Machine Learning for Automatic Performance Tuning
Nicholas Chaimov <nchaimov@cs.uoregon.edu>
Advisor: Allen D. Malony
Computer & Information Science — University of Oregon

Introduction

Automatic Performance Tuning is the problem of producing, from input code, output code which performs better on than the input on some metric, such as time to execute or energy consumption. Empirical Autotuning is an approach to this problem which involves searching a space of transformed code variants (see transformations, below). By measuring the performance of code variants, autotuning produces variants optimized for particular execution environments or input parameters. An exact solution to this problem involves searching a space of transformed code variants (such as Orio [2] or CHILL [3]) which generates the variants, which are substituted into the original program. The transformed programs are then run and their performance measured, and the process repeats until the search termination conditions are met. All of the performance profiles are tagged with metadata describing the transformations applied, the execution environment, and the input data, and are stored in a centralized performance database, TAUdb.

Transformations

Permutation / Loop Interchange

```
for(int i = 0; i < I; ++i) {
  for(int j = 0; j < J; ++j) {
    computation(i,j,k);
  }
}
```

Loop Unrolling

```
for(int i = 0; i < I; ++i) {
  for(int j = 0; j < J; ++j) {
    computation(i,j,k);
  }
}
```

Loop Tiling / Blocking

```
for(int i = 0; i < I; ++i) {
  for(int j = 0; j < J; ++j) {
    computation(i,j,k);
  }
}
```

Loop Fusion

```
for(int i = 0; i < I; ++i) {
  for(int j = 0; j < J; ++j) {
    computation_1(i);
  }
}
```

Loop Splitting

```
for(int i = 0; i < I; ++i) {
  for(int j = 0; j < J; ++j) {
    computation_2(i);
  }
}
```

Decision Tree Learning

We have designed a system for empirical autotuning with an architecture shown in the figure below:

In our system, a baseline performance measurement of the unoptimized code is made using the TAU Performance System. A search engine (such as Active Harmony [1] or Orio [2]) proposes variants based on a parameterized transformation script. A code variant generator (such as Orio [2] or CHILL [3]) generates the variants, which are substituted into the original program. The transformed programs are then run and their performance measured, and the process repeats until the search termination conditions are met. All of the performance profiles are tagged with metadata describing the transformations applied, the execution environment, and the input data, and are stored in a centralized performance database, TAUdb.

The decision tree can be generated using execution environment (such as CPU architecture, number of cores, cache size, etc.) and input parameters (such as input size) as features. For example, the above tree selects a variant of a 3D matrix multiplication routine based upon the size of the input matrix. The variant selected by a decision tree can also be used as the initial configuration for search, resulting in the search process converging earlier, as in the graph below, showing time to converge for default (red) and predicted (green) configurations.

Autotuning is repeated across multiple computers and multiple input datasets, yielding annotated performance data which we can use to generate runtime-adaptive code or to improve the search process. In both cases, this is accomplished by performing decision tree learning over the performance data. To generate runtime-adaptive code, the decision tree is converted into executable code in the form of a wrapper function.

Evaluation

References

