Challenges in Multicore Performance Predictions

Poster Paper

Markus Frank, Floriment Klinaku, Steffen Becker
Universität Stuttgart, Universitätstraße 38, Germany
firstname.lastname@informatik.uni-stuttgart.de

ABSTRACT

Software performance predictions are an established part of an engineering like software development process and therefore relevant to enable high quality and to ensure requirement fulfillment. Software Performance Engineers use for that model-based performance predictions approaches. However, current predictions approaches are based on the assumption of single core CPU systems. To enable Software Performance Engineers to further give accurate predictions also for multicore systems, which are by now state of the art, we need to adapt our current prediction models.

On the poster, we discuss the upcoming challenges to be tackled to increase the accuracy of the performance predictions models.

KEYWORDS

Software Performance Engineering, Multicore, Modeling Language

ACM Reference Format:
Markus Frank, Floriment Klinaku, Steffen Becker. 2018. Challenges in Multicore Performance Predictions: Poster Paper. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering Companion , April 9–13, 2018, Berlin, Germany.* ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/3185768.3185773

1 INTRODUCTION

Building, refactoring, or reverse engineering software is a complex and time-consuming task. To archive high qualitative and requirement fulfilling results it is necessary to follow an engineering-like approach. Thereby, **S**oftware **P**erformance **E**ngineering (SPE) is an important field. In SPE, software architectures are analysed in a model-based manner in early design phases, to evaluate quality attributes (e.g., response time) according to the requirements. To obtain quality attributes for a given architecture, **S**oftware **P**erformance **E**ngineer**s** (SPEs) have to model three aspects of the software system: The software behavior (e.g., architecture, components and interfaces, their interconnection, control flow, etc.), the used hardware (e.g., used hardware architecture, cpu-speed, hard drive access rates, network links), and the usage scenario (e.g., how often each method is called). With these model-based performance predictions SPEs are capable to analyse not only simple systems, but also complex, large-scale, and dynamic cloud-based systems.

However, the current performance prediction models are based on a single metric reflecting the CPU speed. For a long time this was a valid assumption. But most of todays commonly and wide spread CPUs are multicore CPUs. They consist of multiple cores (processing units), complex memory hierarchies (e.g., different caches, and memory levels), and limiting memory bandwidth. We were able to show in [2, 4] that the currently used performance prediction models are not suited to give accurate prediction for these kind of systems. Further, in [3] we performed a full **S**ystematic **L**iterature **R**eview to cover the related work regarding modeling approaches for parallel software and searched for performance prediction approaches not yet known to us. The result of the SLR revealed two insights. (1) The need for improved performance prediction models taking into account the rising number of multicore systems. (2) The absence of modeling approaches for concurrent software and furthermore the absence of performance prediction models for multicore systems in general.

For SPE, this means to re-evaluate and adapt current performance prediction approach. This is a long way. Therefore we use the poster, to present initial thought about the upcoming challenges, which have to be tackled. We can not give a solution yet, but we want to use the poster as discussion starter and to outline future research.

In the following, we will introduce the different challenges, which will be shown on the poster.

2 CHALLENGES

In this section we discuss the challenges for SPE, which we identified based on the finding from two case studies [2, 4] and one full SLR [3] we performed. For the sake of understanding it is useful to group the challenges by the model type they focus on: software model and hardware model.

Figure 1 shows the two models aspect: The software and hardware aspect. Thereby $M0$ and $M1$ indicate the model's abstraction level. $M0$ means reality level and $M1$ the SPE model level.

On the software side, we observed the trend to use frameworks like openMP or paradigms like ACTORs to write concurrent code instead of sequential one. However, these concepts cannot be represented by performance models currently (see challenge C_1 - C_3, which are discussed in a moment).

On $M0$ level on the hardware side, we have the shift from single core CPUs with dedicated memory to multicore CPUs, where multiple cores can—but must not necessary—work on the same memory. On the $M1$ level the performance models consider only a single metric, the CPU speed, till now. However, this is insufficient for multicore systems. In the future performance models have to consider multiple metrics to guarantee prediction accuracy (see challenge C_4 and C_5 below).

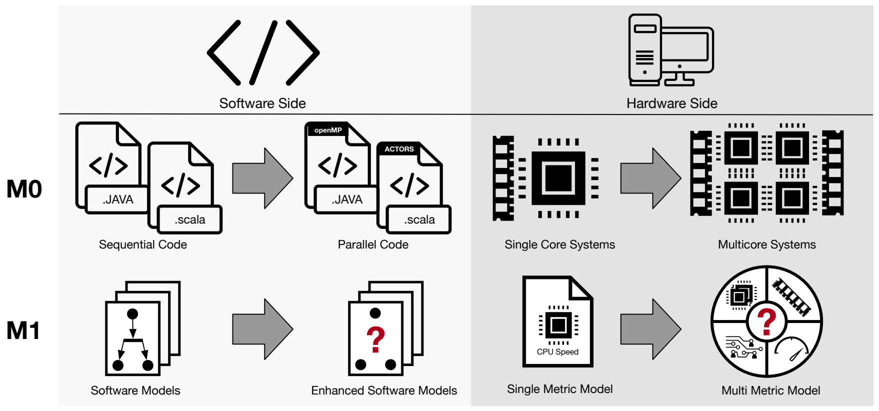

Figure 1: SPE Development: From sequential code to parallel code / From single core systems to multicore systems

2.1 Software Models

To support parallelism on software side it is needed that the software's code is written in a way that it supports concurrency. Originally, programmers directly used and defined threads for this. However, the SLR showed that this is not common today anymore. Developers use: (1) Frameworks and libraries like openMP to automatically generate and handle a massive amount of variable threads. (2) New approaches like ACTOR, which follow new paradigms and facade complexity from the developer. (3) Auto- or manual turners to optimize software for the used hardware. This leads to the following three major challenges on software model side:

C_1 **Parallel Constructs Challenge:** We have seen that parallel constructs are used frequently during parallel programming. To represent these constructs, we need to add constructs on the model level and in our tool support. This way we lower the effort of a software performance engineer during the model creation. E.g., Parallel constructs can be similar to openMP parallel constructs (i.e., parallel section, parallel loop, etc.). The main challenge here is to identify an adequate set of constructs and to integrate them to existing prediction models.

C_2 **Active Resources and Message Passing:** To avoid the complexity to handle multiple threads working on the same memory, multiple approaches rose to ensure that each working unit has its dedicated memory and working set. These units can work concurrently and communicate by messages (e.g., MPI or ACTORs). However, to support such mechanisms it is necessary to have active resources and message passing features available in the models.

C_3 **(Auto-)Tuners** are an important concept and are widely used. Thus, adding tuning concepts to predictions models to increase the accuracy of performance predictions is needed.

2.2 Hardware Models

In the two case studies we show that the accuracy of the software performance prediction models—even for small and manageable use case like a matrix multiplication—drops significantly when using multicore environments. Currently the models only consider CPU speed as a relevant metric, but with the raising complexity of CPU architectures also other factors become relevant [1]. Based on our observations we claim that memory hierarchy and memory bandwidth is a game-changer for multicore CPU predictions. Therefore, additionally to the CPU speed, it will be necessary to include the following two aspects to the prediction models:

C_4 **Memory Bandwidth Challenge:** Similar to network links and their bandwidth, the bandwidth of the memory bus is a limiting factor when it comes to multicore systems since multiple cores share the same memory bus. So, with additional cores and memory levels both the memory bandwidth and the additional synchronization overhead needs to be modeled.

C_5 **Memory Design Challenge:** The architecture of CPU's differs, but in common CPUs there are multiple memory levels like L1 - L3 caches besides the main memory. While the main memory is mostly shared by all cores, caches are specific either for one or multiple cores. In extreme, different cores have different bandwidth and latency to different parts of the main memory (as shown in Fig.1). Also relevant are different cache update and replacement strategies and hardware coherence protocols. This can have a great impact on the software speed and has to be considered in the performance prediction models.

3 SUMMARY

Even thought performance prediction for multicore systems is a complex topic, we cannot deny the need for accurate predictions for those systems. On our poster, we draw the landscape of future challenges in this area and outline future research. Thereby, we see the poster as discussion and research starter, rather than we can present a final solution.

REFERENCES

[1] Steffen Becker, Tobias Dencker, and Jens Happe. 2008. Model-driven Generation of Performance Prototypes. In *SPEC International Performance Evaluation Workshop.* Springer, 79–98.
[2] Markus Frank and Marcus Hilbrich. 2016. Performance Prediction for Multicore Environments—An Experiment Report. In *Proceedings of the Symposium on Software Performance 2016, 7-9 November 2016, Kiel, Germany.*
[3] Markus Frank, Marcus Hilbrich, Sebastian Lehrig, and Steffen Becker. 2017. Parallelization, Modeling, and Performance Prediction in the Multi-/Many Core Area: A Systematic Literature Review. In *Proceedings of The 7th IEEE Int. Symposium on Cloud and Service Computing, 22-25 November 2017, Kanazawa, Japan.*
[4] Markus Frank, Stefan Staude, and Marcus Hilbrich. 2017. Is the PCM Ready for ACTORs and Multicore CPUs? — A Use Case-based Evaluation. In *Proceedings of the Symp. on Software Perf. 2017, 9-10 November 2017, Karlsruhe, Germany.*