HP2C Project Proposal
Operational COSMO Demonstrator (OPCODE)

Principal Investigator	Dr. Oliver Fuhrer Federal Department of Meteorology and Climatology MeteoSwiss Krähbühlstr. 58 Postfach 514 8044 Zürich
Project co-Pis **(in alphabetical order)**	Dr. Isabelle Bey (C2SM, ETH), Dr. Daniel Leuenberger (MeteoSwiss), Dr. Philippe Steiner (MeteoSwiss), Dr. André Walser (MeteoSwiss)
Project title	**Operational COSMO Demonstrator (OPCODE)**
Project partners	Supercomputing Systems AG Technoparkstr. 1 8005 Zurich
Start date	01.07.2011
End Date	30.06.2013
Requested funding	CHF 539'501
Annexes	Budget forms

Executive Summary

It is the aim of this project to leverage the research results of the ongoing HP2C COSMO project and apply them on a co-designed demonstrator implementation of the production environment of MeteoSwiss making aggressive use of GPU technology. In addition to a successful completion of the HP2C COSMO project, two main areas still require significant development in order to achieve this goal. First, several parts of the model code have not been considered in the HP2C COSMO project or still need to be consolidated for a full GPU implementation. Second, for a operational implementation of the COSMO model, the full time-critical part of the production workflow needs to be adapted and tuned to the demonstrator implementation. For these two work packages, additional funding is requested.

If successful, the resulting demonstrator would represent the first full GPU implementation of a full numerical weather prediction and regional climate model encompassing all parts of the code which are necessary in an operational production environment. The demonstrator will have a reduced power consumption and price as compared to the typically employed hardware, and thus potentially drastically reduce the total cost of ownership. As a result, we expect a direct impact on future procurements of HPC hardware in Switzerland, but also an impact on an international level and a high visibility of the HP2C initiative through the presentation of the key achievements in the established international networks.

1 Scientific Background

The societal impacts and economic benefits of weather and climate information have been rapidly increasing over the past decade. In Switzerland, the Swiss Federal Office of Meteorology and Climatology (MeteoSwiss) is by Federal mandate the national provider for weather and climate services. As a central pillar in the tool-chain for providing high-quality, time-critical information to the public and authorities, MeteoSwiss in collaboration with the Swiss National Supercomputing Centre (CSCS) computes several times per day high-resolution weather forecasts over the Alpine region using the weather and climate prediction model of the Consortium for Small-scale Modeling (COSMO) [4] [10].

The COSMO model is developed and maintained by the national weather services of 7 European countries including Switzerland. Furthermore, it is applied for regional climate prediction at over 41 universities worldwide. COSMO typically runs on proprietary platforms with efficiencies between 3% and 13% of peak performance. Expressed in sustained performance, this results in several hundreds of GFLOPS. The limited scalability of weather forecast codes is also reflected in a decreasing efficiency with higher core counts and can go far below 1% for core counts of above 10.000. Low percent of peak performance is typical for stencil code on structured grids [2] [3], as they have a O(1) computational density, executing a few FLOPs per load/store operation. This provides a unique opportunity for a co-designed system, where hardware and software are designed to match more optimally to the requirements of the problem to solve.

Today, graphics processing units (GPUs) are low-cost, low-power (watts per flop) and thus a very high performance alternative to conventional microprocessors. For example, NVIDIA's Tesla C2050 with a theoretical peak of 515 GFLOPS (DP) comes at similar cost as a processor, but an order of magnitude faster and with significantly higher memory bandwidth. Moreover, GPU performance is expected to sustain an increase at a rate of approximately doubling annually.

First attempts to port numerical weather prediction and climate models to GPUs have already been reported in the literature. Pioneering work using both rewrites in CUDA and directive based approaches of specific kernels were done for WRF [5] [6], which is a code very similar to COSMO. The research work done with WRF showed the limitation of the concept due to only partial GPU porting and total forecast resulted in a 1.2 speedup, due to the massive overhead of the data transfers to the GPU and back. The Japanese model ASUCA has been almost fully rewritten in CUDA and large speedups have been reported (with respect to a CPU implementation) [7] [8] [9]. But as the code has been completely rewritten in CUDA it was not taken back by the national weather service and the model community due to lack of CUDA knowhow in the model developer community and the additional complexity of maintaining two code bases. A third approach using an DSL implementation of the HIRLAM weather model and a CUDA backend for the code generator has shown a speedup of over an order of magnitude for the dynamical core including the costly data copying between the GPU and CPU memories [1]. Further exploring the parallelization across several GPUs they showed that the speedup was reduced to 3.6 due to the slow exchanges between the GPUs [11] [12]. The approach using a DSL implementation is only applicable to the dynamical core and has never been adopted by the official version of the HIRLAM code.

The ongoing HP2C COSMO research project is currently investigating novel methods of taking advantage of GPGPU technology for the numerical weather and climate prediction code COSMO. The parts of the model which are undergoing constant development are being ported to the GPU using a directive based approach. The dynamical core, which takes up the largest part of the

runtime but is developed by a small group of researchers, is being re-implemented in C++ based on a stencil library with both a CPU and GPU implementation available.

The main goal of this project is to apply the research results of the HP2C COSMO project and demonstrate their impact on operational weather forecasting of MeteoSwiss, specifically, as well as on the numerical weather and climate prediction community, in general.

2 Current Operational Suite

MeteoSwiss provides the public, government and private clients with weather information, for example time-critical model forecasts. These products need to be delivered within a pre-defined timeframe in order to be useful for decision takers. When co-designing the future hardware and software for the production suite, the full workflow starting from the initial triggering of the suite to the dissemination of the last time-critical product has to be considered. In the following we outline this workflow.

2.1 Operational Workflow

Model chain

MeteoSwiss produces operational numerical weather predictions with two different model setups called COSMO-7 and COSMO-2. The former is the 6.6 km version covering large parts of Europe and a forecast-range of +72h. The latter is the 2.2 km version covering the Alpine region and a forecast-range of +24h. The lateral boundary fields for COSMO-7 are provided by the global model IFS (Integrated Forecast System) from the European Centre for Medium-Range Weather Forecasts (ECMWF). For COSMO-2 they are taken from COSMO-7. This model chain and the operational domains of COSMO-7 and COSMO-2 are shown in Figure 1.

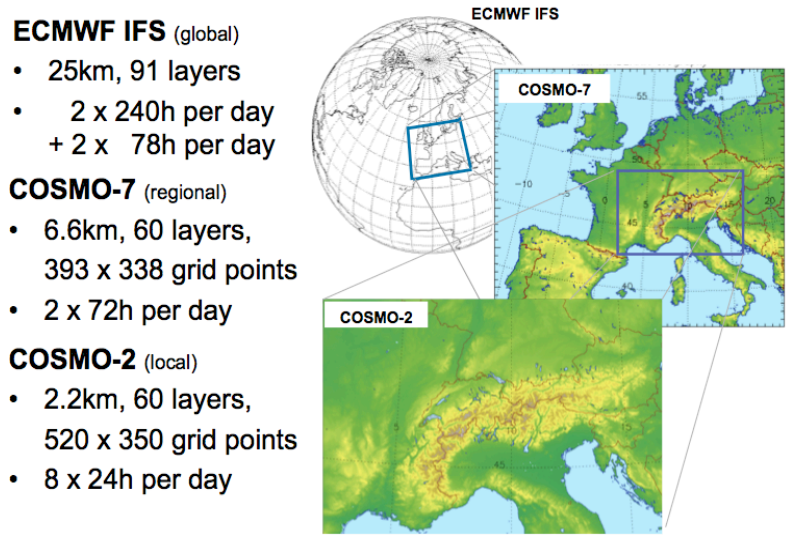


Figure 1: Model chain of the MeteoSwiss operational forecast suite.

Production Scheme

COSMO-2 forecasts up to +24h as well as the COSMO-7 forecasts required to provide the boundary conditions for COSMO-2, are computed every three hours. Three times a day, the COSMO-7 forecasts are extended to +72h (00, 06, 12 UTC runs). Usually, only these +72h forecasts are referred to as COSMO-7 forecasts.

The initial conditions for the forecasts are provided by an assimilation cycle running every three hours both for COSMO-2 and COSMO-7 with a data cut-off time[1] of 45 minutes. Such an assimilation cycle aims at computing an analysis both consistent with the model and as close as possible to observations. To this end, the model is fed by observations (e.g. soundings, radar) during an assimilation run to force the model trajectory towards the observations using a technique called nudging. Figure 2 provides an overview of the COSMO assimilation cycles and forecast runs.

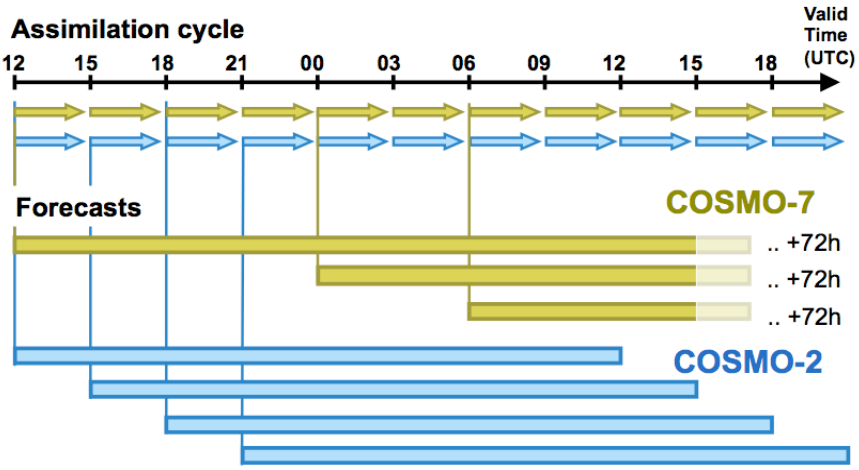


Figure 2: Operational COSMO assimilation cycles and forecasts (without COSMO-7 forecasts used for COSMO-2 boundary conditions only). Only 4 out of the 8 three-hourly COSMO-2 runs per day are sketched.

The COSMO suite is hosted on a Cray XT4 at the Swiss National Supercomputing Centre (CSCS). In the operational configuration COSMO-7 and COSMO-2 are run on 246 nodes populated with an AMD Opteron Budapest quad-core 2.3 GHz CPU. The interpolation of the boundary conditions (see int2lm in Figure 4) is run in parallel to the forecast on two additional nodes. The elapsed times for the individual parts of the suite are:

- COSMO-7 assimilation (3h): 1 min
- COSMO-7 forecast 0-24h: 7 min
- COSMO-2 assimilation (3h): 4 min
- COSMO-2 forecast 0-24h: 23 min
- COSMO-7 forecast 24-72h: 12 min

The schedule of the forecast suites and the workflow of a specific suite including post-processing (see below) are depicted in Figure 3.

[1] The *cut-off time* refers to the time gap between forecast initial time (e.g. 00:00 UTC) and the effective start time of the suite (e.g. 00:45 UTC), which is delayed to wait for latest observations.

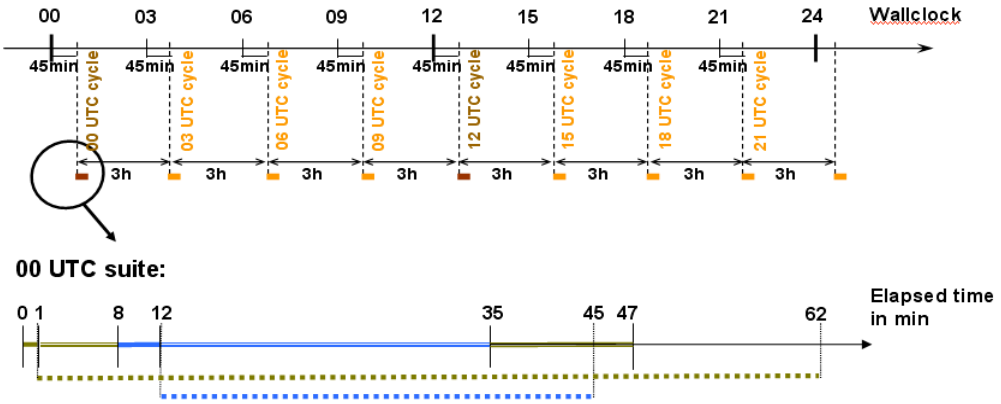


Figure 3: Schedule of the forecast suites and workflow of the 00 UTC suite with COSMO-7 (olive-green) and COSMO-2 (blue) computing (solid lines) and post-processing (dashed lines) tasks.

In addition to the fixed forecast schedule there is also the possibility the start COSMO-2 forecasts on-demand at any time based on the boundary conditions of the latest available COSMO-7 forecast and with an own assimilation cycle.

Pre- and Post-processing

In order to prepare external data and observations (e.g. snow analysis, radar images, sea temperature) for the forecast suite, a few pre-processing tasks are running ahead of the forecast suite. The computational effort required is, however, almost negligible. On the other hand, a comprehensive post-processing runs in parallel to and after the model forecasts producing the forecast products requested from internal and external customers. This pre- and post-processing currently runs on four dual-socket service nodes populated by 2.4 GHz AMD Opteron Shanghai quad-core CPUs (see Figure 4). The main workhorse generating the bulk of all non-graphical products is the fieldextra code, which runs in parallel to the COSMO-2 and COSMO-7 model forecasts and produces products sequentially. After the replacement of metview, the main application for graphical products will be the NCAR Command Language (NCL) and Interactive Data Language (IDL). These are run in parallel to the model forecasts and the products are distributed to as many instances as needed to keep pace with the corresponding model forecast (~12 for COSMO-7 and ~6 for COSMO-2). The trajectory tools lagranto and the dispersion model flexpart are run after the model forecasts and the latter on-demand only. In summary, the post-processing consists of a significant number of tasks reading the model forecast output (or large parts of it) within a short time and thus both requires a certain amount of computing power as well as a storage system with high I/O capacities.

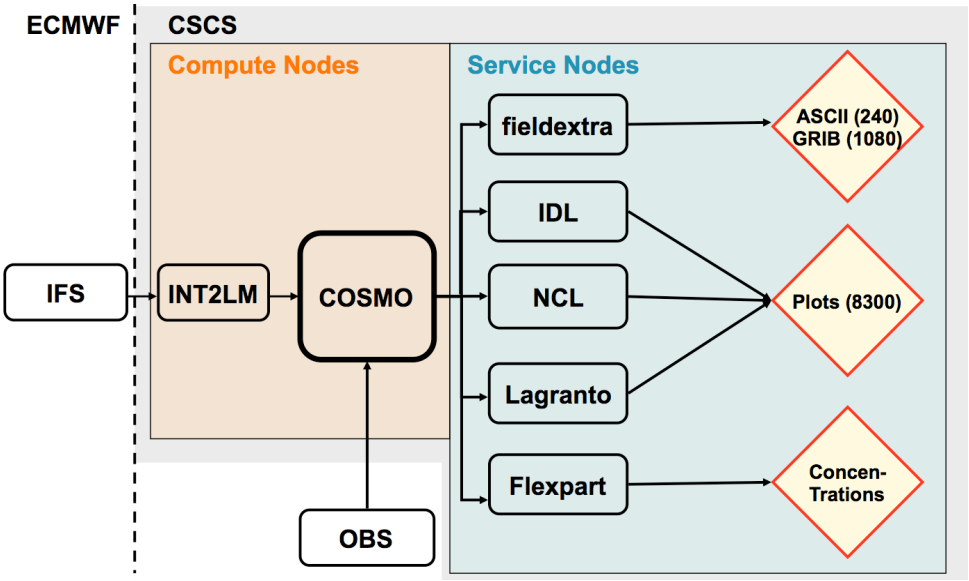


Figure 4: Overview of compute and post-processing tasks and the involved data flow for the operational suite, as it will be implemented until the end of 2011.

Dataflow

The boundary conditions of the global model IFS are disseminated by ECMWF to the operational host and amounts to 4 GB/day. The model forecast output of a full suite is about 42 GB. In addition, the interpolation of boundary conditions produces 8 GB and the post-processing about 20 GB, respectively. The latter data amount has to be transferred to Zürich to the MeteoSwiss dissemination system. The suite requires a short-term storage for initial and boundary condition data of about 200 GB, whereas one part of the model output is archived for the long-term.

Failure Resilience

For the operational suite a slightly smaller but otherwise identical machine is available to act as failover host. The input data for the forecast suites are synchronized to the file system of the failover host in real-time. A failover switch is done, however, only as the last action if other measures to recover a suite fail and it has to be done manually by the operator of the suites. Unsuccessful tasks are automatically retried as a first action. In case input data are missing, a forecast suite waits for it or uses alternative (mostly older) data if appropriate.

2.2 Software

- **COSMO:** The COSMO model is the main CPU consuming application and the heart of the operational forecasting system. It runs on 246 Cray XT4 compute nodes using a MPI parallelization. It is the main focus of the HP2C COSMO project and described in detail in the HP2C COSMO project proposal. Some performance details are given below.
- **int2lm:** The interpolation of driving model output (from the GME, IFS, COSMO or UM models) to supply boundary conditions for a COSMO model forecast is handled by the int2lm software. It is a Fortran 90 code parallelized using MPI library calls and developed by the Consortium for Small-Scale Modeling (COSMO) and strongly linked with the code of the COSMO model (accessing a common code library). For the MeteoSwiss operational suite int2lm runs in parallel on two Cray XT4 compute nodes.
- **COSMO-Package:** The operational suite is steered by the "COSMO-Package" developed at MeteoSwiss. It has a modular structure and is composed of about 50 shell and ruby scripts. In

operational mode COSMO and post-processing are running concurrently; warnings and exits are transmitted to the permanently on-duty operators.

- **fieldextra:** Generic tool to manipulate numerical weather prediction model data and gridded observations used within the Consortium of Small-Scale Modeling (COSMO). It is the main post-processing tool for non-graphical products. It is developed at MeteoSwiss, consists of about 50k lines of Fortran 90 code and has a very modular structure. fieldextra is currently single-threaded and thus runs sequentially in two instances, for COSMO-7 and COSMO-2 post-processing, respectively. It turns out to be a significant bottleneck in the current operational suites.
- **Visualization:** All visualization software packages (IDL, NCL) are third-party software and macro-language or script-based. They read subsets and derived model data provided by fieldextra. The jobs run in parallel in as many instances as needed to keep pace with the corresponding model forecast. Hereto, about 12 instances for COSMO-7 and 6 for COSMO-2 are currently needed. They usually run one to a few minutes.
- **lagranto:** Several real-time products depend on the calculation of trajectories. lagranto is a tool developed at ETH and based on a Fortran code and surrounded by IDL scripts for visualization. The calculation is split into one task for forward and one for backward trajectories. Both tasks read the full model fields and run for about 15 minutes (including visualization which is about 15% of the runtime).
- **flexpart:** flexpart is an atmospheric particle dispersion model developed by the Norwegian Institute for Air Research (NILU). It consists of a sequential Fortran 90 code under the GPL license and is supported by a growing community. flexpart is conducted on-demand, but it is nevertheless an important time-critical tool frequently used by the forecasters. In order to achieve a reasonably acceptable performance, the code has been rudimentary OpenMP parallelized at MeteoSwiss. The tool reads the full model output and runs about 11 min. for a 24h COSMO-2 and 18 min. for a 48h COSMO-7 dispersion run, respectively.

2.3 Hardware

The operational system buin is a Cray XT4 with 260 compute nodes populated by 2.3 GHz AMD Opteron Budapest quad-core CPUs. The machine has two login nodes (buin1, buin2) and one node dedicated to the control of the operational suites (buin8), all three populated with AMD Opteron dual-core CPUs. For the pre- and post-processing 4 nodes provided by an XT5 blade are available, each node populated with 2 AMD Opteron Shanghai quad-core CPUs. The failover machine dole has the same specifications but 172 compute nodes only.

3 The HP2C-COSMO Project

The HP2C COSMO has started in mid-2010 and will last until the end of 2012. It is the goal of this demonstrator project to leverage the developments and results of the HP2C COSMO project and demonstrate their impact on MeteoSwiss' operational weather forecasting system. This project is thus very closely inter-linked with the timeline and outcome of the HP2C COSMO project. In this section, we briefly outline the activities of relevance from the HP2C COSMO project.

3.1 Relevant Tasks

The solver of the Euler equations of fluid motion of an atmospheric model (hereafter referred to as the dynamical core) typically comprises the largest chunk of the runtime of an atmospheric model. As part of task 3 of the HP2C COSMO project, the dynamical core of the COSMO model is being rewritten. This rewrite encompasses several fundamental changes to the COSMO software: shift from Fortran 90 to template meta-programming in C++/CUDA, change of storage

order and data structures, and a code written on a higher abstraction level to allow for rapid prototyping of new algorithms. The algorithms at the base of the dynamical core are stencil methods on structured grids, and as such have a very low arithmetic intensity and are memory bandwidth limited on commodity x86 processors. A strong focus of the dynamical core rewrite is thus to improve data reuse with cache-oblivious algorithms, to use the available memory bandwidth more optimally.
In parallel, the code modules that represent the effect of sub-grid processes by using resolvable scale fields (hereafter referred to as the physical parametrizations) are being adapted to the new data structures and ported to GPUs using a directive based approach. In contrast to the dynamical core, the physical parametrizations are typically compute bound as the algorithms involved frequently employ trigonometric, exponential, root and logarithmic intrinsic procedures. The physical parametrizations code of the COSMO model is shared with another atmospheric model and is undergoing constant development and extension by domain scientists, thus an aggressive shift towards another programming language is currently not feasible for this part of the code.
Since the current I/O layer in the COSMO model is only partly parallel and asynchronous, the associated communication and I/O overhead can be substantial. As part of Task 2 of the HP2C COSMO project, new I/O strategies based on existing parallel I/O libraries as well as in-situ visualization techniques are being explored.

3.2 Expectations and Timeline

For the planning of this project, we expect the following results from the HP2C COSMO project:

1. A fully functional and validated rewritten single-node version of the dynamical core (as compared to the reduced Fortran 90 standalone version of the dynamical core provided to the rewrite team). The dynamical core runs both on CPUs and GPUs. Due to optimized usage of memory bandwidth the dynamical core is at least two times faster than the Fortran standalone version on x86 multi-core processors.

   Q3 2011 – CPU version of new dynamical core available (D1)
   Q4 2011 – CPU version of new dynamical core validated (D2)
   GPU version of new dynamical core available (D3)

2. A fully functional version of the physical parametrizations package with data structures compatible to the data structures of the rewritten dynamical core and ported to GPUs using a directive-based approach.

   Q4 2011 – Physics package with adapted data structures (D4)
   Q3 2012 – GPU implementation of physics package (D5)

3. A refactored I/O layer which reduces the I/O bottleneck on massively parallel architectures using existing parallel I/O libraries.

   Q3 2011 – Implementation strategy for scalable parallel I/O (D6)
   Q2 2012 – Refactored I/O layer (D7)

4 Implementation of COSMO

4.1 Overview and Motivation

With 23 minutes runtime, the 24h COSMO-2 forecast is currently the largest chunk of the total of 47 minutes of compute time for the operational suite on the Cray XT4 system. Using approximately 2.7% of the total peak performance (8.8 TFLOPS for 980 cores) the COSMO model is running at roughly 240 GFLOPS sustained. In terms of memory bandwidth the COSMO code uses approximately 74% of the 2.6 GB/s maximum bandwidth per core, with certain parts of the

dynamical core saturating the available bandwidth. Under the gross assumption that the runtime of the dynamical core will scale onto the GPU with the available memory bandwidth for the dynamical core and with the available peak performance for the physical parametrizations (see also Table 1) we could expect to achieve the same COSMO-2 forecast in 23 minutes on the order of 20 NVIDIA Tesla C2070 GPGPU cards.

Table 1: Comparison of performance characteristics of CPU's used for the current operational implementation and current state-of-the-art GPU.

	Cores	Freq. (MHz)	Peak Perf. DP (GFLOP/s)	Memory BW (GB/s)	Power Cons. (W)
AMD Opteron Budapest	**4**	**2300**	**36.8**	**12.8**	**115**
Tesla C2070 Fermi	**448**	**1150**	**515**	**115**	**238**

The above is only a gross estimation and does not consider several aspects such as the simpler memory subsystem on the GPU, CPU-GPU as well as inter-GPU communication bandwidth and latency, and difficulties in porting the full COSMO code onto the GPU. On the other hand, the above considerations have made no use of the expected speedup of the dynamical core rewrite from the HP2C project. Also, first results from the porting of the physical parametrizations to GPU show that the assumed scaling with peak performance can be used successfully for the most time consuming physical parametrization. The main point of the above estimation is to illustrate that *a successful implementation of the COSMO model on a multi-GPU Server can potentially have a strong impact on the total cost of ownership of the operational HPC system*, namely on the initial investment as well as the power consumption.

4.2 Implementation Tasks

The COSMO model uses a conventional time-stepping scheme to integrate the partial differential equations, which describe the atmosphere. In consequence, the workflow of the COSMO model consists of an initialization and then a sequence of tasks that are repeated every timestep (see Figure 5).

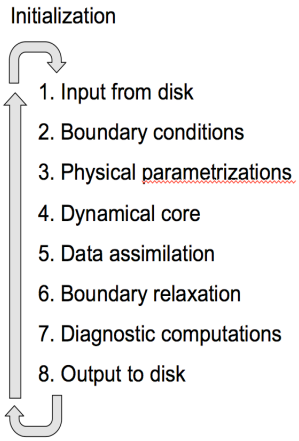


Figure 5: Outline of the basic workflow of COSMO model.

As can be seen in Figure 6, the HP2C COSMO project has targeted the two most time-consuming parts of the COSMO code, the dynamical core (59%) and the physical parametrizations (22%), for optimization and GPU implementation. The example in Figure 7 illustrates that the penalty of

copying the data describing the full three-dimensional state of the atmosphere at a certain timestep from CPU to GPU and back is prohibitively large. An efficient GPU-implementation of the COSMO model has to avoid copy-in/copy-out of three-dimensional fields and *all tasks within a timestep have to run entirely on the GPU*.

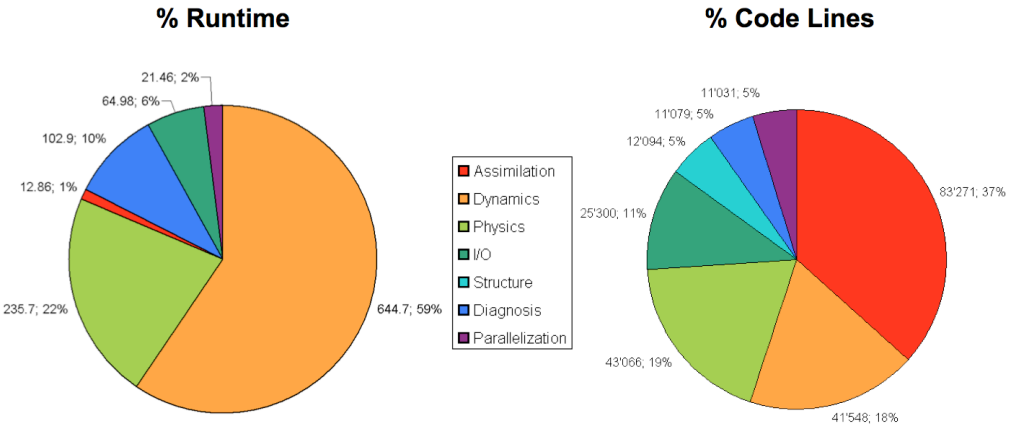


Figure 6: Distribution of different code components of the COSMO model in terms of percent of total runtime (left panel) and percent of total number of lines of code (right panel). Timings are given for a typical operational COSMO-2 forecast.

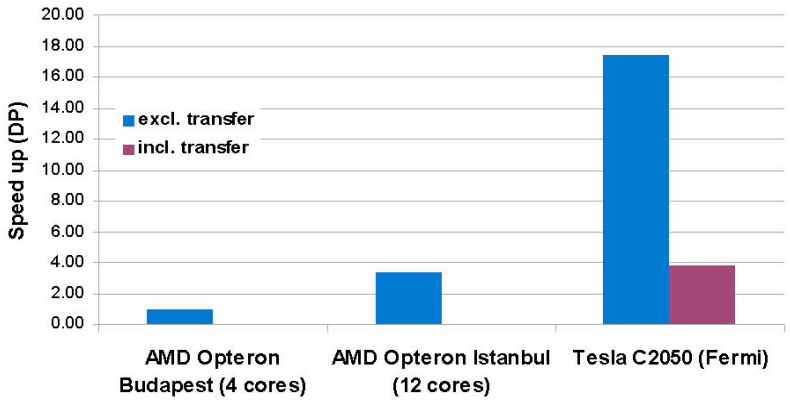


Figure 7: Speedup comparison of the microphysics between a single AMD Opteron quad-core (Budapest) CPU, a dual-socket AMD Opteron hex-core (Istanbul) node, and a PGI directive based GPU implementation on the Tesla C2050 (Fermi) GPGPU card. Purple bars indicate speedup including the CPU-GPU data transfer, blue bars are based on exclusive times.

In order to have the full timestep running on the GPU and to avoid CPU-GPU data transfers the porting efforts within the HP2C COSMO project have to be completed with the following tasks:

A1. Dynamical Core: The new version of the dynamical core has been reduced to a meaningful set of modules for the HP2C COSMO project. In order to run the MeteoSwiss operational suite, several parts of the dynamical core have to be implemented in the new code framework. Namely, the full Strang-splitting of the advection operators, the θ,p-dynamics in the fast wave solver, tracer transport, and the saturation adjustment.

A2. Parallelization: As the HP2C COSMO project will only deliver a single-node version of the dynamical core, a parallelization across several CPUs and GPGPU cards has to be developed and implemented. A library allowing for halo-updates, global reductions, gathers and scatters has to be accessible both in the C++ framework as well as the Fortran part of the code.

A3. Interoperability: The new dynamical core (written in C++) has to be integrated into the Fortran codebase and a common compile system has to be developed. For the GPU implementation, fields in the global memory of the GPGPU device have to be shared between the CUDA code of the new dynamical core and the directive based Fortran code of the physical parametrization. Even though there is currently no solution available to solve this issue, the GPGPU hardware has support for this via Unified Virtual Addressing (UVA) and we have reason to believe that future compiler version will have support for this feature. The code has to be adapted to take advantage of this feature.

A4. Data assimilation: Operational forecasting requires a data assimilation cycle (providing the initial conditions of each forecast) and activates data assimilation during the first hour of the forecast. The data assimilation code is not very time consuming, but consists of approximately 85'000 lines of code which are currently undergoing active development and modularization. Since the data assimilation applies a nudging scheme (relaxation of prognostic variables against interpolated, weighted observations), it requires access to and modifies the full three-dimensional prognostic variables on each timestep. A CPU version of the assimilation code would thus require extensive CPU-GPU data transfers and this is probably prohibitively time-consuming. The code has to be analyzed in detail and other options such as a directive based porting to the GPU have to be explored.

A5. I/O: The current I/O layer is designed for the CPU and assumes direct access to an operating system and file system. An additional software layer has to be introduced which controls the copying of the fields from the CPU to the GPU and back. Both input and output are good candidates for asynchronous I/O. The CPU can read the next required boundary condition file while the GPU is computing the forecast. Similarly, when the GPU is ready for output, the fields can be copied to an output buffer and transferred asynchronously to the CPU which computes additional derived variables and writes the data to disk. This software layer does not exist yet and has to be developed. These activities should be coordinated with and leverage the refactoring of the I/O layer within Task 2 of the HP2C COSMO project.

A6. Other: As can be seen in Figure 5, several other parts of the timestep of the COSMO model have to be ported to the GPU, namely the imposition of the boundary conditions (2.), the boundary relaxation (6.) and the diagnostic computations (7.), as well as the Pollen part of the COSMO-ART module, which is currently run operationally. The porting of these parts should not pose any difficulties and a porting both within the C++ and directive based part of the code is feasible.

For the purposes of the demonstrator, an inter-CPU-node parallelization is not yet strictly required. It should be noted though, that within the HP2C COSMO project an inter-node parallelization is planned and efforts in task A2 should be closely coordinated with this effort.

5 Implementation of the Operational Suite

5.1 Requirements

The demonstrator should – by definition of a demonstration project - fulfill the same requirements as the current operational machine operated by CSCS. These requirements are as follows:

- **Time to solution**: Should be similar or smaller than the time to solution of the current operational suite on the Cray XT4. Thus, the time for a 24h COSMO-2 forecast including the assimilation cycle, the required COSMO-7 forecast and the time-critical post-processing should be smaller or equal to 45 minutes, resp. 62 minutes including the COSMO-7 forecast

(see Figure 3). The required time to solution will determine the dimensioning of the demonstrator hardware (number and type of GPU cards, number and type of CPUs). It should be noted that for the purpose of the demonstrator, the time to solution is not a strict requirement, but will rather serve as a guideline for the dimensioning of a future operational implementation on heterogeneous architectures.

- **Reliability**: The operational suites have a reliability of ~99% for both COSMO-7 (delay < 66 min) and COSMO-2 (delay < 30 min). Required are 99% for COSMO-7 and 95% for COSMO-2. The demonstrator will serve as a baseline to see if these reliability requirements can currently be achieved with a heterogeneous CPU/GPU system.
- **Portability**: The operational suite is migrated to new hardware on a regular basis. It is thus a requirement to keep the software running operationally as architecture independent as possible.

5.2 System Design

The main goal of the current project is to demonstrate the feasibility and benefit of a GPU implementation of the current MeteoSwiss operational suite. To this end, we will construct a "demonstrator system" consisting of a co-design of both of the appropriate hardware and software. Co-designing the hardware and software in order to meet the operational requirements requires broadening the scope of consideration from solely the COSMO model to the full time-critical workflow of the operational suite as described in Section 2. Bottlenecks may lie in the pre- or post-processing and these may even be aggravated on the demonstrator if they are not considered in the design.

Workflow

We plan to implement the full model chain of COSMO-7 and COSMO-2 forecasts (see Figure 1) on the demonstrator. Since almost all other software (COSMO-Package, int2lm, fieldextra, NCL) runs in parallel to the COSMO model, only the COSMO model is ported to the GPUs whereas the other codes will remain on the CPU (see Figure 8). Sufficient CPU power has to be provided for these tasks.

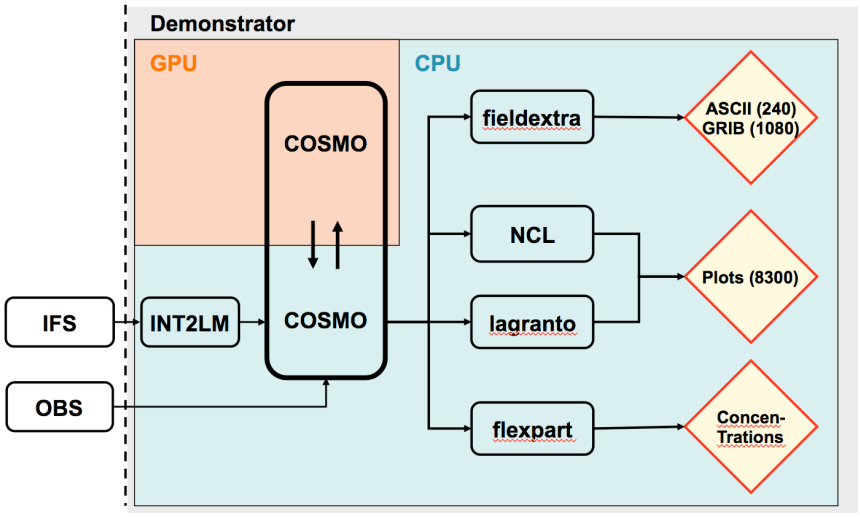


Figure 8: Overview of compute and post-processing tasks and the involved data flow for the demonstrator.

The following parts of the current operational workflow (see Section 2 for a complete description) are ***implemented on the demonstrator system***:

- **COSMO-Package**: The COSMO-Package is ported to the demonstrator system and will control the workflow. Significant modifications are necessary for the demonstrator.
- **COSMO:** The model is implemented with an aggressive shift to a GPU implementation (see detailed description and tasks in Section 4.2).
- **int2lm:** The interpolation software will run in its current state on the CPU in parallel to the COSMO model and is relying on the existing MPI parallelization.
- **fieldextra**: This post-processing tool will run in two instances on the CPU in parallel to the model forecasts. fieldextra is a significant bottleneck (cf. Figure 3) in the current operational suite due to its single-threaded design. A parallelization and optimization is absolutely essential to keep up with the data flow from the COSMO model. A speedup of fieldextra will be a key to achieve more flexibility in the demonstrator system implementation and can substantially speedup the workflow.
- **lagranto**: As the backward trajectory calculation can only be started once the last output file of the forecast has been written, lagranto has to be run after the end of the model forecast, but is part of the time-critical suite. For the demonstrator project, lagranto will be parallelized using OpenMP for multi-threading on a single CPU.
- **flexpart**: The dispersion model is run after the end of the model forecast in on-demand mode and has tight time-to-solution constraints. flexpart will be parallelized using OpenMP for multi-threading on a single CPU.
- **Visualization (NCL)**: The visualization is implemented in its currently existing form, with only slight optimizations to the workflow in order to adapt the graphical product generation to the demonstrator.

The following parts of the current operational workflow (see Section 2 for a complete description) are ***not implemented on the demonstrator***:

- **Pre-processing (except int2lm):** As all pre-processing tasks (except the interpolation of boundary conditions with int2lm) run outside of the COSMO-Package control scope and as the compute time and data volume is typically negligible, these tasks are not implemented on the demonstrator. The required products have to be copied in real-time from the operational machine.
- **IDL Visualization:** As this is only a very small fraction of the visualization tasks, all products generated by IDL are not considered for the demonstrator implementation.
- **COSMO-2 On-Demand:** The operational suite allows for on-demand COSMO-2 forecasts based on the latest COSMO-7 forecast. This functionality will not be included on the demonstrator.
- **Failover Capability:** For the purposes of demonstrating the reliability of the demonstrator, a failover system is not necessary. The experience gathered with the demonstrator will be a key towards designing a failover capability for a operational implementation on a heterogeneous CPU/GPU system.

System Software and Libraries

The following system software components or libraries are required for the demonstrator system:

- **Operating System:** Any standard flavor of Linux operating system.
- **Compiler Suite:** A suite of state-of-the-art compilers (C++, CUDA, Fortran, Fortran + GPU directives) has to be available on the demonstrator system.

- **Scheduling system:** In order to schedule and monitor the suite tasks a scheduling system is required. It is preferably of the same type on the demonstrator as the one on the operational system (currently PBSpro).
- **MPI Library:** Even if the demonstrator hardware will be a single-node, the int2lm software will require an MPI library to run in parallel.

System Hardware

The hardware requirements for the demonstrator system are as follows:

- **GPU:** The COSMO model will require GPUs to provide approximately 2.5 TB/s aggregate memory bandwidth and 10 TFLOPS peak performance.
- **CPU:** The COSMO model will require sufficient CPU power, in order to assure that the tasks executed on the CPU (diagnostic computations, I/O, etc.) are able to keep up with the GPUs. The pre- and post-processing will require at least 24 CPU threads while the model forecasts are running and will use more afterwards if available.
- **Memory:** The model requires about 13 GB of memory when running on a single node, which is the minimal amount of total global GPU memory required. The memory demand of the pre- and post-processing is about 16 GB. Since both run in parallel the available memory on the CPU should be 29 GB at least.
- **Storage:** The operational suite requires a scratch disk space of 500 GB at least (better 1 TB) for cache and working directories.
- **Network:** A Gigabit connection (or better) is required between the demonstrator system and the current MeteoSwiss operational system. In case of inter-node parallelization, a high-performance interconnect might be necessary.

5.3 Implementation Tasks

The adaption, optimization and porting of the reduced operational suite (as described above) to the demonstrator system requires the following tasks:

B1. Hardware: The hardware for the demonstrator will be a unique solution with an extreme high ratio of sockets to be selected to match as well as possible the above requirements, ordered and installed. The system will consist of two dual-socket nodes with 14 GPUs and 2 Solid-state drives (SSDs) in total. This gives an aggregated bandwidth of 1.6 TB/s (ECC on) and a peak performance of 7.5 TFLOPs. Each node will have a main memory of at least 24 GB. Both SSD cards will provide 800 GB scratch space in total. The nodes are integrated in an InfiniBand Fabric to simulate the final data transfer into the global parallel file system GPFS at CSCS.

B2. System Software: The required system will be integrated in the CSCS data center and a complete software stack will be installed, checked and maintained. The system will be managed by CSCS for the duration of the project.

B3. COSMO-Package: A major revision of the scripts is required for the demonstrator, mainly to enhance the flexibility in the workflow, which is required to adapt the package and in fine-tuning the tasks to the demonstrator system. The management of the jobs (submission, monitoring and check-pointing) is a particular bottleneck since the tasks are handled in a sequential fashion and thus the wait times of tasks increase with an increasing number of tasks. The revised COSMO-Package has to be installed, tested, and fine-tuned on the demonstrator.

B4. Post-Processing: The post-processing tools are a bottleneck in the current workflow and this will be aggravated if the COSMO model profits from the results of the HP2C

project or is applied with higher resolutions. A straightforward and effective solution is to parallelize the post-processing tools (fieldextra, lagranto, flexpart) with an appropriate paradigm (e.g. OpenMP for multi-threading on a single CPU). In addition, the post-processing tools will be adapted to the new I/O workflow of the COSMO model. For fieldextra, a GPU acceleration of individual kernels will be explored.

B5. Setup and Testing: A operational forecasting suite adapted/tuned for the demonstrator is implemented. A scheduler allows regular runs for extensive tests during three months. The suite is monitored, maintained and benchmarked. Still existing bottlenecks are identified and potential solutions are proposed.

6 Project Plan

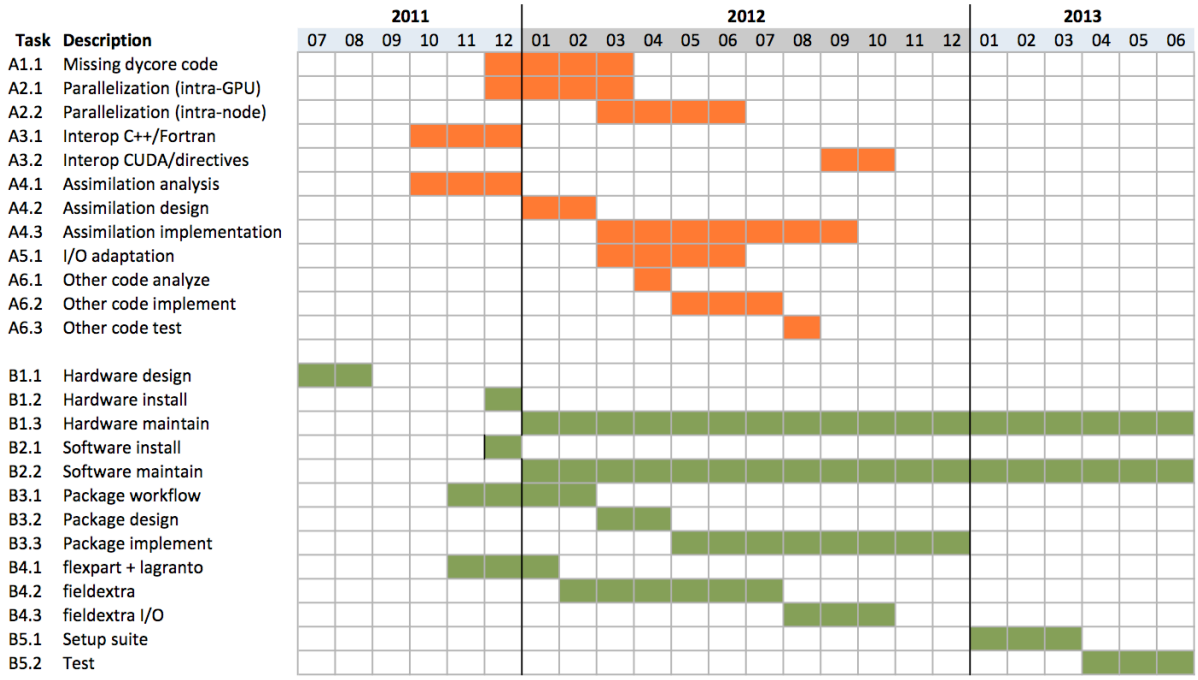
6.1 Tasks and Milestones

The detailed milestones of the work package A "Implementation of COSMO model" and work package B "Implementation of the Operational Suite" are summarized in the following table (FTE = full time equivalent):

Task	Milestone	Effort
A1. Dynamical Core	A1.1 Implement missing code into new dynamical core framework	0.3 FTE
A2. Parallelization	A2.1 Inter-GPU parallelization A2.2 Inter-node parallelization	0.5 FTE (in kind CSCS)
A3. Interoperability C++/CUDA/Fortran	A3.1 Design C++/Fortran build system A3.2 Implement CUDA/Fortran directives data sharing	0.3 FTE
A4. Data Assimilation	A4.1 Code analysis A4.2 Design strategy (GPU or CPU) A4.3 Implementation and testing	0.9 FTE
A5. I/O	A5.1 Adapt new I/O strategy to GPU implementation	0.2 FTE
A6. Other	A6.1 Analyze remaining code and design implementation (stencil framework or directives) A6.2 Implementation A6.3 Testing	0.4 FTE
B1. Hardware	B1.1 Design system and order B1.2 Hardware installation B1.3 Hardware maintenance	0.3 FTE (in kind CSCS)
B2. System Software	B2.1 Software installation B2.2 Software maintenance	0.2 FTE (in kind CSCS)
B3. COSMO-Package	B3.1 Analyze and design workflow B3.2 Design COSMO-Package B3.3 Implement	0.4 FTE
B4. Post-processing	B4.1 Parallelize flexpart and lagranto B4.2 Parallelize/optimize fieldextra B4.3 Adapt fieldextra to new I/O strategy	0.4 FTE 0.25 FTE (in kind COSMO) 0.1 FTE (in kind MeteoSwiss)
B5. Setup and Testing	B5.1 Setup operational suite on demonstrator B5.2 Testing	0.3 FTE
Z1. Project Management		0.2 FTE (in kind MeteoSwiss)
Z2. Support and Advising		0.7 FTE (in kind MeteoSwiss)
TOTAL		**3.2 FTE** 1.0 FTE (in kind MeteoSwiss) 1.0 FTE (in kind CSCS) 0.25 FTE (in kind COSMO)

6.2 Timeline

The detailed timeline of work package A "Implementation of COSMO model" and work package B "Implementation of the Operational Suite" are summarized in the following overview:



6.3 Requested Resources

The project amounts to a total of 5.5 FTEs of which 1.0 FTEs are provided in kind by MeteoSwiss (Z1, Z2), 1.0 FTEs are provided in kind by CSCS (A2, B1, B2) and 0.25 FTE are funded by the Consortium of Small-Scale Modeling (COSMO). Thus, we request funding for a total of 3.2 FTEs. These resources will be distributed as follows:

Tasks	**Effort**	**Location**
A1, A3, A6	1.0 FTE	SCS
A4	0.9 FTE	MeteoSwiss (PostDoc 1)
A5	0.2 FTE	C2SM
B3, B4, B5	1.1 FTE	MeteoSwiss (PostDoc 2)

The total requested funding amounts to CHF 539'501. A detailed budget is given in the Appendix.

6.4 Project Organization

The project will be executed under the auspices of MeteoSwiss in close collaboration with the Swiss National Supercomputing Centre (CSCS), Super Computing Systems AG (SCS) and the Centre for Climate System Modeling (C2SM ETH). The project will be integrated into the existing and efficiently operating network of the running HP2C COSMO project. The link to the COSMO weather and climate modeling communities of MeteoSwiss and C2SM, respectively, will be essential to feed back the project results into the community.

The project will be divided into three sub-projects with designated sub-project leaders (see Figure 9 for detailed view of project management). Secretarial support will be provided by the third-party funding project office at MeteoSwiss.

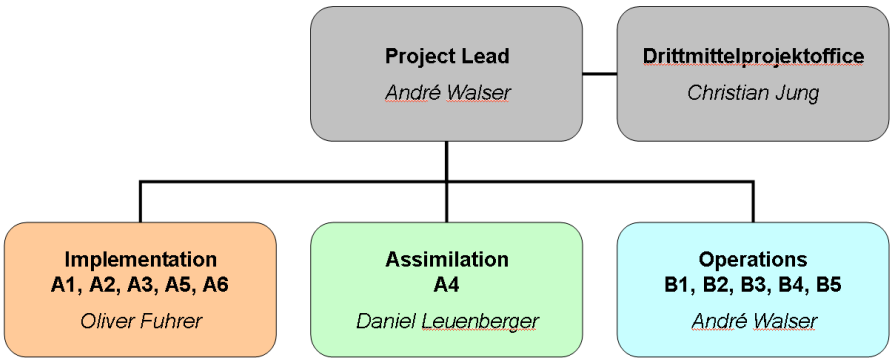


Figure 9: Structure of project management and distribution of the project tasks into the three sub-projects.

The work of tasks A1, A3 and A6 will be done in collaboration with Supercomputing Systems AG (SCS AG), due to the strong time constraints on the project. The details of the collaboration will be determined in a contract between MeteoSwiss and SCS AG upon approval of the project.

6.5 Dependencies and Risks

This project and entailed tasks have several external dependencies and risks. In the following we identify the main dependencies and risks and propose contingency plans in case of occurrence. Dependencies on deliverables of the HP2C COSMO project are marked accordingly and details are given in Section 3.

- Tasks A1 and A6 depend on a successful rewrite and GPU implementation of the dynamical core using a stencil library (D1, D2, D3). The fallback plan of the HP2C COSMO project, should a GPU implementation of the stencil library not be successful, is a hand-written CUDA implementation. Tasks A1 and A6 could also be accomplished with a hand-written CUDA implementation, but the time required would increase the needed human resources.
- Task A3.2 depends on the availability of a version of the physics package, which has been fully ported to GPUs using a directive based approach (D4, D5). In case not all physics modules should be available, a meaningful reduced set of physics modules will be used for the demonstrator. Task A3.2 also depends on the availability of a GPU programming model that allows sharing data on the GPU between CUDA code and Fortran code ported using a directive based approach. In case this should not be available, a copy-in/copy-out strategy would have to be implemented which will significantly reduce the potential speedup of a GPU implementation.
- Task A4 depends on the availability of a stable version of the data assimilation code, which is expected by the end of 2011. Also, it is a priori not clear if a GPU implementation of the assimilation code is feasible and how best it should be accomplished. If task A4.1 should conclude, that a GPU implementation is not feasible, the assimilation would have to be done on a CPU cluster. Since data assimilation is only done for numerical weather prediction, regional climate simulations could still be accomplished by a full GPU implementation.
- Task A5 depends on the results of HP2C COSMO project (D6, D7) and the applicability of the chosen I/O strategy to a GPU-cluster. If the results cannot be applied for the demonstrator, the time allocated for Task A5 could prove to be insufficient.
- The final implementation and testing of the operational suite on the demonstrator (Task B5) depends on the success and timely completion of most of the tasks of this project.

6.6 Expectations and Impact

In case of successful completion, we expect the following output from the project:

- First implementation of a full operational numerical weather prediction suite on a heterogeneous CPU/GPU architecture.
- Proof of concept machine with massively reduced energy consumption and price while keeping within required time to solution.
- Direct impact on HPC procurement of MeteoSwiss in 2013/14. Indirect impact on procurements of all members of the Consortium for Small-scale Modeling (COSMO) and Climate Limited-area Modelling Community consisting of a huge user-community of the COSMO model in over 8 national weather services and over 41 universities worldwide.
- High visibility of HP2C-COSMO project and the Swiss HP2C initiative through direct application of results for a quasi-operational implementation.
- Leap forward in the efficiency of use of HPC resources for numerical weather forecasting and regional climate models will provide new possibilities for the simulation and prediction of weather and climate.

7 References

[1] Cats, G. and L. Wolters, 1996: The HIRLAM project. *IEEE Computational Science & Engineering*, **4** (4), 4-7

[2] Christen, M., Schenk, O., Messmer, P., Neufeld, E., and Burkhart, H., Accelerating Stencil-Based Computations by Increased Temporal Locality on Modern Multi- and Many-Core Architectures. *Proceedings of the 2009 IPDPS conference*.

[3] Colella, P., 2004: Defining Software Requirements for Scientif Computing. See http://www.lanl.gov/orgs/hpc/salishan/salishan2005/davidpatterson.pdf

[4] Consoritum of Small-Scale Modeling (COSMO), http://www.cosmo-model.org/

[5] Michalakes, J. and Vachharajani, M., 2008: GPU Acceleration of Numerical Weather Prediction. *Parallel Processing Letters*, **18** (4), 531-548.

[6] Michalakes, J., J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang: The Weather Research and Forecast Model: Software Architecture and Performance*. Proceedings of the Eleventh ECMWF Workshop on the Use of High Performance Computing in Meteorology*. Eds. Walter Zwieflhofer and George Mozdzynski. World Scientific, 2005, pp 156-168

[7] Shimokawabe, T., T. Aoki, C. Muroi, J. Ishidia, K. Kawano, T. Endo, Akira Nukada, N. Maruyama and S. Matsuoka, 2010: An 80-Fold Speedup, 15 Tflops Full GPU Acceleration of Non-Hydrostatic Weather Model ASUCA Production Code. *SC10* November 2010, New Orleans, USA.

[8] Shimokawabe, T. and T. Aoki, 2010: GPU acceleration of weather prediction model, *ETHZ – Tokyo Tech Workshop: Computing with GPUs, Cells and Multicores*, ETH Zurich, Switzerland, May 10, 2010

[9] Shimokawabe T., T. Aoki, J. Ishida: GPU Acceleration of Meso-scale Atmosphere model ASUCA, *9th world Congress on Computational Mechanics and 4th Asian Pacific Congress on Computational Mechanics*, Sydney, Australia, 19 July 2010.

[10] Steppeler, J., G. Doms, U. Schattler, H. W. Bitzer, A. Gassmann, U. Damrath, and G. Gregoric, 2003, Meso-gamma scale forecasts using the nonhydrostatic model LM, *Meteorol. Atmos. Phys.*, **82**, 75-96

[11] van Engelen, R. A., L. Wolters, and G. Cats, 1997: Tomorrow's Weather Forecast: Automatic Code Generation for Atmospheric Modeling, *IEEE Computational Science & Engineering*, **4** (3), 22-31.

[12] Vu, V. T., G. Cats and L. Wolters, 2010: GPU acceleration of the dynamics routine in the HIRLAM weather forecast model. *Proceedings of the ASCI Conference*, November 2010, Veldhoven, Netherlands.