D2.1

Report on the ExaNoDe mini-applications

Workpackage:	2	Co-Design for Exa-scale HPC systems
Author(s):	Dirk Pleiter	JUELICH
	Guillaume Colin de Verdiere	CEA
Authorized by	Guillaume Colin de Verdiere	CEA
Reviewer	Etienne Walter	BULL
Reviewer	Manolis Marazakis	FORTH
Reviewer	Dominique Drouin	UMI-LN2
Reviewer	Paul Carpenter	BSC
Dissemination Level	Public	

Date	Author	Comments	Version	Status
16.06.2016	Dirk Pleiter	Initial complete draft	0.1	Draft
17.06.2016	Dirk Pleiter	Update based on feedback from Manolis Marazakis, Guillaume Colin de Verdiere and Paul Carpenter	0.2	Draft for internal review
24.06.2016	Dirk Pleiter	Update based on review by Etienne Walter and Dominique Drouin	1.0	Final

Executive Summary

This deliverable defines the application portfolio that will be used for the co-design process within the ExaNoDe project. For this purpose a set of general criteria has been defined, which can be used for selecting applications that are suitable for co-designing future HPC systems. A selection of applications, mini-applications and benchmarks are presented, which we all believe to be good candidates for such a process. The choice does not only impact hardware design in WP4 but also design of programming models for the ExaNoDe architecture within WP3.

After applying the criteria to the list of application candidates, we recommend foreseeing the following applications to become part of the application portfolio of ExaNoDe:

- BQCD: A massively-parallel application for simulating Quantum Chromodynamics, which is the theory for strong interactions.
- HydroC: An application-based benchmark mimicking a 2-dimensional CFD code based on the Finite Volume Method.
- KKRnano: A highly scalable material science application based on the Density Functional Theory (DFT) method.
- MiniFE: A mini-application implementing an Implicit Finite Elements method in 3 dimensions.

Table of Contents

Table of Tables

1 Introduction

The complexity of designing supercomputers is ever increasing. To make sure that the customers really get the machines they need, a strong cooperation process, named co-design, has been in place for a couple of years. Through it, machine architects and end-users interact to refine the definition of a future generation of computers. For this reason co-design is a key aspect of the ExaNoDe project to guide the technology developments performed in work packages WP3 and WP4. For this purpose, suitable applications need to be selected. The original project strategy was to make a choice for relevant applications and then to implement mini-applications that can be ported to the ExaNoDe architecture. After updating the Description of Activities the project no more aims for a prototype system, which could be used for executing mini-applications or even full applications, but rather for a Proof of Concept (PoC). The ExaNoDe PoC will allow demonstrating critical technology components, but it cannot be expected to be suitable for validating the ExaNoDe architecture for large-scale scientific applications in terms of performance and usability. We therefore adjusted the scope of work within work package WP2 such that it will limit the porting efforts to performance critical code sections, so-called kernels.

The concept of mini-applications [Heroux2009] aims at reducing the complexity of the applications while preserving the relevant properties. Which properties are relevant depends on the context and there is thus no simple recipe for creating mini-applications. This is especially true when one tries to design a general purpose architecture that should suit most needs. Reducing the complexity of the applications can have multiple advantages. While the production codes may have restricted access, mini-applications are often easier to place under an open source licence, which makes interactions with a larger research community as well as uptake of the source code by commercial operators easier. The simpler the mini-application the easier it becomes to use these for simulating machine architectures, to port these to early versions of a new architecture, like ExaNoDe, and to optimize it for such a new architecture. The latter might possibly require analysis of different versions of the code, which are, e.g., based on different layouts of the application data in the memory. This would not be feasible for almost any full application code as the code basis for any relevant HPC application has become too big as it exceeds in many cases 100,000 lines of code. Mini-applications can be re-factored and sometimes even be rewritten, which is also relevant for the work on programming models within ExaNoDe's work package WP3.

In section 2 we document the criteria that we used for selecting applications for the ExaNoDe project. In section 3 we provide an overview on all applications that have been considered and then (in section 4) discuss which applications should be used for the ExaNoDe project.

2 Criteria for selecting applications

We suggest selecting the applications for this project on the basis of a set of criteria, which we document in this section. We also provide a rationale on why a given criterion is important in this context. While some of the criteria should be met by all selected mini-applications, some criteria may only apply to a subset of them.

The following table lists criteria, which should be fulfilled by all mini-applications. They are listed in no particular order.

Table 1: Mandatory selection criteria.

Criteria	Rationale
C1.1: Application will need scalable HPC computer resources in the future	The application owner should have an understanding about her/his need for innovative new, massively-parallel HPC architectures once these become available in future.
C1.2: Application is used for large-scale computations on relevant HPC systems today	The application should already be in use on massively-parallel HPC architectures today. We do not want application owners who only believe that they could exploit large-scale compute resources in future.
C1.3: Application is likely granted access to large-scale compute resources in future	Applications which are on long-term research roadmaps are preferred as these are more likely being granted suitable resources, e.g., on massively-parallel architectures based on ExaNode technology.
C1.4: Requirements of the application helps to broaden the set of architectural requirements	It does not make sense to select multiple applications that have a similar performance signature, i.e. all being dominated by stencil computations with similar operational intensity.
C1.5: Close relation to application owners exist	Such relations are important to leverage the expertise of the application owners. Relevant expertise includes knowledge about relevant use cases and problem sizes and future application roadmaps.
C1.6: Appropriate licence conditions for mini-application	The mini-application must be (or become) available under licence conditions which allows porting activities and evaluation to be performed within this project.

The following criteria should apply to a subset of applications, because we want some applications but not applications to meet these criteria:

Table 2: Selection criteria to be fulfilled by a subset of applications.

Criteria	Rationale
C2.1: Mini-application is suitable for evaluation of programming models that are used within ExaNoDe	It does not make sense to aim for providing a port to the ExaNoDe architecture for all of the programming models. This would significantly enhance the efforts without obvious benefits. The goal is rather to have for each programming model at least one mini-application to which this programming model can be applied.
C2.2: At least one application supports MPI	Required for testing the MPI runtime "port" (shmem version) to Unimem (task T3.3 “Parallel programming models and runtime libraries”)
C2.3: At least one application supports OpenCL	Provide test case for the accelerator support work in T3.2 “Virtualization”

C2.4: Application features relatively simple kernel	Simple kernels can be beneficial in several respects. They would allow porting to the ExaNoDe architecture even if the software environment is not yet fully available or stable, e.g. by implementing parts of that code in assembler. Furthermore, such kernels will help to evaluate the architecture as it will be easier to separate hardware and system software related aspects.

The following additional criteria are nice to have, but less critical:

Table 3: Optional criteria.

Criteria	Rationale
C3.1: Application is part of the Unified European Applications Benchmark Suite (UEABS) established by PRACE[1]	Being able to communicate results on established benchmarks may facilitate communication with important stakeholders like PRACE.

3 Application candidates

In this section we provide an overview on the applications that had been considered during the first months of the project. Selected properties have been collected in Table 4. We note, however, that the methodology of selecting applications described in this deliverable would allow work package WP2 to analyse the suitability of other applications at a later point of project execution based on requirements coming from other work packages. On potential candidate is Alya,[2] which had been proposed after the analysis of the applications described below had been completed and which may be of interest for WP3 because of its support of different programming models.

3.1 Abinit

Abinit[3] is an ab initio code that is largely used for material science. It has a broad community of users. Abinit uses internally FFTs which are very important kernels for HPC. This code is written in FORTRAN and is parallelized using MPI and some OpenMP. The current version has a proven scalability of thousands of cores on the Curie system. A mini-application version of the code already exists.

3.2 BQCD

BQCD (Berlin Quantum ChromoDynamics program) is a hybrid Monte-Carlo code that simulates Quantum Chromodynamics on a lattice (LQCD) with dynamical Wilson-type fermions [Nakamura2010]. It is written in Fortran 90 and uses MPI and OpenMP for parallelisation. A relatively simple kernel, where mainly sparse matrix-vector multiplications are performed, dominates overall performance. The application is part of the UEABS and one of the PRACE-3IP benchmark applications. It is currently used for large-scale projects on different Tier-1 systems. LQCD is on different future research roadmaps and an application area which is in need for exascale computing resources.

[1] http://www.prace-ri.eu/ueabs/
[2] http://www.bsc.es/es/computer-applications/alya-system
[3] http://www.abinit.org

3.3 HydroC

HydroC[4] is not an application, but rather a mini-application. It is considered here as it represents a large class of relevant codes. It is a simplified version of the astrophysical code RAMSES. It is a 2-dimensional CFD using the Finite Volume Method with a Godunov's scheme and a Riemann solver at each interface on a regular 2D mesh. The code basis is O(1,000) lines of code and thus small. Another aspect that is interesting in this context is the support of accelerators through an OpenCL version of the code [Lavallee2012].

3.4 KKRnano

KKRnano is based on the Density Functional Theory (DFT) method, which is a very popular method in condensed matter physics and material science [Thiess2012]. It is written in Fortran 90 and uses MPI and OpenMP for parallelisation. Dense matrix and other linear algebra tasks dominate the overall performance. The application is optimized for scaling to a very large number of atoms and thus for execution on massively-parallel HPC systems. It is, e.g., member of Jülich's High-Q Club[5], which is a list of applications that could demonstrate scalability using 28 racks of Blue Gene/Q, i.e. 458,752 cores. Material science is an area that will in future be in need of exascale computing resources.

3.5 MiniFE

MiniFE[6] is already a mini-application, which is widely used for co-design projects in the USA. MiniFE mimics the finite element generation, assembly and solution for an unstructured grid problem. The calculations are performed using a 3-dimensional box of configurable size. While the discretisation is structure, MiniFE treats it as an unstructured grid. The numerical problem is linear and the resulting matrix is symmetric. Therefore conjugate gradient can be applied, which is a popular algorithm for solving sparse linear systems.

3.6 NEST

NEST (NEural Simulation Tool) is a simulation code for the investigation of the dynamics of brain-scale neuronal network models [Gewaltig2007]. It does so at the level of resolution of neurons and synapses, where neurons are brain cells which are connected to each other by the synapses. It is implemented in C++ and uses MPI and OpenMP for parallelisation. The application is optimized for high scalability and is used on systems which are among the largest available. It is, e.g., member of Jülich's High-Q Club, i.e. it scales to at least 28 Blue Gene/Q racks, i.e. 458,752 cores, and has been executed on the full K-Computer, a system with 705,024 cores. It is one of the key applications in the Human Brain Project (HBP) and part of the HBP's PCP. It will need significantly large systems than available today in order to be able to simulate brain networks of realistic size.

Table 4: Application properties.

Application	Programming language	Programming models
Abinit	Fortran90	MPI, OpenMP, CUDA
BQCD	Fortran90	MPI, OpenMP
HydroC	C/C++	MPI, OpenMP, OpenACC, OpenCL, CUDA
KKRnano	Fortran90[7]	MPI, OpenMP

4 https://github.com/HydroBench/Hydro/tree/master/HydroC
5 http://www.fz-juelich.de/ias/jsc/EN/Expertise/High-Q-Club/_node.html
6 https://mantevo.org/packages/
7 An initial implementation of a mini-application has been started using C++.

MiniFE	C++	MPI, OpenMP, CUDA, Cilk
NEST	C++	MPI, OpenMP

4 Evaluation of the criteria

The following table shows the outcome of applying previously defined criteria to the list of applications described in the previous section:[8]

Table 5: Application of mandatory criteria.

	Abinit	BQCD	HydroC	KKRnano	MiniFe	NEST
C1.1	Y	Y	Y	Y	Y	Y
C1.2	Y	Y	N/A	Y	Y	Y
C1.3	Y	Y	N/A	Y	Y	Y
C1.4	Y	Y	Y	Y	Y	Y
C1.5	-	Y	Y	Y	-	Y
C1.6	Y	Y	Y	-	Y	Y

Criterion C2.1 is fulfilled when including BQCD and MiniFE. BQCD has earlier been used for evaluating the programming model GPI [Gruenwald2012]. MiniFe has already been refactored and rewritten in order to explore different programming models [Heroux2009].

Criterion C2.2 can be met by including any of the proposed applications, which all use MPI for parallelisation over multiple nodes.

Criterion C2.3 is fulfilled if we add HydroC to the application portfolio. This is the only application for which to our knowledge a port to OpenCL is available.

Criterion 2.4 can be met by using BQCD, HydroC and KKRnano. For both applications overall performance is dominated by relatively simple kernels. In the former case, typically most of the execution time is spent in a sparse linear solver. Also in case of KKRnano a solver is most performance critical, however in this case mainly dense matrix-matrix multiplications are executed.

The following table gives an overview on how the criteria apply to the individual criteria:

Table 6: Application of criteria to be fulfilled by a subset of applications.

	Abinit	BQCD	HydroC	KKRnano	MiniFe	NEST
C2.1	-	Y	Y	-	Y	-
C2.2	Y	Y	Y	Y	Y	Y
C2.3	-	-	Y	-	-	-
C2.4	-	Y	Y	Y	-	-

We would meet criterion C3.1 by when BQCD is part of the application portfolio.

[8] Note that for MiniFE we applied the criteria to the application mimicked by this mini-application.

5 Conclusions

Based on the analysis of the previous section we propose to include the following applications to the application portfolio of ExaNoDe:

- BQCD: This application meets all criteria C1.1 to C1.6 as well as C2.1, C2.4 and C3.1.
- HydroC: While this application does not meet criteria C1.2 and C1.3 (being only a mini-application) it helps to meet criteria C2.1 and C2.2.
- KKRnano: While the application currently does not meet criterion 1.6, we expect it to be feasible to extract a mini-application that can be placed under an open source licence. Using this application would broaden the set of architectural requirements as it is the only application involving dense matrix-matrix multiplications. Furthermore, it helps to meet criterion C2.4.
- MiniFE: Although the application does not meet criterion 1.5 it is expected to be useful for exploring programming models (criterion C2.1).

As described in section 3 this list might be extended depending on needs arising during further project execution applying the criteria defined in this document.

6 References

[Gewaltig2007] M.-O. Gewaltig, M. Diesmann, „NEST (Neural Simulation Tool),“ Scholarpedia, Vol. 2, No. 4, 2007.

[Gruenwald2012] D. Grünwald, “BQCD With GPI. A Case Study,” HPCS’12, 2012 (doi:10.1109/HPCSim.2012.6266942).

[Heroux2009] M.A. Heroux et al., “Improving Performance via Mini-applications,” Sandia Report SAND2009-5574, 2009 (doi: 10.2172/993908).

[Lavallee2012] Pierre-François Lavallée et al., “Porting and optimizing HYDRO to new platforms and programming paradigms – lessons learnt,” Technical report, PRACE, December 2012 (http://www.prace-ri.eu/IMG/pdf/porting_and_optimizing_hydro_to_new_platforms.pdf).

[Nakamura2010] Y. Nakamura, H. Stüben, „BQCD – Berlin Quantum Chromodynamics program,” PoS LAT2010, 040, 2010 (arXiv:1011.0199).

[Thiess2012] A. Thiess et al., "Massively parallel density functional calculations for thousands of atoms: KKRnano," Physical Review B, Vol. 85, 235103, 2012 (doi:10.1103/PhysRevB.85.235103).