LOAD-AWARE SMART CONSUMER FOR ADAPTIVE RESOURCE MANAGEMENT

Sai Guruju, Abhishek Niranjan, Krishnachaitanya Gogineni, Jithendra Vepa

Observe.AI

ABSTRACT

In modern systems, especially those dealing with multimedia content like audios and videos, managing varying workloads efficiently is crucial for maintaining performance and cost-effectiveness. This paper presents a Load-Aware Smart Consumer system designed to handle the transcription of customer-agent conversations, a process that is often hampered by the wide variance in the duration of audio calls. Our system dynamically adjusts the concurrency of processing based on the current load, thereby ensuring stability and efficient resource utilization. By monitoring the processing load across instance-workers, the system can make informed decisions about accepting and processing new tasks, leading to improved resilience and cost savings. This approach is not limited to transcription engines but can be applied to any multimedia processing system facing similar challenges of input variability and resource constraints.

KEYWORDS

high latency systems, concurrency control, cost-efficiency, stability, dynamic resource allocation, multimedia processing

1. INTRODUCTION

In the realm of multimedia processing systems, services designed to process variable-sized inputs, such as multimedia content in call centers, face a set of unique and complex challenges. These systems, which include Automatic Speech Recognition (ASR), Audio Segmentation, Speaker Diarization, and Video Redaction for example, must be equipped to handle a wide range of input sizes, from short audio clips to hours-long recordings. The variability in input size can significantly impact resource utilization, system stability, and overall performance, leading to increased latency and higher resource consumption for larger files.

Traditional approaches to managing these systems often struggle to balance the trade-offs between processing efficiency, cost-effectiveness, and maintaining a high quality of service. In the context of ASR, for example, the transcription of long recordings requires substantial computational resources, which can strain the system and lead to delays. Similarly, audio segmentation and speaker diarization tasks become more complex and resource-intensive as the length of the audio increases. Video redaction, with its need for analyzing and processing visual and audio components, also faces challenges in handling large video files efficiently.

To effectively address these challenges, it is essential to develop innovative solutions that can dynamically adapt to the variability of input sizes while optimizing resource usage. In this paper,

David C. Wyld et al. (Eds): AdNLP, CDKP, NCO, SAI, MLT, Signal, SOFT, ICAITA, CMC - 2024
pp. 205-212, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141018

we introduce a novel Load-Aware Smart Consumer (LASC) system, which exemplifies such a solution. By intelligently adjusting concurrency levels based on the current workload, our system ensures that resources are allocated and used efficiently, leading to cost savings, improved system resilience, and enhanced performance.

The concepts and methodologies presented in this paper, while illustrated through the lens of a transcription engine for call centers, are broadly applicable to a variety of software systems dealing with similar challenges. These systems include, but are not limited to, video processing platforms, data analytics pipelines, and real-time communication tools. Our approach demonstrates the potential for significant improvements in both performance and cost-efficiency, making it a valuable contribution to the field of software engineering.

Furthermore, this paper delves into the technical details of the Load-Aware Smart Consumer system, including its architecture, algorithms, and implementation strategies. We also present a comprehensive evaluation of the system's performance, showcasing its effectiveness in real-world scenarios. Through this exploration, we aim to provide insights and guidance for software engineers and system designers seeking to enhance the adaptability and efficiency of their systems.

2. RELATED WORK

The challenges associated with processing variable-sized inputs in software systems have been a topic of interest in the field of software engineering for several years. Several studies and approaches have been proposed to address these challenges, particularly in the domains of multimedia processing and real-time data analysis.

Dynamic Resource Allocation: Research in dynamic resource allocation has explored various strategies for adapting resource usage in response to changing workloads. For example, Jayanthi S (2014) proposed a dynamic resource scheduling algorithm for cloud-based multimedia processing systems, which adjusts the allocation of cloud computing resources based on strategies with respect to Quality Of Service (QOS), efficiency, performance, and cost.

Load Balancing in Distributed Systems: Load balancing techniques have been widely studied to distribute workloads evenly across multiple processing units. Sakshi et al (2022) developed a load balancing mechanism that first analyzes the resource requirements such as CPU, Memory, Energy and Bandwidth usage and allocates an appropriate number of VMs for each application. Second, the resource information is collected and updated where resources are sorted into four queues according to the loads of resources i.e. CPU intensive, Memory intensive, Energy intensive and Bandwidth intensive. This method demonstarates that SLA-aware scheduling not only facilitates the cloud consumers by resources availability and improves throughput, response time etc. but also maximizes the cloud profits with less resource utilization and SLA (Service Level Agreement) violation penalties

Scalable Architectures for Video Processing: Scalable architectures have been proposed to handle the processing of large video files efficiently. For instance, Kim et al. (2018) introduced a scalable video processing framework that dynamically adjusts the number of processing nodes based on the size and complexity of the input video, ensuring efficient resource usage and reduced latency.

The Load-Aware Smart Consumer system proposed in this paper builds upon these previous works by providing a comprehensive solution that dynamically adapts to variable input sizes across a range of software systems. By integrating concepts from dynamic resource allocation,

load balancing, and scalable architectures, our approach offers a novel and efficient way to manage the challenges associated with processing multimedia content and other variable-sized inputs. The final solution has been adapted in multiple internal services at Observe.AI like our Audio Segmentation module described in [8] and Transcript Embeddings model detailed in [7] where the length of the audio file considerably alter the processing requirements per request; Sentiment detection module described in [10] and Silence detection module described in [9] where the length of the transcription considerably affect the processing requirements per request.

3. ARCHITECTURE

In this section, we provide the architecture details:

1. *ASR Components:* description about components that form an automatic speech recognition (ASR) system, how they are deployed
2. *Problem:* highlight stability issues that plague the system due to wide variance in audio duration lengths
3. *Load-Aware Smart Consumer*: our solution to ensure both stability and cost-effectiveness

While the focus of this paper is on the application of the Load-Aware Smart Consumer (LASC) pattern to Automatic Speech Recognition (ASR) systems, it is important to acknowledge the inherent generality and versatility of the LASC system. The architectural principles and resource management strategies proposed herein have been proven effective across various systems characterized by high variability in processing requirements at the request level. These include, but are not limited to, video processing, sentiment detection and LLM networks. The LASC system is designed with a generic framework in mind, adaptable to a wide spectrum of distributed computing environments where workload unpredictability is a central concern. However, for the purpose of providing a tangible and detailed context, this paper will concentrate on its implementation within an ASR system, highlighting how LASC successfully contributes to enhancing stability and cost-efficiency in such a complex processing scenario.

3.1. ASR Components

The transcription engine is a key component of our system, responsible for converting audio streams into text. The engine's architecture is designed to handle asynchronous processing of audio files, ensuring scalability and efficient resource utilization.

Processing Pipeline: The transcription process begins by polling a request message from a queue, which contains information about the audio file to be processed. The audio file is then downloaded from cloud storage, and its speech segments are identified using Voice Activity Detection (VAD). These segments are packed into batches and passed through a series of pre-processing steps to prepare them for transcription.

The core of the transcription process involves an inference stage, where encoder and decoder models are used to convert the audio segments into text. Finally, the output is post-processed to generate a complete transcript of the audio file.

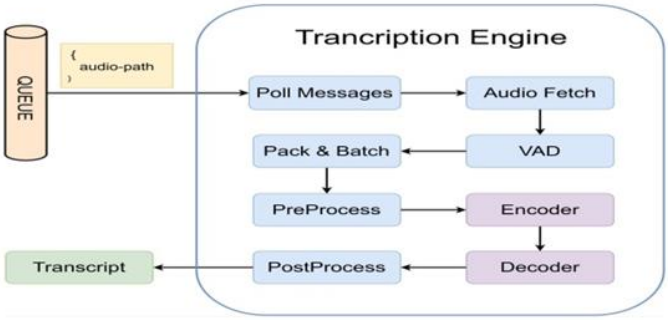

Figure 1: Architectural diagram for Transcription Engine

Parallel Processing: To achieve high throughput, the engine is designed to process multiple audio files in parallel. This is accomplished by deploying multiple instances of the transcription engine, each capable of handling its own set of request messages. The system's scalability is further enhanced by hosting it on cloud instances equipped with NVIDIA-T4 GPUs, which provide the necessary computational power to handle the transcription workload.

Dynamic Scaling: The system is capable of scaling horizontally by adding more instances as the demand increases. This ensures that the transcription engine can maintain high performance levels even during peak loads, without experiencing significant delays or downtime.

3.2. Problem

In distributed systems, particularly those dealing with multimedia processing, managing variable workloads poses significant challenges. In our context, the transcription engine must handle audio recordings of varying lengths, from a few seconds to several hours. This variability leads to uneven resource demands, with longer recordings requiring more computational resources compared to shorter ones.

When multiple consumers within the same instance simultaneously process a surge of longer calls, it can result in resource contention, leading to request timeouts or server overload. This not only impacts system stability but also leads to downtime, affecting the overall throughput and efficiency of the system.

One approach to mitigate this issue is to reduce worker concurrency to a level where even long calls can be processed smoothly. However, this solution drastically reduces throughput for shorter calls, resulting in underutilization of computational resources. Alternatively, upgrading to more powerful hardware can increase concurrency levels for longer calls, but this too leads to inefficiencies as shorter calls can now be processed at much higher concurrency levels, again leading to resource underutilization.

These challenges highlight the need for a more dynamic and adaptive approach to resource management in distributed systems dealing with variable workloads.

3.3. Solution : Load-Aware Smart Consumer

Effective resource management in a distributed system requires a dynamic approach to handle variable workloads. Our solution, the Load-Aware Smart Consumer (LASC), addresses this by intelligently adjusting concurrency levels based on the current load and the characteristics of

incoming tasks. This approach ensures both stability and cost-effectiveness, optimizing resource utilization while maintaining high throughput.

Dynamic Concurrency Adjustment: The system considers several factors to determine the resource requirements for processing audio files, such as the duration of the call, sampling rate, and the number of channels. Based on these indicators, the system dynamically adjusts concurrency levels, allowing for higher concurrency for shorter calls or tasks with lower compute needs, and lower concurrency for longer calls. This adaptive mechanism prevents resource contention and ensures that each task is processed efficiently without overloading the system.

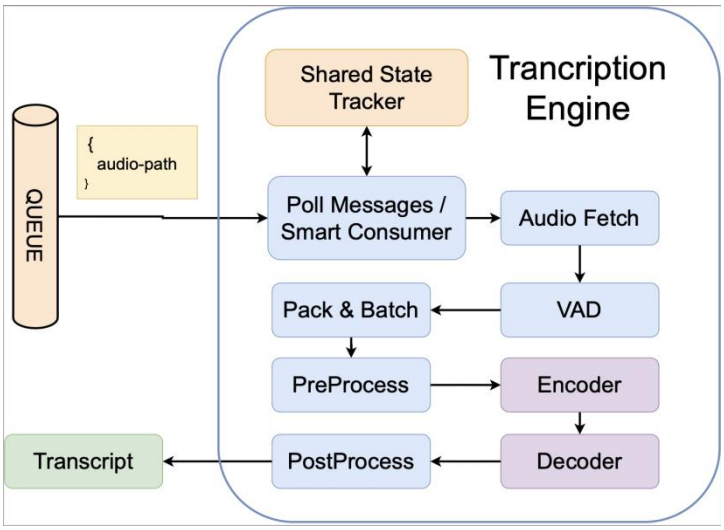

Figure 2: Transcription Engine with Smart Consumer

Shared State Tracker: A crucial component of our solution is a shared state tracker, which monitors the load being processed across workers in an instance. This tracker is implemented by extending the functionalities of Python's managers. Base Manager from the multiprocessing package. It provides a centralized view of the system's load, enabling real-time adjustments to the concurrency levels based on the current workload. By maintaining a shared state, the system can make informed decisions about resource allocation, ensuring balanced and efficient utilization.

System Architecture: Figure 2 illustrates the architecture of the transcription engine with the Load-Aware Smart Consumer components integrated. The system is designed to process different call lengths efficiently by dynamically adjusting the load across workers. The architecture includes the following key components:

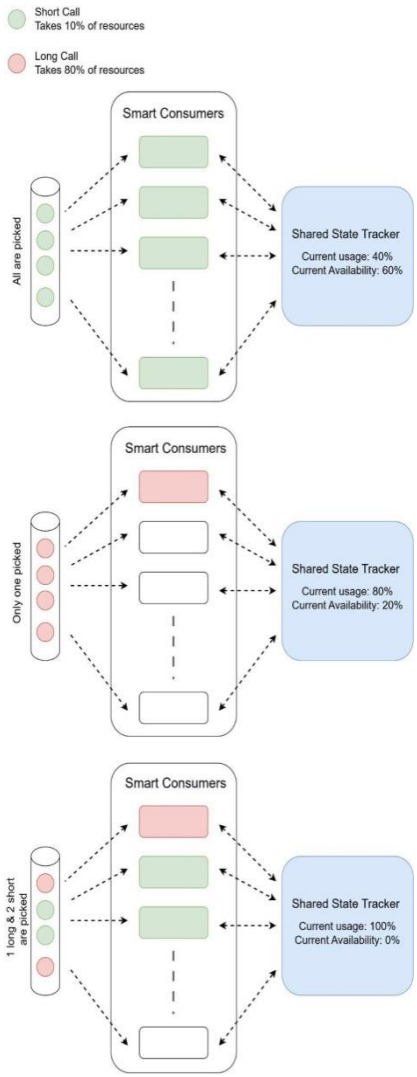

Figure 3: Processing with Load-Aware Smart Consumer under different scenarios

3.4. Impact and Metrics

After deploying the Load-Aware Smart Consumer system as a wrapper on multiple services, including automatic speech recognition, audio segmentation, and diarization, we observed significant positive trends across various dimensions. The system's impact was particularly pronounced in two key areas:

1. **Service Downtime Reduction:** We observed a significant reduction in service downtime by a factor of almost 8x. This improvement indicates the system's ability to maintain operational continuity and enhance overall service reliability.
2. **Cost Reduction:** For high variance workloads, the system achieved a cost reduction of 45% compared to choosing a high availability configuration with low processing concurrency. This cost savings highlights the system's efficiency in resource utilization and cost-effectiveness.

These results demonstrate the effectiveness of the Load-Aware Smart Consumer system in improving service reliability and reducing operational costs without compromise, making it a

valuable addition to multimedia processing systems facing similar challenges of input variability and resource constraints

4. CONCLUSIONS AND OTHER APPLICATIONS

In this paper, we have presented a solution to address the challenges of managing variable workloads in distributed systems, particularly in the context of multimedia processing. The Load-Aware Smart Consumer system dynamically adjusts concurrency levels based on the current load and the characteristics of incoming tasks, ensuring efficient resource utilization and system stability.

The shared state tracker plays a crucial role in this system, enabling real-time monitoring of resource availability and facilitating smart decision-making for task processing. This approach has enabled us to process millions of hours of audio files on a monthly basis in a cost-effective and resilient manner, with a transcription cost of less than 1 cent per hour of audio and no server downtime.

The concept of the Load-Aware Smart Consumer is not limited to ASR systems. It can be applied to other multimedia processing tasks, such as speaker diarization and audio segmentation, which also exhibit variability in input sizes. Moreover, this approach can be extended to video processing systems, where the length of the video significantly impacts resource requirements. Overall, the Load-Aware Smart Consumer concept demonstrates a scalable and adaptable solution for managing variable workloads in distributed systems, with potential applications across various domains in software engineering.

REFERENCES

[1] NVIDIA T4 GPU, https://www.nvidia.com/en- in/data- center/ tesla- t4/.
[2] Python multiprocessing.managers library/multiprocessing.html#managers.
[3] Mao, M., Humphrey, M., & Brewer, D. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11).
[4] Kim, Yoon-Ki & Kim, Yongsung & Jeong, Chang-Sung. (2018). RIDE: real-time massive image processing platform on distributed environment. EURASIP Journal on Image and Video Processing. 2018. 10.1186/s13640-018-0279-5.
[5] Jayanthi S, "Literature review: Dynamic resource allocation mechanism in cloud computing environment," 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, India, 2014, pp. 279-281, doi: 10.1109/ICECCE.2014.7086627.
[6] Chhabra, Sakshi, and Ashutosh Kumar Singh. "Dynamic resource allocation method for load balance scheduling over cloud data center networks." Journal of Web Engineering 20.8 (2021): 2269-2284.
[7] Nandan, Apoorv, and Jithendra Vepa. "Language agnostic speech embeddings for emotion classification." ICML 2020 Workshop on Self-supervision in Audio and Speech. 2020.
[8] Gogineni, K., Yadama, T.R., Vepa, J. (2021) Audio Segmentation Based Conversational Silence Detection for Contact Center Calls. Proc. Interspeech 2021, 2349-2350
[9] Ingle, D., Kumar, A., Gogineni, K., Vepa, J. (2022) Real-Time Monitoring of Silences in Contact Center Conversations. Proc. Interspeech 2022, 1951-1952
[10] A. Kumar and J. Vepa, "Gated Mechanism for Attention Based Multi Modal Sentiment Analysis," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 4477-4481, doi: 10.1109/ICASSP40776.2020.9053012.

AUTHORS

Sai Guruju is a Lead Engineer at Observe.AI, he works primarily on productionizing complex Machine learning models.

Abhishek Niranjan is a Sr. Engineer at Observe.AI, he specialises in audio machine learning models like ASR and diarization

Krishna Gogineni is a Principal Engineer at Observe.AI, he leads the Platform team and serves as an architect for the company

Jithendra Vepa is the CTO and Co-Founder of Observe.AI.