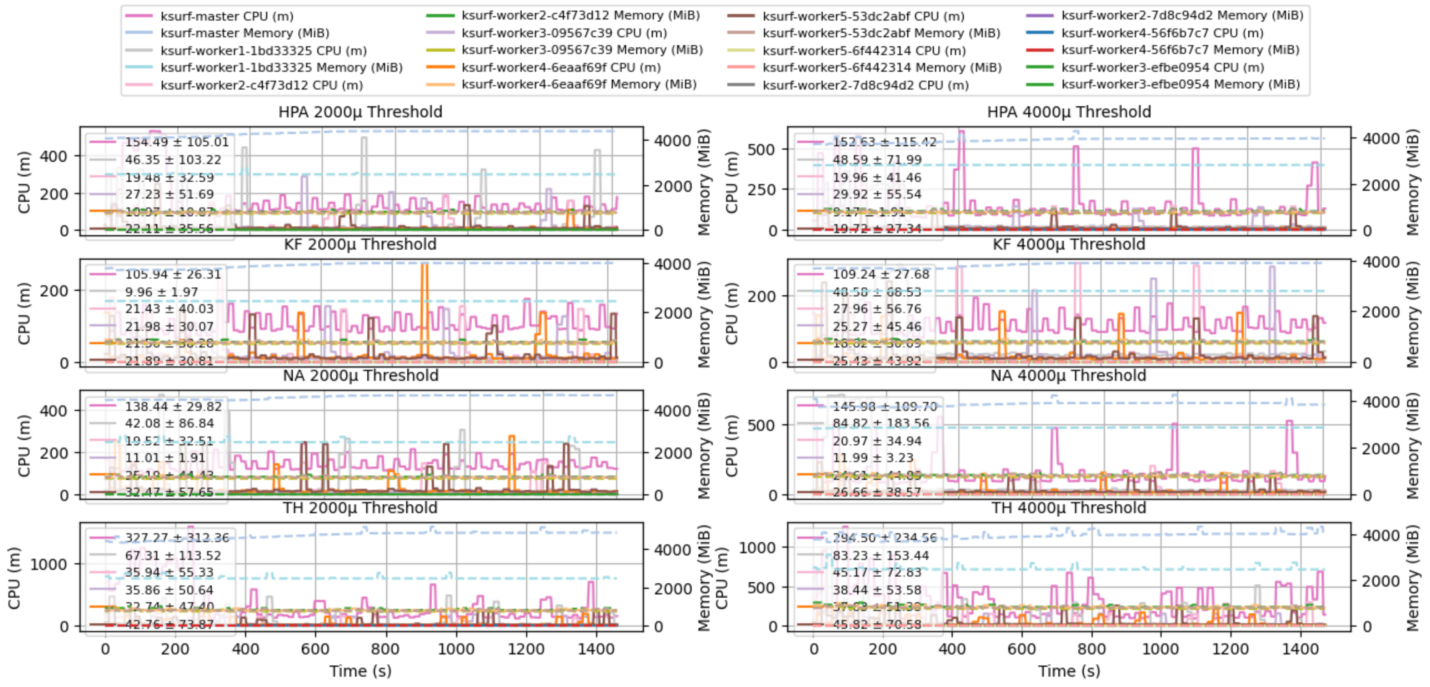

Figure 4: Kafka Node Metrics at 2 & 4 Millipod Thresholds

5 Evaluations

The evaluation involves benchmark experiments using Drone for resource allocation and Prometheus for metric retrieval, computation and persistence. The experiments use ten 4 core x86 64bit Intel Platinum 6548Y+ 2.5GHz 60MB cache (Broadwell) CPU and 15.4GB DDR5 memory compute nodes running Ubuntu Linux, Kubernetes 1.32 and Apache Kafka 2.8.2 in Kubernetes pod containers on Arbutus Compute Canada [7]. Datasets are 1-dimensional resource traces with temporal sliding windows used as a second dimension. The Attention neural networks are trained using a 10 thousand-value training set selected from the respective experiment time series historical data. Ksurf auto-scaler (KF), Drone (DR), Drone+Ksurf (DRKF), threshold crossing auto-scaler (TH) and non-threshold crossing baseline (NA) perform orchestration in the experiments.

5.1 VarBench Kafka Benchmark Experiments

VarBench is a tool for benchmarking real-time message queue processing microservice applications with variable workloads at the application and underlying cloud platform layers [9]. *VarBench* consists of a Kubernetes cluster with 1 master, 5 scaler and 50 worker nodes, and an Apache Kafka application consisting of an Apache Zookeeper instance, 2 Kafka producer nodes, and consumer nodes running on Kubernetes worker pods with one consumer per pod, a Prometheus instance to collect metrics from worker pods, together with Drone for pod orchestration as shown in Figure 3 [18] [3].

5.1.1 Kafka Workload & Application. Typical online web traffic arrives in intervals that follow exponential distributions over small time scales fitting a Poisson process model [31]. The experiments use a Poisson workload at $\lambda = 0.125$, where the Kafka producer targets the Kafka service endpoint using a single topic with n partitions and $n = num\_brokers$ brokers [32] [8]. The Kafka client is a simple application running on the worker pods receiving the workload as notifications which are written to a log file [8].

5.1.2 VarBench Configuration. The *VarBench* configuration is augmented with a Prometheus pod instance for metrics collection and aggregation and a Drone instance for orchestration. The Drone instance is configured to use the private cloud algorithm with GP or Ksurf model depending on the experiment type [39]. The workload is driven by the producer nodes orchestrating all provisioned testing resources and compilation of test results. The testing goal is to compare Drone and Drone+Ksurf to four *VarBench* auto-scaler types (HPA, KF, NA, TH) at selected CPU thresholds (0.5, 1, 2, 4 millipods) [9].

5.1.3 Results. DRKF demonstrates a reduction of pod scaling actions regarding the number of pods when compared to Drone, leading to a 12.5% reduction in worker pod count. DRKF displays lower worker pod variability represented by pod count variance reduction of over 110.1% demonstrating the effectiveness of Ksurf at mitigating Drone cloud variability as shown in Figure 7.

DRKF also demonstrates reduced resource variability demonstrated by a significant 3.81% reduction in worker pod CPU usage and 7.74% reduction in variance when compared to Drone as shown in Figure 9 and Table 2. The manually configured threshold-crossing *VarBench* auto-scalers are more effective at maintaining lower worker pod CPU usage at all time periods demonstrated in Figure 7. Drone accurately models the performance-action relationship including broad sets of variables besides past resource usage which is enhanced by lower variability predictions provided by Ksurf.

DRKF mitigates a limitation of Drone regarding "flash crowd" workloads which are characterized by short-term bursty traffic which breaks the Gaussian Process prior assumption, due to the capability of Ksurf to support nonlinear models maintaining model consistency and reducing the risk of over-provisioning and under-provisioning under these conditions.

CPU (Millipod) & Memory (MiB)				
Metric	DR Scaler CPU	DRKF Scaler CPU	DR Scaler Mem	DRKF Scaler Mem
Mean	545.7	545.2	2788.5	2829.9
Std Deviation	387.74	367.10	143.15	130.96

Table 1: ***VarBench*** **Scaler Node Metric Summary**

CPU (Millipod) & Memory (MiB)				
Metric	DR Worker CPU	DRKF Worker CPU	DR Worker Mem	DRKF Worker Mem
Mean	65.26	**62.93**	827.2	830.3
Std Deviation	42.71	**39.64**	20.91	20.71

Table 2: ***VarBench*** **Worker Pod Metric Summary**

5.2 Discussion

5.2.1 Worker Pod Memory and CPU Usage Tradeoff. DRKF results in consistently higher memory usage on average across all worker pods in the variability experiments. However, the significance is 10x lower with than the CPU usage improvement only a 0.373% increase in memory usage with a 0.96% reduction in memory usage variance. In the private cloud algorithm, Drone uses the CPU metric as the action variable in order to optimize memory as the reward variable [39]. Outliers in the CPU usage data cause the Drone optimizing function to have fewer safe actions, leading to increased scaling, which impacts Drone GP more than DRKF due to the Ksurf attention mechanism. The scaling of pods increases resource variability, the reduced scaling activity of DRKF is associated with reduced CPU and memory usage variance across all nodes as shown in Table 1 and Table 2. For the Kafka application under test, reduced pod scaling activity leads to significantly lower CPU usage mean and variance due to worker pod application being more CPU than memory bottlenecked with CPU utilization 63% $\gg$ 6% memory utilization.

5.2.2 Master and Scaler Node Resource Footprint. DRKF has a 1.46% higher memory footprint on the scaler nodes because Ksurf is more memory intensive than GP due to the transformer network attention layers. The DRKF CPU footprint is lower due to Ksurf being more compute efficient leading 0.092% lower CPU usage. Ksurf has

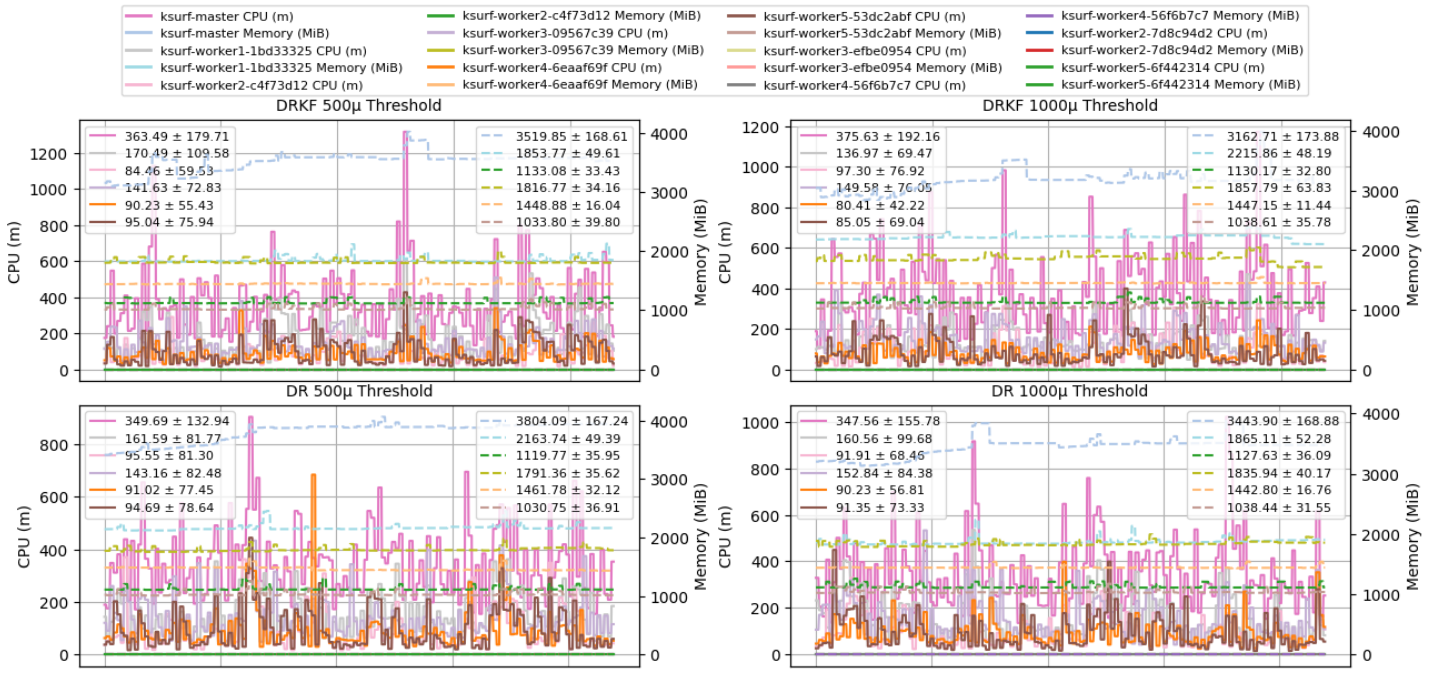

Figure 5: Drone Node Metrics at 1 Millipod Threshold

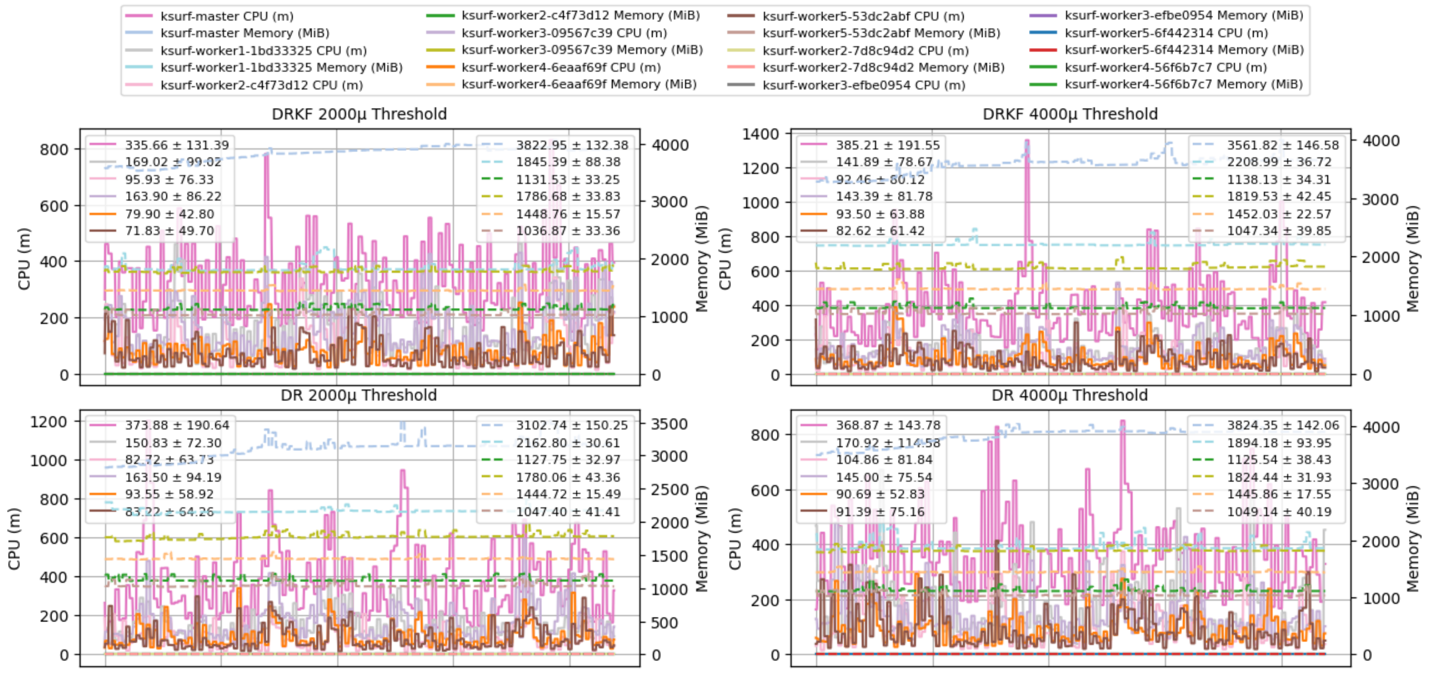

Figure 6: Drone Node Metrics at 2 & 4 Millipod Thresholds

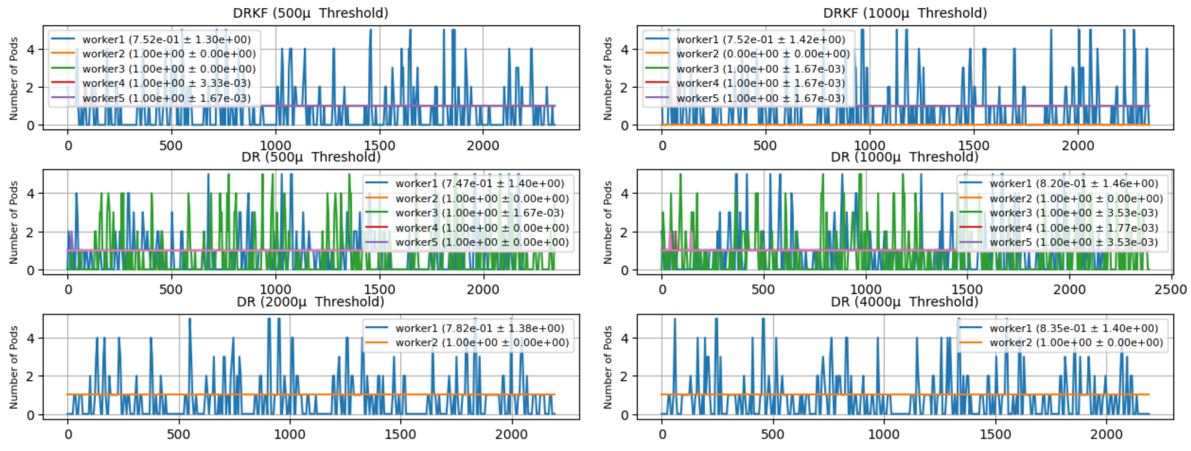

Figure 7: Kafka Worker Pod Counts

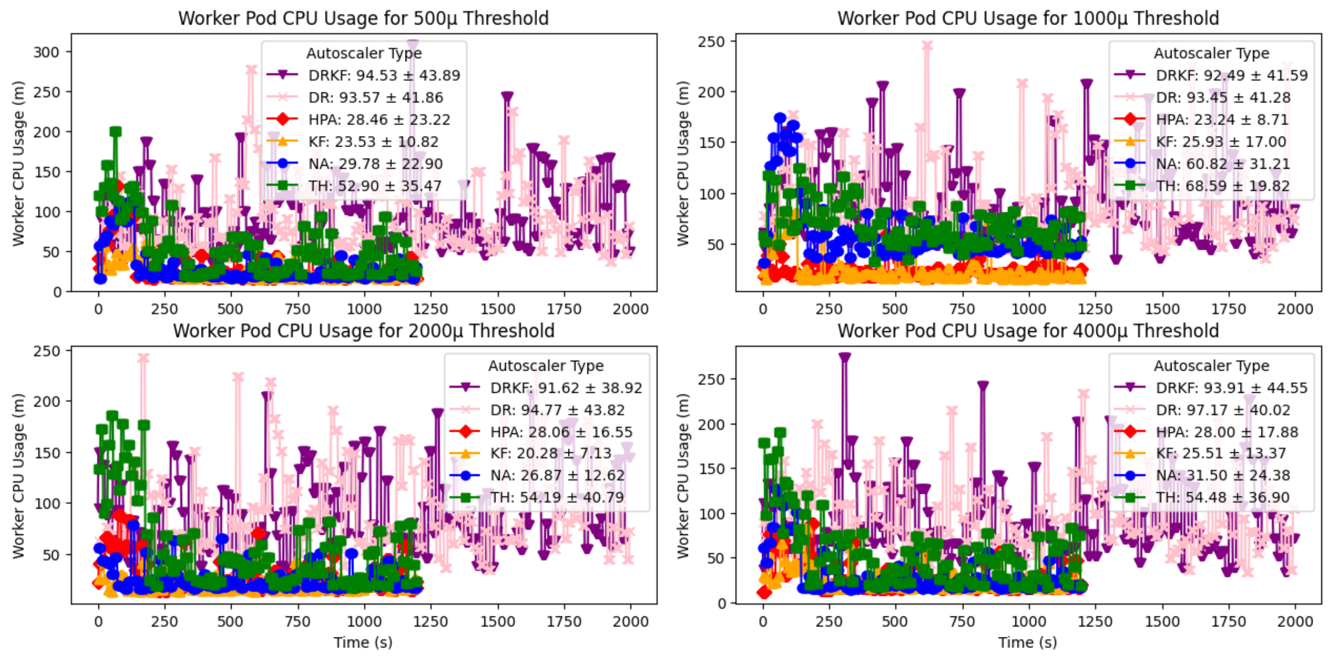

Figure 8: Kafka Worker Pod CPU Usage By Threshold

$O(n^3) \gg O(LN^2d)$ time complexity gain and $O(n^2) \gg O(N^2)$ memory complexity gain since GP batch size n grows at each time step and the Ksurf data size N is fixed $n \gg N$, however the memory used may be higher for Ksurf when the underlying implementation is operating in batch mode [13] [33]. This is a reasonable cost with regards to the significant 3.81% CPU usage reduction on the worker pods. The key lesson is that online methods including Ksurf are better suited to real-time time series inference than batch learning algorithms as demonstrated by GP.

6 Conclusions

In this work, the Drone+Ksurf orchestrator is introduced to tackle the problem of container orchestration under conditions of high variability. Drone+Ksurf is rigorously compared to Drone and baseline auto-scaler algorithms on a set of experiments using the $VarBench$ benchmark. Ksurf uses a pre-trained attention layer and principal component analysis to mitigate outliers and noise in time series data [34] [17]. A key benefit of Drone is its use of a non-parametric surrogate objective function GP, which makes it easier to implement and deploy, and provides robustness to user error introduced when selecting initial parameters.

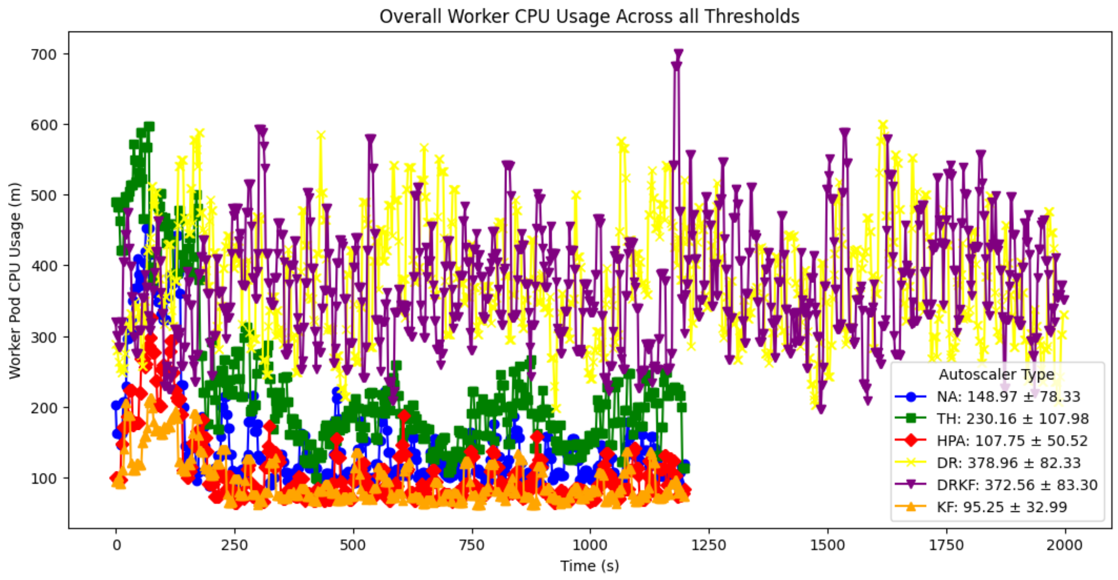

Figure 9: Kafka Mean Worker Pod CPU Usage For All Thresholds

The $VarBench$ benchmark experiments show the impact of the Ksurf predictor on the orchestration performance of Drone. Although the Kubernetes scaler node hosting Drone showed 1.48% increased memory usage, the CPU usage was slightly lower 0.09% with a 90% reduction in CPU usage variance on the Drone host. The worker pods demonstrated a significant reduction in mean CPU usage of 3.59% and a corresponding reduction in CPU usage variance of 7.18%. The reductions in memory and CPU usage variance reduce the probability over- and under-provisioning Drone orchestration environments. This is captured in a 6.89% reduction in average worker pod count for Drone+Ksurf. The reduction in worker pod CPU usage demonstrates significant positive impact of Ksurf on the orchestration quality of Drone by increasing robustness to variability in resource metrics used for orchestration decisions. The key lesson is that online estimation methods such as Ksurf are more accurate than batch learning methods such as GP on real-time short horizon time series inference tasks. Also, the scaling period of the HPA, TH and KF auto-scalers defined by the worker CPU usage convergence duration after scaling actions is approximately half as long, indicating the effect of the higher computation time of each iteration of Drone.

Future work should involve adopting Ksurf for graphical inference such as using graph neural networks to complete model parameters for Ksurf by computing initial Kalman gain as a graph filter that minimizes prediction error when estimating the hidden states of the microservice dependency graph structure using partial measurements [5] [28]. The graphical Ksurf would be instrumental in addressing a key limitation in Drone for microservice-oriented resource management [39].

References

[1] O. Alipourfard et al. "{CherryPick}: Adaptively unearthing the best cloud configurations for big data analytics". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 2017, pp. 469–482.
[2] "Amazon EC2 Spot Instances Pricing– Amazon Web Services". In: AWS, 2023. URL: https://aws.amazon.com/ec2/spot/pricing/.
[3] *Apache Kafka*. http://kafka.apache.org/. Version 1.29, Accessed: 2024-01-24. 2024.
[4] M.-A. Bandits. "Introduction to Multi-Armed Bandits". In: ().
[5] I. Buchnik et al. "GSP-KalmanNet: Tracking Graph Signals via Neural-Aided Kalman Filtering". In: *IEEE Transactions on Signal Processing* 72 (2024), pp. 3700–3716.
[6] "Burstable performance instances– Amazon Web Services". In: AWS, 2023. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/.
[7] "Compute Canada". In: Digital Alliance Canada, 2025. URL: https://www.alliancecan.ca/en.
[8] M. Dang'ana. "AKF-PCA Source Code". In: *Github* (2025). URL: https://github.com/msrg/kfpca.
[9] M. Dang'ana and H.-A. Jacobsen. "Ksurf: Attention Kalman Filter and Principal Component Analysis for Prediction under Highly Variable Cloud Workloads". In: *11th International Conference on Electrical Engineering, Computer Science and Informatics*. 2024, pp. 302–308. DOI: 10.1109/EECSI63442.2024.10776180.
[10] J. Dean and L. A. Barroso. "The tail at scale". In: *Communications of the ACM* 56.2 (2013), pp. 74–80.
[11] D. Duvenaud. "Automatic Model Construction with Gaussian Processes". PhD thesis. University of Cambridge, 2014.
[12] S. Eismann et al. "Microservices: A performance tester's dream or nightmare?" In: *Proceedings of the ACM/SPEC international conference on performance engineering*. 2020, pp. 138–149.
[13] K. Gregor et al. "Draw: A recurrent neural network for image generation". In: *International conference on machine learning*. PMLR. 2015, pp. 1462–1471.
[14] J. Hartikainen and S. Särkkä. "Kalman filtering and smoothing solutions to temporal Gaussian process regression models". In: *2010 IEEE International Workshop on Machine Learning for Signal Processing*. 2010, pp. 379–384. DOI: 10.1109/MLSP.2010.5589113.
[15] H. D. Kabir et al. "Uncertainty-aware decisions in cloud computing: Foundations and future directions". In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–30.
[16] C. Ke et al. "ProScale: Proactive Autoscaling for Microservice With Time-Varying Workload at the Edge". In: *IEEE Trans. on Parallel and Distributed Systems* 34.4 (2023), pp. 1294–1312. DOI: 10.1109/TPDS.2023.3238429.

[17] Y. Kim et al. “Structured attention networks”. In: *International Conference on Learning Representations* (2017). URL: https://console.cloud.google.com/storage/browser/external-traces/charlie.

[18] *Kubernetes: Production-Grade Container Orchestration.* https://kubernetes.io/. Version 1.29, Accessed: 2024-01-24. 2024.

[19] Q. Li et al. “Kalman Filter and Its Application”. In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS).* 2015, pp. 74–77. DOI: 10.1109/ICINIS.2015.35.

[20] Q. Li et al. “RAMBO: Resource allocation for microservices using Bayesian optimization”. In: *IEEE Computer Architecture Letters* 20.1 (2021), pp. 46–49.

[21] Y. Liu, H. Xu, and W. C. Lau. “Accordia: Adaptive cloud configuration optimization for recurring data-intensive applications”. In: *Proceedings of the ACM Symposium on Cloud Computing.* 2019, pp. 479–479.

[22] T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano. “A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments”. In: *J Grid Computing* 12 (2014), pp. 559–592. DOI: 10.1007/s10723-014-9314-7.

[23] S. Luo et al. “Characterizing microservice dependency and performance: Alibaba trace analysis”. In: *Proceedings of the ACM symposium on cloud computing.* 2021, pp. 412–426.

[24] S. Luo et al. “The power of prediction: microservice auto scaling via workload learning”. In: *Proceedings of the 13th Symposium on Cloud Computing.* 2022, pp. 355–369.

[25] F. Opolka et al. “Adaptive gaussian processes on graphs via spectral graph wavelets”. In: *International Conference on Artificial Intelligence and Statistics.* PMLR. 2022, pp. 4818–4834.

[26] “Prometheus”. In: SoundCloud, 2025. URL: https://prometheus.io.

[27] “Resource Management for Pods and Containers | Kubernetes”. In: Kubernetes, 2025. URL: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/.

[28] G. Revach. et al. “KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics”. In: *IEEE Trans. on Signal Processing* 70 (2022), pp. 1532–1547. DOI: 10.1109/TSP.2022.3158588.

[29] D. M. Roijers et al. “Interactive Multi-objective Reinforcement Learning in Multi-armed Bandits with Gaussian Process Utility Models”. In: *Machine Learning and Knowledge Discovery in Databases.* Ed. by F. Hutter et al. Cham: Springer International Publishing, 2021, pp. 463–478.

[30] J. Shi et al. “Clash of the titans: Mapreduce vs. spark for large scale data analytics”. In: *Proceedings of the VLDB Endowment* 8.13 (2015), pp. 2110–2121.

[31] H. Shigeki et al. “Web server access trend analysis based on the Poisson distribution”. In: *Proceedings of the 6th International Conference on Software and Computer Applications.* 2017, pp. 256–261. ISBN: 9781450348577. DOI: 10.1145/3056662.3056701.

[32] *Twitter Developer API hashtag #Ukraine.* https://developer.twitter.com. Accessed: 2023-05-24.

[33] J. Wenger et al. “Computation-aware gaussian processes: Model selection and linear-time inference”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 31316–31349.

[34] S. Wold, K. Esbensen, and P. Geladi. “Principal Component Analysis”. In: *Chemometrics and Intelligent Laboratory Systems* 2 (1987), pp. 37–52.

[35] G. e. a. Yu. “Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach”. In: *IEEE Trans. on Cloud Computing* 10.2 (2020), pp. 1100–1116.

[36] H. Zhang et al. “Contextual Gaussian Process Bandits with Neural Networks”. In: *Advances in Neural Information Processing Systems.* Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 26950–26965. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/5526c73e3ff4f2a34009e13d15f52fcb-Paper-Conference.pdf.

[37] H. Zhang et al. “Contextual Gaussian process bandits with neural networks”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 26950–26965.

[38] J. Zhang. “Improvement and application of Extended Kalman filter algorithm in target tracking”. In: *Modern Computer* 32 (2012), pp. 11–16.

[39] Y. Zhang et al. “Lifting the fog of uncertainties: Dynamic resource orchestration for the containerized cloud”. In: *Proceedings Of The 2023 ACM Symposium On Cloud Computing.* 2023, pp. 48–64.