Self-Adapting Software as a Means of Meeting the Multicore Programming Challenge

Konstantin Nedovodeev
St-Petersburg University of Aerospace Instrumentation
Saint-Petersburg, Russia
Konstantin.Nedovodeev@guap.ru

Abstract

Recently, according to the growth of internal complexity of modern computing platforms, a new approach has been shaped – automatic tuning of data processing algorithms while meeting the underlying platform characteristics. Such libraries are often called in literature as “auto-tuners” and “self-adapting software”. Despite the fact that there exist many adaptive libraries, the task of adaptive library construction for heterogeneous multicore processor (HMP) based systems seems to stay unsolved. To fill the gap between hardware and software a new approach is offered, which is based on automatic program source synthesis. An approach is incorporated into an initiative, called SAMPL (*Self-Adapting Matrix Processing Library*). This article gives an overview of SAMPL.

INDEX TERMS: SELF-ADAPTING SOFTWARE, MULTICORE, SAMPL.

I. INTRODUCTION

Emergence of multicore processors made it much harder to achieve high performance [1]. It's widely known that using abstraction is a way to struggle against complexity. One way to increase level of abstraction while constructing the software is to use *software libraries*. The most widely applicable approach to create a library is to hand-craft different versions of it for different versions of computing platforms. The aforementioned approach has the following shortcomings: poor portability, and high time-to-market for each version of the library. Recently, according to the growth of internal complexity of modern computing platforms, a new approach has been shaped – automatic tuning of data processing algorithms while meeting the underlying platform characteristics. Such libraries are often called in literature as “auto-tuners” and “self-adapting software” [3, 4]. Among others the following representatives could be mentioned: ATLAS [9], FLAME [6], FFTW [5], PLASMA [2], etc. There is a lot of evidence that automatic software adaptation is an essential part of future software [3, 4].

II. MAIN PART

Despite the fact that there exist many adaptive libraries [2, 5, 6, 9], as far as author knows, the task of adaptive library construction for heterogeneous multicore processor (HMP) based systems [7] still stays unsolved. To fill the gap between hardware and software a new approach is offered. This approach is based on a formal ground developed by the author earlier, and is incorporated into an initiative, called SAMPL (*Self-Adapting Matrix Processing Library*). In order to adapt to the platform characteristics SAMPL automatically generates source codes of its subprograms (see fig. 1).

Tuning and adaptation of a subprogram is a multi-stage process in SAMPL. During the first stage information about subprograms to be called is extracted from the user program source code, calls of subprograms that could be generated by SAMPL are extracted, the strings of their formal parameters are parsed according to the header files provided by

SAMPL. During the second stage each call of the SAMPL-program is linked with an instance of an internal representation, which is called an *abstract macro-flow graph* (AMFG) [8]. Abstract macro-flow graph mirrors data dependencies between different calls of subprograms, whose main responsibility is in managing data processing inside local memory of a computational core of an HMP (from now on we mention such subprograms as *computational granules*). The following stage implies resource allocation, namely: mapping the computational granules calls to the computational cores of HMP, distribution of information transfers among information exchange channels involving different DMA-cores, and memory allocation.

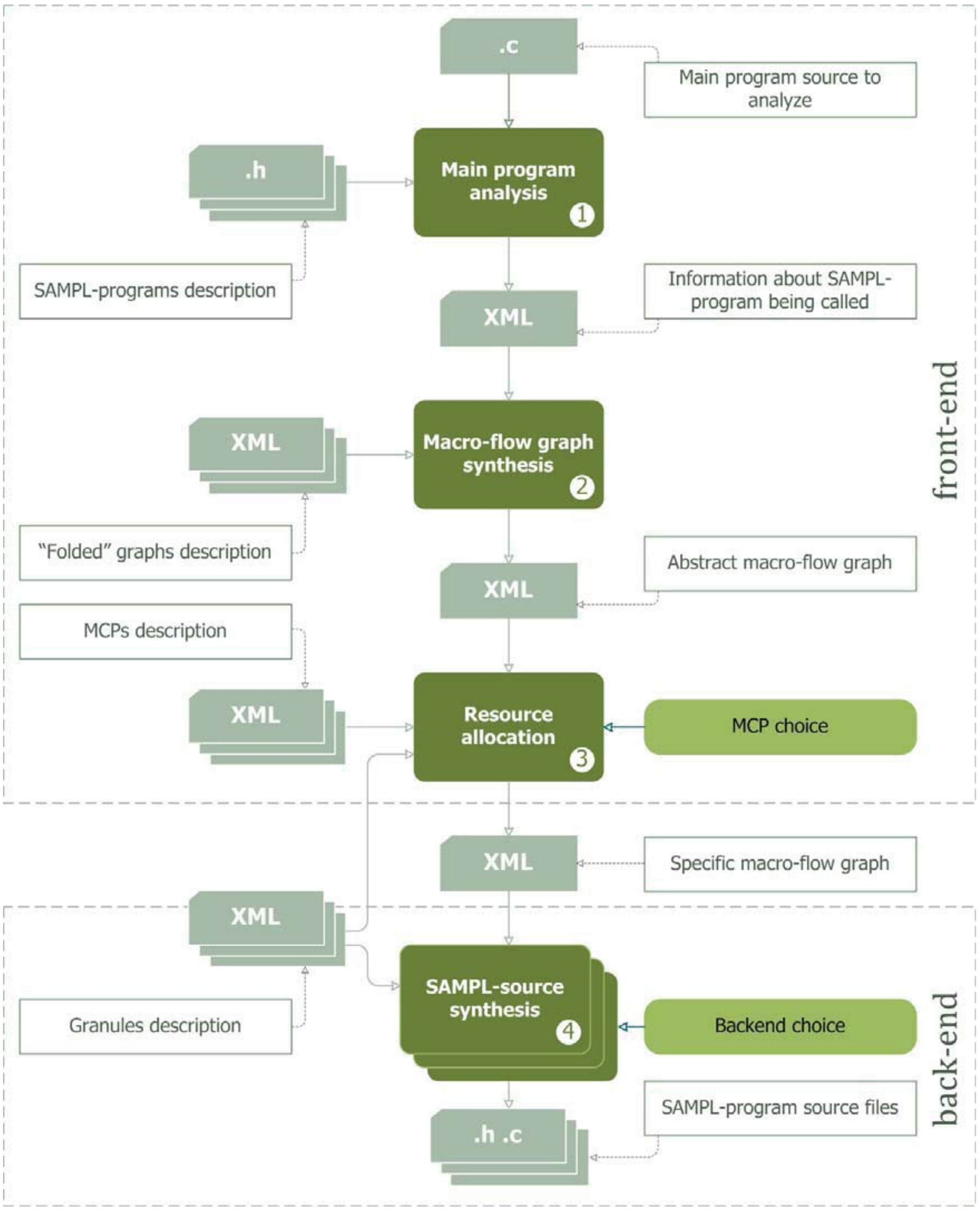
.c
Main program source to analyze
.h
Main program analysis 1
SAMPL-programs description
Information about SAMPL-program being called
XML
XML
Macro-flow graph synthesis 2
"Folded" graphs description
Abstract macro-flow graph
XML
MCPs description
XML
Resource allocation 3
MCP choice
front-end
XML
Specific macro-flow graph
XML
SAMPL-source synthesis 4
Granules description
Backend choice
SAMPL-program source files
.h .c
back-end

Fig. 1. The flowchart of SAMPL

The resulting information is incorporated into an exemplar of internal representation of a SAMPL-program, which is called *specific macro-flow graph* (SMFG). Specific macro-flow graph is tightly bound to the architecture of an HMP, which the SAMPL-program should be run on. The aforementioned representation encapsulates enough information to synthesize the source files.

The source code of a subprogram for the control core [7] is defined in term of abstraction layer components. The layer of abstraction is called SIL (*SAMPL Intermediate Layer*). The SIL incorporates those program components, whose tasks are in: managing the computational granules calls, managing DMA-transfers, and managing coarse-grained synchronization. For the computation and data transfer to overlap the multi-buffering scheme is used in local memory allocation. While generating the source codes of a SAMPL-program, for each node of the SMFG the SIL component call is added.

An approach which is incorporated in SAMPL gives the user an opportunity to vary the internal structure of a SAMPL-program in accordance with the following characteristics:

- workload distribution configuration;
- number of the computational cores in HMP;
- structure and characteristics of an information exchange subsystem;
- characteristics of a memory subsystem.

III. Conclusion

A new approach to adaptive libraries construction (SAMPL) was presented in this article. The aforementioned approach is based on an automatic source codes synthesis. Synthesis in SAMPL is a multi-stage process; it includes an internal program representation synthesis and static resource allocation. Internal representation does not depend on particular multicore processor characteristics. Internal representation mentioned takes into account the fact that there is a DMA-transferring mechanism incorporated into the multicore processor. While allocating resources the program is automatically restructured according to the platform characteristics. Layer of abstraction (SIL) makes it easier for SAMPL to be portable among different multicore processors. To adjust SAMPL to a new multicore processor it will probably be needed to develop SIL components and computational granules.

So far the back-end for the domestic “Multicore” family of processors [10] is being finished. The next milestone is the resource allocation component implementation.

Acknowledgment

The author would like to thank his scientific adviser Yuriy Sheynin for useful comments.

References

[1] Herb Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software," appeared in *Dr. Dobb's Journal*, March 30, 2005.

[2] Agullo E., Demmel J., Dongarra J., Hadri B., Kurzak J., Langou J., Ltaief H., Luszczek P., Tomov S., "Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects," *Journal of Physics: Conference Series*, vol. 180, 2009.

[3] Asanovic Krste et al., "The Landscape of Parallel Computing Research: A View from Berkeley," *Technical Report No. UCB/EECS-2006-183*, University of California, Berkeley, December, 2006.

[4] David Bailey, Allan Snavely, “Performance modeling, metrics, and specifications,” in *Workshop on the Roadmap for the Revitalization of High-End Computing*, June, 2003.

[5] Frigo, M. and Johnson, S.G., “FFTW: An Adaptive Software Architecture for the FFT,” *Proc. ICASSP*, vol.3, 1998.

[6] John A. Gunnels, Fred G. Gustavson, Greg M. Henry, and Robert A. van de Geijn, "FLAME: Formal Linear Algebra Methods Environment," *ACM Transactions on Mathematical Software* 27(4), December, 2001

[7] Nedovodeev K., Sheynin Y., “High-performance processing of large data chunks in heterogeneous multicore processors,” *Electronic components*, №9, 2006 (in russian)

[8] Nedovodeev K., “Using the dataflow model for the description of a tiled algorithm,” *Proceedings of all-Russian forum of students, PhD students and young researchers «Science and innovation in technical universities»*, Saint-Petersburg, October, 2007 (in russian)

[9] R. C. Whaley, A. Petitet, J. J. Dongarra, “Automated Empirical Optimization of Software and the ATLAS project,” *Parallel Computing*, vol. 27, №1-2, 2001

[10] Solokhina T., Alexandrov Y., Glushkov A., etc., “Domestic triplecore digital signal microcontrollers with 1.5 GFlop/s performance,” *Electronic components*, №6, 2005 (in russian)