Programming Support for Heterogeneous Parallel Systems

Siegfried Benkner
Department of Scientific Computing
Faculty of Computer Science
University of Vienna
http://www.par.univie.ac.at

Outline

- Introduction
- The Challenge of Heterogeneous Manycore Architectures
- The EU Project PEPPHER
- Performance Portability & Programmability
- Conclusion

S. Benkner, Department of Scientific Computing, University of Vienna. International Workshop on Peta-Scale Computing Programming Environment, Languages and Tools (WPSE 2010), Kyoto, Feb. 18, 2010

Department of Scientific Computing

One of three departments at the **Faculty of Computer Science.**

Organization

- Software Science Group (Nordbergstrasse)
- Data Analysis Group (Universitätsstrasse)

Staff

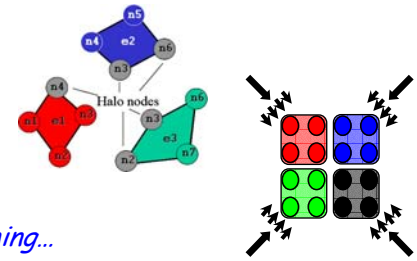
- 15 Faculty
- 25 Research
- 5 Admin/Support

Department of Scientific Computing

Parallel Computing / HPC

- Programming Models and Languages
- Compiler and Runtime Technologies
- Programming Environments and Tools

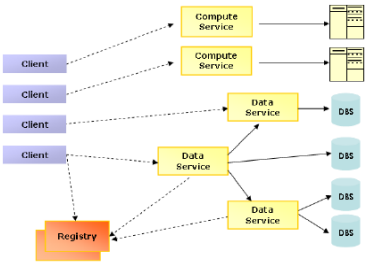
Vienna Fortran, HPF, HPF+, Hybrid Programming…



Grid/SOA/Cloud Computing

- Parallel Application Services
- On-demand supercomputing, QoS
- Data Virtualization & Integration & Mining

Grid Miner, Vienna Grid Environment, …

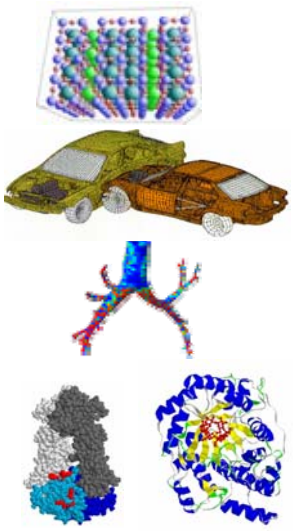


S. Benkner, Department of Scientific Computing, University of Vienna. International Workshop on Peta-Scale Computing Programming Environment, Languages and Tools (WPSE 2010), Kyoto, Feb. 18, 2010

Department of Scientific Computing

Application-oriented Research

- Crash Simulation
- Engine Design Opt.
- Weather Prediction
- Material Sciences
- Financial Optimization
- Transport Logistics
- Biomedical Simulations
- Molecular Dynamics
- Image Processing
- Bioinformatics



Research Infrastructure @ DSC

Clusters

- SUN X4100 Cluster, 288 processor cores, Infiniband
- Vienna Scientific Cluster, SUN X2270, 3488 cores, Infiniband QDR

  (shared between three universities in Vienna)

Heterogeneous Multicore Systems

- IBM BladeCenter QS22/LS22 (8x PowerXCell 8i, 2x Opteron), 80 cores
- 4 Sony Playstation3, NVIDIA GPUs, AMD GPUs
- GPU Cluster, 16x Nehalem X5550, 16x NVIDIA C1060,

  16x NVIDIA C2050 (Fermi) soon

Multicore (R)evolution

Core Scaling instead of Frequency Scaling

- Doubling of cores every 18/24 month
- Problem of extracting performance shifted to programmers (ILP→TLP)
- Need strong scaling to increase application performance

Multicore Pervasivness

- Parallel computing everywhere
- Embedded, General Purpose, and HPC
- Increasing architectural diversity and complexity

Homogeneous versus *Heterogeneous* Multicore Architectures

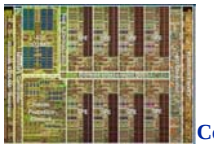
Heterogeneous Manycore Architectures

Number of ***active* cores** on chip soon **limited** by power constraints.

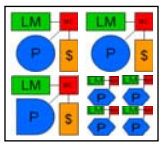
- ☞ different types of cores on a chip, or
- ☞ same cores but different clock frequencies, or
- ☞ partitioning groups of homogeneous cores, or …

Examples:

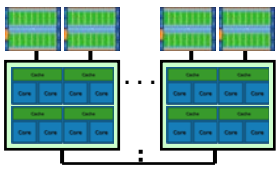
- Cell Processor: PPU + 8 SPUs
- SARC Research Processor Architecture
- CPU + GPU/Accelerators
- FIRST (GRAPE), Roadrunner (Cell), TSUBAME(GPU), …



Cell BE



SARC



GPU Cluster

Programming Heterogeneous Manycore

Much harder than for homogeneous systems

- Need of allocating and managing resources
- Explicit memory management (DMA transfers, double buffering, …)
- Functional partitioning of code for different core types
- Different memory models, ISAs, compilers, programming models

Current Solutions?

- IBM CellSDK, NVIDIA CUDA, ATI Stream SDK, OpenCL, …

Support for Computation Offloading

- Rapidmind, HMPP, PGI Accelerator, StarSs, …
- Codeplay Offload++

```
...
__offload {
   func1(); //compiles func1 and all functions it calls to run on the SPU
}
func2() // runs on the PPU
...
```


Programming Heterogeneous Manycore

Will we need new programming models?

- No „one-size-fits-all" model
- Need to integrate different models

Portability of major importance

- Increasing complexity of architectures
- Increasing architectural diversity

Performance Portability

- Consider different aspects not just FLOPs
- Power as important as performance

Programmability/Productivity

- Raise level of abstraction
- Automate low-level optimization tasks

Compositional Approach

Abstract Hardware Models

Abstract Performance Models

Adaptation
Auto-Tuning
Algorithmic Choice

EU Project PEPPHER

Performance Portability & Programmability for Heterogeneous Manycore

- EU ICT Call 4, Computing Systems; 3.5 M€; Start: Jan. 2010, 36 Months;
- Coordinated by Universität Wien, Department of Scientific Computing

Goal: Enable efficient, productive and portable programming of heterogeneous multicore systems.

Holistic Approach

- Higher-Level Support for Parallel Program Development
- Algorithms & Data Structures
- Advanced Compilation & Auto-tuning
- Runtime Systems
- Hardware Mechanisms

Crosscutting Domains: Embedded – General Purpose – HPC

http://www.peppher.eu

PEPPHER Partners

Universität Wien, Austria (Coordinator) [S. Benkner, J. Traeff, S. Pllana]

Chalmers Tekniska Högskola AB, Sweden [P. Tsigas]

Codeplay Software Limited, UK [A. Richards]

INRIA, France [R. Namyst]

Intel GmbH, Germany [H.C. Hoppe]

Karlsruher Institut für Technologie, Germany [P. Sanders]

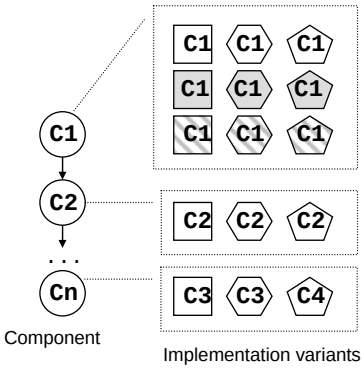
Linköpings Universitet, Sweden [C. Kessler]

Movidius Ltd., Ireland [D. Moloney]

PEPPHER – Key Research Issues

Compositional Approach

- Parallel components with annotations to capture non-functional properties
- Support different programming models/ vendor-specific APIs/ algorithms
- Coordination & Composition
- Mainstream vs. Expert Programmer



Advanced compilation techniques

- Compilation to portable low-level substrate (e.g. OpenCL)
- Auto-tuning, runtime feedback, …

PEPPHER – Key Research Issues

Adaptive algorithms and data structures

- Support for lock-free programming
- Parameterized wrt. heterogeneous multi-core architecture models
- Amenable to static/dynamic adaptation

Advanced Runtime Support

- Data-aware scheduling for heterogeneous architectures
- Collection of runtime data to feed back into compilation and auto-tuning

Hardware Issues

- Abstract architecture models
- HW support for performance tuning, synchronization and scheduling

Performance Portability?

Notions for performance portability required!

> **Implementation I of some algorithm runs on architecture A with efficiency X.**
>
> **Q: Will I run on architecture B with efficiency Y, with some simple relation between X and Y?**
>
> **A: I may not be suitable for B or may not run at all.**

cf. Self-Consistent MPI Performance Guidelines, Traeff et al, IEEE TPDS, 2009.

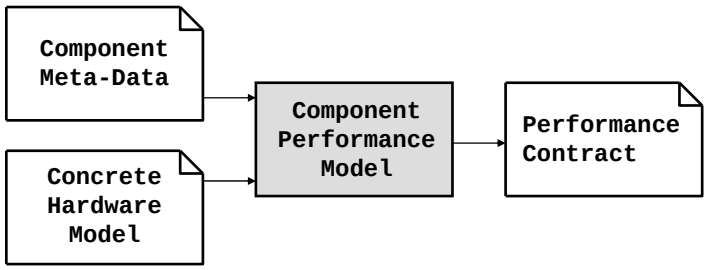
Strategies for enhancing performance portability

- Specify tasks at a **higher level of abstraction** → components
- Provide framework that supports **adaptation of components**
- Provide support for **algorithmic choice** and selection of data structures
- Utilize advanced compilation, **auto-tuning**, and runtime scheduling

Performance-Aware Component Model

Each component is associated with a ***performance model*** that is parameterized with ***component meta-data*** and a ***hardware model***.

- **Meta-data** captures non-functional properties relevant for selecting best component implementation. (problem size, data layout, etc.)
- **Hardware model** specifies main HW/SW characteristics (cores, memory, …)
- **Performance contract** should include (relative) runtime, power, … estimates



Component Implementations

Components written in C/C++ are parallelized using different low-level interfaces (OpenMP, threads, task model, …).

For each component multiple implementation variants may exist, that offer different performance characteristics (e.g. performance/power tradeoff) and may run on different core architectures.

Implementation variants may utilize different algorithms and/or data structures from PEPPHER library.

Implementation variants either written by expert programmers or generated (semi-)automatically by means of auto-tuning.

Selection and scheduling of "best" component variant done at runtime.

Compilation and Auto-tuning

☐ **Compilation**

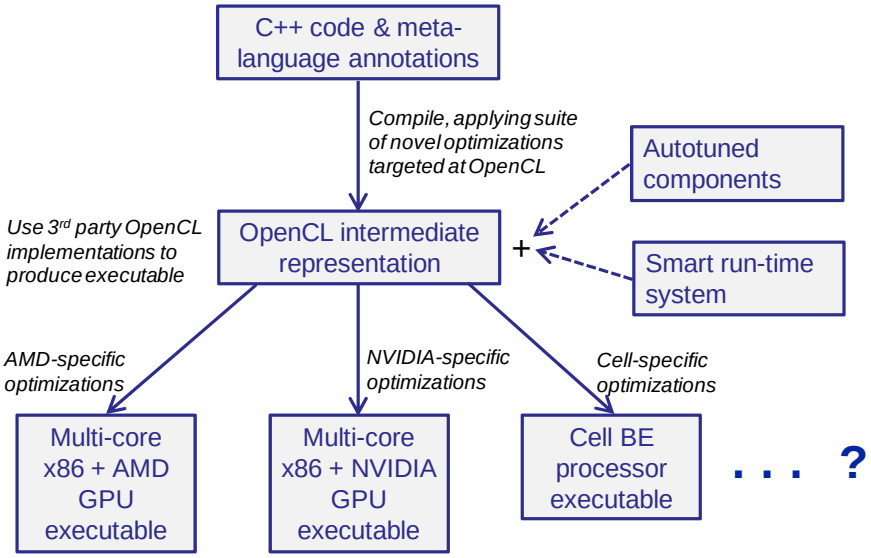
- Compile Annotated C++ to **OpenCL** as a low-level portability substrate.
- Automatic **generation of glue-code** for component selection.
- **Optimization of data management** in heterogeneous multi-core systems

☐ **Auto-Tuning**

- Automate intricate task of optimizing component implementations for a specific architecture.
- Optimization space for heterogeneous architectures is unmanageable even by expert programmer.
- Extend auto-tuning techniques by **integrating abstract hardware models**

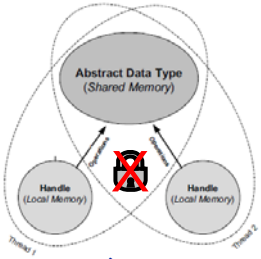
Related Work: ATLAS, Spiral, FFTW, …

Compilation and Auto-tuning



Adaptive Data-Structures & Algorithms

- Toolbox of **non-blocking shared data structures and parallel algorithms** written by expert programmers.
  E.g: stack, queue, list, dicitionary, …



- Parameterize algorithms and data structures with architecture models, (e.g. degree of parallelism, memory usage, …)

- Investigate static and dynamic adaptation and auto-tuning methods.

Based on existing work:

- Parallel C++ STL for GNU compiler (P. Sanders, Karlsruhe)
- Lock-Free Shared Abstract Data Types (P. Tsigas, Chalmers)

Advanced Runtime Support

- Selection of component variants based on available hardware resources.
- Data-aware runtime scheduling.
- Capture & feed-back of runtime information to higher levels.
- Support for calibration of performance models.
- Support for auto-tuning mechanisms.

Work led by R. Namyst (U. Bordeaux/INRIA)

Hardware-related Issues

- **Abstract HW models** for different classes of heterogeneous architectures
  - Memory/Processor Topology
- Investigate **HW-support mechanisms** for
  - performance tuning,
  - data-structures and synchronization,
  - memory management and utilization,
  - runtime scheduling.
- Simulation and experimental evaluation on prototype board developed by partner Movidius (www.movidius.com).

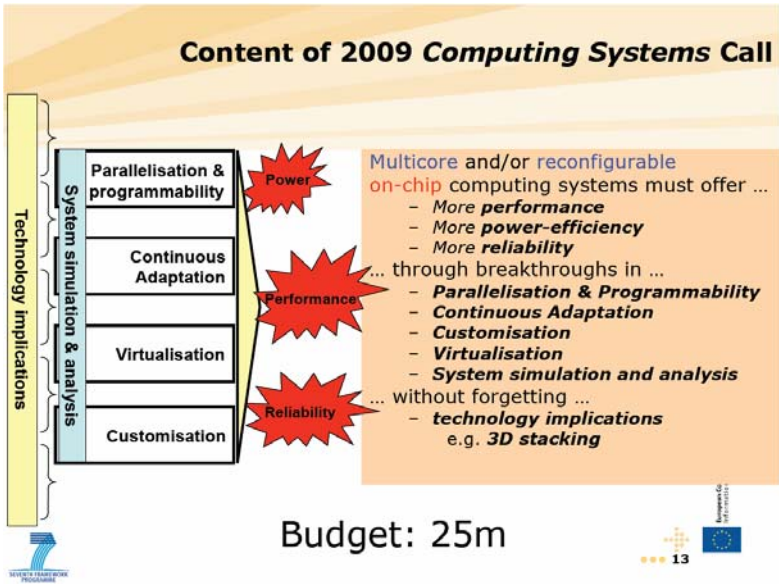
Heterogeneous Multicore ←→ Distributed Systems

- Programming models
  - MPI, OpenMP, threads, PGAS ?
  - GPU specific: Cuda, Ng, OpenCL …
  - Streaming Languages
- Computation partitioning
  - host processor
  - accelerators, special cores/units
- Data management
  - to/from shared memory; local stores
  - between functional units
  - impact on scheduling

- Programming models
  - Sequential
  - MPI, OpenMP, threads, PGAS
  - Components, Services, Workflow
- Computation partitioning
  - Coarse-grain tasks
  - Sequential and parallel tasks
- Data management
  - moving files btw. machines
  - management of intermediate data
  - impact on scheduling

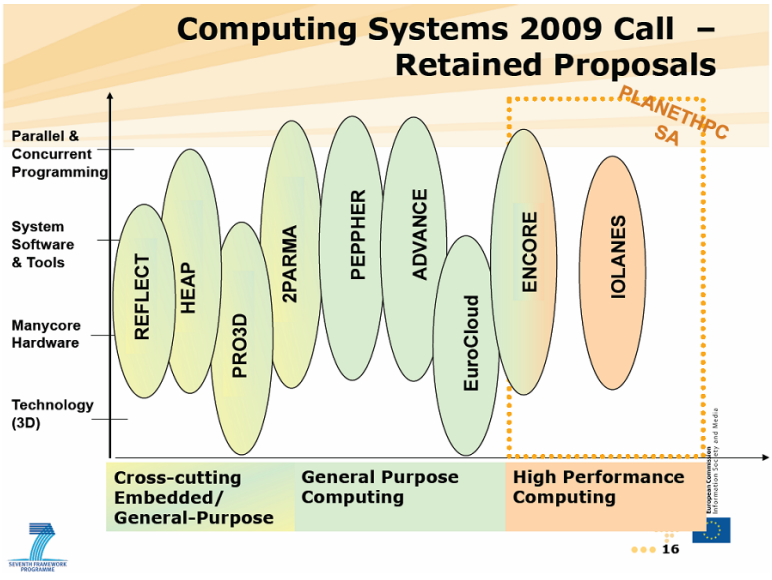
Compose applications from components!

PEPPHER – Related European Projects



Source: Dr. Panagiotis Tsarchopoulos, Computing Systems Objective Coordinator, European Commission

PEPPHER – Related European Projects



Source: Dr. Panagiotis Tsarchopoulos, Computing Systems Objective Coordinator, European Commission

Related Research Efforts

- HiPEAC NOE (Europe)
- ParLab (Berkeley)
- Pervasive Parallelism Laboratory (Stanford)
- Sequoia (Stanford)
- PetaBricks (MIT)
- GrADS (Rice)
- CellSs (Barcelona)

... and many more ...

Conclusion

Paradigm shift to (heterogeneous) multi-core architectures offers huge research opportunities for the next decade.

- Raise level of abstraction in parallel programming
- Memory management (locality) most critical issues
- Portability of major concern
- Auto-tuning & adaptivity to fight architectural complexity

Rethink process of designing and optimizing parallel software.