D6.2 – iPIC3D implicit particle-in-cell code for space weather applications preparation and evaluation (a)

WP6: Integration, testing and pilot applications

Version: 1.0
Author(s): Roman Iakymchuk (KTH)
Date: 31/03/17
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications
preparation and evaluation (a)

<table>
<thead>
<tr>
<th><strong>Due date:</strong></th>
<th>PM18</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Submission date:</strong></td>
<td>31/03/2017</td>
</tr>
<tr>
<td><strong>Project start date:</strong></td>
<td>01/10/2015</td>
</tr>
<tr>
<td><strong>Project duration:</strong></td>
<td>36 months</td>
</tr>
<tr>
<td><strong>Deliverable lead organization:</strong></td>
<td>KTH</td>
</tr>
<tr>
<td><strong>Version:</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Final</td>
</tr>
<tr>
<td><strong>Author(s):</strong></td>
<td>Roman Iakymchuk (KTH)</td>
</tr>
<tr>
<td><strong>Reviewer(s):</strong></td>
<td>Philipp Gschwandtner (UIBK)</td>
</tr>
<tr>
<td></td>
<td>Thomas Heller (FAU)</td>
</tr>
<tr>
<td></td>
<td>Arne Hendricks (FAU)</td>
</tr>
<tr>
<td><strong>Dissemination level</strong></td>
<td>PU</td>
</tr>
<tr>
<td><strong>CO</strong></td>
<td>CO - Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>

**Disclaimer**

This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement Nr 671603. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the Project and to the extent foreseeable in such agreements.
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications preparation and evaluation (a)

Acknowledgements

The work presented in this document has been conducted in the context of the EU Horizon 2020. AllScale is a 36-month project that started on October 1st, 2015 and is funded by the European Commission.

The partners in the project are UNIVERSITÄT INNSBRUCK (UBIK), FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN NÜRNBERG (FAU), THE QUEEN'S UNIVERSITY OF BELFAST (QUB), KUNGLIGA TEKNISKA HÖGSKOLAN (KTH), NUMERICAL MECHANICS APPLICATIONS INTERNATIONAL SA (NUMEXA), IBM IRELAND LIMITED (IBM).

The content of this document is the result of extensive discussions within the AllScale Consortium as a whole.

More information
Public AllScale reports and other information pertaining to the project are available through the AllScale public Web site under http://www.allscale.eu.

Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comments, Changes, Status</th>
<th>Authors, contributors, reviewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>31/01/17</td>
<td>Document structure</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>0.2</td>
<td>02/03/17</td>
<td>Introduction and functionality complete</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>0.3</td>
<td>03/03/17</td>
<td>Benchmarks: particle-wave interaction</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>0.4</td>
<td>04/03/17</td>
<td>Porting to AllScale API/SDK: numerical schemes and API description</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>0.5</td>
<td>09/03/17</td>
<td>Benchmarks: Earth’s radiation belts; Status and future work: complete</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>0.6</td>
<td>10/03/17</td>
<td>Ready for internal review</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>0.7</td>
<td>27/03/17</td>
<td>Addressed reviewers’ comments</td>
<td>Roman Iakymchuk</td>
</tr>
<tr>
<td>1.0</td>
<td>30/03/17</td>
<td>Final version</td>
<td>Roman Iakymchuk</td>
</tr>
</tbody>
</table>
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications
preparation and evaluation (a)

Table of Contents
More information .................................................................................................................. 3
Executive Summary ............................................................................................................. 5
1 Introduction ....................................................................................................................... 6
2 Functionality of the iPIC3D Pilot .................................................................................... 6
3 Porting to the AllScale API/SDK .................................................................................. 8
   3.1 Numerical Scheme ...................................................................................................... 8
      3.1.1 Field Solver ......................................................................................................... 9
      3.1.2 Particle mover ...................................................................................................... 9
   3.2 API Description ......................................................................................................... 10
   3.3 Validation .................................................................................................................. 11
   3.4 Relative Scalability .................................................................................................. 11
4 Benchmarks .................................................................................................................... 12
   4.1 Particle-wave interaction .......................................................................................... 12
   4.2 Earth’s Radiation Belts ............................................................................................. 13
5 Status and Future Work .................................................................................................. 14
6 Bibliography .................................................................................................................... 15

Index of Figures
Figure 1: Structure of the iPIC3D pilot .............................................................................. 8
Figure 2: Task-based nested recursive parallelism for the iPIC3D pilot on a grid
with particles, where T.x.y.z stands for the task number x with its
corresponding sub-tasks y and z ....................................................................................... 10
Figure 3: Strong scaling of the particle mover part of the iPIC3D code
implemented with the AllScale API/SDK and compared against the original
version on an Intel Ivy Bridge node at PDC ........................................................................ 12
Figure 4: A trajectory of a proton trapped in the radiation belt ..................................... 13
Figure 5: Earth’s (Van Allen) radiation belts with satellites. Credit: NASA ............... 13

Index of Tables
Table 1: Realistic test cases for iPIC3D .............................................................................. 14
Executive Summary
This document describes the incremental preparation, deployment and validation of the iPIC3D pilot application with the AllScale API/SDK. First, we present the pilot, coupled with a high level description of the underlying functionality and its numerical implementation. Second, we outline the effort, both algorithmic and development, regarding porting the pilot to the AllScale API/SDK. Finally, we provide a set of benchmarks to verify correctness and performance of the new implementation.
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications
preparation and evaluation (a)

1 Introduction
The iPIC3D pilot was already introduced in D2.1 (Roman Iakymchuk (KTH) 2016) when we derived requirements imposed by the pilot applications towards the AllScale Toolchain. That deliverable was focused on iPIC3D’s scientific and commercial impact, stakeholders, Exascale potential, current parallelization approach, as well as AllScale potential. In the current document, we discuss the algorithmic functionality of the pilot, and describe the effort and the results of porting iPIC3D to the AllScale API/SDK.

The underlying method in the iPIC3D pilot is the Particle-in-Cell (PIC) method. The PIC method is one of the most common and powerful numerical techniques for the simulation of fusion, astrophysical, and space plasmas. For instance, PIC simulations (G. Lapenta 2013) are used to study the interaction of the Earth’s electromagnetic field with the hot plasma emanated by the sun, the so-called solar wind. Since the high-energy plasma in space can damage spacecrafts and put in danger the life of astronauts in space, it is important to enable efficient large-scale PIC simulations that are capable of predicting different phenomena in space.

This document is organized as follows. Section 2 describes the original iPIC3D functionality and its algorithmic structure. Section 3 focuses on the porting and restructuring effort to enable task-based nested recursive parallelism for the iPIC3D pilot. Section 4 presents a set of benchmarks to be used during the course of the project. Finally, Section 5 summarizes the current effort and outlines future plans.

2 Functionality of the iPIC3D Pilot
In kinetic methods the evolution of the distributions function for a species, $f$, is solved using the transport equation without the collisional term, the so-called Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m} (E + \frac{\mathbf{v} \times \mathbf{B}}{c}) \cdot \nabla \mathbf{v} = 0,$$

Equation 1: Vlasov equation.

where $q$, $m$ are the charge and mass of the species, respectively. $\mathbf{v}$ is the particle velocity, $r$ is the particle position, and $\mathbf{B}$, $\mathbf{E}$ are the magnetic and electric fields, accordingly. The Vlasov equation is solved in combination with Maxwell’s equations

$$\frac{\partial \mathbf{B}}{\partial t} = -c \nabla \times \mathbf{E}$$

$$\frac{\partial \mathbf{E}}{\partial t} = c \nabla \times \mathbf{B} - \mathbf{J}$$

Equation 2: Maxwell’s equations.

The links between the Vlasov equation and Maxwell’s equations are the charge $\rho$
and current $J$ density

$$\rho = \Sigma q \int f d\mathbf{v},$$

Equation 3: Charge density.

$$J = \Sigma q \int \mathbf{v} f d\mathbf{v}.$$  

Equation 4: Current density.

One of the approaches to solve the Vlasov-Maxwell system is to employ the PIC method. In the PIC method, the Vlasov equation is simplified by describing the distribution by a certain amount of computational particles picked at random from an initial Maxwellian distribution.

The PIC method solves the kinetic equation of plasmas by first sampling plasma distribution functions with computational particles and then following their trajectories by solving the equation of motion for each particle

$$\frac{d\mathbf{r}}{dt} = \mathbf{v},$$

$$\frac{d\mathbf{v}}{dt} = \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}).$$

Equation 5: Equation of motion.

Particles position and velocity are updated in the particle mover stage. The electromagnetic field determining the particle trajectory is calculated by computing the Maxwell’s equations (see Equation 2) on a grid (field solver): each grid cell is characterized by an electric and magnetic field defined on nodes (electric and magnetic fields) and in the center (only magnetic field). The coupling between particles and both electric and magnetic fields is provided by the so-called interpolation functions: the electric fields and magnetic fields acting on the particle are calculated by interpolating the values on the grid cells at the particle position; the current and charge densities are computed by aggregating the values from the particles to the grid. Figure 1 shows these four computational stages in iPIC3D.

In general, the workflow of iPIC3D can be summarized in two steps: 1) electric $E$ and magnetic $B$ fields as well as the particles velocity $\mathbf{v}$ and position $\mathbf{r}$ are initialized on the grid using the set-up defined in the input file; 2) The Maxwell equation and equation of motion are calculated simultaneously on the grid for several cycles. The amount of cycles to run the simulation is also specified in the input file (Eriksson 2016).
Typically, parallel iPIC3D simulations divide the simulation box into several domains, equal in size, with initially the same number of particles (S. Markidis 2010). Each domain is assigned to a process that carries out the computation for the particles in the domain. When a particle exits the domain, it is communicated to a different domain. Because of the non-uniform configuration of the electromagnetic field in space, computational particles concentrate in relatively small spatial regions while few particles cover other spatial regions. This results in having more particles in certain simulation domains than other and consequents the work-imbalance problem: processes with fewer particles have to wait for other processes with more particles to finish their computations at every time step; we also name them as synchronization points.

Workload-imbalance is the most severe and limiting problem in large-scale PIC simulations. For instance, the parallel simulation of the magnetic reconnection (a phenomena occurring on Earth’s magnetotail) on 8,192 cores leads to processes spending 22.2% of their time waiting for all processes to become available for collective operation. The other simulation set-up shows even more severe work-imbalance: two-dimensional simulations of planetary magnetospheres on 2,048 cores present the process imbalance of 61.7% (Peng, et al. 2015). To achieve Exascale performance, this problem of work-imbalance needs to be solved.

3 Porting to the AllScale API/SDK

To fully utilize the AllScale Environment’s ability of conducting dynamic load balancing and latency hiding, both, the parallel algorithm and the underlying data structure have to be constructed according to the AllScale program model. Fortunately, in both cases those modifications are readily comprehensible.

3.1 Numerical Scheme

The algorithmic part of the PIC simulation code covers four steps: the projection of particle properties to the grid nodes, the solution of force-field equations over those grid nodes, the application of those forces on the involved particles, and finally the movement of particles in space. In order to tailor these parts for the task-based nested recursive parallelism, we redefined the numerical scheme of the current iPIC3D pilot. Our major effort was on the following two components: the field solver for the Maxwell’s equation, Equation 2, and the particles mover for the equation of motion, Equation 5.
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications
preparation and evaluation (a)

3.1.1 Field Solver
The field solver for the Maxwell’s equation, Equation 2, which corresponds to the
GMRES algorithm, induces global communication and synchronization on each
iteration of the method while computing the residual. Hence, we started
developing a prototype version of iPIC3D allowing us to minimize, if not omit
entirely, global synchronization points and to use the AllScale Toolchain
potential more efficiently. Taking into account the scope of the Earth's magnetic
dipoles (see Section 4.2), a problem we aim to simulate, we started with a
primitive static field solver by computing fields only at the beginning of the
simulation. That is partially advocated by the strong field forces that imprison
particles in two dipoles. However, for more realistic scenarios, we provided an
explicit (direct) field solver, which requires synchronization and communication
only with neighboring nodes, implementing the so-called forward approach.
However, this scheme has its limitations as it requires smaller time steps and is
only a first order method. Therefore, we propose for the consideration, with the
potential to be implemented within the AllScale project, two other explicit
approaches with stronger numerical stabilities:

1. The second order leapfrog approximation (Birdsall 1991);
2. The classic Runge-Kutta method of the fourth order (RK4).

These two methods are currently being investigated. However, by eliminating
global synchronization and communication, we introduce much more (due to the
reduced time step) computational load. This may render the approach to be ill-
suited for large scale simulations.

As it is often the case that certain numerical schemes are beneficial for a set of
scenarios, we consider also to provide implementations of Krylov-type solvers as
GMRES, but with the relax (reduced) requirements on synchronization and
communication. We consider two possible solutions:

1. BiCGStab (Vorst 1992) (Krasnopolsky 2010) – which is a pipelined
Krylov-type solver that has a strong potential to eliminate some fraction
of global communication;
2. Polynomial Krylov-type solvers (M. Eiermann 1985). This type of solvers
was designed back in 80s when communication was an issue as it is now.
These solvers require to construct polynomials based on the information
concerning the eigenvalues spectrum. This approach has a strong
potential to reduce global communication and synchronization to its
minimum.

These two types of methods have very promising features. Furthermore, they do
not require any changes to the time step or a grid, so the same constraints
remain as for GMRES.

3.1.2 Particle mover
Many PIC implementations try to use explicit solvers – such as leapfrog
approximations or Tajima’s scheme (Particle-in-Cell Consulting LLC 2011) – to
solve the equation of motion (see Equation 5). Obviously, these approaches have
their advantages, however they may not assure numerical stability as it was
shown in the above reference. To aim for latter, we propose to employ a second
order scheme, which is called Boris mover (Boris 1970), for solving the equation
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications
preparation and evaluation (a)

of motion. This scheme from 70s involves also computations on a half time step
points and strengthens numerical stability. In addition, it is well suited for task-
based nested recursive parallelism as each particle is processed independently.

3.2 API Description
All four stages of iPIC3D are essentially parallel loops iterating over all cells in
the 3D-space grid and such parallel loops can be directly mapped to a recursive
formulation.

To implement a parallel loop using nested recursive parallel tasks, each tasks
covers a range of elements. The initial task covers the entire loop range. If split,
the task’s range is split in half, producing two new tasks. This general principle is
encoded into a generic pfor operator offering the same abstract interface like a
conventional loop.

Thus, by utilizing the pfor operation on the structure of the 3D grid of the PIC
code, tasks covering sub-regions of the overall 3D grid are generated and
distributed throughout the system. Furthermore, operations on particles within
cells are additionally parallelized by utilizing parallel loops over the

Figure 2 illustrates an example of the resulting task-decomposition for the

The grid of cells needs to be adapted to facilitate its dynamic distribution among
multiple address spaces. To this end, facilities to address sub-regions of the
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications preparation and evaluation (a)

overall grid have to be provided to the runtime system to be able to address fractions of the overall data block as well as to manage and manipulate the data distribution. While several different formats for addressing sub-regions are conceivable, for the PIC simulation unions of axis-aligned boxes are utilized. Thus, arbitrary sub-blocks of the grid may be moved between address spaces by the AllScale runtime system.

Since n-D grids are a common concept, the AllScale infrastructure provides a predefined, generic grid implementation covering the corresponding implementation details similar to containers in the standard template library (STL) provided by any C++ implementation. Thus, the adaptation of the data structure narrows down to using a container type instead of a raw array within the simulation code.

### 3.3 Validation

Porting applications from one environment to another is a tedious process that requires frequent validation of its correctness. Hence, while writing the four key parts of the iPIC3D code we follow the task-based nested recursive approach as the AllScale Toolchain provides. We utilized unit tests initially to cover every sub-function. In addition, we compared the computed numerical results of larger parts – such as the particle mover – against the MATLAB version of the code. Thus, by following this approach, we ensured both a high unit test coverage of the code and its correctness.

Validation of the entire AllScale iPIC3D pilot is focused on comparisons against the original pilot. There are two techniques for this comparison:

1. Compare results of computations with results of a certain time step.
2. Compare only the final results.

The first is applicable for relatively small problems such as the particle-wave interaction, see Figure 4: there we compare the position and the velocity of particles on every intermediate time step. As we target large scale simulations, we consider comparing only intermediate stages of simulations by, for instance, taking the snapshots of the entire simulation, including the particle distribution and/or field values. The most appropriate way is to verify the final result of simulation, for example, by comparing the total accumulated energy for both particles and fields.

### 3.4 Relative Scalability

As adjustment of the iPIC3D pilot towards better utilization of the AllScale API/SDK capabilities is still ongoing, we do not show scalability results of the entire pilot runs here as they might be erroneous and may lead to an incorrect interpretation of the overall approach. However, we present our preliminary scalability results for the particle mover part of the code, see Figure 3. This part is the most compute-intensive segment of the code, especially in the Earth’s radiation belts simulations, see Section 4.2. For both the original iPIC3D and the AllScale PIC3D particle mover, we used the same input files that corresponds to the tiny benchmark, see Table 1. But, the compilers were of different versions: gcc v.4.9.2 for the original particle mover since the code failed to compile with the most recent version of gcc; gcc v.6.2.0 for the AllScale particle
mover. The experiments were carried out on an Intel Ivy Bridge CPU with 1TB RAM per node on the Tegner cluster at PDC, KTH. We fixed the problem size and varied the number of cores used for the AllScale implementation – the strong scaling test. The original code was executed using a configuration of eight MPI processes (2x2x2) due to the specific 3D domain splitting mechanism employed in the original code. These preliminary results indicate that AllScale is capable to boost the performance of the iPIC3D compute-intensive part. More detailed study as well as large simulations, involving the entire pilot, are on their way.

![Graph showing strong scaling of the particle mover part of the iPIC3D code implemented with the AllScale API/SDK and compared against the original version on an Intel Ivy Bridge node at PDC.](image)

Figure 3: Strong scaling of the particle mover part of the iPIC3D code implemented with the AllScale API/SDK and compared against the original version on an Intel Ivy Bridge node at PDC.

We foresee to present the fully functional pilot with more scalability results on shared-memory architectures at the mid-term review.

## 4 Benchmarks

We list below two main test cases to be used within the project: particle-wave interaction and Earth’s radiation belts. The main focus is on the latter.

### 4.1 Particle-wave interaction

Figure 4 presents the particle trajectory in the Earth radiation belts simulation at the example of one proton. The initial particle position is located at $x = 4R_e$, $y = 0$, $z = 0$, where $R_e = 6,378$ km is the Earth radius, with the pitch angle equal to 30 degrees. This result demonstrates that under the strong magnetic field forces any particle (electron or proton) with its position relatively close to the radiation belts and a certain value of the pitch angle is trapped in the Earth radiation belts. This simulation is a preliminary step for the Earth’s radiation belts simulation.
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications preparation and evaluation (a)

4.2 Earth’s Radiation Belts
The AllScale iPIC3D code has been designed to tackle simulation with severe work-imbalance. The simulation of Earth’s radiation belts (see Figure 5) is a good example of such problems. The Earth dipolar magnetic field traps high-energetic particles in two spatial regions around the Earth. These regions have a doughnut shape and for this reason they are called belts. The trapping of particles in the radiation belts depends on the initial pitch angle (angle between the magnetic field and the particle velocity vector) and on the energy of the particle. Because particles are trapped in these regions, there is a high concentration of particles in the radiation belts and a small number of particles outside the radiation belts. For this reason, tasks, which are responsible for the computations on the cells inside the belts regions, have very high workload compared to the rest of the grid.

Figure 4: A trajectory of a proton trapped in the radiation belt.

Figure 5: Earth’s (Van Allen) radiation belts with satellites. Credit: NASA.
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications preparation and evaluation (a)

In order to facilitate efficient tuning of the iPIC3D code within an intra- and inter-node level, we provide in Table 1 some realistic test cases.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Number of cells</th>
<th>Particles per cell</th>
<th>Time steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny</td>
<td>$16^3$</td>
<td>8$^3$</td>
<td>640</td>
</tr>
<tr>
<td>Small</td>
<td>$32^3$</td>
<td>8$^3$</td>
<td>1,280</td>
</tr>
<tr>
<td>Medium</td>
<td>$64^3$</td>
<td>10$^3$</td>
<td>2,400</td>
</tr>
<tr>
<td>Large</td>
<td>$160^3$</td>
<td>12$^3$</td>
<td>4,800</td>
</tr>
<tr>
<td>Real world</td>
<td>$320^3$</td>
<td>16$^3$</td>
<td>9,600</td>
</tr>
<tr>
<td>Exascale</td>
<td>$5,000^3$</td>
<td>&lt;22$^3$</td>
<td>&lt;100,000</td>
</tr>
</tbody>
</table>

*Table 1: Realistic test cases for iPIC3D.*

The idea of these test cases is to gradually increase the simulation domain as well as the number of particles per cell in order to reach the PIC simulation of the entire radiation belts formation, which is the last test case in the table.

5 Status and Future Work

All the four core parts of iPIC3D have been ported to the AllScale API/SDK. This process required considerable time at the beginning to get familiar with API as well as to restructure the entire numerical scheme of the pilot. But, once started, the process went smoother and much faster. My productivity was boosted by using the AllScale API that offers implicit recursive parallelism, e.g. the \texttt{pfor} operator. Another useful property, which scientists tend to forget, is test-driven development that is one of the key concepts in the project. So, developers are confident that segments of their codes produce correct results. This greatly reduces the chance of otherwise frequently encountered bugs in numerical codes. Furthermore, the idea behind the AllScale engineering is to preserve the structure of mathematical equations. For instance, the particle mover using the Boris method was implemented originally in MATLAB without preserving the structure of mathematical equations. Instead, we restructured and encapsulated this code via operator overloading to roughly four lines, making the four main formulas in the Boris method easy to recognize.

We presented the main concepts of the AllScale Toolchain and the corresponding iPIC3D implementation using a simple example (see Figure 4) at the EASC conference (Roman Iakymchuk 2016) held in Stockholm, Sweden, April 25-29, 2016.

Our future work is focused on verifying the current implementation on the proposed set of test cases and comparing its performance against the original version. Moreover, we plan to provide multiple implementations of field solvers depending on the needs. In addition, we will also verify more sophisticated boundary conditions, e.g. open boundary conditions as in the original version of the pilot. One of our main focused would be on performance optimization and tuning. We plan to overlap the four main computing stages as well as time steps, which would be a sort of parallel in time implementation of the iPIC3D pilot.
D6.2 – iPIC3D implicit particle-in-cell code for space weather applications preparation and evaluation (a)

6 Bibliography


Roman Iakymchuk (KTH), Herbert Jordan (UIBK), Peter Thoman (UIBK), Thomas Heller (FAU), Stéphane Monté (NUMECA), Benoît Leonard (NUMECA), Emanuele Ragnoli (IBM), Fearghal O. Donncha (IBM), Khalid Hasanov (IBM), Xavier Aguilar (KTH), Dana Akhmetova (KTH), Kiril Dichev (QUB), Thomas Fahringer (UIBK). D2.1 – Requirement Specifications and Reports on External Technological Developments (a). Innsbruck: AllScale Project, 2016, 64.

