Manuscript prepared for Geosci. Model Dev. Discuss.
with version 2014/07/29 7.12 Copernicus papers of the LaTeX class copernicus.cls.
Date: 27 May 2015

# Experiences with distributed computing for meteorological applications: Grid computing and Cloud computing

**F. Schüller[1], S. Ostermann[2], R. Prodan[2], and G. J. Mayr[1]**

[1]Institute of Meteorology and Geophysics, University of Innsbruck, Innrain 52, 6020 Innsbruck, Austria
[2]Institute of Computer Science, University of Innsbruck, Innsbruck, Austria

Correspondence to: F. Schüller (felix.schueller@uibk.ac.at)

**Abstract**

Experiences with three practical meteorological applications with different characteristics are used to highlight the core computer science aspects and applicability of distributed computing to meteorology. Presenting Cloud and Grid computing this paper shows use case scenarios fitting a wide range of meteorological applications from operational to research studies. The paper concludes that distributed computing complements and extends existing high performance computing concepts and allows for simple, powerful and cost effective access to computing capacity.

## 1 Introduction

Meteorology has an ever growing need for substantial amounts of computing power, be it for sophisticated numerical models of the atmosphere itself, modeling systems and workflows like e.g. coupled ocean and atmospheric models or the accompanying activities such as visualization or dissemination. In addition to the increased need for computing power, more data are being produced, transferred and stored, which increases the problem complexity. Consequently, concepts and methods to supply the compute power and data handling capacity have to evolve, too.

Until the beginning of this century high performance clusters, local consortia and/or buying cycles on commercial clusters were the main methods to acquire sufficient capacity. Starting in the mid 1990s, the concept of Grid computing, in which geographical and institutional boundaries only play a minor role, became a powerful tool for scientists. Foster and Kesselman (2003) published the first and most cited definition of the Grid: *A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.* In the following years the definition changed to viewing the Grid not as a computing paradigm, but as an infrastructure that brings together different resources in order to provide computing support for various applications, emphasizing the social aspect (Foster and Kesselman, 2004;

Bote-Lorenzo et al., 2004). Grid initiatives can be classified as *Compute Grids*, i.e. solely concentrated on raw computing power, or *Data Grids* concentrating on storage/exchange of data.

Many initiatives in the atmospheric sciences have utilized Compute Grids. One of the first climatological applications to use a Compute Grid is the Fast Ocean Atmospheric Model (FOAM) (Nefedova et al., 2006). They performed ensemble simulations of a coupled climate model on the Teragrid, a US based Grid project sponsored by the National Science Foundation. More recently, Fernández-Quiruelas et al. (2011) provided an example with the Community Atmospheric Model (CAM) for a climatological sensitivity study investigating the connection of sea surface temperature and precipitation in the El Nino area. Todorova et al. (2010) presents three Bulgarian projects investigating air pollution and climate change impacts. WRF4SG utilizes Grid computing with the Weather Research and Forecast Model WRF (Blanco et al., 2013) for various applications in weather forecasting and extreme weather case studies. TIGGE, the THORPEX Interactive Grand Global Ensemble, partly uses Grid computing to generate and share atmospheric data between various partner (Bougeault et al., 2010). The Earth system Grid ESGF is a US-European data Grid project concentrating on storage and dissemination of climate simulation data (Williams et al., 2009).

Cloud computing is a slightly newer concept than Grid computing. Resources are also pooled, but this time usually within one organisational unit, mostly within commercial companies. Similar to Grids, applications range from services based on demand to simply cutting ongoing costs or determining expected capacity needs.

The most important characteristics of Clouds are condensed into one of the most recent definitions by Mell and Grance (2011): *Cloud computing is a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* Further definitions can be found in Hamdaqa and Tahvildari (2012); Vaquero and Rodero-Merino (2008) or Vaquero and Rodero-Merino (2008). One of the few papers to apply Cloud

technology to meteorological research is Evangelinos and Hill (2008), who conducted a feasibility study for Cloud computing with a coupled atmosphere–ocean model.

In this paper, we discuss advantages and disadvantages of both infratructures for atmospheric research, show the supporting software ASKALON, and present three examples of meteorological applications, which we have developed for different kinds of distributed computing: projects *MeteoAG* and *MeteoAG2* for a Compute Grid, and *RainCloud* for Cloud computing. We look at issues and benefits mainly from our perspective as users of distributed computingPlease note we show our experiences but not a direct, quantitive comparison, as we did not have the resources to run experiments on both infrastructures with the exact same application..

## 2 Aspects of distributed computing in meteorology

### 2.1 Grid and Cloud computing

Our experiences in Grid computing come from projects *MeteoAG* and *MeteoAG2* within the national effort AustrianGrid (AGrid), including partners and supercomputer centers from all over Austria (Volkert, 2004). AGrid Phase 1 started in 2005 and concentrated on research of basic Grid technology and application. Phase 2, started in 2008, continued to build on research of Phase 1 and additionally tried to make AGrid self-sustaining. The research aim of this project was not to develop conventional parallel applications that can be executed on individual Grid machines, but rather to unleash the power of the Grid for single distributed program runs. To simplify this task, all Grid sites are required to run a similar Linux operating system. At the height of the project AGrid consisted of 9 clusters distributed over 5 locations in Austria including various smaller sites with ad-hoc Desktop PC networks. The progress of the project, its challenges and solutions were documented in several technical reports and other publications (Bosa and Schreiner, 2009).

For Cloud computing, plenty of providers offer services, e.g. Rackspace or Google Compute Engine. Our Cloud computing project *RainCloud* uses Amazon Web Services (AWS),

simply because it is the most well known and widely used. AWS offers different services for computing, different levels of data storage and data transfer, as well as tools for monitoring and planning. The services most interesting for meteorological computing purposes are Amazon Elastic Compute Cloud (EC2) for computing and Amazon Simple Storage Service (S3) for data storage. For computing, so called *instances* (i.e.virtual computers) are defined according to their compute power relative to a reference CPU, available memory, storage and network performance.

Figure 1 shows the basic structure of Cloud computing on the right side and AGrid as Grid example on the left side. In both cases an additional layer, so called Middleware, is applied between the compute resources and the end user. The Middleware layer handles all necessary scheduling, transfer of data and setup of Cloud nodes. Our Middleware is ASKALON (Ostermann et al., 2008), which is described in more detail in Sect. 2.2.

In the following sections, we list advantages and disadvantages of Grid and Cloud concepts, which affected our research most (see Table 1 for a brief overview). Criteria are extracted from literature, most notably Foster et al. (2008), which contains a general comparison with all vital issues, but also Mell and Grance (2011), Hamdaqa and Tahvildari (2012), and Foster and Kesselman (2003). The often discussed issue of security of sensitive and valuable data did not apply to our research and operational setting. However, for big and advanced operational weather forecasting this might be an issue due to its monetary value. Because the hardware and network is completely out of the end user's control, possible security breaches are harder or even impossible to detect. If security is a concern detailed discussions can be found in Cody et al. (2008) for Grid computing, and Catteddu (2010) and Feng et al. (2011) for Cloud computing.

### 2.1.1 Advantages/disadvantages Grid

+ *Handle massive amounts of data*. The full atmospheric model in MeteoAG generated large amounts of data. Through Grid tools like *gridftp* Allcock et al. (2002) we were able to efficiently transfer and store all simulation data.

+ *Access to HPC which suits parallel applications (e.g. Message Passing Interface; MPI).* The model used in MeteoAG, as many other meteorological models, is a massive parallel application parallelized with MPI. On single systems they run efficiently, however across different HPC clusters latencies become too high. A middleware can leverage the advantage of access to multiple machines and run applications on suitable machines and appropriately distribute parts of workflows in parallel.

− *Different hardware architectures.* During tests in MeteoAG we discovered problems due to different hardware architectures, which can be substantial (Schüller et al., 2007). We tested different systems with exactly the same setup and software and got consistently different results. In our case this affected our complex full model, but not our simple model. The exact cause is unclear, but most likely a combination of programming, the used libraries and setup down to the hardware level.

− *Difficult to setup and maintain as well as inflexible handling.* For us, the process of getting necessary updates, patches or special libraries needed in meteorology onto all Grid sites was complex and lengthy or sometimes even impossible due to operating system limitations.

− *Special compilation of source code.* To get the most out of the available resources, the executables in MeteoAG needed to be compiled for each architecture, with possible side effects. Even in a tightly managed project like AGrid, we had to supply three different executables for the meteorological model, however with changes only during compilation, not in the model code itself.

Other typical characteristics are not as important for us. The *limited amount of resources* never influenced us as they were always vast enough to not hinder our models. The *need to bring your own hardware/connections* is also a small hindrance since this is usually negotiable or the Grid project might have different levels of partnership.

### 2.1.2 Advantages/disadvantages Cloud computing

+ *Cost*. Costs can easily be determined and planned. More about costs can be found in Sect. 4.

+ *Full control of software environment, including operating system (OS) with root access*. This proved to be one of the biggest advantages for our workflows. It is easy to install needed software, special libraries or modify any component of the system. Cloud providers usually offer most standard operating systems as *images/AMI (Amazon Machine Image)*, but tuned images can also be saved permanently and made publicly available (with additional storage costs).

+ *Simple on-demand self-service.* For applications with varying requirements for compute resources or with repeated but short needs for compute power, this is an important characteristic. As long as funds are available the required amount of compute power can be purchased. Our workflow was never forced to wait for instances to be available. Usually our standard on-demand Linux instances were up and running within 5–10 s (Amazon's documentation states a maximum of 10 min).

− *Slow data transfer and hardly any support for MPI computing.* Data transfer to and from Cloud instances is slow as well as a higher network latency between the instances. Only a subset of instance types are suitable for MPI computing. This limitation makes Cloud computing unsuitable for large-scale complex atmospheric models.

*Missing information about underlying hardware* has no impact on our workflow, as we are not trying to optimize a single model execution. *No common standard between Clouds* and the possibility of *a Cloud provider going out of business* is unimportant for us, too. Our software relies on common protocols like ssh and adaptation to a new Cloud provider could be done easily by adjusting the script requesting the instances.

## 2.2 Middleware ASKALON

To make it as simple as possible for a (meteorological) end user to use distributed computing resources, we make use of a so called Middleware system. ASKALON, an existing Middleware from the Distributed and Parallel Systems group in Innsbruck, provides integrated environments to support the development and execution of scientific workflows on dynamic Grid and Cloud environments (Ostermann et al., 2008).

To account for the heterogeneity and the loosely coupled nature of resources from Grid and Cloud providers, ASKALON has adopted a workflow paradigm (Taylor et al., 2007) based on loosely coupled coordination of atomic activities. Distributed applications are split in reasonably small execution parts, which can be executed in parallel on distributed systems, allowing the runtime system to optimize resources usage, file transfers, load balancing, reliability, scalability and handle failed parts. To overcome problems resulting from unexpected job crashes and network interruptions, ASKALON is able to handle most of the common failures. Jobs and file transfers are resubmitted on failure and jobs might also be rescheduled to a different resource if transfers or jobs failed more then 5 times on a resource (Plankensteiner et al. (2009a)). These features still exist in the Cloud version but play a less important role as resources showed to be more reliable in the Cloud case.

Figure 1 shows the design of the ASKALON system. Workflows can be generated in a scientist-friendly Graphical User Interface (GUI) and submitted for execution by a service. This allows long lasting workflows without the need for the user to be online throughout the whole execution period.

Three main components handle the execution of the workflow:

- *Scheduler.* Activities are mapped to physical (or virtualized) resources for their execution with the end user deciding which pool of resources are used. A wide set of scheduling algorithms is available e.g. HEFT (Zhao and Sakellariou, 2003) or DCP-C (Ostermann and Prodan, 2012). HEFT for example takes as input tasks, a set of resources, the times to execute each task on each resource, and the times to communicate results between each job on each pair of resources. Each task is assigned

a priority and then distributed onto the resources accordingly. For the best possible scheduling, a training phase is needed to get a function that relates the problem size to the processing time. Advanced techniques in prediction and machine learning are used to achieve this goal (Nadeem and Fahringer, 2009; Nadeem et al., 2007).

– *Resource Manager.* Cloud resources are known to *scale by credit card* and theoretically an infinite amount of resources is available. The resource manager has the task to provision the right amount of resources at the right moment to allow the execute engine to run the workflow as the scheduler decided. Cost constraints must be strictly adhered to as budgets are in practice limited. More on costs can be found in Sect. 4.

– *Execute Engine.* Submission of jobs and transfer of data to the compute resources is done with a suitable protocol, e.g. ssh or GRAM in a Globus environment.

– *System Reliability.* An important feature which is distributed over several components of ASKALON is the capability to handle faults in distributed systems. Resources or network connections might fail any time and mechanisms as described in Plankensteiner et al. (2009a) are integrated in the execution engine Qin et al. (2007) allowing workflows to finish even when parts of the system fail.

## 3 Applications in meteorology

In the following subsections, we detail the three applications we developed for usage with distributed computing. All projects investigate orographic precipitation over complex terrain. The most important distributed computing characteristics of the projects are shown in Table 3.

### 3.1 MeteoAG

MeteoAG started as part of the AGrid computing initiative. Using ASKALON we created a workflow to run a full numerical atmospheric model and visualization on a Grid infrastruc-

ture (Schüller, 2008; Schüller et al., 2007; Schüller and Qin, 2006). The model is the non hydrostatic Regional Atmospheric Modeling System (RAMS; version 6), a fully MPI parallelized Fortran based code (Cotton et al., 2003). The NCAR Graphics library is used for visualisation. Due to all AGrid sites running a similar Linux OS, no special code adaptations to Grid computing were needed.

We simulated real cases as well as idealised test cases in the AGrid environment. Most often these were parameter studies testing sensitivities to certain input parameters with many slightly different runs. The investigated area in the realistic simulations covered Europe and a target area over western Austria. Several nested domains are used with a horizontal resolution of the innermost domain of 500 m and 60 vertical levels (approx. 7.5 million grid points). Figure 2 shows the workflow deployed to the AGrid. Starting with many simulations with a shorter simulation time, it was then decided which runs to extend further. Only runs where heavy precipitation occurs above a certain threshold were chosen. Post-processing done on the Compute Grid includes extraction of variables and preliminary visualization, but the main visualization was done on a local machine.

The workflow characteristics relevant for distributed computing are: few (20-50) model instances but highly CPU intensive as well as lots of interprocess communications. Results of this workflow require a substantial amount of data transfer between the different Grid sites and the end-user (O(200Gb)).

Upon investigation of our first runs it was necessary to provide different executables for specific architectures (32bit, 64bit, 64bit Intel) to get optimum speed. We ran into a problem while executing the full model on different architectures. Using the exact same static executable with the same input parameters and setup led to consistently different results across different clusters (Schüller et al., 2007). For real case simulations, these errors are negligible compared to errors in the model itself. But for idealized simulations e.g. investigation of turbulence with an atmosphere initially at rest, where tiny perturbations play a major role, this might lead to serious problems. We were not able to determine the cause of these differences. It seems to be a problem of the complex code of the full model and its interaction with the underlying libraries. While we can only speculate on the exact cause, we

strongly advise to use a simple and quick test such as simulating an atmosphere at rest or linear orographic precipitation to test for such differences.

### 3.2 MeteoAG2

MeteoAG2 is the continuation of MeteoAG and also part of AGrid (Plankensteiner et al., 2009b). Based on the experience from the MeteoAG experiments, we hypothesize that it would be much more effective to deploy an application consisting of serial CPU jobs. ASKALON is optimized for submission of single core parts of a workflow, which avoids internal parallelism and communication of activities and allows best control over the execution within ASKALON.Thus MeteoAG2 uses a simpler meteorological model, the Linear Model of orographic precipitation (LM) (Smith and Barstad, 2004). The model computes only very simple linear equations of orographic precipitation, is not parallelized, and has short runtime, O(10s), even with high resolutions (500 m) over large domains. LM is written in Fortran. ASKALON is again used for workflow execution and Matlab routines for visualisation.

With this workflow, rainfall over the Alps was investigated by taking input from European Centre for Medium-Range Weather Forecasts (ECMWF) model, splitting the Alps into subdomains (see Fig. 3a) and running the model within each subdomain with variations in the input parameters. The last step combines the results from all subdomains and visualises them. Using Grid computing allowed us to run many O(50 000) simulations in a relatively short amount of time O(h). This compares to about 50 typical, albeit a lot more complex runs in current operational meteorological setups.

The workflow deployed to the Grid (Fig. 3b) is simple with only two main activities: preparing all the input parameters for all subdomains and then the parallel execution of all runs. One of the drawbacks of MeteoAG2 is the very strict setup that was necessary due to the state of ASKALON at that time, e.g. no robust if-construct yet, and the direct use of model executables without wrappers. The workflow could not easily be changed to suit different research needs, e.g. change to different input parameters for LM or to using a different model.

### 3.3 RainCloud

Switching to Cloud computing, RainCloud uses an extended version of the same simple model of orographic precipitation as MeteoAG2. The main extension to LM is the ability to simulate different layers, while still retaining its fast execution time (Barstad and Schüller, 2011). The software stack includes ASKALON again, the Fortran-based LM, python scripts and Matplotlib for visualisation.

The inclusion of if-constructs in ASKALON and a different approach to the scripting of activities (e.g. wrapping the model executables in python scripts and calling these) allows RainCloud to be used in different setups. We are now able to run the workflow in 3 flavours without any changes: idealised, semi-idealised and realistic simulations as well as different settings: operational and research. Figure 4b depicts the workflow run on Cloud computing. Only the first two activities, *PrepareLM* and *LinearModel* have to be run, the others are optional. This workflow fits a lot of meteorological applications as it has the building blocks:

- preparation of the simulations (*PrepareLM*)
- execution of a meteorological model (*LinearModel*)
- post processing of each individual run, e.g. for producing derived variables (*PostProcessSingle*)
- post processing of all runs (*PostprocessFinal*)

All activities are wrapped in Python scripts. As long as the input and output between these activities are named the same, everything within the activity can be changed. We use archives for transfer between the activities, again allowing different files to be packed into these archives.

The operational setup produces spatially detailed, daily probabilistic precipitation forecasts for the Avalanche Service Tyrol (Lawinenwarndienst Tirol) to help forecast avalanche danger. Figure 4a shows the schematic of our operational workflow. Starting with data from

ECMWF, we forecast and visualize precipitation probabilities over Tyrol with a spatial resolution of 500 m. Additionally, research type experiments are used to test, explore and run experiments with new developments in LM through parameter studies.

Our workflow invocations vary substantially in required computation power as well as data size. The operational job is run daily during winter, whereas research types are run in bursts. Data usage *within* the Cloud can be substantial O(500Gb) with all flavours, but with big differences of data transfer from the Cloud back to the local machine. Operational results are small, in the order of O(100Mb), while research results can amount to O(100Gb), influencing the overall runtime and costs due to the additional data transfer time.

## 4 Costs, performance and usage scenarios

### 4.1 Costs

To define the exact costs for a **dedicated server** system or the participation in a **Grid** initiative is not trivial, and often even unknown to the provider. We contacted several of them, but due to complicated budgeting methodologies the final costs are not obvious. Greenberg and Hamilton (2008) discuss costs for operating a server environment for data services from a provider perspective. Costs discussed there include servers, infrastructure, power requirements and networking. However, the authors did not include the cost of human resources for e.g. system administration. Patel and Shah (2005) include human resources and establish a cost model for setup and maintenance of a data center. Grids may have different and negotiable levels of access and participation, with varying associated costs to the user. Some initiatives, e.g. PRACE (Guest et al., 2012), offer free access to Grid resources after a proposal/review process.

**Cloud** computing on the other hand offers simpler and transparent costs. Pricing varies depending on the provider, capability of a resource and also on the geographical region. Prices (as of November 2014) of AWS *on-demand* compute instances for Linux OS can be found in Table 2 and range from 0.014 up to $\sim 5\,\mathrm{USD\,h^{-1}}$ (region Ireland). Cheaper

instance pricing is available through *spot* instances where one bids on spare resources. These resources might get cancelled if demand rises, but are a valid option for interruption-tolerant workflows or for developing a workflow.

Figure 5 shows the difference between spot and on-demand pricing for 25 test runs of our operational workflow (circle and x, right y-axis). All runs use 32 cores but a different number of instances, i.e. only one c3.8xlarge (32 cores) instance, but 32 m1.medium (1 core) instances. Runtime only includes the actual workflow, not the spin up needed to prepare the instances. It usually takes 5–10 s for an instance to become available and another 2–5 min to setup the system and install necessary libraries and software. Spot and on-demand only differ in the pricing scheme not in the computational resources themselves. With spot pricing we achieved savings between 65–90 %, however with an additional startup latency of 2–3 min (compared to 5–10 s).

To give an idea, a very simplified cost comparison can be done with the purchasing costs of dedicated hardware, excluding costs for system administration, cooling or power. The operational part of RainCloud runs on 32 cores for approximately 3 h per day for six months of the year, i.e. 550h per year.

- A dedicated 32 core server with 64GB RAM costs around 5500 USD (various brands, excluding tax, Austria, November 2014).
- A comparable on-demand AWS instance (c3.x8large; 32 cores, 60 GB RAM) could run for $\sim 2800$ h at 1.91 USD $h^{-1}$ pricing.

Assuming no instance price variance, our operational workflow could be run on AWS for approximately five years, the usual depreciation time for hardware. This suggests AWS being the cheaper alternative for RainCloud, since hardware is only one part of the total cost of ownership of a dedicated system.

### 4.2 Performance

For our operational RainCloud workflow, Figure 5 shows the effect of different instance types on the runtime. First, a clear difference between the instance types is evident, with

the longest running taking nearly twice as long as the shortest one. Second, even within one instance type, runtime varies by 10-20 percent. Serial execution on a one core desktop PC takes about 12 h, i.e. a speedup of $\sim 18$ ( compared to 1 $\sim 0.66$ h as seen in Fig. 5). Based on these experiments our daily operational workflow uses four m3.2xlarge instances.

To put this into relation, Schüller et al. (2007) show a speedup for MeteoAG of multiple cores vs. 1 core for a short running test setup of $\sim 5$, with higher speedups possible for a full complex workflow run. For MeteoAG2, Plankensteiner et al. (2009b) show a speedup of $\sim 120$ when executing that workflow on several grid machines compared to the execution on a single desktop PC. However, as these are different workflows, no comparison between the type of computing resources can be made from these performance measures.

### 4.3 Usage scenarios

Different usage scenarios are commonly found in meteorology. For choosing the right type of computing system, several issues need to be taken into account. Only above a certain workflow scale is it worth the effort to move away from a local machine. Grids usually have a steep learning curve, Clouds offer simple (web) interfaces and local clusters are somewhere in the middle. To make the most out of Cloud computing (and to some extent out of Grid computing), it is best to have a workflow which can be split into small, independent components.

In an *operational scenario with frequent invocations*, either Clouds and Grids might be suitable depending on the amount of data transfered and the complexity of the model. Time critical data dissemination of forecast products can be sped up with (data) Grids. *Operational scenarios with infrequent invocations* might benefit from using Grid or even Cloud computing, avoiding the need for a local cluster. Examples are recalculation/reanalysis of seasonal/climate simulations or updating of model output statistics (MOS) equations. One important consideration for operational workflows is the scheduling latency, i.e. the time between submitting a job and its actual execution. Berger et al. (2009) and Lingrand and Montagnat (2009) show median latencies of 100s for EGEE Grid, but with frequent outliers upwards to 30 minutes and more (RainCloud 10s-120s).

For a *research scenario with bursts of high activity with many small tasks*, Cloud computing fits perfectly. The costs are fully controllable and only little setup is required. Examples of such use cases include parameter studies with simple models or computation of MOS. If a lot of data transfer is needed, Grid computing is the better alternative. *Research applications with big, long running, data intensive simulations* such as high resolution complex models are best run on Grids or local clusters.

## 5 Conclusions

We successfully deployed meteorological applications on distributed computing infrastructure of both Grids and Clouds. Our meteorological applications range from a complex atmospheric limited-area model to a simplified model of orographic precipitation. Adhering to some limitations/considerations, distributed computing can cater to both.

A consideration to be taken into account for both concepts is security. With Grids, it is relatively easy to determine users and potential access to data as all resources and locations are known. With Clouds, this is nearly impossible/impractical to do this and potential breaches are hard to detect.

If the Grid is seen as an agglomeration of individual supercomputers, complex parallelized models are simple to deploy and efficient to use in a research setting. The compute power is usually substantially larger than what a single institution could afford. However, in an operational setting the immediate availability of resources might not be a given. This is an issue that needs to be addressed in advance. For data storage and transfer, e.g. dissemination of forecasts, Grids are a powerful tool.

Taking Grid as a structure, workflows involving MPI are not simple to exploit. As with Clouds, it is much more effective to deploy an application consisting of serial jobs with as little interprocess communication as possible.

Heterogeneity of the underlying hardware cannot be ignored for Grid computing as quality tests showed (Schüller et al., 2007). Differences arising solely based on the used hardware

might influence very sensitive applications. However this is application-specific and needs to be tested for each setup.

The setup and access to Cloud infrastructure is a lot simpler and involves less effort than participation in a Grid project. Grids require hardware and more complex software to access whereas access to Clouds is usually kept as simple as possible.

(Commercial) Cloud computing is very effective and cost saving tool for certain meteorological applications. Individual projects with high-burst needs or an operational setting with a simple model are two examples. Elasticity, i.e. access to a larger scale of resources, is one of the biggest advantages of Clouds. Undetermined or volatile needs can be easily catered for. One option is to use Clouds to baseline workflow requirements and then build and move to a correctly sized in-house cluster/setup based on this prototyping.

Disadvantages of Clouds include above mentioned security issues, but one of the biggest problems for meteorological applications is data transfer. Transfer to and from the Cloud and within the Cloud infrastructure is considerably slower than for a dedicated cluster setup or Grids. Recently new instance types for massively parallel computing have been emerging (e.g. Amazon) but high computation applications with only modest data needs are best suited for most Clouds.

Private Clouds remove some of the disadvantages of public Clouds, security and data transfer are the most notable ones. However, using private Clouds also removes the advantage of not needing hardware and system administration. We used a small private Cloud to develop our workflow before going full-scale on Amazon AWS with our operational setup.

In a meteorological research setting with specialised software, Clouds offer a flexible system with full control over operating system, installed software and libraries. Grids on the other hand are managed on individual Grid sites and are more strict and less flexible. The same is true for customer service. Clouds offer one contact for all problems and offer (paid) premium support as opposed to having to contact each system administration for every Grid site.

In conclusion, both concepts are an alternative or a supplement to self-hosted high performance computing infrastructure. We have laid out guidelines with which to decide whether one's own application is suitable to either or both alternatives.

*Acknowledgements.* This research is supported by Austrian Grid, funded by the bm:bwk (Federal Ministry for Education, Science and Culture) (BMBWK GZ 4003/2-VI/4c/2004) (MeteoAG), (GZ BMWF 10.220/002-II/10/2007) (MeteoAG2), and Standortagentur Tirol: RainCloud.

## References

Allcock, B., Bester, J., Bresnahan, J., Chervenak, A. L., Foster, I. T., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., and Tuecke, S.: Data management and transfer in high-performance computational Grid environments, Parallel Comput., 28, 749–771, 2002.

Barstad, I. and Schüller, F.: An extension of Smith's linear theory of orographic precipitation: introduction of vertical layers, J. Atmos. Sci., 68, 2695–2709, 2011.

Berger, M., Zangerl, T., and Fahringer, T.: Analysis of overhead and waiting time in the EGEE production Grid, in: Proceedings of the Cracow Grid Workshop, 2008, 287–294, 2009.

Berriman, G. B., Deelman, E., Juve, G., Rynge, M., and Vöckler, J.-S.: The application of cloud computing to scientific workflows: a study of cost and performance, Philos. T. R. Soc. A., 371, doi:10.1098/rsta.2012.0066, 2013.

Blanco, C., Cofino, A. S., and Fernandez-Quiruelas, V.: WRF4SG: a scientific gateway for climate experiment workflows, Geophys. Res. Abstr., EGU2013-11535, EGU General Assembly 2013, Vienna, Austria, 2013.

Bosa, K. and Schreiner, W.: A supercomputing API for the Grid, in: Proceedings of 3rd Austrian Grid Symposium 2009, edited by: Volkert, J., Fahringer, T., Kranzlmuller, D., Kobler, R., and Schreiner, W., Austrian Grid, Austrian Computer Society (OCG), 38–52, available at: http://www.austriangrid.at/index.php?id=symposium, 2009.

Bote-Lorenzo, M. L., Dimitriadis, Y. A., and Sanchez, E. G. A.: Grid Characteristics and Uses: A Grid Definition, Vol. 2970, Springer, Berlin, Heidelberg, 2004.

Bougeault, P., Toth, Z., Bishop, C., Brown, B., Burridge, D., Chen, D. H., Ebert, B., Fuentes, M., Hamill, T. M., Mylne, K., Nicolau, J., Paccagnella, T., Park, Y.-Y., Parsons, D., Raoult, B., Schuster, D., Dias, P. S., Swinbank, R., Takeuchi, Y., Tennant, W., Wilson, L., and Worley, S.: The THORPEX interactive grand global ensemble, B. Am. Meteorol. Soc., 91, 1059–1072, 2010.

Catteddu, D.: Cloud computing: benefits, risks and recommendations for information security, in: Web Application Security SE – 9, edited by: Serrão, C., Aguilera Díaz, V., and Cerullo, F., Vol. 72 of Communications in Computer and Information Science, Springer, Berlin, Heidelberg, p. 17, 2010.

Cody, E., Sharman, R., Rao, R. H., and Upadhyaya, S.: Security in grid computing: a review and synthesis, Decis. Support Syst., 44, 749–764, 2008.

Cotton, W. R., Pielke Sr., R. A., Walko, R. L., Liston, G. E., Tremback, C. J., Jiang, H., McAnelly, R. L., Harrington, J. Y., Nicholls, M. E., Carrio, G. G., and McFadden, J. P.: RAMS 2001: Current status and future directions, Meteorol. Atmos. Phys., 82, 5–29, 2003.

Deelman, E., Singh, G., Livny, M., Berriman, B., and Good, J.: The cost of doing science on the cloud: the montage example, in: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, p. 50, 2008.

Evangelinos, C. and Hill, C.: Cloud computing for parallel scientific HPC applications: feasibility of running coupled atmosphere-ocean climate models on Amazon's EC2, Ratio, 2, 2–34, 2008.

Feng, D.-G., Zhang, M., Zhang, Y., and Xu, Z.: Study on cloud computing security, Journal of Software, 22, 71–83, 2011.

Fernández-Quiruelas, V., Fernández, J., Cofiño, A., Fita, L., and Gutiérrez, J.: Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model, Environ. Modell. Softw., 26, 1057–1069, 2011.

Foster, I. and Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

Foster, I., Zhao, Y., Raicu, I., and Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared, 2008 Grid Computing Environments Workshop, 1–10, 2008.

Foster, I. T. and Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, 2nd Edn., Morgan Kaufmann, Amsterdam, 2004.

Greenberg, A. and Hamilton, J.: The cost of a cloud: research problems in data center networks, ACM SIGCOMM Computer Communication Review, 39, 68–73, 2008.

Guest, M., Aloisio, G., and Kenway, R.: The scientific case for HPC in Europe 2012–2020, tech. report, PRACE, 2012.

Hamdaqa, M. and Tahvildari, L.: Cloud computing uncovered: a research landscape, Adv. Comput., 86, 41–85, 2012.

Lingrand, D. and Montagnat, J.: Analyzing the EGEE production grid workload: application to jobs submission optimization, Lect. Notes Comput. Sc., 5798, 37–58, 2009.

Mell, P. and Grance, T.: The NIST definition of cloud computing recommendations of the National Institute of Standards and Technology, Special Publication 800–145, NIST, Gaithersburg, available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (last access: 9 February 2015), 2011.

Nadeem, F. and Fahringer, T.: Predicting the execution time of grid workflow applications through local learning, in: High Performance Computing Networking, Proceedings of the Conference on Storage and Analysis, 1, 1–12, doi:10.1145/1654059.1654093, 2009.

Nadeem, F., Prodan, R., and Fahringer, T.: Optimizing Performance of Automatic Training Phase for Application Performance Prediction in the Grid, in: High Performance Computing and Communications, Third International Conference, HPCC 2007, Houston, USA, 26–28 September 2007, 309–321, 2007.

Nefedova, V., Jacob, R., Foster, I., Liu, Z., Liu, Y., Deelman, E., Mehta, G., Su, M.-H., and Vahi, K.: Automating climate science: large ensemble simulations on the TeraGrid with the GriPhyN virtual data system, e-science, 0, 32, doi:10.1109/E-SCIENCE.2006.261116, 2006.

Ostermann, S. and Prodan, R.: Impact of variable priced cloud resources on scientific workflow scheduling, in: Euro-Par 2012 Parallel Processing, edited by: Kaklamanis, C., Papatheodorou, T., and Spirakis, P., Vol. 7484 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 350–362, 2012.

Ostermann, S., Plankensteiner, K., Prodan, R., Fahringer, T., and Iosup, A.: Workflow monitoring and analysis tool for ASKALON, in: Grid and Services Evolution, Barcelona, Spain, 73–86, 2008.

Patel, C. D. and Shah, A. J.: Cost Model for Planning, Development and Operation of a Data Center, Technical Report HP Laboratories Palo Alto, HPL-2005-107(R.1), 9 June, 2005.

Plankensteiner, K., Prodan, R., and Fahringer, T.: A new fault tolerance heuristic for scientific workflows in highly distributed environments based on resubmission impact, in: eScience'09, 313–320, 2009a.

Plankensteiner, K., Vergeiner, J., Prodan, R., Mayr, G., and Fahringer, T.: Porting LinMod to predict precipitation in the Alps using ASKALON on the Austrian Grid, in: 3rd Austrian Grid Symposium, edited by: Volkert, J., Fahringer, T., Kranzlmüller, D., Kobler, R., and Schreiner, W., Vol. 269, Austrian Computer Society, 103–114, 2009b.

Qin, J., Wieczorek, M., Plankensteiner, K., and Fahringer, T.: Towards a light-weight workflow engine in the ASKALON Grid environment, in: Proceedings of the CoreGRID Symposium, Springer-Verlag, Rennes, France, 2007.

Schüller, F.: Grid Computing in Meteorology: Grid Computing with – and Standard Test Cases for – A Meteorological Limited Area Model, VDM, Saarbrücken, Germany, 2008.

Schüller, F. and Qin, J.: Towards a workflow model for meteorologcial simulations on the Austrian Grid, Austrian Computer Society, 210, 179–190, 2006.

Schüller, F., Qin, J., Nadeem, F., Prodan, R., Fahringer, T., and Mayr, G.: Performance, Scalability and Quality of the Meteorological Grid Workflow MeteoAG, Austrian Computer Society, 221, 155–165, 2007.

Smith, R. B. and Barstad, I.: A linear theory of orographic precipitation, J. Atmos. Sci., 61, 1377–1391, 2004.

Taylor, I. J., Deelman, E., Gannon, D., and Shields, M.: Workflows for e-Science, Springer-Verlag, London Limited, 2007.

Todorova, A., Syrakov, D., Gadjhev, G., Georgiev, G., Ganev, K. G., Prodanova, M., Miloshev, N., Spiridonov, V., Bogatchev, A., and Slavov, K.: Grid computing for atmospheric composition studies in Bulgaria, Earth Science Informatics, 3, 259–282, 2010.

Vaquero, L. and Rodero-Merino, L.: A break in the clouds: towards a cloud definition, ACM SIGCOMM Computer Communication Review, 39, 50–55, 2008.

Volkert, J.: The Austrian Grid Initiative – high level extensions to Grid middleware, in: PVM/MPI, edited by: Kranzlmüller, D., Kacsuk, P., and Dongarra, J. J., Vol. 3241 of Lecture Notes in Computer Science, Springer, p. 5, 2004.

Williams, D. N., Drach, R., Ananthakrishnan, R., Foster, I. T., Fraser, D., Siebenlist, F., Bernholdt, D. E., Chen, M., Schwidder, J., Bharathi, S., Chervenak, a. L., Schuler, R., Su, M., Brown, D., Cinquini, L., Fox, P., Garcia, J., Middleton, D. E., Strand, W. G., Wilhelmi, N., Hankin, S., Schweitzer, R., Jones, P., Shoshani, A., and Sim, A.: The Earth System Grid: enabling access to multimodel climate simulation data, B. Am. Meteorol. Soc., 90, 195–205, 2009.

Zhao, H. and Sakellariou, R.: An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm, in: Euro-Par Conference, 189–194, 2003.

**Table 1.** Overview of advantages/disadvantages of Grids and Clouds affecting our applications most. For a detailed discussion see section 2.

| Grids | Clouds |
|---|---|
| + Massive amounts of data<br>+ Access to parallel computing enabled HPC | + Cost<br>+ Full control of software setup<br>+ Simple on-demand access |
| - Inhomogeneous hardware architectures<br>- Complicated setup and inflexible handling<br>- Special compilation of source code needed | - Slow data transfer<br>- Not suitable for MPI computing |

**Table 2.** Prices and specifications for Amazon EC2 on demand instances mentioned in this paper, running Linux OS in region *EU-west* as of November 2014. m1.xlarge, m1.medium and m2.4xlarge are previous generations which were used in our experiments. Storage is included in the instance, additional storage is available for purchase. One Elastic Compute Unit (ECU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

| Instance Family | Instance Type | vCPU | ECU | Memory (GiB) | Storage (GB) | Cost USD $h^{-1}$ |
|---|---|---|---|---|---|---|
| General purpose | m3.medium | 1 | 3 | 3.75 | 1 x 4 SSD | 0.077 |
| | m3.2xlarge | 8 | 26 | 30 | 2 x 80 SSD | 0.616 |
| Compute optimized | c3.8xlarge | 32 | 108 | 60 | 2 x 320 SSD | 1.912 |
| Storage optimized | hs1.8xlarge | 16 | 35 | 117 | 24 x 2048 | 4.900 |
| Micro instances | t1.micro | 1 | Var | 0.615 | EBS only | 0.014 |
| General purpose | m1.xlarge | 4 | 8 | 15 | $4 \times 420$ | 0.520 |
| | m1.medium | 1 | 2 | 3.75 | $1 \times 410$ | 0.130 |
| Memory optimized | m2.4xlarge | 8 | 26 | 68.4 | $2 \times 840$ | 1.840 |

**Table 3.** Overview of our projects and their workflow characteristics.

| Project | MeteoAG | MeteoAG2 | RainCloud |
|---|---|---|---|
| Type | Grid | Grid | Cloud |
| Meteorological model | RAMS (Regional Atmospheric Modeling System) | single layer linear model of orographic precipitation | double layer linear model of orographic precipitation |
| Model type | complex full numerical model parallelized with Message Passing Interface (MPI) | simplified model | double layer simplified model |
| Parallel runs | 20–50 | approx 50 000 | $>$ 5000 operational, $>$ 10 000 research |
| Runtime | several days | several hours | 1–2 h operational/ $<$1 h research |
| Data transfer | O(200GB) | O(1GB) | O(MB) – O(1GB) |
| Workflow flexibility | strict | strict | flexible |
| Applications | parameter studies, case studies | downscaling | parameter studies, downscaling, probabilistic forecasts, model testing |
| Intent | research | research | operational, research |
| Frequency | on demand | on demand | operational: daily, research: on demand |
| Programming | shell scripts, Fortran, NCAR Graphics, MPI | shell scripts, Fortran, Matlab | python, Fortran |

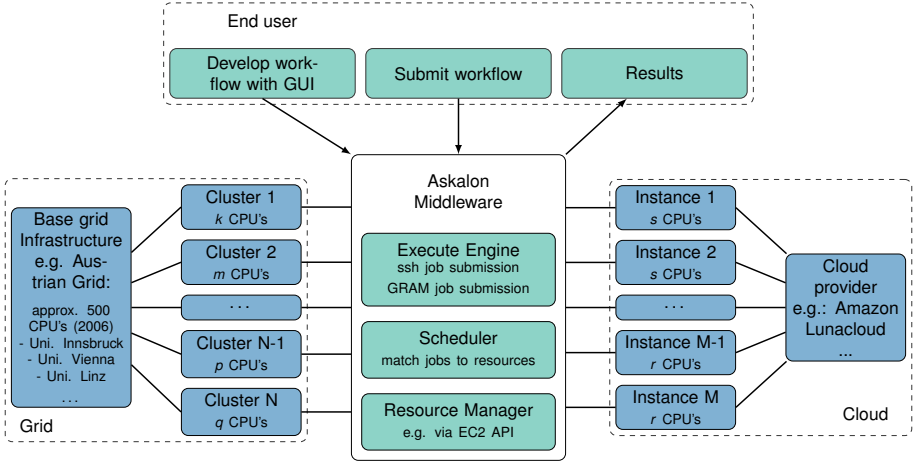



**Figure 1.** Schematic setup of our computing environment for Grid (left) and Cloud (right) computing. End users interact with the ASKALON Middleware via a Graphical User Interface (GUI). The number of CPUs per cluster provided by the base Grid varies, whereas the instance types of Cloud providers can be chosen. Execute Engine, Scheduler and Resource Manager interact to effectively use the available resources and react to changes in the provided computing infrastructure.

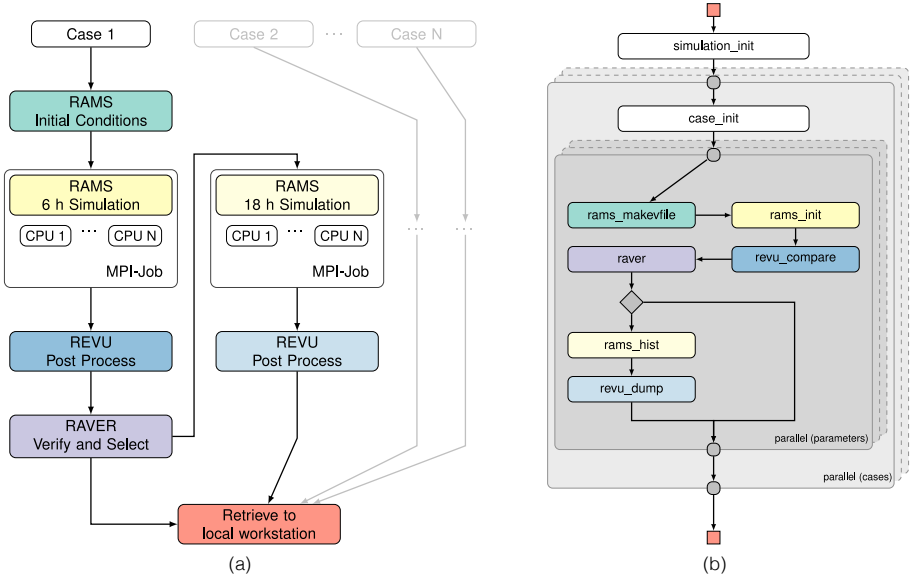
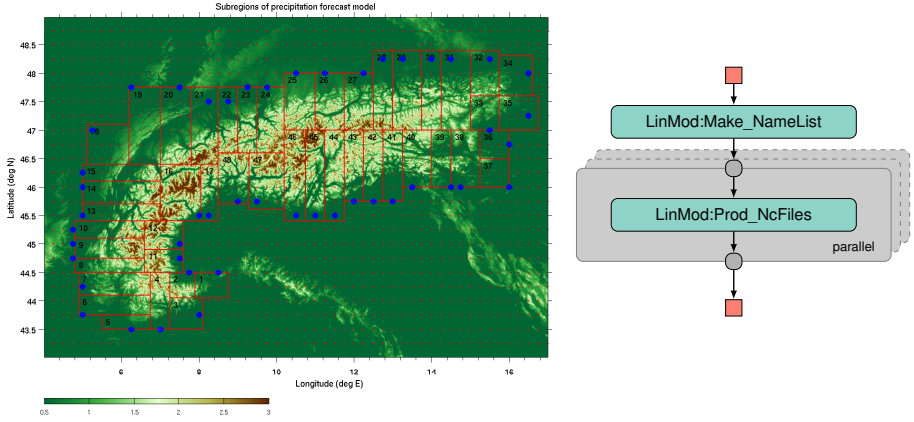



**Figure 2.** Workflow of MeteoAG using the Regional Atmospheric Modelling System (RAMS) and supporting software REVU (extracts variables) and RAVER (analyses variables). Each case represents a different weather event. **(a)** Meteorological representation with indication which activities are parallelized using Message Passing Interface (MPI). **(b)** Workflow representation of the activities as used by ASKALON Middleware. In addition to the different cases, selected variables are varied within each case. Same colors between the subfigures.




(a) (b)

**Figure 3.** Setup and workflow of MeteoAG2 using the Linear Model (LM) of orographic precipitation. **(a)** Grid setup of experiments in MeteoAG2 with dots representing grid points of the European Center of Medium Range Weather Forecast (ECMWF) used to drive the LM. Topography height in km a.m.s.l. **(b)** Workflow representation of the activities as used by ASKALON. Activity MakeNML prepares all input sequentially. ProdNCfile is the main activity with the linear model run in parallel on the Grid. Figure **(a)** courtesy of Plankensteiner et al. (2009b).

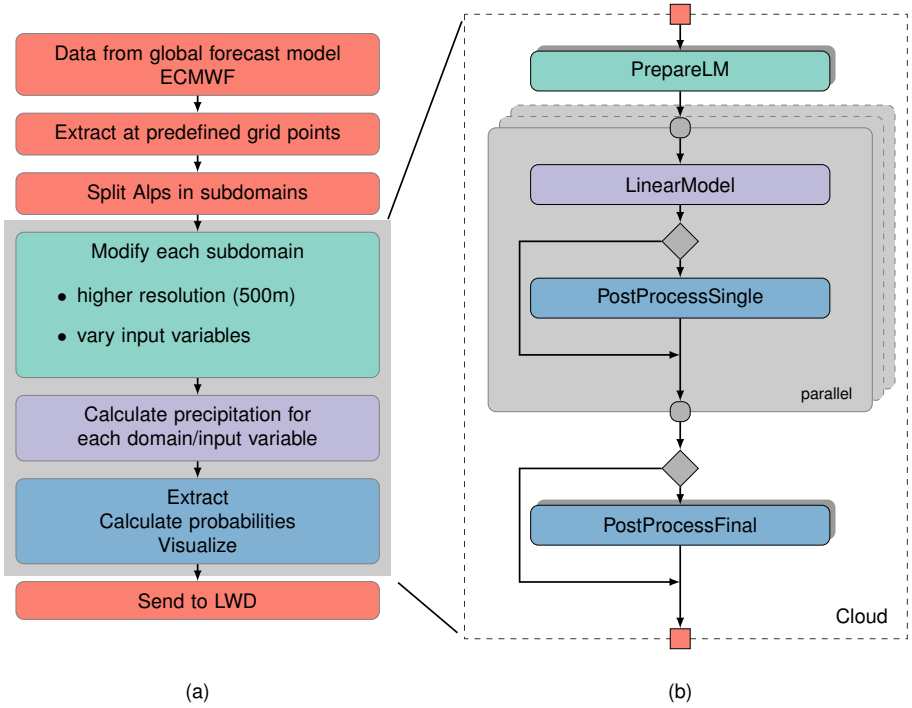



**Figure 4.** Workflow of RainClouds operational setting for the Avalanche Warning Service Tyrol (LWD) using the double layer Linear Model (LM) of orographic precipitation. Input data from the European Center for Medium Range Weather Forecast (ECMWF). **(a)** Meteorological flow chart with parts not executed on Cloud (in red). **(b)** Workflow with activities as used by ASKALON. Same colors between subfigures.

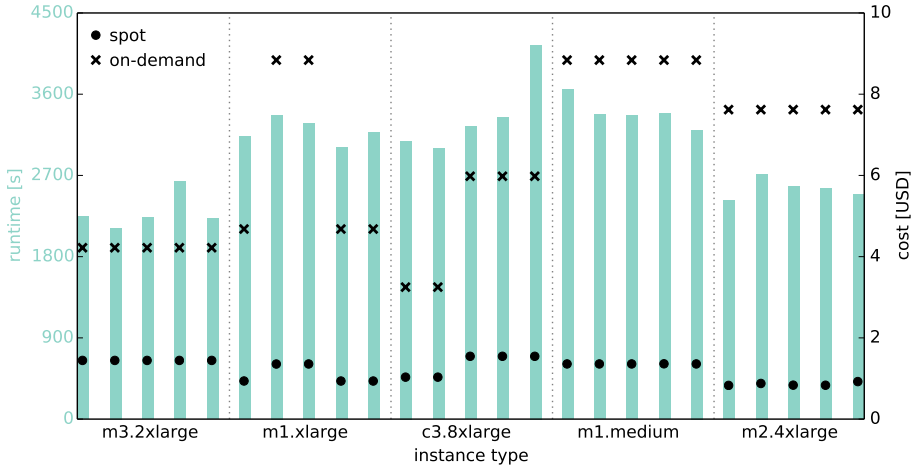



**Figure 5.** Bars show overall runtime of one operational run on various Amazon EC2 instance types, each with a total of 32 cores (left $y$ axis). Each bar represents one workflow invocation with the corresponding instance type. Dots show costs for on-demand instances (x) and spot instances (circle; right $y$ axis). Only the execution part is shown, spin up time i.e. preparation and installation (2–5 min) is not included. See Table 3 for exact specifications. All experiments were run during March 2014 with the exact same setup.