International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 09 | Sep 2023 www.irjet.net p-ISSN: 2395-0072

Forecasting Capacity Issues in Stateful Systems: A Proactive Approach

Anuj Phadke, Parth Santpurkar, Meenakshi Jindal

[1]*Senior Software Engineer, Netflix, USA*
[2]*Senior Software Engineer, Netflix, USA*
[3] *Senior Software Engineer, Netflix, USA*

Abstract - *Managing capacity in systems operating at scale poses significant challenges, making it difficult to proactively plan for potential capacity issues. Scaling critical systems in response to capacity limitations entails risks and can lead to stressful situations. To address this concern, this paper presents a novel forecasting system designed to proactively predict capacity issues. By adopting this proactive approach, organizations can mitigate the likelihood of encountering such situations and ensure the seamless performance of their stateful systems. The proposed forecasting system offers valuable insights, enabling timely resource allocation and efficient management to maintain optimal system operations.*

Keywords: **Forecasting, Facebook Prophet, Time series, Capacity management, Pre-emptive scaling**

1. INTRODUCTION

In the contemporary digital landscape, the operation of large-scale systems has become an integral part of modern business and technology ecosystems. These systems, often characterized by their complexity and scale, underpin critical services, from cloud computing platforms to e-commerce websites, from financial institutions to social media networks. Ensuring the uninterrupted performance of these systems is not merely a matter of operational efficiency; it is a strategic imperative for organizations worldwide.

1.1 Importance of Capacity Planning

Capacity planning is very critical for organizations that operate at scale. Here are some of the reasons why capacity planning is critical:

1. Optimal resource allocation: Capacity planning guarantees that systems have adequate computing power, storage capacity, network bandwidth, and CPU resources available, thereby preventing overprovisioning or underutilization of these resources.

2. Cost: Running systems at scale can lead to substantial infrastructure costs. However, by implementing effective capacity planning, organizations can control and manage these costs efficiently.

3. Performance: Effective capacity planning ensures that one plans and scales the systems for peak usage. This ensures that your systems remain performant even during peak periods.

4. Handling seasonal spikes: It is essential to consider occasional spikes and seasonal fluctuations in traffic, such as increased activity during holidays, when designing your system.

5. Meeting SLAs: Adequate provisioning of systems is necessary to ensure they meet user SLAs, such as latency and throughput requirements.

Large-scale automated systems, whether in a datacenter or the cloud, can encompass thousands of nodes. These nodes generate diverse time-series metrics, such as disk usage, memory usage, and total network traffic, which provide operational insights. It can be very difficult to manually track this data and identify patterns and look for capacity issues before it happens. In summary, capacity planning is not merely a technical exercise but a strategic imperative for organizations. It ensures that resources are allocated efficiently, risks are mitigated, and the organization is well-prepared to adapt to changing circumstances and demands, ultimately contributing to overall success and sustainability.

1.2 Literature Survey

Sanjeev Vijaykumar et al. [1] developed workload forecasting using neural network and artificial lizard search optimization. They conducted experiments using a benchmark of Google cluster trace. E.G.Radhika et al., [2] developed forecasting techniques to autoscale web applications using auto-regressive integrated moving averages and Recurrent neural network-long short-term memory (RNN-LSTM) techniques. They found that RNN-LSTM gives a lower error rate compared to using ARIMA. Yexi Jiang et al., [3] proposed an intelligent cloud capacity management system using IBM smart cloud trace data. They used the ensemble method for forecasting. M.S. Aslanpour et al., [4] developed the proactive auto-scaling algorithm PASA with heuristic predictor and ran the simulations in cloudSim. There are various other models developed for forecasting cloud resources, but they do not reflect actual user interactions [5] that happen in the real world, and all are designed to work in the cloud.

2. Proposal

We propose a generic system that would adapt according to usage pattern changes, resources usage and forecast any capacity issues beforehand. This system can work in the cloud as well as in non-cloud deployments.

2.1 Architecture Diagram

In the diagram below, we show the overall architecture of the system we are proposing:

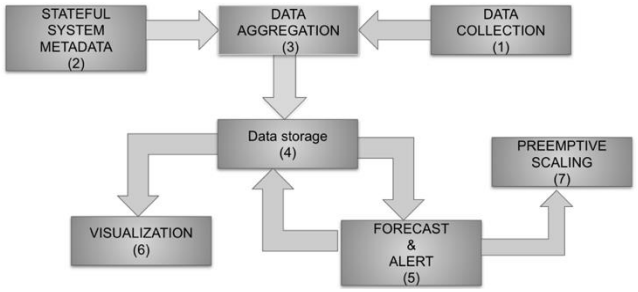

Fig 1: *Proposed Forecasting system*

3. The entire system can be broken down into following 7 components

3.1 Data Collection (telemetry)

In order to discern the typical trends of a metric within a stateful system, it becomes imperative to establish a means of acquiring the pertinent metrics for predictive analysis. This can be accomplished either by leveraging an existing telemetry system or by configuring a tailored agent designed to collect the precise metrics required for forecasting purposes. Frequently, summarized metrics like minimum or maximum values, collected over extended intervals such as daily or hourly, prove sufficient for forecasting trends over time.

To identify the general pattern of a metric in a stateful system, it is essential to have a method for capturing the desired metrics for prediction. This can be achieved by utilizing an already deployed telemetry system or setting up a custom agent to gather the specific metrics needed for forecasting. In many instances, aggregated values (such as minimum or maximum) collected over an extended period (daily or hourly) suffice to predict the trend over time.

3.2 Stateful system metadata

This system is used to store metadata like SLOs and thresholds for the systems.

3.3 Data Aggregation

Data collection and collation is one of the most important parts of this entire process. Without good and reliable data, we cannot have good predictions. The method of aggregation over the data is just as important. For a given metric, we can have hundreds of time series for a stateful system which is scaled up anywhere from a few single digit nodes to hundreds of nodes. We tested various data aggregation methods to come up with a single reliable time series to feed into the forecaster. We found that the aggregation that gave us the most accurate prediction results over a period of time was to take the mean of the metric, for which we have the max of the timeseries for every given node in the system. This produced the most actionable results for us, rather than just taking the average or the max across the entire system. This stage joins data from the telemetry data from the data collection stage and the system metadata.

3.4. Data storage

The data gathered from the data collection processes undergoes an ETL (Extract, Transform, Load) process and is then consolidated and stored in a centralized database. When the data collection agent operates on multiple nodes within a distributed system, centralizing the data enables straightforward querying for the forecasting process.

3.5. Forecasting and alerting

We used the Facebook Prophet system in our proposed system to predict metric values. Prophet exhibits various features that are useful for forecasting metrics for a real-world application:

Seasonality detection - Prophet can automatically detect and handle various types of seasonal patterns, including daily, weekly, and yearly seasonality.

- Holiday effect - It allows you to include custom holiday effects that might impact your time series data.
- Capture trend - This allows you to detect upwards or downward trends which is critical to predict the overall usage over time. The algorithm works well in capturing the trend over time and is not very efficient in predicting random intermittent spikes in a metric.
- Uncertainty interval - FB prophet provides uncertainty intervals for the forecast that helps you understand the range of the predictions.
- Outlier detection - It is very robust in handling missing data and outliers.
- Scalable - The algorithm works very well with large datasets.

FBProphet [6] uses an additive model, which means that the components are added together to form the time series. The

model then employs a Bayesian approach to fit the components to the observed data. The basic equation for forecasting algorithm is as follows:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

where:

y(t): Observed value at time interval (t)

s(t): Seasonal component

h(t): Holiday effect

ε(t): Noise

The trend component i.e. g(t) can be modelled using the growth trend or the logistic growth trend. The seasonal component s(t) includes Fourier series terms to capture seasonalities. The holiday effect h(t) allows you to incorporate the effect of holidays on the time series data. By identifying and accounting for these repetitive patterns, you can better capture the underlying structure of the data and improve the accuracy of your predictive models.

Each metric that we generate for any stateful system was combined into a single time series and fed to the Prophet library to generate a future time series. Generated forecasts are fed back again to the data storage layer.

3.5. Visualization

We used the forecasted values for building visualizations and dashboards to visualize the predicted metric usage in the future.

3.6 Pre-Emptive Scaling

For any given deployment of a stateful system, for the forecasted metric to be actionable, the metric needs to have some threshold. This threshold is important to be defined and set correctly for the pre-emptive scaling to be effective. We regularly run standard benchmarks against the datastores and fine tune the thresholds as needed.

Once we have the forecasted time series, we compare each individual data point to figure out if the thresholds will be met or crossed anytime in the future. If we find that the thresholds are crossed, then we trigger a notification and a downstream system that takes care of scaling up the given stateful system (outside the scope of this paper).

4. Testing the system

We currently have this system in production, and it looks at hundreds of stateful clusters across various datastore types (Cassandra, Elasticsearch etc.). For all clusters we have predefined thresholds and a system in place for data collection, aggregation and ingestion into the forecaster. Fig 1. Below is an example of the graph we generate for the admin of our systems to verify that the predictions are accurate and how soon we can expect a datastore to hit our predefined thresholds.

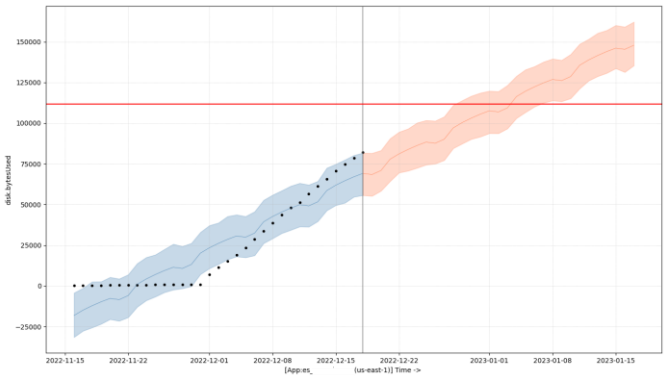

Fig 2: *A graph showing the increase of disk space plotted over time. Blue portion is the aggregated data points. The Orange portion are the forecasts with upper and lower bounds. The horizontal red line is the predefined threshold which we would like to avoid reaching.*

5. Case Study: Proactive Capacity Management in Large-Scale Machine Learning Environments

In this section, we delve into a real-world case study that exemplifies the critical importance of proactive capacity management in large-scale systems. Our case study revolves around a comprehensive database of machine learning (ML) results, a high-demand computing environment where accurate capacity planning is paramount.

6.1 Background

In the realm of machine learning research and development, the demand for computational resources has surged in recent years. Training complex neural networks, processing large datasets, and fine-tuning models require substantial computational power. This led to the creation of a dedicated cluster for ML experiments, serving researchers across the organization. However, the dynamic nature of ML workloads made it increasingly challenging to predict and manage capacity effectively.

6.2 Challenges in Storage and Computation

In addition to the challenges previously mentioned, this ML environment faced unique challenges related to storage of training datasets:

- Large Datasets: ML experiments often require access to extensive datasets, sometimes spanning terabytes of data. Storing and managing these datasets efficiently while ensuring fast access for training jobs was a significant challenge.

- Data Transfer Bottlenecks: Moving large datasets to and from storage systems could lead to network bottlenecks and slow down job execution, affecting overall system performance.

- Data Versioning: Maintaining version control for datasets was crucial to ensure reproducibility in research. This introduced complexity in data storage and tracking.

6.3 The Proactive Forecasting Solution

To address these challenges, we extended the proactive capacity forecasting system introduced in this paper to encompass storage and data-related aspects. This system was configured not only to predict compute resource needs but also to anticipate data storage requirements for upcoming ML experiments.

7. Results and Impact

The holistic approach of the forecasting system, covering both compute and data aspects, yielded significant benefits:

- Optimized Data Management: Predictive analysis allowed for proactive storage allocation based on upcoming job requirements. This reduced data transfer bottlenecks and improved data access times.

- Cost-Efficient Storage: With precise predictions, storage resources were allocated efficiently, reducing storage costs associated with over-provisioning.

- Enhanced Data Version Control: The forecasting system integrated version control for datasets, streamlining data management and ensuring reproducibility.

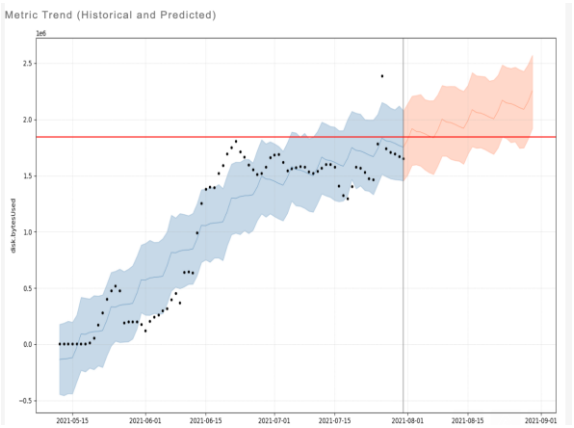

Fig 3: *A graph showing the increase of disk space plotted over time where the system detected that the disk usage would exceed the threshold 30 days in advance.*

8. CONCLUSIONS

We have demonstrated a method for pre-emptive scaling of stateful systems which can be applied to any datastore deployed across any environment (local, public cloud, non-public cloud etc.). This system was successfully able detect capacity issues for more than 500 stateful systems at scale with an accuracy of almost 86% over 2 years.

REFERENCES

[1] Sanjeev Vijayakumar, Jitendra Kumar2, "Cloud Resource Usage Forecasting using NeuralNetwork and Artificial Lizard Search Optimization"

[2] E.G.Radhika, G.S.Sadasivam, and J.F.Naomi , "An Efficient Predictive technique to Autoscale the Resources for Web applications in Private cloud"

[3] Yexi Jiang, Chang-Shing Perng†, Tao Li∗, Rong Chang†, "Intelligent Cloud Capacity Management"

[4] M. S. Aslanpour and S. E. Dashti, "Proactive auto-scaling algorithm (PASA) for cloud application,'' Int. J. Grid High Perform. Comput., vol. 9, no. 3, pp. 1–16, Jul. 2017.

[5] MOHAMED SAMIR , KHALED T. WASSIF , AND SOHA H. MAKADY, "Proactive Auto-Scaling Approach of Production Applications Using an Ensemble Model"

[6] Taylor SJ, Letham B. 2017. Forecasting at Scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2

9. BIOGRAPHIES

1. Anuj Phadke is Senior Software Engineer at Netflix. He received his Master's degree in Computer Engineer from Stony Brook University. His areas of interest include distributed systems and databases.

2. Parth Santpurkar is a Senior Software Engineer at Netflix. He received his Master's Degree in Information Assurance from Northeastern University. His primary areas of interest are Distributed systems and Software Engineering.

3. Meenakshi Jindal is a seasoned software engineer with experience designing software solutions across multiple domains, including banking, insurance, travel, and media. She specializes in designing high-performance, scalable, and reliable distributed systems