Automatic Mapping of the HPEC Challenge Benchmarks*

Nadya Travinin Bliss, Jason Dahlstrom, Daniel Jennings, Sanjeev Mohindra
{nt, jdahlstrom, dsj, smohindra}@ll.mt.edu
MIT Lincoln Laboratory, Lexington, MA 02420

Abstract

This talk presents the automatic mapping results of the HPEC Challenge benchmarks onto an embedded tiled architecture using pMapper. pMapper is a novel automatic mapping system originally designed for automatic mapping of MATLAB programs onto commodity cluster systems. Here, the pMapper architecture is extended to address the mapping of applications onto tiled architectures. These extensions are achievable due to the modularity and flexibility of the pMapper architecture, demonstrating the versatility of the design. After reviewing the HPEC Challenge benchmarks, which are developed to evaluate the performance of multiprocessor HPEC systems, the effectiveness of pMapper is discussed. In the talk, mapping results for all of the benchmarks are presented. As an example, for the convolution benchmark, pMapper finds the optimal mapping which yields an ideal linear speedup.

Introduction

As bandwidths and signal processing complexity increase in modern sensor systems, the computational performance requirements also increase. In order to handle the increased processing needs, embedded systems are architected with multiple processing elements. This trend is accelerating – not only do current systems have multiple CPUs, the CPUs of emerging tiled architectures combine multiple compute elements in a single chip. The HPEC Challenge benchmark suite contains kernel level and system level benchmarks for measuring performance of multiprocessor systems. To derive the performance, traditionally the benchmarks are mapped onto the system by an expert parallel algorithm developer. Mapping of algorithms to architectures is a non-trivial task.

pMapper

pMapper is based on a 2-phase automatic mapping architecture designed to distribute signal processing applications onto multi-processor systems [1]. During the first phase, Initialization, pMapper collects performance information on a given system and stores that information in a performance model. The initialization is done once for a given system. The second phase, Mapping and Execution, is performed for every program submitted to the system. pMapper uses dynamic code analysis in order to achieve global or program flow optimization. Specifically, pMapper chooses an efficient set of maps for the entire program, not a single function.

pMapper was originally designed for automatically mapping MATLAB programs onto large-scale grid computing systems, such as the LLGrid [2]. However, due to its versatility, pMapper can also be used as a tool for mapping algorithms onto embedded systems. pMapper also has the ability to simulate and predict the performance of such systems. Figure 1 illustrates the Mapping and Execution phase set up to predict mappings onto the IBM Cell processor [3].

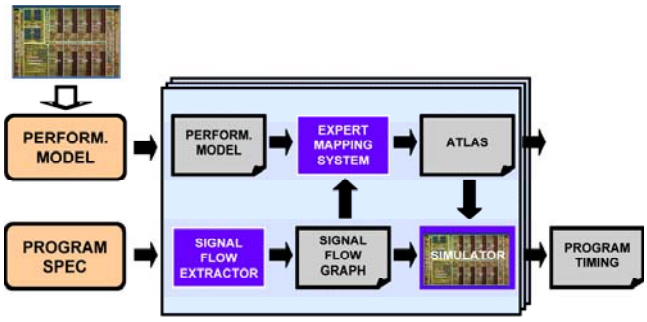

Figure 1: Mapping and Execution with Simulated Cell.

IBM's Cell processor is chosen as a representative tiled architecture. Let us consider each component of the mapping and execution phase in detail. The Performance Model is a lookup table of timing data derived from the Cell processor machine model. The machine model consists of processor characteristics such as latency, bandwidth, and processor speed. The Program Spec is a MATLAB specification for the program being mapped. The Signal Flow Extractor extracts the program parse tree and, by utilizing lazy evaluation, collects as much information as possible about the program prior to the mapping. The Expert Mapping System uses the Signal Flow Graph (SFG) together with the Performance Model to produce an efficient set of maps, or an atlas, for the program. For more information on these components see [1]. Note that when being used as a mapping tool, pMapper does not execute the program, but instead simulates it. The Simulator produces the expected runtime of the program on the simulated architecture.

HPEC Challenge

The HPEC Challenge benchmark suite [4] was developed to allow for quantitative evaluation of multiprocessor systems. It consists of both kernel level and compact application benchmarks. Originally the kernel level benchmarks were designed to evaluate the performance of polymorphous computing architectures (PCAs). However, MIT Lincoln Laboratory is currently using them to evaluate performance of the Cell processor, which consists of a PowerPC processor and 8 Synergistic Processing Elements (SPEs). The kernel benchmarks consist of signal processing kernels, information processing kernels, and a communication kernel. The compact application benchmark combines all of the kernels in the Scalable Synthetic Compact Application #3 (SSCA #3), a benchmark used in the HPCS program [5]. The SSCA #3 is based on an implementation of Synthetic Aperture Radar (SAR).

*This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

pMapper Mapping and Results

pMapper with the Cell machine model is used to produce mappings for the HPEC Challenge Benchmarks. This section presents the mapping results for one of the signal processing kernels, the convolution, and one of the information processing kernels, the genetic algorithm. Figure 2 is the Program Spec for the convolution benchmark.

```
%Initialize variables
1. N...; M...; K...;
2. A = rand(1, N, M, p);
3. B = zeros(1, N, M, p);
4. filter = rand(1, K);
%Iterate over the bank of convolutions
5. for ii = 1:M
6.    B(:,:,ii) = conv2(A(:,:,ii), filter);
7. end
8. B
```

Figure 2: pMapper bank of convolutions code.

This benchmark consists of a bank of convolutions. In the MATLAB code, this bank of convolutions can be represented with a 3-dimension array (line 2 in Figure 2). Note that the ***p*** in the constructor call is used to indicate to pMapper that the object should be considered for distribution. Once the array is created, each convolution can be represented by a FOR loop iterating over the third dimension of the array. The last statement in the program, with the omitted semi-colon, is a data display statement in MATLAB syntax. At this point, the data must be returned to the user. This causes the extraction and the mapping of the SFG. pMapper produces ideal linear speedup for this application as illustrated in Figure 3.

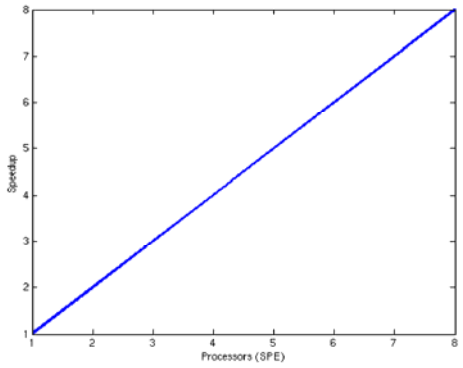

Figure 3: Convolution speedup.

pMapper produces an embarrassingly parallel mapping for this benchmark, i.e. divides the convolutions in the bank between all eight of the Cell SPEs. This is the best mapping for this benchmark and is the mapping chosen by an expert mapper.

The genetic algorithm (GA) benchmark requires the implementation a simple genetic algorithm which consists of population initialization, and then for each generation, selection of parents, crossover, and mutation. The mapping results for the genetic algorithm benchmark are more interesting. In this case the linear speedup is not achievable as the benchmark requires communication between each generation of the genetic algorithm. The code is not shown here to save space; however, let us note that the Program Spec in MATLAB stores the population as an N by M matrix where N is the size of the population and M is the length of the chromosome. Figure 4 shows the speedup curve produced by pMapper.

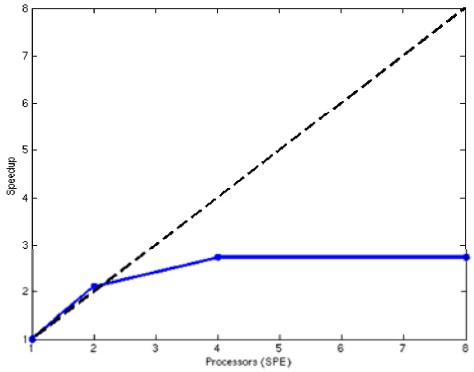

Figure 4: Genetic algorithm speedup.

In the case of this benchmark, pMapper only chooses to use three of the available 8 processors in order to balance communication and computation. Higher speedup is not achievable for this benchmark, so while the theoretical optimum is not achieved, pMapper does achieve the maximum possible speedup for this algorithm.

Summary

pMapper is an automatic mapping architecture that allows prediction of mapping onto and performance of multiprocessor embedded systems, including tiled architectures. HPEC Challenge benchmarks were developed for evaluating the performance of HPEC multiprocessor systems. Mapping the benchmarks onto the systems can be a laborious task. However, pMapper can automatically generate efficient mappings for the benchmarks by removing much of this complexity. The convolution benchmark mapped by pMapper yielded ideal linear speedup, the genetic algorithm mapping achieved the maximum speedup for the algorithm. More detailed results for both the convolution and the GA will be presented in the talk, as well as the results for other kernel benchmarks and the SSCA #3 compact application benchmark.

References

[1] N. Travinin, H. Hoffmann, R. Bond, H. Chan, J. Kepner, E. Wong, "pMapper: Automatic Mapping of Parallel Matlab Programs," *HPEC 2005 Workshop*, Lexington, MA, September 2005.

[2] A. Reuther, T. Currie, J. Kepner, H. Kim, A. McCabe, M. Moore and N. Travinin, "LLgrid: Enabling On-Demand Grid Computing with gridMatlab and pMatlab," *HPEC Workshop 2004*, Lexington, MA, September 2004.

[3] T. Chen, R. Raghavan, J. Dale, E. Iwata, "Cell Broadband Engine Architecture and its first implementation," http://www-128.ibm.com/developerworks/power/library/pa-cellperf/, November 2005.

[4] R. Haney, T. Meuse, J. Kepner, J. Lebak, "The HPEC Challenge Benchmark Suite," *HPEC 2005 Workshop*, Lexington, MA, September 2005.

[5] HPCS: High Productivity Computer Systems, http://www.highproductivity.org.