HiFlow^3- A Flexible and Hardware-Aware Parallel Finite Element Package

Engineering Mathematics and Computing Lab (EMCL), Karlsruhe Institute of Technology (KIT), Germany

1 Introduction

Numerical simulation based on e.g. finite element (FE) discretizations of partial differential equations (PDEs) describing physical processes is a well established method for gaining deeper scientific insight. It aims at delivering simulation results, allowing to replace costly experimentation, enabling comprehensive parameter studies, and allowing for optimization of production processes and development cycles. Complexity of geometries and models and the demands for high accuracy typically result in a huge number of degrees of freedom (DoFs) with complex couplings. In this setting short solving run times require both powerful hardware and efficient parallel numerical methods compliant with multi-level parallelism and hierarchical memory systems of modern computing platforms. Due to the shift towards the multi- and many-core era both aspects need to be considered in the big picture following the approach of hardware-aware computing. The wide variety of available platforms and the associated challenges of software portability result in a strong need for portable and self-adaptable concepts providing high performance and moderate programming effort for domain specialists and software users.

HiFlow^3, a modular software package based on finite element methods (FEM), is actively developed by a team of approximately 20 persons at the *Engineering Mathematics and Computing Lab* (EMCL) at Karlsruhe Institute of Technology (KIT). It is based on ten years of experience and development and currently re-engineered and restructured in order to consider new paradigms in parallel computing, recent hardware technologies and state-of-the-art programming techniques. The vision of HiFlow^3 is to design a multi-purpose tool, which provides performance and flexibility for the solution of complex problems with high relevance in science and impact on society for instance in the fields of meteorology, climate prediction, energy research, medical engineering, and bioinformatics.

The document is structured as follows. First, the main design ideas of HiFlow^3 are outlined. Then the workflow and the main modules are introduced.

2 Motivation and Concepts

2.1 Fields of Application

A typical application incorporating the full complexity of physical and mathematical modeling is the *United Airways*[*] project. Within an interdisciplinary framework its objective is the simulation of the full human respiratory tract including complex and time-dependent geometries with different length scales, turbulence, fluid-structure interaction (e.g. fine hairs and mucus), and effects due to temperature and moisture variations. Due to the complex interaction of the different physical effects it is a great challenge to construct robust numerical methods giving accurate simulation results within a moderate time. As a further vision, interactive and real time simulations should give medical advice on personalized data during examination

[*] *http://united-airways.org/*

and surgery. Another example is the project *Goal Oriented Adaptivity for Tropical Cyclones*[†]. Many weather phenomena such as the development and motion of tropical cyclones are influenced by processes on scales that may range from hundreds of meters to thousands of kilometers. Due to computational requirements and memory limitations it is often impossible to resolve all relevant scales on globally refined meshes. Application dependent requirements and varying demands with respect to the quantity which is of interest lead to specific treatment of modern adaptive numerical methods that need to be fitted to the associated problem settings.

2.2 Flexibility

The conceptual goal of HiFlow^3 is to be a flexible multi-purpose software package that can be adapted to a wide range of user scenarios. To this end, the core of HiFlow^3 is divided into three main modules: Mesh, DoF/FEM and Linear Algebra; see Figure 2.1. These three core modules, which are intrinsically parallel, are essential for the solution procedure based on FEM for PDEs.

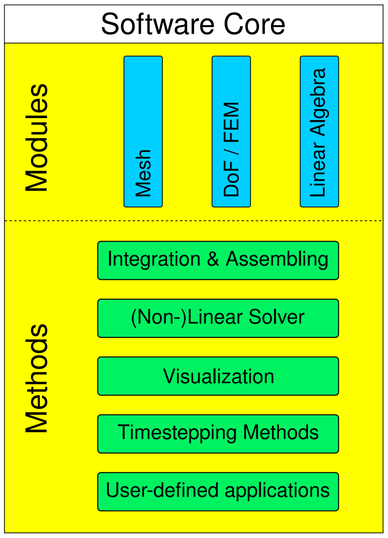

Fig. 2.1: Structure of the HiFlow^3 core divided into modules and methods.

Other building blocks for HiFlow^3 consist for instance of routines for numerical integration of element integrals, assembling of system matrices, inclusion of boundary conditions, setting up nonlinear and linear solvers, providing output data for visualization of solutions, error estimators. This modular structure ensures the flexibility to employ the library to solve a wide variety of problems. Another aspect of modularity is to enable extensibility of the HiFlow^3 package. There exist two basic ways to extend HiFlow^3. Further modules and methods for user-defined applications can be added to the HiFlow^3 core. Existing modules and methods can be augmented by adding further functionality. For instance in the Mesh module a new type of mesh can be developed by inheriting the abstract mesh base class. The existing H^1 and L^2 FE-spaces can be extended by $H(div)$ or $H(curl)$ applying the same paradigm. HiFlow^3 relies on a generic implementation in order to be able to use different modules and methods for a variety of problems following the approach of a versatile library.

2.3 Performance, Parallelism, Emerging Technologies

An important goal associated with the design of HiFlow^3 is the full utilization of available resources on heterogeneous platforms – from large HPC systems to a stand-alone workstation or coprocessor-accelerated

[†] *http://metstroem.mi.fu-berlin.de/*

machine and it is also geared towards software as a service in a cloud-computing framework. To achieve this goal all three modules (Mesh, DoF/FEM, Linear Algebra) are based on distributed data structures. The communication between different nodes and processors is realized by means of an MPI-layer. On the linear algebra level we use concepts of hardware-aware computing for acceleration of basic routines like local matrix-vector and vector-vector operations. Here, the main challenge is to fully utilize the available computing power of emerging technologies like multi-core CPUs, graphics processing units (GPUs), multi-GPUs. The advantage of the structure provided by the Linear Algebra module is the potential to build solvers without having detailed information on the underlying platform. All methods are laid out with respect to scalability in a generic sense. In the software design best performance may conflict with the constraints associated with a flexible software. The design of HiFlow^3 aims at obtaining high standards with respect to the efficiency without sacrificing the flexibility for the use as well as extension of the software. HiFlow^3 provides the user with a transparent and documented structure enabling him to set up and get a simulation for his applications running with a reasonable effort.

2.4 Hardware-aware Computing

The huge demand on fast and accurate simulation results for large-scale problems poses considerable challenges on the implementation on modern hardware. Supercomputers and emerging parallel hardware like GPUs offer impressive computing power in the range of Teraflop/s for desktop supercomputing up to Petaflop/s for cutting edge HPC machines. The major difficulty is the development of efficient parallel code that is scalable with respect to exponentially increasing core counts and portable across a wide range of computing platforms. With the advent of the many-core era new platforms like the GPUs, the Cell BE, FPGA-based systems like Convey's Hybrid Core Computer or Intel's tiled architectures (Polaris, Rock Creek) or the Larrabee incarnation Knight Ferry have emerged. These technologies come along with impressive capabilities but different programming approaches, different processing models, and different tool chains. Moreover, all numerical methods need to be compliant with multiple levels of parallelism within hybrid systems and with hierarchical memory subsystems. Typically, manual tuning and parameter variation is necessary for optimal performance on a dedicated system. In this context, hardware-aware computing is a multi-disciplinary approach to identify the best combination of applications, physical models, numerical schemes, parallel algorithms, and platform-specific implementations that is giving the fastest and most accurate results on a particular platform. Design goals are maximal flexibility with respect to software capabilities, mathematical quality, efficient resource utilization, performance, and portability of concepts. Hardware-aware computing not only comprises highly-optimized platform-specific implementations, design of communication-optimized data structures, and maximizing data reuse by temporal and spatial blocking techniques – it also relies on the choice and development of adequate mathematical models that express multilevel-parallelism and scalability up to a huge number of cores. The models need to be adapted to a wide range of platforms and programming environments. Data and problem decomposition on heterogeneous platforms is a further objective. Memory transfer reduction strategies are an important concept for facing the bottlenecks of current platforms. Since hardware reality and mathematical cognition give rise to different implementation concepts, hardware-aware computing means also to find a balance between the associated opponent guiding lines while keeping the best mathematical quality (e.g. optimal work complexity, convergence order, error reduction and control, accuracy, robustness, efficiency). All solutions need to be designed in a reliable, robust and future-proof context. The goal is not to design isolated solutions for particular configurations but to develop methodologies and concepts that preferably apply to a wide range of problem classes and architectures or that can be easily extended or adapted. The HiFlow^3 project has implemented related concepts in the framework of the local multi-platform LAtoolbox within the Linear Algebra module, see also 2.7.

2.5 Object-oriented Programming Approach

In order to be able to handle the complexity of solving such a great variety of problems HiFlow^3 is implemented in C++. The object-oriented paradigm of C++ with its support for static and dynamic polymorphism and inheritance allows all involved developers to contribute with their individual specialized knowledge. End-user and researcher can handle opaque and problem-oriented objects, computer scientists and hardware specialists produce optimized low-level implementations, while mathematicians provide new numerical solver algorithms. Furthermore, the intensive use of the template features of C++ gives the compiler a lot of opportunities for compile-time optimizations. Our modular approach with utilization of the object-oriented concepts of C++ is the a well-suited way to handle the complexity of such a sophisticated software project with many contributors. The approaches of data abstraction, encapsulation and clear interfaces between various modules not only ease maintainability of the code but also enable reusability of specific parts for the extension of functionalities and features. An additional benefit of this approach is the possibility to use parts of the software as stand-alone libraries or modules for other projects like e.g. the LAtoolbox. In such a way, our project provides building blocks for the development of new solvers and the extension to new problem domains.

2.6 Modeling and Workflow

Many physical problems are modeled by means of PDEs. Typical examples are the Poisson equation (e.g. for electric or gravity fields), stationary or time-dependent convection diffusion equations (e.g. transfer of particles or energy), and the Navier-Stokes equation (e.g. fluid flow around an obstacle, simulation of a tropical storm, air flow in the lung). A classical approach for solving associated problems relies on discretization of the equations in the domain of interest by means of FEM on spatial grids. Searching for approximations in finite dimensional function spaces by means of weak formulations results in linear or nonlinear systems of equations that are typically solved by Newton-type methods, time-stepping schemes and iterative solvers. Coupling between DoFs is mainly expressed by nearest neighbor interaction with a high degree of locality. Due to the required accuracy of the discrete approximations large numbers of DoFs are required resulting in huge and sparse matrices with bad condition numbers in general.

For the FE solution procedure the following steps have to be performed to simulate a physical process with HiFlow^3. First, the physical problem has to be modeled by PDEs. The next theoretical step is to derive a weak formulation in a variational sense. In a pre-processing step the domain of interest is discretized by means of a FE triangulation and converted to a format readable by the Mesh module. Once the spatial grid is read in it can be adjusted e.g. by means of a refinement procedure. Then, a problem-adapted FE space with appropriate ansatz functions has to be chosen. For the Navier-Stokes equations Taylor-Hood elements are an appropriate choice. Once these two informations are provided the DoFs can be determined first locally on each cell, and then globally for the whole mesh. In the case of distributed memory platforms the DoF-partitioner is used to create a global DoF numbering throughout the whole computational domain by only using its local information and MPI communication across the nodes. Once all DoFs are identified the matrix can be assembled by local integration over all elements within the triangulation. Data structures for matrices and vectors are provided by the Linear Algebra module. In case of nonlinear problems a Newton method is combined with a linear solver for the problem solution. Finally, output in either sequential or parallel format is provided for the visualization. This is an important aspect of the simulation cycle for an assessment and exploration of simulation data. There exist several visualization tools like Paraview[‡] which can be used for this post-processing step.

2.7 Modules in HiFlow^3

The implementation of a parallel and flexible FE software package aiming at general purpose deployment and portability across a wide range of platforms comes along with various challenges in the respective modules.

The Mesh module interacts with the discretized computational domain. It is designed primarily for unstructured meshes, and can handle several different cell types in different dimensions. In order to support

[‡] `http://www.paraview.org/`

adaptive algorithms, the Mesh module can work with both non-conforming meshes and meshes containing cells of mixed types. Additionally, it provides functionality to refine and coarsen a mesh, both globally and locally, and to retrieve the history of these modifications. Furthermore, the use of meshes distributed over several processors is supported, in order to reduce the memory requirements for large-scale simulations running on a high-performance cluster. For this purpose, the module also contains functionality for dealing with partitioning and communication of the mesh data. The link between the local mesh and its neighbors is a layer of ghost cells that are shared between each pair of processors.

The DoF/FEM module treats the problem of numbering and interpolating the DoFs. In the first step the local DoFs of each cell for a chosen FE space need to be determined. Then the local DoFs of each cell have to be numbered globally. We are using a generic approach handling the implemented cell types and FE spaces. In the case of distributed data structure the DoF module further handles the neighborhood relations between cells which are distributed on different processors. To this end, the module takes advantage of ghost cells created by the Mesh module. Another task is to distribute the DoFs in a balanced way such that each processor is responsible for almost the same number of DoFs. Here, different complexities and work load contributions need to be considered since each cell might use different ansatz functions.

The Linear Algebra module provides distributed and platform-optimized data structures for vectors and matrices as well as implementations for corresponding matrix-vector and vector-vector operations. Due to localized interactions between FE basis functions resulting matrices are typically sparse. For that purpose, adequate sparse matrix formats are provided. The structure of the Linear Algebra module is based on two levels: the inter-node level communication layer utilizes MPI and the intra-node communication and computation model is based on platform-specific programming environments that are accessed by generic and unified interfaces. In order to reduce the cost of communication on the upper layer the Linear Algebra module takes advantage of ghost cells provided by the Mesh module (and also used in the DoF module). On the local intra-node layer the focus is set to methods of hardware-aware computing aiming at best choice of platform-specific implementations. The concept of the Linear Algebra module allows to compute local matrix vector operations on hybrid systems and accelerators such as multi-core-CPUs, GPUs, and OpenCL-enabled devices. By providing unified interfaces with back-ends to different platforms and accelerators the module allows seamless integration of various numerical libraries and devices. The user is freed from particular hardware knowledge – the final decision on platform and chosen implementation is taken at run time. Naturally, scalability plays an important role in this module.

3 Conclusion

The creation of a multi-purpose FE software package that is portable across a wide variety of platforms including emerging technologies like hybrid CPU and GPU platforms is a challenging and multi-facetted task. By our modular approach that utilizes the concepts of object-orientation, data abstraction, polymorphism and inheritance we have created a highly capable piece of software that provides a powerful means for gaining scientific cognition. By utilizing several levels of parallelism by means of a two-level communication and computation model and following the concepts of hardware-aware computing HiFlow^3 is a flexible numerical tool for solving bleeding-edge scientific problems on the basis of FEM optimized for high performance computers. Furthermore, the modules Mesh, DoF/FEM and Linear Algebra complemented by auxiliary methods provide a broad suite of building blocks for development of modern numerical solvers and application scenarios. The user is freed from a detailed knowledge of the hardware – he only has to familiarize with the provided interfaces and needs to customize the available modules in order to adapt HiFlow^3 to his domain-specific problem settings. As an open source project HiFlow^3 further supports extensibility of modules and methods. Within large scale projects like e.g. the United Airways project HiFlow^3 has already proven its potential. In the next steps the efficiency of the methods considered in the different modules especially related to the scalability will be evaluated. Based on these results HiFlow^3 will improve, following further the path of object oriented techniques in software design for scientific computing.