Self-adapting software for meeting the multicore programming challenge

Konstantin Nedovodeev,
research assistant,
SUAI, HPCaNTI

Architecture

Multicore processors (MPSoCs) in question comprise the following components:

- Control core;
- Computational cores;
- DMA-cores.

explicitly-managed memory (EMM) architecture;
small local (scratchpad) memories;
heterogeneous cores (cntrl + comp. + DMAs);
big common (off-chip) memory;

Representatives:
OMAP, DaVinci TI, Cell IBM+Toshiba+Sony, Diopsis Atmel, “Multicore” Elvees (Russia)

Class of problems to be solved

1. Sequence of algorithm steps does not depend on values of particular data elements;
2. Manipulating matrices, vectors and scalars in mathematical sense;
3. Problem could be formulated in terms of block(tile) processing steps.

Matrix-vector product is an example (BLAS):

$y = \alpha A x + \beta y$ (*matrix form*), $A \in R^{NxN}$, $x,y \in R^N$, $\alpha, \beta \in R$

$(y_i = \alpha A_{i,0} x_0 + \beta y_i$
$(y_i = \alpha A_{i,j} x_j + y_i \forall 1 \leq j \leq N')$
$\forall 0 \leq i \leq N')$ (*block form*),
$N' = ceil(N / NB)$, NB – blocking factor

Issues addressed by an approach

1. workload distribution among computational cores;
2. information transfers distribution among different channels:
3. trying to reuse data in the local store (locality-awareness);
4. trying to use LS <-> LS (bypassing) as much as possible;
5. using multi-buffering to hide memory latency;
6. local memory allocation without fragmentation;
7. managing synchronization of parallel processes;
8. avoiding WaW, WaR dependencies by allocating temporary store in common memory (results renaming).

An approach

1. Using graph representation (“folded” graph) of mass problem;

mirrors the informational structure of a problem;

2. Offline construction of representation of particular problem (abstract macro-flow graph (AMFG));

corresponds to particular blocking choice;

3. Using hypergraph representation of an HMP;

representation comprises:

- communication subsystem characteristics;
- memory subsystem characteristics;
- characteristics of different cores;

4. Resource allocation using particular hypergraph and AMFG;

computation mapping + computation scheduling + transfers mapping + transfers scheduling + memory allocation;

An approach (continued)

5. Automatic source code generation;

source codes are written in C language;

6. Run-time library support as an abstraction layer;

Abstraction Layer (AL) is responsible:

1. managing computational cores operation;
2. managing DMA transfers;
3. managing synchronization;

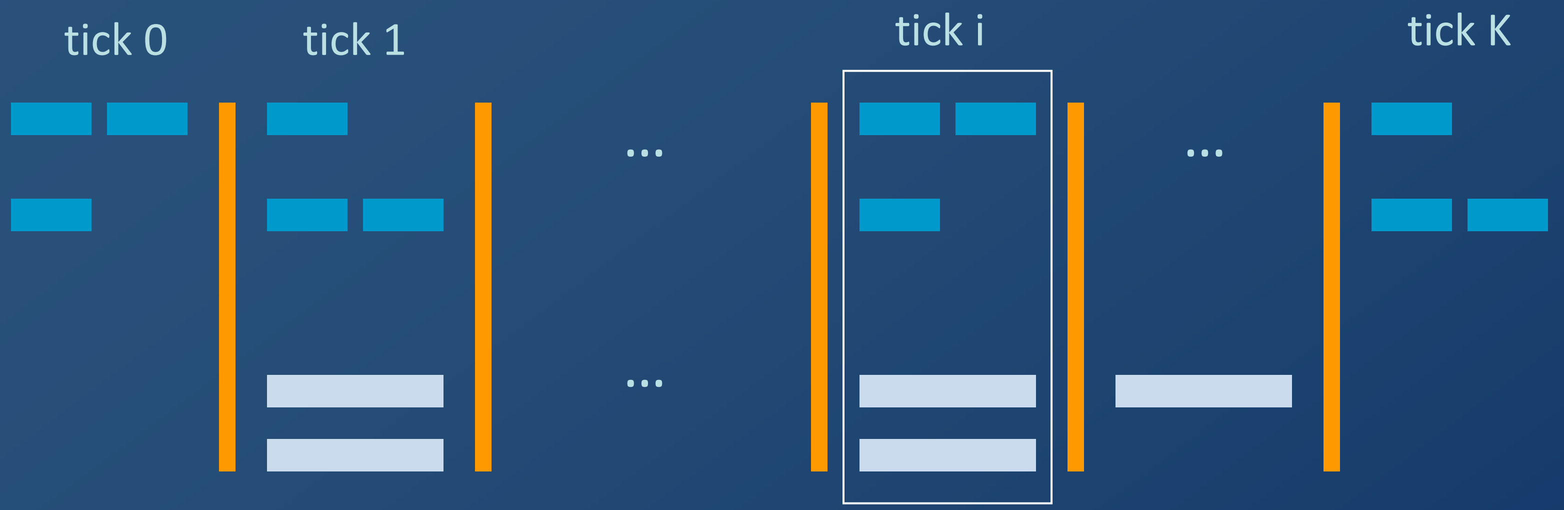
7. Using hand-crafted subprograms (computational granules) for computational cores;

highly optimized codes fit entirely into the local store.

Source code snippet (C language)

```
#define HMP_MODEL_NAME
#include "include\SIL.h"
#include …
…
#ifndef DMA_NUM  #define DMA_NUM M #endif
#ifndef DSP_NUM  #define DSP_NUM N #endif
...
/* SAMPL-program implementation */
retType samplProgramName(..., par_N_type parN){
/* Creating fragments of the DMA-queues tick 0 */
  ...
  DSP_0_INIT(…);
  ...
  DMA_0_INIT_RUN(…);
  ...
/* Creating fragments of the DMA-queues tick 1 */
  ...
  SAMPL_BARRIER(DMA_NUM);
  DSP_0_RUN();
  ...
  DMA_0_RUN();
  ...
/* Creating fragments of the DMA-queues tick 2 */
  ...
  SAMPL_BARRIER(DMA_NUM + DSP_NUM);
  ...
  return retVal;
}// end sampl_program_name()
...
#undef DMA_NUM
#undef DSP_NUM
#undef HMP_MODEL_NAME
```

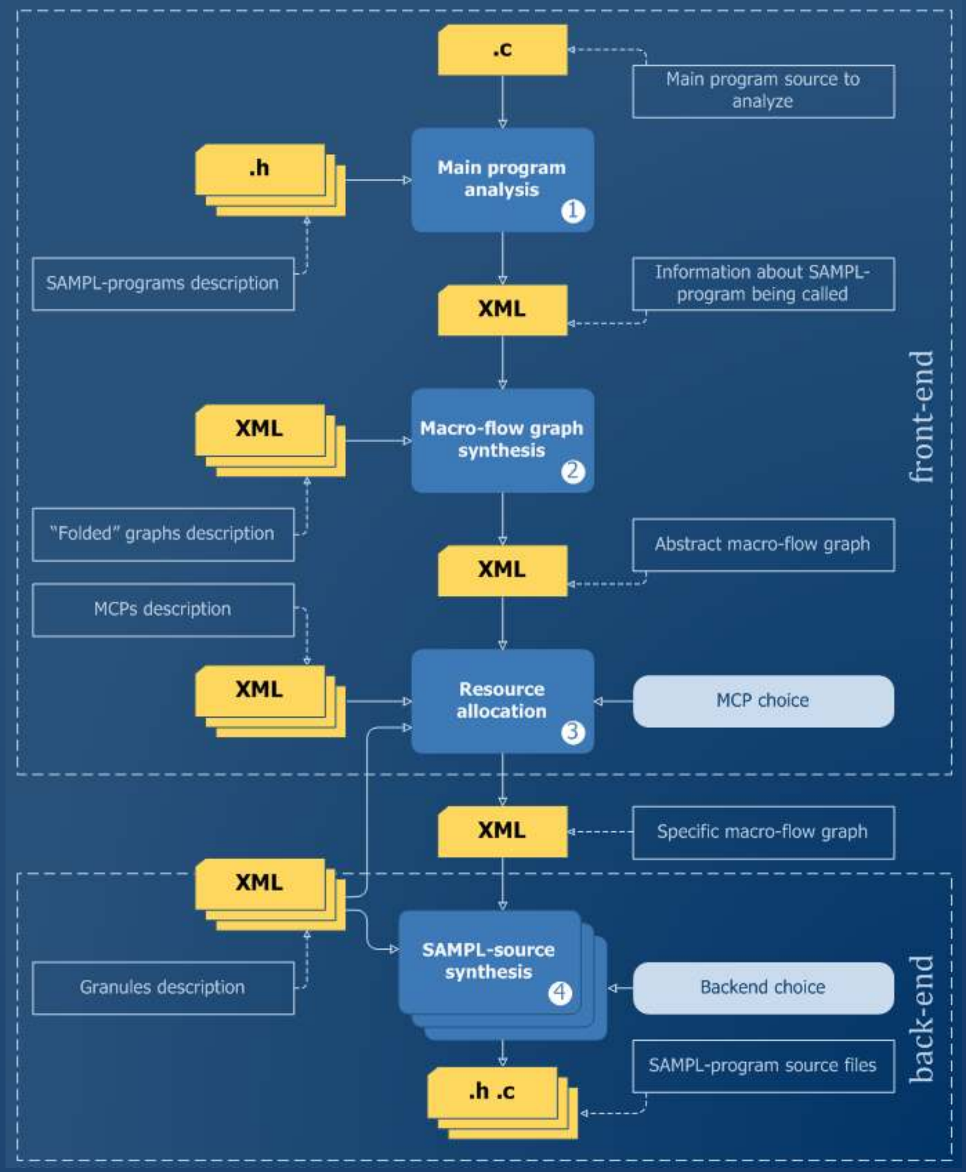
Flowchart of program execution



Computation ⬜
Transfers 🟦
Barrier 🟧

Overlapping computation and communication by using multi-buffering of local store

Self-Adapting Matrix Processing Library (SAMPL)


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

Adaptation in **SAMPL** (**S**elf-**A**dapting **M**atrix **P**rocessing **L**ibrary) is a multi-stage process. Parallel application program is synthesized through the usage of formal representation of computation, namely macro-flow graph which is generated automatically.

Abstract macro-flow graph depends solely on data partitioning but not on a specific multicore processor characteristics.

While performing scheduling and allocation a specific macro-flow graph is constructed which is related to a particular multicore processor chosen.

After the **schedule** is constructed it contains all the information needed (computational tasks distribution across the computational cores, data exchanging distribution among DMA-cores and information exchange channels, etc.) to synthesize an application program to perform parallel data processing on a specific multicore-based system.