Mini-Project Progress Review 2

Project Title : TailBench
Project Guide : Dr. K.V. Subramaniam

Project Team (Names & USN) :
Supreeth G Kurpad - PES1UG19CS520,
Abhishek Aditya BS - PES1UG19CS019,
Vishal R - PES1UG19CS571.

Project Abstract and Scope

Tailbench is a benchmark suite and evaluation methodology for latency-critical applications. Tailbench consists of 9 different applications that can be used to obtain the tail latencies. The 9 applications are harness, img-dnn, masstree, moses, shore, silo, specjbb, sphinx, xapian.

Out of all the applications, we have limited the scope of the project to a single application - Sphinx.

Sphinx is a speech recognition system written in C++. Speech recognition systems are an important component of speech-based interfaces and applications such as Apple Siri, Google Now, and IBM Speech to Text. Sphinx performs speech recognition using random instances from the CMU AN4 alphanumeric database.

This phase of the project is dedicated to extensively profiling sphinx on varied parameters of cpu stress, memory stress, cache availability and other metrics shown in the upcoming slides.

PES UNIVERSITY

Suggestions from Review - 1

For Review - 1, benchmarking for all the 8 applications present in tailbench were presented. The panelists recommended the following improvements with respect to our project

- Instead of analyzing all the 8 tailbench applications at the same time, pick a few applications and profile them extensively.
- Identify the dataflow and analyze how latency information propagates through different applications.
- Profile the applications on different configurations and server loads.
- Generate function call graphs to identify how each function spends its time processing.
- Stress the server resources and run the applications to determine how they perform under such conditions.

Project Requirements

- Profile one or more applications from the tailbench application suite.
- Determine the root cause for tail latencies based on the profiling performed.
- Once the root cause is determined, create an environment to prevent the tail latencies, consequently improving the performance of the latency critical applications.

Literature Survey

● *Kasture, Harshad, and Daniel Sanchez. "Tailbench: a benchmark suite and evaluation methodology for latency-critical applications." In 2016 IEEE International Symposium on Workload Characterization (IISWC), pp. 1-10. IEEE, 2016.*
Link : https://ieeexplore.ieee.org/abstract/document/7581261

● *Xiong, Xingwang, Lei Wang, Wanling Gao, Rui Ren, Ke Liu, Chen Zheng, Yu Wen, and Yi Liang. "DCMIX: generating mixed workloads for the cloud data center." In International Symposium on Benchmarking, Measuring and Optimization, pp. 105-117. Springer, Cham, 2018.*
Link : https://link.springer.com/chapter/10.1007/978-3-030-32813-9_10

● *Kasture, Harshad, Xu Ji, Nosayba El-Sayed, Nathan Beckmann, Xiaosong Ma, and Daniel Sanchez. "POSTER: improving datacenter efficiency through partitioning-aware scheduling." In 2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 134-135. IEEE, 2017.*
Link : https://ieeexplore.ieee.org/abstract/document/8091227

● *K. -. Lee, H. -. Hon and R. Reddy, "An overview of the SPHINX speech recognition system," in IEEE Transactions on Acoustics,
Speech, and Signal Processing, vol. 38, no. 1, pp. 35-45, Jan. 1990, doi: 10.1109/29.45616.*
Link : https://ieeexplore.ieee.org/abstract/document/45616

● *Tu, Stephen, Wenting Zheng, Eddie Kohler, Barbara Liskov, and Samuel Madden. "Speedy transactions in multicore in-memory
databases." In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pp. 18-32. 2013.*
Link : https://dl.acm.org/doi/abs/10.1145/2517349.2522713

Detailed Literature Survey

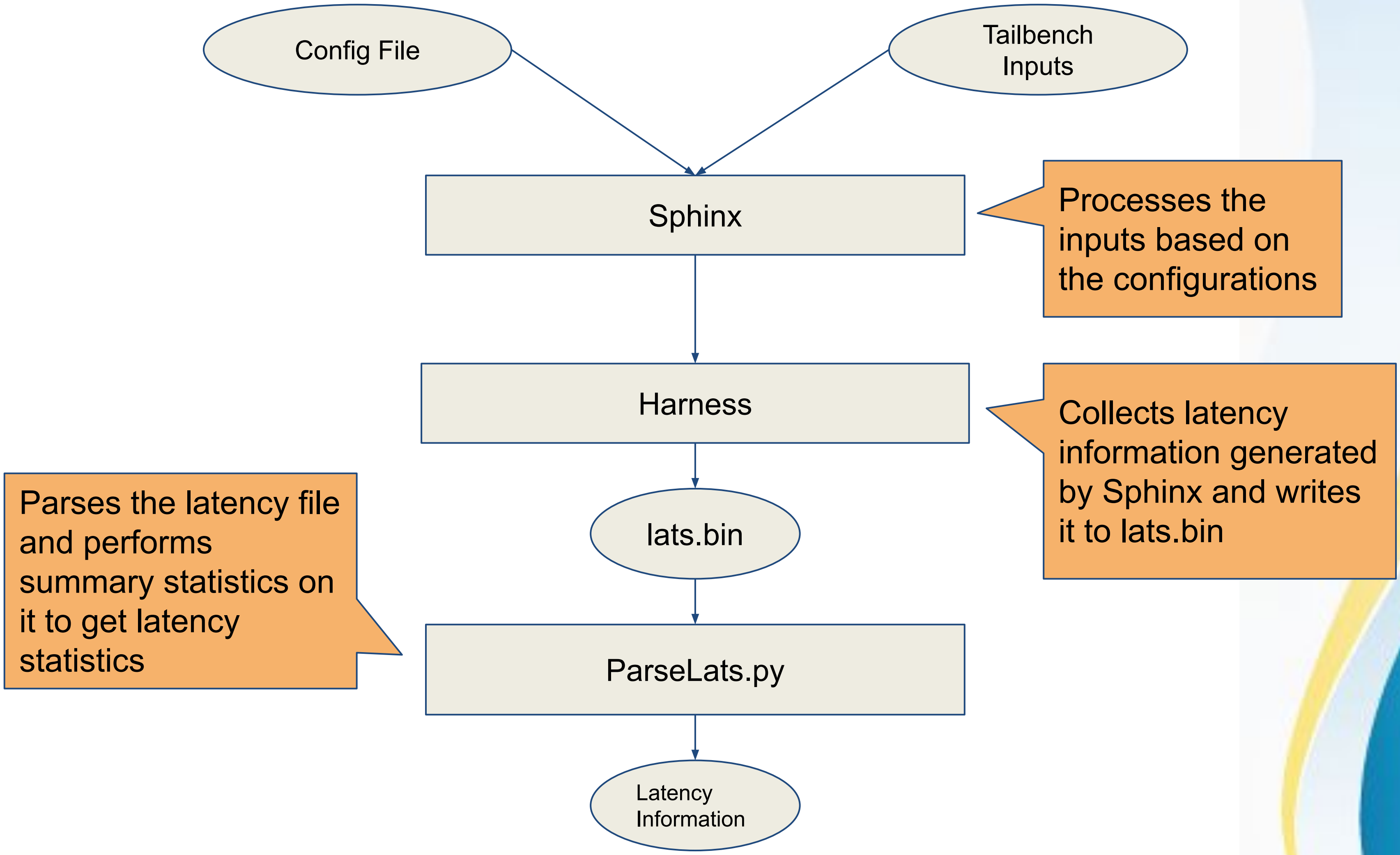
Tailbench is modular - Majority of the applications on Tailbench are independent of one another.

A common application “harness” provides a variety of load-testing scenarios in order to benchmark and validate the applications.

After evaluating the applications on different workloads, it was found that compared with the Service-Standalone mode (only executing the online service workload), 99th percentile latency of the service workload under the Mixed mode (workloads mix execution) increased 3.5 times, and the node resource utilization under that mode increased 10 times.

A description is given of SPHINX, a system that demonstrates the feasibility of accurate, large-vocabulary, speaker-independent, continuous speech recognition. Speaker independence, continuous speech, and large vocabularies pose three of the greatest challenges in automatic speech recognition. Previously, accurate speech recognizers avoided dealing simultaneously with all three problems. SPHINX, a system that demonstrates the feasibility of accurate, large-vocabulary speaker-independent, continuous speech recognition. SPHINX is based on discrete hidden Markov models (HMM’s) with LPC-derived parameters. To provide speaker independence, knowledge to these HMM’s were added in several ways: multiple codebooks of fixed-width parameters, and an enhanced recogniar with carefully designed models and word duration modeling. To deal with coarticulation in continuous speech, yet still adequately represent a large vocabulary, two new subword speech units-function-word dependent phone models and generalized triphone models were introduced. With grammars of perplexity 997, 60, and 20, SPHINX attained word accuracies of 71, 94, and 96 percent on a 997-word task.

System Design


Config File
Tailbench Inputs
Sphinx
Processes the inputs based on the configurations
Harness
Collects latency information generated by Sphinx and writes it to lats.bin
lats.bin
Parses the latency file and performs summary statistics on it to get latency statistics
ParseLats.py
Latency Information

Technologies Used

Operating System: Ubuntu 18.04

System Configuration : 2 x 8 core Intel Xeon 2.3Ghz, 64 GB DDR4 RAM, 1TB HDD

Languages used by the 8 applications in Tailbench:

- C
- C++
- Java
- HTML
- Python

Tools:

- Bash
- Perf
- Ftrace
- Valgrind
- taskset
- iostat
- dstat
- stress-ng
- cachestat
- htop
- matplotlib
- pandas
- scipy

Project Progress So far

What we have done so far:

- Compiled all tailbench applications
- Ran all the applications and tabulated the runtime statistics of each application.
- Generated CPU and Memory Utilization Graphs.
- Analyzed if each application is CPU intensive/ Memory Intensive by observing the resident set size and the percentage CPU utilization for each of the 8 applications.
- Used perf and valgrind to generate function call graphs.
- Generated latency graphs for all applications.
- Performed extensive analysis on Sphinx.
- Generated Function Call Reports to figure out the amount of time spent in each function called by sphinx.
- Tabulated latency details under various stress simulations like CPU Stress, Memory Stress, Disk I/O stress and a combination of the above factors.
- Monitored the disk and cache usage of Sphinx
- Analyzed the performance of Sphinx with disk caching enabled/disabled.

Conclusion

- Sphinx is a CPU Intensive Process and can be configured to use 100% of all the 32 cores on the server.
- The mean of response times for the requests increases on increasing the stress on the CPU and Memory.
- Caching is not a major factor for the performance of the application. The mean response time remains unchanged irrespective of the number of page faults.
- Although the mean response time for the requests increases, the ratio of the 95th and 99th percentile to the mean remains approximately the same.
- In the next phase, the plan is to profile the system along the life cycle of a request/response.
- Once this is achieved, we will be able to map the system state to the latency that is observed, which will better help us pinpoint to the root cause of the tail latencies.

References

Link to GitHub Repository:
https://github.com/supreethkurpad/Tailbench

Progress Report 1:
https://github.com/supreethkurpad/Tailbench/blob/main/Tailbench-Report-1.pdf

Progress Report 2:
https://github.com/supreethkurpad/Tailbench/blob/main/Tailbench-Report-2.pdf

Progress Report 3:
https://github.com/supreethkurpad/Tailbench/blob/main/TailBench-Report-3.pdf

Generated Graphs and Latency files :
https://github.com/supreethkurpad/Tailbench/tree/main/TailBench%20Graphs%20+%20Statistics

Thank You