Laboratory for Parallel Programming

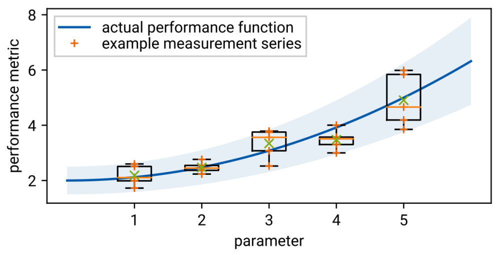
TECHNISCHE UNIVERSITÄT DARMSTADT

Master's Thesis

Automated performance modelling via basic block instrumentation

Motivation

Performance is one of the most important non-functional requirements of an application. We developed Extra-P to create performance models of applications. With these models one can identifiy the performance behavior of the application according to multiple parameters (e.g., process count, input size). Extra-P uses measurements of the application (e.g., runtime) to construct the model. Unfortunately, many systems expose their processes to substantial amounts of interference (aka noise), leading to significant run-to-run variation. On noisy systems, one usually has to repeat performance measurements several times and then apply statistics to capture trends. To avoid this, we want to extend Extra-P to create models in a modified approach using noise-resilient basic block instrumentation to generate a prior for the Extra-P modeller.



An example of a noisy measurement series and its actual underlying function.

Task

Your overall task in this thesis will be to implement basic block instrumentation for performance modelling and test its capabilites. Basic block instrumentation can be seen as a high level approach of understanding program structure. In it we divide the code into instruction blocks without diverging paths, called basic code blocks, and control structures, like loops and conditional jumps. As the number of times a basic code block is executed mainly depends on the behavior of the control structures, the asymptotic complexity of an application or subroutine mostly arises by the correlation between the control structures and the input parameters.

First, you should implement a way to divide a given program into its basic code blocks and their control structures using the code instrumentation framework llvm and be able to track the execution path during a run, e.g., count the number of loop iterations. Second, you should use this capability to find the correlation between the input parameters, e.g., input size, and the behavior of the control structures. Then, you should use your found correlations to automatically generate estimates for the asymptotic complexity of the application and its major subroutines and then use these as a basis for a noise-resilient performance model. Finally, you should compare your new approach with our currently used one.

In short your task consists of the following parts.

- Implement a framework for basic code block instrumentation and use it to deduce asymptotic complexity
- Develop a noise-resilient modeling approach with these found correlations
- Evaluate your implementation with noisy measurement data (synthetic and real)
- Compare this approach with our existing noise-resilient approach

Requirements

- Python
- C++
- Performance analysis, parallel programming

Contact

Gustavo de Morais <gustavo.morais@tu-darmstadt.de>

Issue date: **January 2022**