A DISTRIBUTED FRAMEWORK FOR ENTERPRISE LEVEL SPEECH RECOGNITION SERVICES

I. Arizmendi[1] *and R. C. Rose*[2]

[1]AT&T Labs – Research, Florham Park, NJ
iker@research.att.com

[2]McGill University, Dept. of ECE, Montreal, Canada
rose@ece.mcgill.ca

ABSTRACT

This paper presents methods for improving the efficiency of automatic speech recognition (ASR) decoders in multi-user applications. The methods involve allocating ASR resources to service human-machine dialogs in deployments that make use of many low cost, commodity servers. It is shown that even very simple strategies for efficient allocation of ASR servers to incoming utterances has the potential to double the capacity of a multi-user deployment. This is important because, while there has been a great deal of work applied to increasing the efficiency of individual ASR engines, there has been little effort applied to increasing overall efficiency at peak loads in multi-user scenarios.

1. INTRODUCTION

Many automatic speech recognition applications and services are deployed in multi-user scenarios. In general, the major challenge in designing a framework to support multi-user applications is to develop a system that can scale to serve a large user population while simultaneously minimizing the degradation in quality of service under peak load conditions [2].

Designing multi-user automatic speech recognition applications in particular involves additional issues that distinguish this case from more generic services. These issues arise from the high variability in processing effort that exists for ASR in human-machine dialog scenarios which results from a number of sources. These sources of variability include the infrequent occurrence of user utterances as a fraction of the total length of the human-machine dialog, the high variance in processing effort that exists over time in an utterance, and the high variance in processing effort that exists between different ASR tasks. They will be discussed in more detail in Section 3.

This paper presents efficient methods for allocating ASR resources to service human-machine dialogs in deployments that make use of many low cost, commodity servers. It is shown that exploiting knowledge of the above sources of variability in allocating ASR resources can lead to significant increases in resource utilization and quality of service. While there has been a great deal of effort devoted to increasing the efficiency of individual ASR engines through techniques such as improved search and pruning strategies [3], very little work has been reported on techniques to increase efficiency in multi-user ASR scenarios. An experimental study is presented demonstrating the performance gains that are achievable for varying user populations and ASR resources. The study was performed using a platform based on the single threaded, non-blocking distributed speech enabled middleware framework that was originally developed for client-server based mobile ASR services [1].

The remainder of this paper is organized as follows. Section 2 provides basic definitions of a call in the context of human-machine dialogs and quality of service for ASR servers. Section 3 presents a simple theoretical model for efficient ASR resource allocation. This model will be used to predict the total number of users that can be supported by the proposed framework under different assumptions while maintaining a given quality of service. Finally, the experimental results are presented in section 4.

2. MULTI-USER ASR SCENARIO

There are several assumptions that are made in this work concerning the means by which a user interacts with a speech dialog system and how both ASR quality of service (QOS) and system overload are defined. The most general assumption about the overall implementation is that calls are accepted from multiple users and are serviced by pools of ASR servers each of which can return a recognition string for any given utterance with some latency. The manner in which these ASR servers are allocated is described in Section 3.

A typical interaction, or *call*, in human-machine dialog applications consists of several steps. The user first establishes a channel with the dialog system over a public switched telephone network (PSTN) or VoIP connection. Once the channel is established, the user engages in a dialog that consists of one or more turns during which the user speaks to the system and the system responds with information, requests for disambiguation, confirmations, etc. During the periods in which the system issues prompts to the user, the user will generally remain silent and the system will be mostly idle with respect to that channel. Finally, when the user is done, the channel is closed and the call is complete.

The quality of service (QOS) of an overall implementation is defined here in terms of the latency a system ex-

hibits in generating a recognition result for an utterance. For utterances processed on a server, there are a number of factors that contribute to this latency. When the multi-server system is operating at near peak capacity, the number of concurrent utterances, or utterance load, the server is handling can be the dominant factor. The focus of this paper rests on the observation that, irrespective of all other factors, implementing simple strategies for reducing the instantaneous load on ASR servers will result in a significant decrease in the average response latency observed by the user.

A server's maximum utterance load is defined here as the maximum number of concurrent utterances which can be processed with an acceptable average response latency. A server that handles more than its maximum utterance load is said to be *overloaded*.

3. ASR RESOURCE ALLOCATION STRATEGIES

This section presents two strategies for allocating ASR servers to incoming calls. It will be shown that an intelligent approach for allocating utterances to servers in a typical multi-server deployment can dramatically reduce the incidence of overload with respect to more commonly used allocation strategies.

3.1. Call-Level Allocation

A common approach for indirectly balancing the utterance load across the hardware resources an allocator has at its disposal is *call-level allocation.* Using this approach, an allocator assigns a call to an ASR process running on a decoding server for the duration of the call. This process is responsible for all feature extraction, voice activity detection, and decoding. For example, consider the hardware configuration shown in Figure 1. The figure illustrates a typical setup where a source of call traffic (a PBX, or VoIP gateway) routes user calls to ASR processes residing on two servers. The time-lines in Figure 1 depict six calls, each containing two utterances per call which are indicated by the rectangular labels along the time-line. As calls arrive, a simple allocator tracks the number of calls on each server and ensures that they all handle an equal number of calls.

However, as the number of calls handled by an ASR deployment increases, use of such a simple allocator can lead to an unacceptably high utterance load on some servers even when other servers are underutilized. In Figure 1 we see that during the highlighted interval the first server will need to handle an utterance load of 3 even though the second server is only handling a load of 1. Assuming that the maximum utterance load for each server is two and assuming that the processing of each utterance requires identical computational complexity, the first server will be overloaded. If the computational complexity of the ASR task is sufficiently high, this may result in unacceptably high latencies for users assigned to the overloaded first server. A simple probabilistic argument can

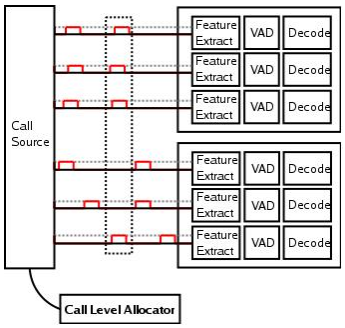

Figure 1: Example of call-level allocation showing calls being routed directly to two ASR servers.

be made that generalizes the example to an arbitrary deployment and makes this deficiency explicit. Assume, for simplicity, that each utterance is of some fixed duration, d, and each call is of some fixed duration, D. A call is then assumed to consist of L randomly occurring utterances so that at any time t, the probability that an utterance is active is given by

$$p_t = L\frac{d}{D}. \tag{1}$$

If we assume that a server that handles an utterance load of more than Q is overloaded, then the probability of overload if it services M calls, with $M > Q$, is given by

$$P_q = \sum_{k=Q+1}^{M} \binom{M}{k} p_t^k \left(1 - p_t\right)^{M-k}. \tag{2}$$

This is simply the probability that more than Q users out of M calls on a server will speak at any given moment. This probability is obviously zero when the server is handling Q calls or less. The probability P_q can then be used to calculate the probability, P_C, that one or more servers in a deployment of S servers (with $S > 1$), each handling M calls, will be overloaded.

$$P_C = 1 - \left(1 - P_q\right)^S \tag{3}$$

In Section 4, Equations 2 and 3 will be used to determine the number of calls, M, that can be supported by the call-level allocation strategy when the probability of overload, P_C, is fixed at an acceptable value. It will be shown that the fundamental difficulty with this approach arises from the fact that the call-level allocator knows nothing of what transpires within a call.

3.2. Utterance-Level Allocation

One way to reduce the probability of overload is to let the allocator look within calls to determine when utterances begin and end. This additional information can be used to implement an allocator that assigns computational resources to utterances instead of entire calls. This will be referred to as *utterance-level allocation*. Figure 2

illustrates this approach. In order to inspect the audio stream of incoming calls the allocator is placed between the source of call traffic and the ASR decoding servers. In addition, feature extraction and voice activity detection are moved to the allocator so that it may determine when utterances begin and end. Of course, it is possible to perform feature extraction in several locations including the client, the allocator as shown here, or in the ASR server. From this vantage point the allocator can keep track of activity across the deployment and intelligently dispatch utterances and balance the incoming utterance load. This allows that same deployment of S servers to be viewed as a single virtual server that can handle an aggregate utterance load of SQ concurrent utterances.

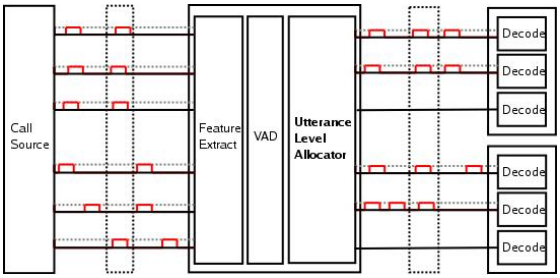

Figure 2: An utterance level allocator can look within dialogs to determine when utterances begin and end. This information is used balance the load on ASR decoding servers.

Under this model, an overload on any server can only occur if more than SQ utterances are active, an event that is considerably less likely than any individual server being overloaded. More specifically, for a deployment handling SM calls, with $SM > SQ$, the probability, P_U, that an overload will occur is given by

$$P_U = \sum_{k=SQ+1}^{SM} \binom{SM}{k} p_t^k \left(1 - p_t\right)^{SM-k} \qquad (4)$$

Equation 4 will be used in Section 4 to determine the number of calls that can be supported by the utterance-level allocation strategy when the probability of overload, P_U, is fixed at an acceptable value.

Note that although the allocator in this scenario acts as a gateway to the decoding servers it generally is not a bottleneck as the processing required to detect utterances is very small [1]. However, we must introduce an allocator that can monitor all traffic, which may be a potential bottleneck. We look at the effects of such an allocator in Section 4.

3.3. Refining Utterance-Level Allocation

Incorporating knowledge of additional sources of variability in ASR computing effort can further improve the efficiency of multi-user ASR deployments. Two examples of these sources of variability are illustrated by the plots displayed in Figure 3. The first is the high variance in computational load exhibited by a decoder over the length of an utterance. It is well known that the instantaneous branching factor associated with a given speech recognition network can vary considerably. This fact, coupled with the pruning strategies used in decoders, results in a large variation in the number of network arcs that are active and must be processed at any given instant. This is illustrated by the plot in Figure 3a which displays the number of active network arcs in the decoder plotted versus time for an example utterance in a 4000 word proper name recognition task. The plot demonstrates the fact that the majority of the computing effort in such tasks occurs over a fairly small portion of the utterance. Knowledge of this time dependent variability in the form of sample distributions could potentially be used to allocate utterances such that peak processing demands do not overlap.

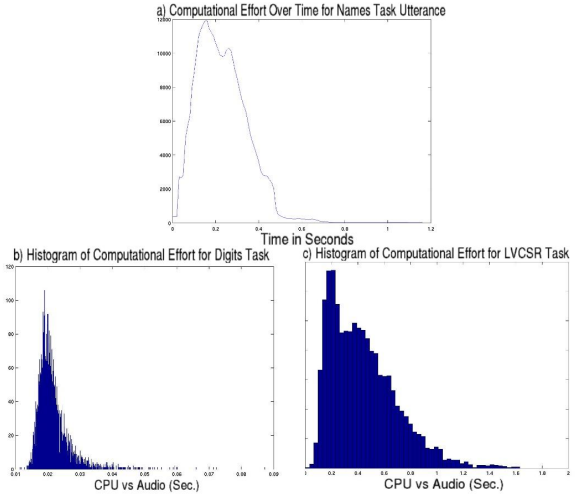

Figure 3: a) Computational effort measured as the number of active arcs versus time for an example utterance. The distribution of the ratio of decoding time to audio duration (CPU vs. audio) for b) digit task and c) LVCSR task.

The second source of variability comes from the variation in computational complexity that exists between different ASR tasks. This is illustrated by the histograms displayed in Figures 3b and 3c. The plots display the distribution of average computational effort measured as the ratio of the decoding time to the utterance duration. The distributions correspond to continuous digit and large vocabulary continuous speech recognition (LVCSR) tasks with means of 0.022 and 0.44 respectively on a 2.6 GHz server. As would be expected, the high perplexity stochastic speech recognition network associated with the LVCSR task demands a higher and more variable level of computational resources than the small vocabulary deterministic network. Distributions characterizing this inter-task variability could be incorporated into server allocation strategies. In addition to the obvious efficiency improvements beyond those discussed in Section 3.2, servers with large CPU caches can be dedicated to a single ASR task to achieve improved cache utilization.

4. EXPERIMENTAL RESULTS

This section presents the results of two comparisons of the call-level allocation (CLA) and utterance-level allocation (ULA) strategies. The first compares the efficiencies of the CLA and ULA strategies that are predicted by the model presented in Sections 3.1 and 3.2. The second compares the two strategies using an actual deployment where ASR decoders are run on multiple servers processing utterances from an LVCSR task.

A comparison of the efficiencies as predicted by the model can be made by plotting the number of incoming calls with respect to the probabilities of overload, P_C in Equation 3 for the CLA strategy and P_U in Equation 4 for the ULA strategy. The difference in overall efficiency for the two strategies can be measured as the difference between the number of calls that are supported at a given probability of overload.

Figure 4 shows a plot of this comparison for an example where the multiuser configurations illustrated in Figures 1 and 2 are configured with ten ASR servers each of which can service a maximum of four simultaneous utterances without overload. It is also assumed that, on the average, there are active utterances to be processed by an ASR server for only one third, $\frac{d}{D} = 1/3$, of the total duration of a call. It is clear from Figure 4 that, at a probability of overload equal to 0.1, the utterance-level allocation strategy can support approximately two times the number of calls that can be supported by the call-level allocation strategy.

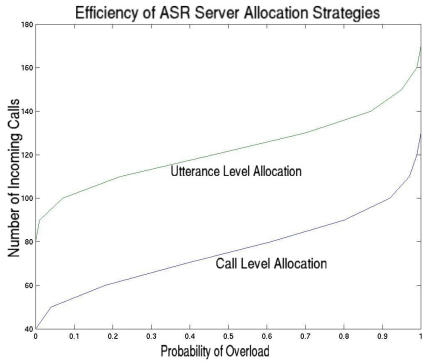

Figure 4: Number of calls supported by CLA and ULA strategies plotted versus predicted probability of overload.

A comparison of the efficiencies that are obtainable in an actual deployment was made using the distributed speech enabled middleware framework [1]. The framework was configured with ten 1 GHz Linux based servers running instances of the AT&T Watson ASR decoder. Calls were formed from utterances that were natural language queries to a spoken dialog system with speech active for an average of 35 percent of the total call duration and each server able to service approximately two simultaneous utterances without overload. A rather aggressive load of four hundred of these calls were presented simultaneously to the multi-user system. An overall performance measure was used that is derived from the latency based QoS defined in Section 2. For a given number of incoming calls, a count is obtained for the percentage of utterances where the latency in generating a recognition result falls below a specified threshold. Figure 5 shows a plot of these percentages plotted versus the threshold that is placed on the maximum response latency. The maximum response latency ranges from 0.5 to 3.0 sec. Curves are shown for both the CLA and ULA strategies.

The system implemented with the ULA strategy is shown in Figure 5 to support a significantly larger call load than the CLA system. It can be seen that the ULA strategy is able to service approximately twice as many requests with a one second maximum latency.

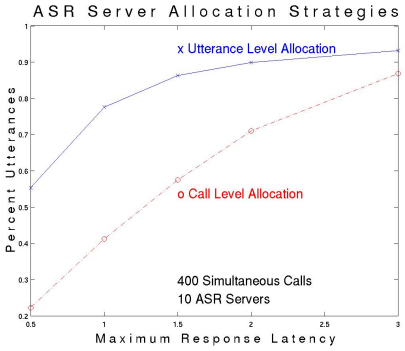

Figure 5: Percentage of actual calls serviced within specified latencies for CLA and ULA strategies.

5. CONCLUSIONS

This paper has demonstrated the importance of efficient strategies for allocating ASR servers in multi-user ASR applications. Section 3.3 outlined the sources of variability in processing effort that exists for ASR decoders servicing human-machine dialogs. It was shown in Section 4, using both theoretical and experimental results obtained for actual dialog utterances, that simply using an utterance-level strategy for assigning ASR servers there is a potential for increasing the efficiency of the overall multi-user ASR deployment by a factor of two. Future work will involve the development of allocation strategies that exploit additional knowledge of the variation in computational complexity for human-machine dialogs.

6. REFERENCES

[1] R. C. Rose, I. Arizmendi, and S. Parthasarathy. An efficient framework for robust mobile speech recognition services. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, April 2003.

[2] Matt Welsh, David E. Culler, and Eric A. Brewer. SEDA: An architecture for well-conditioned, scalable internet services. In *Symposium on Operating Systems Principles*, pages 230–243, 2001.

[3] S. Wendt, G. A. Fink, and F. Kummert. Dynamic search-space pruning for time-constrained speech recognition. *Proc. Int. Conf. on Spoken Language Processing*, September 2002.