MaSiF: Machine Learning Guided Auto-tuning of Parallel Skeletons

MaSiF: Machine Learning Guided Auto-tuning of Parallel Skeletons

Alexander Collins
a.collins@ed.ac.uk

Christian Fensch
c.fensch@ed.ac.uk

Hugh Leather
hleather@inf.ed.ac.uk

University of Edinburgh, School of Informatics

ABSTRACT

We present MaSiF, a novel tool to auto-tune parallelization parameters of skeleton parallel programs. It reduces the cost of searching the optimization space using a combination of machine learning and linear dimensionality reduction. To auto-tune a new program, a set of program features is determined statically and used to compute k nearest neighbors from a set of training programs. Previously collected performance data for the nearest neighbors is used to reduce the size of the search space using Principal Components Analysis. This results in a set of eigenvectors that are used to search the reduced space.

MaSiF achieves 88% of the performance of the oracle, which searches a random set of 10,000 parameter values. MaSiF searches just 45 points, or 0.45% of the optimization space, to achieve this performance. MaSiF provides an average speedup of 1.18x over parallelization parameters chosen by a human expert.

Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming—*Parallel programming*

General Terms

Measurement, Performance, Experimentation

Keywords

Auto-tuning, FastFlow, Machine Learning, Multi-core, Parallel Skeletons

1. INTRODUCTION

Parallel skeletons provide a predefined set of parallel templates that can be combined, nested and parameterized with sequential code to produce complex parallel programs. The implementation of each skeleton includes parameters that have a significant effect on performance; so carefully tuning them is vital. The optimization space formed by these parameters is complex, non-linear, exhibits multiple local optima and is program dependent. This makes manual tuning impractical. Effective automatic tuning is therefore essential for the performance of parallel skeleton programs.

PACT'12, September 19–23, 2012, Minneapolis, Minnesota, USA.
ACM 978-1-4503-1182-3/12/09.

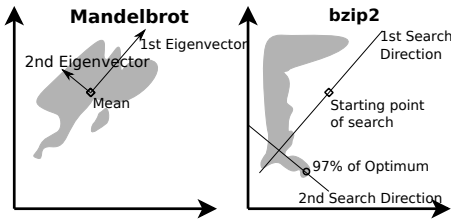

Figure 1: Optimizing a new program using eigenvector search (right) using training data from a similar program (left).

We present MaSiF, a tool that auto-tunes the FastFlow parallel skeleton framework [1]. We have identified five implementation parameters within FastFlow that have an effect on performance. MaSiF auto-tunes them using machine learning guided search applied to performance data for a subset of the parameter space collected a priori. Given a new program, a set of static features are extracted that categorize its behavior. A nearest neighbor classifier then chooses k programs from the training set that are the closest match for the new program.

The best performing points in the optimization space for these k programs are collected together. Dimensionality reduction is applied to these points, to produce a reduced sized search space. The search is then performed within this smaller space, along the directions of greatest variation in the best points provided by the dimensionality reduction.

Our results show that MaSiF achieves 88% of the performance of the oracle, which searches a random set of 10,000 parameter values. MaSiF searches just 45 parameter values on average. MaSiF also achieves an average speedup of 1.18× over parallelization parameters chosen by a human expert. These results demonstrate that our technique is effective at automatically optimizing parallel skeleton programs without the need for human expertise.

2. MOTIVATION

Figure 1 illustrates how we optimize a new program (bzip2), given training data from a similar program (mandelbrot). The left plot in Figure 1 shows the optimization space of the mandelbrot program with 2 parallelization parameters. Shaded areas mark parallelization parameter values that obtain 95% or more of the best performance. Applying Principal Components Analysis (PCA) [2] to these best parameter values provides a mean and two eigenvectors. These are shown in the plot.

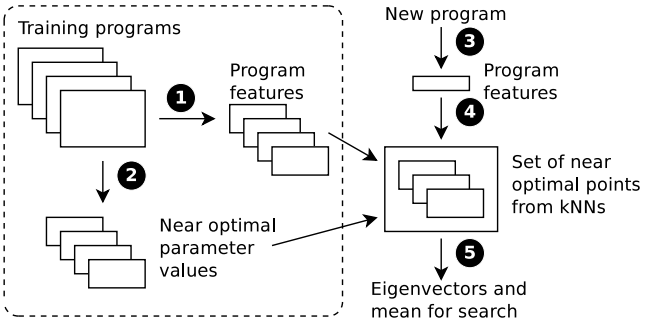

Figure 2: Schematic of how machine learning guides the search. The dotted box indicates the offline training phase. ❶ Extract program features from training programs. ❷ Find a set of near optimal parameter values for each training program, by measuring a random subset of the space. ❸ Extract program features from the new program. ❹ Compute the k-nearest neighbors, using the program features, to create a set of near optimal parameter values for all nearest neighbors. ❺ Apply PCA to the set of near optimal parameter values. The resulting mean and eigenvectors are used to perform the search.

The right plot in Figure 1 shows the optimisation space of a different program: bzip2, with the same 2 parallelization parameters as the mandelbrot program. Again, the grey shaded area indicates parameter values with 95% or more of the best available performance. Even though the optimization space looks completely different, we can apply a search to optimise bzip2 using the mean and eigenvectors from mandelbrot. Starting at that mean, we exhaustively search along the first eigenvector in both directions to find the best point. From there, we now search along the direction of the second eigenvector. The search finds a point that obtains 97% of the best available performance.

This motivating example shows that the eigenvectors provided by the PCA for one program can be used to guide the search in the optimisation space of a different program, if the programs are a good match. In Section 3, we show how similar programs can be found.

3. AUTO-TUNING USING MaSiF

MaSiF starts by collecting all 'near optimal' parameter values from similar training programs, using machine learning detailed in Figure 2. We define 'near optimal' parameter values as those at 95% or more of the best performance of the oracle (a random search over 10,000 parameter values).

Applying PCA to this set of parameter values provides a new set of orthogonal basis vectors, or eigenvectors, for the parameter space.

PCA also returns a measure of the variance of the data in the direction of each eigenvector, called the eigenvalues. These are used to scale the eigenvectors so that they cover the region of the space where the majority of the 'near optimal' parameter values lie. The search space is reduced by only searching within the plane defined by the two eigenvectors that capture the most variation. The search space is further reduced by searching along each eigenvector in turn, as discussed in Section 2.

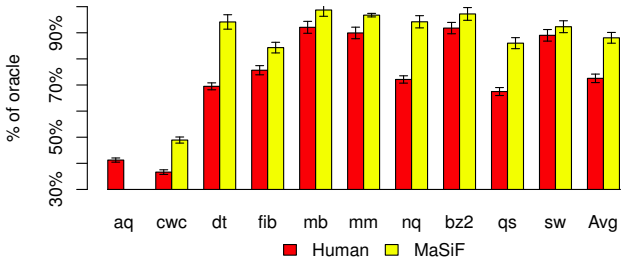

Figure 3: The percentage of the oracle performance achieved by a human expert and MaSiF. MaSiF searches along two eigenvectors, exploring 45 parameter values on average. The error bars show 99% confidence intervals for the mean.

4. RESULTS

We evaluate MaSiF using a 32-core shared memory machine and 10 existing skeleton parallel programs implemented using FastFlow.

Figure 3 shows the percentage of the oracle performance achieved by MaSiF and parallelization parameters chosen by the FastFlow authors. This plot demonstrates that MaSiF achieves performance closer to the oracle than the expert in all but one case. For 7 of the 10 programs, MaSiF achieves at least 90% of the oracle performance, and in some cases is close to 100%. On average over all programs, MaSiF achieves 88% of the oracle performance.

It is important to note that MaSiF only measures the performance at 45 points on average, in order to obtain this performance. This is far fewer than the size of the space, which consists of 127,008 points.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated that MaSiF achieves 88% of the performance of the oracle; which searches a random subset of the entire space. The results also show that MaSiF provides a 1.18× speedup over human expert chosen parameters, on average over all programs.

These results show that MaSiF does not significantly impact the maximum possible performance achievable—in fact it does better than a human expert—whilst markedly reducing the number of parameters that need to be searched. The PCA space reduction reduces the size of the search space by 275× on average, and searching along just two eigenvectors reduces the space by 1,925×.

Our technique is fully automatic and not constrained to only work with FastFlow. It can be easily adapted to tune different parameters. In the future, we will investigate searching the reduced parameter space at runtime. We will also investigate more sophisticated techniques to improve over the exhaustive search along each eigenvector.

6. REFERENCES

[1] M. Aldinucci, M. Torquati, and M. Meneghin. FastFlow: Efficient parallel streaming applications on multi-core. Technical Report TR-09-12, Università di Pisa, Dipartimento di Informatica, Italy, 2009.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.