Ecs251 project proposal

Harshil Patel[1] **, Anugya Sharma**[1]
[1]Department of Computer Science, University of California, Davis

1 Introduction & Motivation

The advancements in multicore architecture and multiprocessor computers are increasing and ongoing (Adam, 2022). To keep up with this trend, we need techniques like multithreading to maximize the performance of ever-advancing cpu's (ni). Multithreading is of vital importance for this because it can harness the power of multiprocessor computer, improve system reliability by preventing one operation from negatively affecting another in a program(user interface event affecting time-critical operation), and maximize cpu use (in programs that read/write from a file, perform I/O, or poll the user interface) (ni).

While a very efficient tool for harnessing the power of multiprocessor computer, multithreading does come with some challenges. The top most important challenge in multithreading is the management of thread pools (Lee et al., 2011), namely the size of the thread pool. The number of threads in a thread pool determine the changes in response times and resource utilization (Lee et al., 2011). In our research, we have come up with a "dynamic thread pool" with varying number of threads such that we can optimize minimum response time for maximum resource utilization.

Start Rough note for Q2 ....Current heuristics that are used to determine the number of threads are flawed (Ling et al., 2000). There has been some researches on a dynamic model for thread size. In the beginning, they just had the watermark model(adjusts the number of threads according to the current number of incoming requests) (Kim et al., 2007) .While the watermark model solves the problem of "efficient usage of resources", it doesn't efficiently obtain the required number of threads in advance (Kang et al., 2008). There are some prediction based models, but In the prediction based models, only factor such as the request rate or number of worker threads is considered in the management. (Lee et al., 2011) One research that has proposed dynamic thread pool(prediction based) model, themselves point out that the model lacks testing for a longer period of time and on general server environment which provides various services simultaneously (Ling et al., 2000).Trendy exponential moving average model proposed in one study doesn't have the desired accuracy in its predictions (Lee et al., 2011).

All these researches into dynamic sizes of pool, but none look at the memory usage/CPU overheads and whether that are significantly lower than static pool to signify the performance cost, whatever they may be. We have to look at this , so that this informs the people making decision to choose dynamic thread pools moving forward. ....End Rough note for Q2

example citation (Hu et al., 2024).

References

[link].

George K Adam. 2022. Co-design of multicore hardware and multithreaded software for thread performance assessment on an fpga. *Computers*, 11(5):76.

Hanxu Hu, Simon Yu, Pinzhen Chen, and Edoardo M Ponti. 2024. Fine-tuning large language models with sequential instructions. *arXiv preprint arXiv:2403.07794*.

DongHyun Kang, Saeyoung Han, SeoHee Yoo, and Sungyong Park. 2008. Prediction-based dynamic thread pool scheme for efficient resource usage. In *2008 IEEE 8th International Conference on Computer and Information Technology Workshops*, pages 159–164. IEEE.

Ji Hoon Kim, Seungwok Han, Hyun Ko, and Hee Yong Youn. 2007. Prediction-based dynamic thread pool management of agent platform for ubiquitous computing. In *International Conference on Ubiquitous Intelligence and Computing*, pages 1098–1107. Springer.

Kang-Lyul Lee, Hong Nhat Pham, Hee-seong Kim, Hee Yong Youn, and Ohyoung Song. 2011. A novel predictive and self–adaptive dynamic thread pool management. In *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications*, pages 93–98. IEEE.

Yibei Ling, Tracy Mullen, and Xiaola Lin. 2000. Analysis of optimal thread pool size. *ACM SIGOPS Operating Systems Review*, 34(2):42–55.