HPC Software Design for Computational Engineering Simulations

H. Köstler, Ch. Feichtinger, J. Götz, S. Donath, U. Rüde

Reno, US
1.3.2011
WaLBerla: parallel block-structured grid framework
Contents

- Software and Performance Engineering in HPC
- WaLBerla: Numerical Results
- Future Work
WaLBerla

SOFTWARE AND PERFORMANCE ENGINEERING
Performance at all costs?

**Performance optimization**
- optimization techniques
- architectural factors
- programming techniques

**Performance engineering**
- simplicity
- extendability
- performance
- effort
WaLBerla: Basic idea

waLBerla (C++)
Code management, standard implementations

Low-level kernels for optimized architecture-specific computations (in C++, CUDA, Assembler)
Performance at different scales

waLBerla

heterogeneous devices, distributed memory parallel

Low-level kernels

Local device, shared memory parallel (OpenCL, OpenMP)
WaLBerla structure

- **The Core**
  - Responsible for sequence control and data management

- **Modules**
  - Common functionality, used by several applications

- **Applications**
  - Each user works on an own application
WaLBerla framework

- **Main goal:** provide a massive parallel and efficient software framework for multi-physics simulations
- WaLBerla is mainly designed for HPC clusters
- Software design concepts
  - Patch Concept
  - Sweep Concept
  - Functionalities
  - Communication Concept
WaLBerla: Patch concept

Simulation Domain

Patch

Unknown / Cell

Block

Block Info:
- Application
- Rank
- Is Allocated
- AABB
- BlockID

Block Data:
- Simulation Data:
  - Cartesian Data
  - Unstructured Data
- Configurable:
  - Data Structures
  - Algorithms
  - Optimizations
WaLBerla: Blocks

- Pure LBM
- Free-Surface
- Particulate Flows
- Free-Surface + Particulate Flows
WaLBerla: Sweep concept

Sweep Concept

Sweep Chain I (Time loop)
- Sweep I
- Sweep II
- Sweep III

Sweep Chain II
- Sweep I
- Sweep II

Sweep
- Preprocessing
  - Communication
  - Timing
- Block Sweep
- Global Sweep
- Postprocessing
  - Visualization
  - Timing

↓: Execution Order  ◀: Iteration  ➤: Dependency
WaLBerla: Functionalities

- Different granularities
- Realized by adding meta data to each functionality consisting of three unique identifiers (UID)

<table>
<thead>
<tr>
<th>UID</th>
<th>Name</th>
<th>Granularity</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>Functionality Selector</td>
<td>Simulation</td>
<td>Gravity on/off, collision model</td>
</tr>
<tr>
<td>hs</td>
<td>Hardware Selector</td>
<td>Process</td>
<td>CPU, GPU, CPU+GPU</td>
</tr>
<tr>
<td>bs</td>
<td>Block Selector</td>
<td>Block</td>
<td>Application</td>
</tr>
</tbody>
</table>
WaLBerla: Heterogeneous simulations

Hardware: hsCPU

FieldData:
Velocity: Layout AoS
Density: Layout AoS
PDF: Layout SoA

Copy Functions:
copyToBuf(fs, hsCPU, bs) → copyFromBuf(fs, hsCPU, bs)
copyToBuf(fs, hsGPU, bs) → copyFromBuf(fs, hsGPU, bs)

MPI Buffers

Functionality fs (Standard)
Application bs (Pure LBM)

Hardware: hsGPU

FieldData:
Velocity(CPU + GPU): Layout AoS
Density(CPU + GPU): Layout AoS
PDF Buffers(CPU + GPU)
PDF(GPU): Layout SoA
WaLBerla: Communication concept

![Diagram showing communication concept between GPU and CPU](Image)

- **GPU Buffers**
  - Local Communication
  - Swap
  - GPU - GPU Copy Operations

- **CPU**
  - MPI Buffers
  - MPI_Isend
  - MPI_Irecv
  - InfiniBand Transfer
  - PCI Express Transfer

- **Neighboring Process**

---

Computer Science X - System Simulation Group
Harald Köstler (harald.koestler@informatik.uni-erlangen.de)
WaLBerla: Software quality factors

- Efficiency
  - Portability
  - Scalability
- Reliability
- Usability
- Expandability
- Maintainability
Efficiency

- One of the major software quality measurement criteria
- Balance between a modular and flexible framework and highly optimized single purpose implementations
- Implementation of parallelization and sweep kernels crucial for good efficiency
- Functionality management allows exchange and optimization of kernels for various simulation scenarios and hardware
- Good performance results are obtained with waLBerla
  → talks Ch. Feichtinger, S. Donath, J. Habich, and J. Götz
Scalability

- On shared-memory level OpenMP and new parallel languages like CUDA and OpenCL are supported.
- On distributed-memory level different MPI implementations are supported.
- Scales well on various HPC clusters e.g. Woodcrest (10.4 TFlops/s) in Erlangen, the HLRB II (62.3 TFlop/s) in Munich and Juropa (207 TFlop/s) and Jugene (1 PFlop/s) in Jülich.
Make it easy to start to work with waLBerla by

- Full doxygen documentation
- Cross-platform build-system CMake
- User interface is a single, clearly structured input file configuring the applications
Enable fast and distributed code development by

- Subversion
- WaLBerla has a clear structure, which helps to identify relevant parts programmers have to change
- Use rest of the framework as black box
Avoid coding errors and ensure software quality by

- C++ coding rules, naming conventions
- common design patterns
- Boost Library
- most parts of the application code is written in a C-like style without more advanced programming techniques
Expandability

- It is easy to extend waLBerla because
  - Many multi-physics simulations can be mapped to the patch and sweep concepts
  - It allows to exchange the underlying data structures
  - It allows to incorporate special optimized kernels in different programming languages
  - All routines can be implemented as “plug-and-play” by using the functionality management
To maintain the growing waLBerla code we

- Apply a strict software quality control, where few main developers are responsible for committing changes to core routines.
- Module and application developers can change only their own parts and suggest changes to the core routines.
- Provide clearly structured and well documented interfaces.
Ensure robust simulations by

- **Debug mode**: checks e.g. array index bounds or memory allocation and initialization

- *(Parallel)* logging system to report runtime information
  - Supports four levels of logging granularity
  - Coding errors and problems originating from the physical model can be analyzed on a per cell / unknown level
Portability

- WaLBerla is easily portable to other platforms because
  - Boost library makes it independent from file system issues
  - Wrapper functions control choice of built-in system functions
  - Cross-platform build-system **CMake** is used 😊
  - Tested on several HPC clusters
SELECTED NUMERICAL RESULTS
Fluid-structure interaction: particle segregation

- Biggest run 64.3 million objects in LBM grid with 151 billion lattice cells
- Runs on 294,912 cores of a Blue Gene/P at Jülich Supercomputing center
Future Work

- Performance Engineering
  - Load balancing for heterogeneous simulations
  - OpenCL

- Applications
  - Adaptive refinement
  - Particulate Flows
  - For more details refer to other talks of our group
    - Ch. Feichtinger: MS121 Thu 3:30
    - S. Donath: MS133 Fr 9:30
    - J. Götz: MS108 Thu 1:00
Acknowledgements

- Regional Computing Center of Erlangen (support for computer systems)

- Funded by
  - SKALB, German project,
  - Bundesministerium für Bildung und Forschung
  - KONWIHR, Bavarian project