Parallelizing Geospatial Tasks in Grid Computing

Dipl.-Ing. Stefan Werder
Research Assistant
Leibniz Universität Hannover
Institute of Cartography and Geoinformatics
Appelstraße 9a
30167 Hannover
Germany
E: stefan.werder@ikg.uni-hannover.de

Dipl.-Inf. Andreas Krüger
Research Assistant
Technische Universität Berlin
Department for Geodesy and Geoinformation Science
Straße des 17. Juni 135
10623 Berlin
Germany
E: krueger@igg.tu-berlin.de

Keywords
Grid Computing, Parallelization, Data Quality

Abstract
The available number and amount of geospatial data are increasing. So is the complexity of performed calculations on these datasets. This leads to the need for efficient parallelization of geospatial tasks. Another aspect that plays an important role in connection with parallelization is distributed computing. In this paper six (research) questions are presented, that have to be answered in the context of parallelizing geospatial tasks in a grid computing environment. However, they are not restricted to the grid architecture. The questions include aspects of whether to parallelize the data or the tasks, data quality issues, system architecture decisions, and workflow orchestration.

1 Motivation

The demands that both spatial data infrastructures and GIS have to meet are rapidly increasing. Three main factors can be identified as the sources for these needs. Firstly, the available number and amount of spatial data sets are rapidly increasing. This is due to the availability of more detailed data acquisition techniques, such as airborne laser scanning, detailed models in 3D and also 4D, affordable geo-enabled sensors and such devices as mobile phones equipped with GPS and inertial navigation systems (INS), as well as data acquisition by the masses (crowdsourcing). Secondly, the complexity of applied operations or calculations on these data sets is increasing due to more complex models (e.g. optimization problems in generalization). The third factor is the consideration of security aspects, which are not only used for controlling who can access data to which extent and under which restrictions, but also as a basis for accounting and billing.

Demands increasing and so are the available infrastructures for data storage and computation in terms of performance and flexibility. One indicator for the increasing performance is the biannually released list of the top 500 supercomputers in the world, which shows an exponential growth (Meuer 2008). Another indicator is the dynamic development of the fields of grid computing and cloud computing. Grid computing has been discussed in the scientific community for almost a decade and is well defined by Foster's (2002) three-point checklist. In contrast, cloud computing is relatively new and not strictly defined (Vaquero et al. 2009). However, the two fields share the same vision of making high-performance computing available for a larger user base.

The different approaches for distributed computing have the need for efficient parallelization in common, which can be based on both data and computations. This paper focuses on the topic of parallelization of geospatial tasks in a grid computing environment, which raises two questions. The first one is how to divide a given task into several subtasks, which then can be executed concurrently. In most cases the solution to this problem depends only on the algorithms and processes of the task itself, and is therefore not specific or limited to grid computing. The second question is how to make use of the grid in order to accelerate processing, in which aspects such as available grid middleware and orchestration of web services have to be considered.

The remainder of this paper is structured as follows. In the next section the Spatial Data Infrastructure Grid (GDI-Grid), which is the project the authors are working in, is presented shortly. Thereafter six research questions that have to be answered in the context of parallelization in a grid computing environment are listed and explained in more detail in the consequent sections.

2 GDI-Grid

In Germany the D-Grid initiative builds the foundation for a sustainable development of grid technology and eScience methods. The project is funded by the German Federal Ministry of Education and Research (BMBF) with 100 Million Euro and runs from 2005 to 2010. Part of D-Grid is the Spatial Data Infrastructure Grid (GDI-Grid), which aims at the efficient mining and processing of spatial data for simulation of noise dispersion and disaster management. The focus is on processing services, which are able to handle massive amount of geospatial data and also compute complex operations based on three real-world scenarios. The processing services are thereby both traditional Web Processing Services (WPS) defined by the Open Geospatial Consortium (OGC) and grid services according to the Web Service Resource Framework (WSRF) defined by the Open Grid Forum (OGF). One of the mentioned scenarios uses noise propagation simulations for the assessment and management of environmental noise. Based on a directive by the European Union (2002) all EU member states are obliged to inform the public every five years about environmental noise emitted by major transport routes (road, railway, air), and from sites of industrial activity. Therefore noise maps have to be produced, which also serve as a basis for further acoustic planning and may have significant economical impacts.

3 Overview of research questions

Having introduced the relevant background as well as some basic concepts and terminology, we want to present six (research) questions that, from our perspective, have to be answered in the context of parallelization of geospatial tasks in a grid computing environment. The questions as well as the used terminology are more precisely outlined in the following sections.

1. Which existing geospatial tasks (potentially) benefit from parallelization?
2. Which of the two concepts of task or data parallelism yields the best results for a given problem, or is it a combination of both?
3. How does parallelization affect the quality of the task result?
4. Where do the actual parallel tasks run in the overall system architecture – on different cores of a CPU, in a GPU, in a cluster, or on different worker nodes in a grid environment? And how are the data distributed in that architecture?
5. How to efficiently orchestrate several parallelized tasks in a complex workflow?
6. How complex and time-intensive is the adoption of existing code to gain advantage from parallelization?

The research questions two and six address the question how to divide a given task into several subtasks, whereas the research questions four and five address the question how to make use of the grid in order to accelerate processing.

3.1 Geospatial tasks benefiting from parallelization

The question which existing geospatial tasks benefit from parallelization includes two aspects. Firstly, a list of geospatial operators, tasks and complex processes which benefit from parallelization, e.g. by reducing runtime or increasing stability, should be created. Based on that list it would be easier for a GIS developer to decide which parts of a given program should be rewritten or exchanged by already parallelized code fractions or libraries.

Secondly, it also includes an in-depth performance analysis based on reproducible metrics. These metrics are not specific to geospatial tasks, but are suitable for comparing different approaches for parallelizing the same geospatial task. For analyzing performance three measures can be estimated. Speedup measures the factor of increased speed between parallel and sequential processed calculation. Efficiency measures the utilization of the available processors. Scalability measures the ability to increase performance while increasing the number of processors. For a quantification of speedup, efficiency, and scalability, several estimation metrics can be used. These are for instance Amdahl's Law, Gustafson-Barsis' Law, the Karp-Flatt Metric, or the Isoefficiency Metric (Quinn 2003).

3.2 Task and data parallelism

Two high-level concepts exist for parallelizing a task (Culler et al. 1999). Firstly, a task can be divided into several independent subtasks operating on the same or different dataset (task parallelism). Secondly, it can be divided into several subtasks that each processes a part of the whole dataset (data parallelism). Data parallelism corresponds to tiling or partitioning of geospatial data. If the individual subtasks require no communication between each other in data parallelism, the problem is also called embarrassingly parallel or nicely parallel (Foster 1995).

For data parallelism explicit processing steps for both tiling and subsequent merging of the dataset are necessary. A generic approach for tiling is sufficient for our applications, and we suppose this applies also to other applications. The parameter set that satisfies our requirements are the width and height of a single tile, the coordinates of one corner of a start tile, the tile border size, and rules for objects located near or on tile borders (fig. 1). The tile border leads to the inclusion of additional data beyond the actual tile size.

For simulations of noise propagation within the GDI-Grid the tile border size is 3km for a tile size of 1km² leading to a total of 49km². This overhead is necessary, because all noise emitters in these areas contribute to the total noise level in the centre tile. Tiling rules specify how objects near or on tile borders are distributed between adjacent tiles, for instance they can be put into the tile where their centre point lies in, cloned for all tiles in which their bounding box is in, or simply being cut at the tile borders.

In contrast to the generic approach for a tiling service, the merging process incorporates application specific logic for most of our applications. The logic has to take the inter-dependencies at the tile borders into account in order to obtain a continuous result.

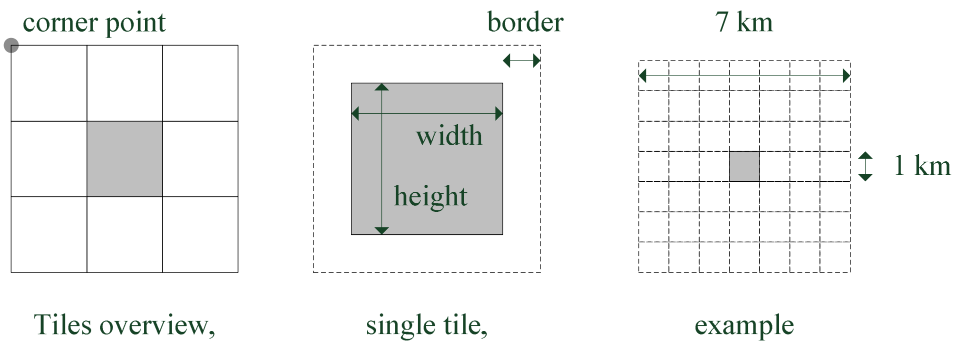

Figure 1: Parameters for a tiling service

Parallelization can be achieved using different criteria – besides others by object type (e.g. buildings, streets), by geometry type (e.g. point, line, area), by operation type (e.g. calculate length, calculate angles), by spatial criteria (e.g. tiling), and by Level of Detail. Therein the geometry type is often tied to the object type or subtype. The following example shows a parallelization for checking the integrity of geospatial datasets (fig. 2). In the example buildings have to fulfil the constraints that their border consists of only orthogonal lines and that each building's area is greater than 100m². For a detailed description of the constraints as well as their formalization in the context of the GDI-Grid see Werder (2009). The choice of the optimal parallelization criteria as well as their chaining can have significant influence on the performance. For checking data integrity, for instance the parallelization by object type is performed first, because integrity constraints differ in most cases for different object types.

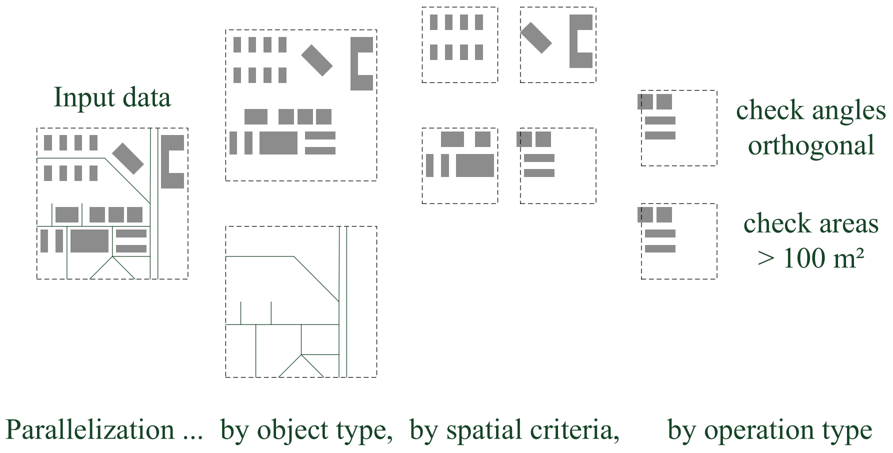

Figure 2: Parallelization of massive geodata for checking data integrity

The following example is concerned with map generalization (Müller 2008). There the concept of data parallelism is used in order to speed up the processing. Maps with land use information, which are traditionally acquired from satellite imagery, are derived from existing datasets of a national mapping agency offering more details as well as higher resolution. The overall task includes several processing steps, such as re-classification, adjustment of geometries, and aggregation. As a result, the runtime of the program is evaluated in relation to several tile sizes (fig. 3).

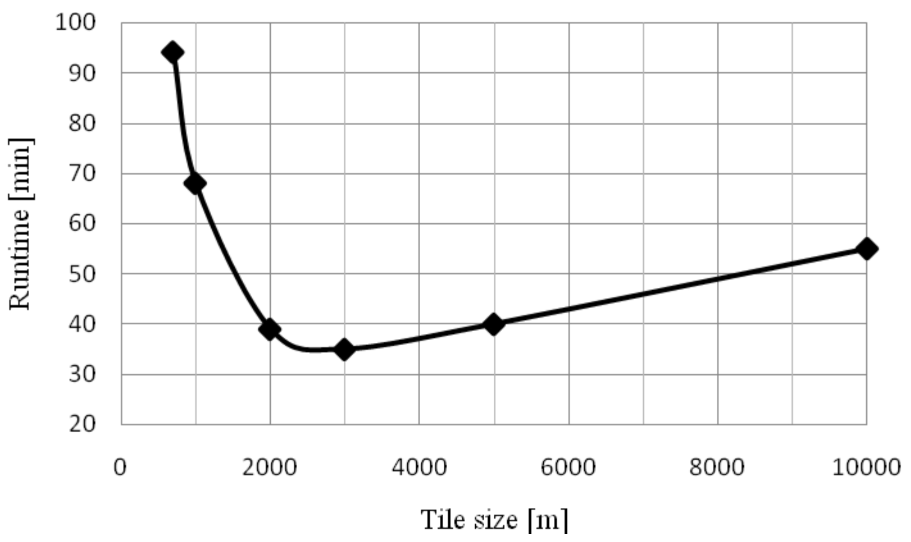

Figure 3: Runtime [min] in relation to tile size [m] (based on Müller (2008))

The runtime reaches the optimum for a tile size with a width and height of approximately 3000 m. For smaller tile sizes the data input and output operations slow down the overall task, while for larger tile sizes the algorithm needs to compare more objects with each other. This leads to the conclusion that general recommendations concerning individual parallelization aspects, such as optimum tile size, are not reasonable, because they depend on the underlying algorithms, data, and system architecture.

3.3 Parallelization and quality

Parallelization can affect the quality of obtained results in a positive or negative way. An increase in quality can be obtained, for instance, by not only processing the standard model, but also calculating alternative models in parallel. After processing, the best model can be chosen or the results of several models can be combined. The quality of the result can also decrease, however, due to parallelization (Müller 2008). The result of the map generalization is compared to a reference dataset by matching identical features. The match rate has a range from 58.4% for the smallest tile size to 59.6% for the biggest tile size, with 59.3% for the tile size of 3000 m. The partitioning leads to a small quality loss, because the algorithm has no information for objects beyond the partition borders, which itself influences the quality of objects near the borders. Therefore the smallest tile size results in a quality loss of 1.2%, whereas the optimum tile size results only in a not significant quality loss of 0.3%.

Based on these considerations the (research) question is not only formulated in terms of whether the quality is affected but how it is affected. This incorporates analyzing whether the used parallelization affects the quality globally, that is the whole dataset, or whether the quality changes locally (e.g. near tile borders, for a specific type of object, etc).

3.4 Overall system architecture

Tasks can be executed in parallel within several system architectures (e.g. on different cores of CPUs, on graphics processing units (GPUs), in clusters, or in grid environments). Choosing the appropriate system architecture depends on many factors, such as already available infrastructure, amount of money that can be invested in new hardware or in their rental, desired speed, types of used algorithms, data size and distribution, software licensing issues, and security considerations. Concerning spatial data infrastructures (SDI), grid computing is a reasonable choice, because it complements the SDI. The distributed nature of the numerous and massive geospatial datasets in an SDI harmonizes with their distributed computing in a grid environment.

In order to specify the overall system architecture of a grid system more precisely, the distribution of both the parallelized (sub-) tasks and of the corresponding data as well as their combination in a workflow have to be defined. Several individual services are orchestrated in a workflow. The actual execution of a workflow is then performed by a workflow engine. In a grid environment three different approaches can be distinguished (Krüger, Kolbe 2008).

In the first approach the workflow engine is not aware of the grid (fig. 4), i.e. it does not know of the existence of the grid infrastructure. The services in the grid are therefore encapsulated by traditional web services. Subtasks are executed sequentially and the input and result datasets of the subtasks are always moved to a central location outside the grid. With this solution, calculations of single services can be distributed on several worker nodes, but no parallel data and information flow between them is realized. Worker nodes are individual computers on which jobs actually run. For communication and data transfer between two services no grid technologies can be used.

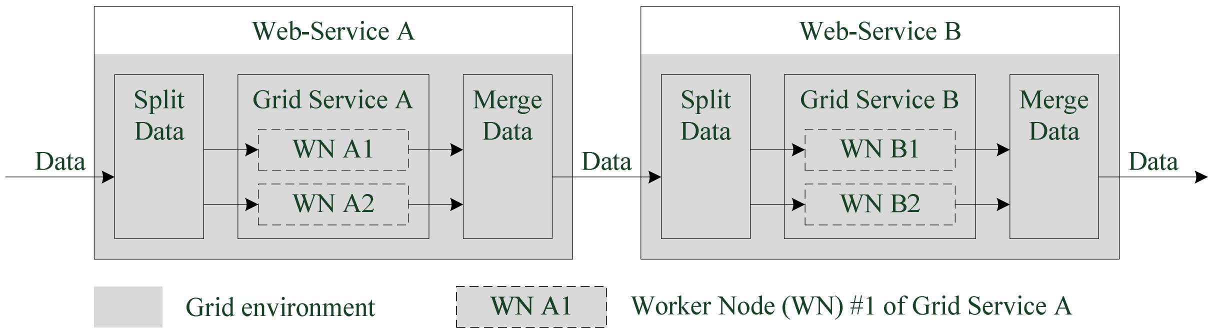

Figure 4: Workflow engine is not grid-aware

In the second approach the workflow engine is aware of the grid, i.e. it knows of the existence of the grid infrastructure and is able to make use of it accordingly. The engine executes different subtasks sequentially, but it is able to create several parallel processes for the same subtask (fig. 5). This solution allows full use of grid technologies for communication and data transfer between two services. However, in the long term an efficient coupling of grid services should be achieved.

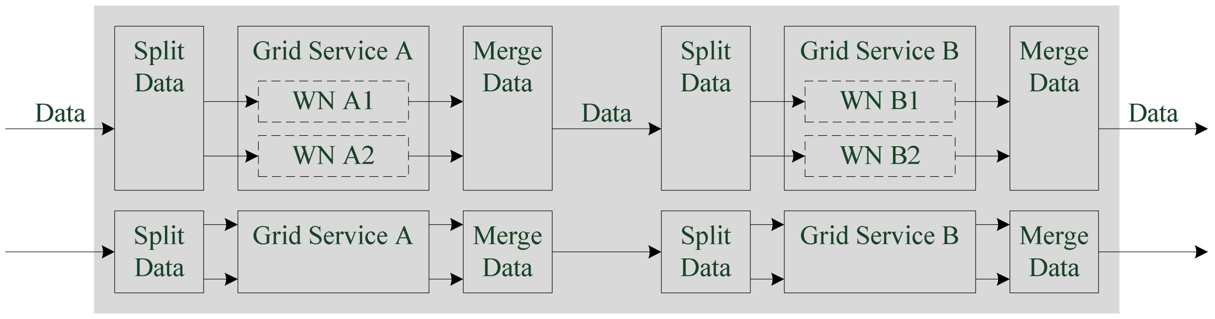

Figure 5: Workflow engine is grid-aware

The third approach allows for the efficient coupling of grid services. In order to achieve this, grid services must be enabled to directly communicate with each other, especially concerning their accepted type of parallelization. Fig. 6 shows the two grid services A and B that are compatible concerning their parallelization and are therefore able to directly exchange the data between the respective worker nodes (WN). This way a time-intensive merge and subsequent split of the different datasets of the worker nodes is not necessary. Grid technologies can be utilized for parallel communication and data transfer directly between active worker nodes of subtasks. The concept of this parallelization approach has been named “transcendent gridification” by Krüger & Kolbe (2008).

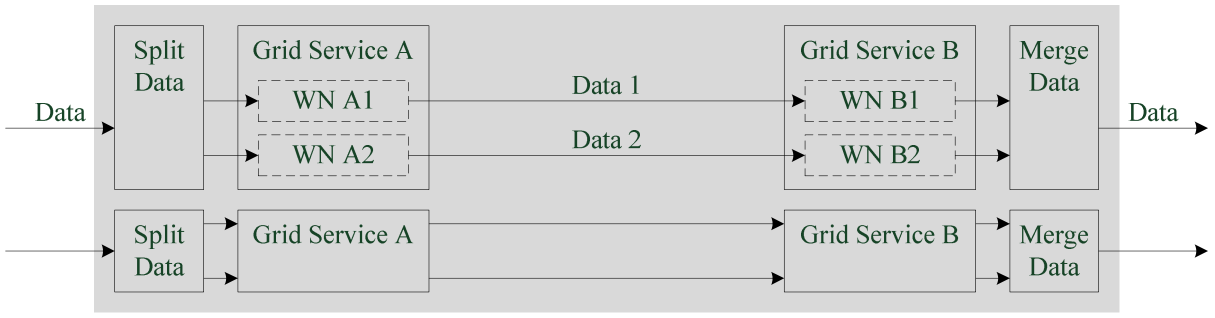

Figure 6: Transcendent gridification by compatible grid services

Another aspect for an optimized use of the grid environment is the spatial distribution of files and algorithms within a grid environment. Choosing appropriate worker nodes within a grid environment depends, for instance, on the required memory size, the CPU speed and load, and the provided operating system. Requirements on the hardware must be fulfilled for high-performance workflow processing. The transfer of very large datasets can produce significant overhead, which has already been highlighted for the example of noise simulation within the GDI-Grid. Thus, services that process algorithms on very large datasets should be deployed near databases that store the datasets, and vice versa. In this context, ”near” means not a real distance, but a connection providing preferably a high-speed data transfer. The actual selection of appropriate worker nodes for deploying a workflow within a grid environment is the work of a meta-scheduler. However, meta-schedulers are not incorporated in all grid environments and still require more research (Dong, Akl 2006).

3.5 Orchestration

The efficient orchestration of tasks in a complex workflow has two aspects. The first aspect covers the requirements that a workflow engine should satisfy in general. An important issue is thereby the support for basic control-flow patterns (e.g. split, merge, choice, loop), which are for instance defined by Russell et al. (2006).

The second aspect covers the technical issues that have to be solved in the context of grid computing for an SDI, which are highlighted in the remainder of this section. The most important aspect is that the types of (web) services used in a workflow differ between the GIS and the grid community. However, the respective standardization bodies, that are the Open Geospatial Consortium (OGC) for the GIS community and the Open Grid Forum (OGF) for the grid community, already collaborate in order to develop open standards that address the distributed computing needs of geospatial applications (Lee, Percivall 2008). The two approaches as well as the corresponding standards are presented in the following and an existing solution for the combination of both approaches is outlined.

In GIS the delivery and processing of spatial data are provided by special web services, which are standardized by the OGC. Such OGC compliant web services (OWS) are besides others:

- Web Processing Services (WPS) provide calculations on (very extensive) spatial data and GIS functionality, including access to calculations and/or computation models (OGC 2007a).
- Web Coverage Services (WCS) describe and deliver multidimensional coverage data (OGC 2008).
- Web Feature Services (WFS) provide data access functionality and operations on geographic features (OGC 2005).
- Catalogue Services (CS-W) handle the discovery and retrieval of spatial data, data stores and provide metadata of services (OGC 2007b).

In general, OWS are stateless and use proprietary security concepts. A web service description is stored in the corresponding capabilities document and is not (yet) realised by the Web Services Description Language (WSDL). The access to the services is performed over a “GetCapabilities” and a “describeX” operation, whereas “X” is a placeholder for the provided functionality. The service discovery is realised by CS-W. Messaging is realized by a combination of HTTP GET key value pairs using different data formats, like ASCII, binaries and OGC XML formats, e.g. the Geography Markup Language (GML) (Hobona et al. 2007).

Within a grid environment the Open Grid Services Architecture (OGSA) describes the architecture which is based on grid services. A grid service is a web service that provides a set of well-defined interfaces and that follows specific conventions. The interfaces address discovery, dynamic service creation, lifetime management, notification and manageability. Additionally grid services address authorisation and concurrency control (Foster et al. 2002). OGSA and SDI differ in some technologies. OGSA requires stateful web services and the service description is realised by WSDL. Most grid environments provide the Simple Object Access Protocol (SOAP) as messaging format. For service discovery the Monitoring and Discovery Service (MDS) and Universal Description Discovery and Integration (UDDI) are used. In contrast to the OWS, the security is based on the Grid Security Infrastructure (GSI), which uses asymmetric encryption per credentials (certificates which contains a public key) and a private key for decryption. (Foster, Kesselman 2004; Hobona et al. 2007).

In the GDI-Grid the approaches from the GIS and grid community are combined in a comprehensive workflow engine (Fleuren, Müller 2008). The engine is based on the Business Process Execution Language (BPEL). The aim of the GDI-Grid project is to be able to combine four different service types. These are standard web services described by WSDL, WSRF-based grid services, a special grid service that allows submitting and monitoring of simple executables as jobs in the grid, and traditional OWS which do not (yet) support WSDL. In order to grid-enable existing OWS providing computation-intensive functionality, they have to be ported to grid services. However, it is important that the workflow engine can also handle traditional OWS, for which a conversion is either impossible or not reasonable, because they are being provided by third parties or offer only simple functionalities.

3.6 Adoption of existing code

The last research question is a short one. The time investment for parallelizing an already existing task ranges from setting a compiler switch to a thorough code analysis and reconstruction process. In finance the measure of return on investment exists, which is calculated as the ratio of profit earned in relation to the amount of money invested (Feibel 2003). This concept can be transferred to parallelization either incorporating money or time.

4 Conclusions

Parallelization in a grid computing environment raises two questions: (1) how to parallelize tasks, which is independent from the grid, and (2) how to efficiently execute these tasks in the grid. We have identified six (research) questions that have to be answered by actual work in the context of parallelization in a grid computing environment. These questions have been explained in more detail by covering efficiency metrics, task and data parallelism, quality issues, system architectures, and workflows.

The question is not whether to parallelize, because we face massive geospatial data and more complex algorithms, but how to parallelize efficiently. But thereby we have to keep in mind that the absolute improvement is hard to measure exactly, because it depends on many factors.

5 Acknowledgements

The research described in this paper is part of the GDI-Grid project (Spatial Data Infrastructure Grid) funded by the German Federal Ministry of Education and Research (BMBF). The support is gratefully acknowledged.

6 References

Culler, D.E.; Singh, J.P.; Gupta, A. (1999): Parallel Computer Architecture - A Hardware/Software Approach. Morgan Kaufmann Publishers.

Dong, F.; Akl, S.G. (2006): Scheduling algorithms for grid computing: State of the art and open problems. http://research.cs.queensu.ca/TechReports/Reports/2006-504.pdf, accessed 08/09.

European Union (Ed.) (2002): Directive 2002/49/EC of the European Parliament and of the Council of 25 June 2002 relating to the assessment and management of environmental noise. Official Journal of the European Communities, L189.

Feibel, B.J. (2003): Investment Performance Measurement.Wiley.

Fleuren, T.; Müller, P. (2008): BPEL Workflows Combining Standard OGCWeb Services and Grid-enabled OGC Web Services. In: Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications, Sep 1-5, Parma, Italy.

Foster, I. (1995): Designing and Building Parallel Programs. Addison Wesley.

Foster, I. (2002): What is the Grid? A three point checklist. GridToday 1(6).

Foster, I.; Kesselman, C. (2004): The Grid 2: Blueprint for a new Computing Infrastructure. Elsevier.

Hobona, G.; Fairbairn, D.; James, P. (2007): Workflow Enactment of Grid-Enabled Geospatial Web Services. In: Proceedings of the 2007 UK e-Science All Hands Meeting, Sep 10-13, Nottingham, UK.

Krüger, A.; Kolbe, T.H. (2008): Mapping Spatial Data Infrastructures to a Grid Environment for Optimised Processing of Large Amounts of Data. In: Proceedings of the ISPRS Congress Beijing 2008, Jul 3-11, Beijing, China.

Lee, C.; Percivall, G. (2008): Standards-Based Computing Capabilities for Distributed Geospatial Applications. IEEE Computer 41(11).

Meuer, H.W. (2008): The TOP500 Project: Looking Back Over 15 Years of Supercomputing Experience. Informatik-Spektrum 31(3), Springer.

Müller, P.G. (2008): Ableitung von Corine Land Cover Daten aus dem ATKIS-Basis-DLM. Master Thesis, Institute of Cartography and Geoinformatics, Leibniz Universität Hannover.

OGC (2005): Web Feature Service Implementation Specification: Version 1.1.0. OGC Document 04-094.

OGC (2007a): OpenGIS Web Processing Service: Version 1.0.0. OGC Document 05-007r7.

OGC (2007b): OpenGIS Catalogue Services Specification: Version 2.0.2. OGC Document 07-006r1.

OGC (2008): Web Coverage Service (WCS) Implementation Specification: Version 1.1.2. OGC Document 07-067r5.

Quinn, M.J. (2003): Parallel Programming in C with MPI and OpenMP. McGraw-Hill Professional.

Russell, N.; ter Hofstede, A.H.M.; van der Aalst, W.M.P.; Mulyar, N. (2006): Workflow Control-Flow Patterns: A Revised View. http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf, accessed 08/09.

Vaquero, L.M.; Rodero-Merino, L.; Caceres, J.; Lindner, M. (2009) A Break in the Clouds: Towards a Cloud Definition. ACM SIGCOMM Computer Communication Review 39(1).

Werder, S. (2009): Formalization of Spatial Constraints. In: Proceedings of the 12th AGILE International Conference on Geographic Information Science, Jun 2-5, Hannover, Germany.