Orchestration: Turning material breakthroughs into application performance

Jeronimo Castrillon
(on behalf of the Orchestration team)
TU Dresden – Cfaed
Chair for Compiler Construction
jeronimo.castrillon@tu-dresden.de
Cfaed: Research program (revisited)

Goal

to explore new technologies for electronic information processing which overcome the limits of CMOS technology
Orchestration: Materials for performance

Cfaed: Research program (revisited)

Goal
“to explore new technologies for electronic information processing which overcome the limits of CMOS technology”

Cfaed: Research program (revisited)

Information Processing

Systems research to handle heterogeneity

Devices & Circuits

Materials & Functions

CMOS (industry focus)

A Silicon Nanowires

B Carbon

C Organic

D Biomolecular Assembly

E Chemical

F Orchestration

H HAEC: Highly Adaptive Energy-Efficient Computing

G Resilience

Biology to inspire solutions

Material research for post-CMOS technologies
Orchestration Path

Mission
Turn materials breakthroughs into application performance
Orchestration Path

Underlying algorithms: self-adjusting
Code synthesis: variant generation
Heterogeneous “template”
Cross-layer strategies

Materials-Inspired Paths (Paths A – E)

Programming abstractions
“Micro-kernels”
System analysis: modeling, verification & reasoning
Orchestration Path: the team

- Prof. Uwe Aßmann
- Prof. Franz Baader
- Prof. Christel Baier
- Prof. Jeronimo Castrillon
- Prof. Gerhard Fettweis
- Prof. Christof Fetzer
- Prof. Jochen Fröhlich
- Prof. Hermann Härtig
- Prof. Wolfgang Lehner
- Prof. Wolfgang E. Nagel
- Dr. Marcus Völpa
- Prof. Axel Voigt
Orchestration Path: the team (2)

- PhD/Postdocs
  - Nils Asmußen
  - Andres Goens
  - Sebastian Haas
  - Dirk Habich
  - Immo Huismann
  - Tomas Karnagel
  - Sven Karol
  - Sascha Klüppelholz
  - Linda Leuschner
  - Matthias Lieber
  - Siqi Ling
  - Steffen Märker
  - Johannes Mey
  - Benedikt Nöthen
  - Michael Raitza
  - Norman Rink
  - Annett Ungethüm
Outline

- Introduction
- HW/SW stacks
  - Orchestration stack
- Outlook
- Concluding remarks
What is a stack

- Software components that form a complete platform (where you can run applications on)
  - Web app: Operating system, web server, database, programming language
- Provide means to handle complexity
HW/SW Stack: Example 1

Android sack
Source: https://source.android.com/source/index.html
HW/SW Stack: Example 2

HSA Stack
Source: Heterogeneous System Architecture
http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/
Stacks and wild heterogeneity

- Abstract/hide complexity
- Leave as much visible as needed for efficiency
- Emerging techs.: rethink established layers
  - Languages, compilers, operating systems, ...
Example: Compiler – Runtime

Compiler

“protocol” assumptions

Runtime environment

Code

Data

Stack

Heap

0x0

Endurance problem: try not to use same locations that often!

0xF..F
Example: Flash FTL

File system

↓

Hard disk

Flash Translation Layer (FTL)
- Mapping
- Wear leveling

→

File system

Flash drive
Outline

- Introduction
- HW/SW stacks
- **Orchestration stack**
- Outlook
- Concluding remarks
Recall: orchestration stack

- Heterogeneous materials
  - In Cfaed: Along with orchestration (see other DMA presentations)
- Research plan
  - Start with heterogeneous CMOS
  - Tiled for scalability, integrability
  - Pilot projects with materials as they become available
Applications/Drivers

- Mobile communication: typically run on heterogeneous
- Data-bases: benefits of heterogeneity (see below)
- HPC: high level languages, abstractions for scalability, productivity and portability
  - Adaptive Finite Element Methods (FEM)
  - Computational Fluid Dynamics (CFD)
Applications/Drivers (2)

- Adaptive Finite Element Methods (FEM)
  - Working with multiple-meshes
  - Communication across subdomains
  - Requires repartitioning ➔ load balancing
  - Working on
    - Portable *templatized* library
    - First results on heterogeneous platforms

[Witkowski15]
Applications/Drivers (3)

- Computational Fluid Dynamics (CFD)
  - Large applications \(10^{10}\) unknowns
  - Communication/computation is key

- Working on
  - Heterogeneous implementation
  - More versatile fundamental concepts

[Huismann15]
Hardware

**Tomahawk: heterogeneous CMOS**

- Tiled (with NoC) for scalability
- Allows to integrate wildly heterogeneous components
- CoreManager (CM): HW support for fine-grained task dispatching

[Arnold13]
Hardware

- DB-processor for sorting algorithms

Selectivity:

[Arnold14]

Final processor

+1 Load-Store unit

+ Partial loading

+ Extended ISA

Data bus: 32->128 bit
Operating Systems

- Microkernel for heterogeneity
  - Isolation (at NoC level)
    - Dynamic compartments
    - Isolation at network boundary
  - Remote control: no need for OS everywhere
    - Simplify integration of novel materials
    - OS services on remote cores
Operating Systems

- Microkernel for heterogeneity

- Isolation & remote control

[Asmussen15]
Compilers

- Parallel dataflow programming languages and compilers

```plaintext
__PNkpn AudioAmp __PNin(short A[2]) __PNout(short B[2])
    __PNparam(short boost){
        while (1)
            __PNin(A) __PNout(B) {
                for (int i = 0; i < 2; i++)
                    B[i] = A[i]*boost;
            }
    __PNprocess Amp1 = AudioAmp __PNin(C) __PNout(F) __PNparam(3);
    __PNprocess Amp2 = AudioAmp __PNin(D) __PNout(G) __PNparam(10);
```

[Castrillon14]
Compilers

- SW synthesis for Tomahawk
  - Static/dynamic mapping
  - Automatic code generation
**Skeletons**

- Basic building blocks for parallelism
- “Heterogeneous” programming
  - Different possible implementations of same principles (CUDA, Fortran, ...)
  - Advanced “pre-compiler” to generate and manage variants

⇒ Productivity
Skeletions

- Basic building blocks for parallelism

**Style Definitions**

- **Style**: OpenACC
- **Type**: parallelization
- **Fragment**
  ```
  Target: DoConcurrent
  !$acc loop private(#PRIVATE_VARS#)
  #INNER#
  !$acc end do
  ```

**Sequential Program**

```plaintext
do concurrent (i = 1:n, j = 1:m)
  mat(i,j) = x - y(i,j,k) * 2
end do
```

**Parallel Program**

```plaintext
!$acc parallel
!$acc loop private(i, j)
do k = 1, num_ele
  do i = 0, po
    mat(i,j) = x - y(i,j,k) * 2
  end do
end do
!$acc end loop
!$acc end parallel
```

**Style Application Recipe**

```plaintext
RECIPE {parallelization: OpenACC} OpenMP
```

[Karol15]
Algorithms

- Raise the level of abstraction
  - High-level **domain specific languages** (more later)
  - Algorithmic transformations: more optimization room
  - Algorithmic specification: allow for adaptability
Formal analysis

- Probabilistic model checking (PMC)
  - Analyze approaches across layers of the stack
  - Work on models for heterogeneous systems
  - Examples:
    - Probability of finishing N tasks in T time
    - Probability of finishing N tasks with energy budget B
    - Trade-off analysis cost-utility (energy vs. performance)
  - Model protocols: Spinlock, barriers, eBond, ...
Formal analysis

- Probabilistic model checking (PMC)

eBond

Minimal expected energy vs. Maximal required bandwidth

[Baier14]

Speaker: J. Castrillon
Date: September 21, 2015
Page: 30
Outline

- Introduction
- HW/SW stacks
- Orchestration stack
- Outlook
- Concluding remarks
Current and future work

- Further work on languages and methodologies
  - Computational biology
  - Chemical information processing

- Technology analysis at system-level
  - Pros (USPs) and cons
  - Attempt skip incremental improvement
  - System-level benefit based on assumptions
Language for computational biology

- Particle and mesh-based Simulations
  - Discrete models of physical phenomena
- Goal: Abstract programming language + optimization for heterogeneous HPC-clusters
Language for computational biology

import LaplacePSE2D from stencils as Lap
import BcdefPeriodic from topologies as bcdef
import RungeKutta from schemes as rk4

module GrayScott
  external real kRate= 1.0;  "reaction rate"
  external real F;  "reaction parameter"
  external real Du= 1.0;  "diffusion constant of U"
  external real Dv= 1.0;  "diffusion constant of V"

  phase initialize
    << ... >>
  end initialize

  phase solve
    field <real , l> U;
    field <real , l> V;
    integer t;

    |6U| = Du * V^2 * U - U * V^2 + F *(1 - U);
    6t
    |6V| = Dv * V^2 * V + U * V^2 - (F + kRate) * V;
    6t

  end solve

  phase finalize
    << ... >>
SiNW: Components and reconfigurability

- Edge of SiNW over other technologies
  - P/N reconfigurability
  - Multi-gate devices (with low latency penalty)

- Components based on assumptions
  - New uArchitectures?
  - New pipeline structures and compiler optimizations?
Outline

- Introduction
- HW/SW stacks
- Orchestration stack
- Outlook

- Concluding remarks
Summary

- Orchestration path: Multi-disciplinary effort to turn *materials breakthroughs* into *application performance*
- HW/SW stack to handle heterogeneity
  - Using CMOS as vehicle
  - Prepare for "wildly" heterogeneous systems
- Interact with material experts, understand technologies and analyze at system-level
Thanks!

Retreat: Orchestration and resilience paths (03.2015)
References


