Datenbank Spektrum (2015) 175–184
DOI 10.1007/s13222-015-0197-y

SCHWERPUNKTBEITRAG

VAT: A System for Visualizing, Analyzing and Transforming Spatial Data in Science

Christian Authmann · Christian Beilschmidt · Johannes Drönner · Michael Mattig · Bernhard Seeger

Received: 10 June 2015 / Accepted: 5 September 2015 / Published online: 30 September 2015

Abstract The amount of available data changes the style of research in geo-scientific domains, and thus influences the requirements for spatial processing systems. To support data-driven research and exploratory workflows, we propose the **V**isualization, **A**nalysis & **T**ransformation system (VAT). We first identify ten fundamental requirements, which span from supporting spatial data types over low latency computations to visualization techniques. Based on these we evaluate state-of-the-art systems from the domains of spatial frameworks, GIS, workflow systems, scientific databases and Big Data solutions. The goal of the VAT system is to overcome the identified limitations by a holistic approach to raster and vector data, demand-driven and tiled processing, and the efficient usage of heterogeneous hardware architectures. A first comparison with other systems shows the validity of our approach.

1 Introduction

The importance of spatial data has grown significantly, especially in the fields of biodiversity and environmental sciences. The Millenium Ecosystem Assessment [9] of the United Nations testified a state of degradation for earth's ecosystems. This yields a dramatic loss of species and will ultimately impact human well-being. In order to mitigate this threatening trend, it is of utmost importance to capture and analyze the state of the environment to introduce effective counteractions. Satellites, surfaced-based sensors and human observations deliver a high amount of relevant information. Current approaches however have severe limitations with respect to the performance and usability in handling data sets of this magnitude and heterogeneity.

Our work is motivated by two ongoing research projects with different focus: GFBio and IDESSA. The German Federation for Biological Data (GFBio[1] [12]) strives to facilitate data sharing and reuse in the biological sciences. The goal is to establish a sustainable service oriented data infrastructure for German research projects. Key partners are national archives and data centers that provide strong expertise in data management and data curation. Our task is to provide added-value services for the data portal that is currently being built. This first of all means visualizing and aggregating the available data. In addition, we develop techniques that enable users to correlate selected data sets with public and private ones.

IDESSA[2] is a focused research project that aims to facilitate sustainable rangeland management in South African savannas. A variety of geo-spatial data is used to model savanna dynamics. Land degradation caused by woody plant encroachment, overgrazing and climate change needs to be detected and measured. Our task is to provide an integrative system for providing decision-support to local farmers. The system shall also assist local scientists in further research about savanna dynamics.

This is an extended version of the paper "Rethinking Spatial Processing in Data-Intensive Science" [3] selected for the special DASP issue *Best Workshop Papers of BTW 2015*.

M. Mattig (✉) · C. Beilschmidt · J. Drönner · C. Authmann · B. Seeger
Department of Mathematics and Computer Science
University of Marburg,
Marburg, Germany
e-mail: mattig@mathematik.uni-marburg.de

C. Authmann
e-mail: authmanc@Mathematik.Uni-Marburg.de

[1] www.gfbio.org
[2] www.idessa.org

The two projects exemplify that processing of spatial data meets the three key properties of Big Data [20]: Variety, Volume and Velocity.

Variety: Scientists have to deal with varying data formats, coordinate reference systems, resolutions and uncertainties [4]. There are raster images which arrange the earth's surface in a regular grid with each cell containing a measurement value. They represent continuous phenomena like elevation, temperature or precipitation. There are also point data sets that associate attributes to locations of interest, e.g. occurrences of a species or measurements of a sensor. In addition, there are line and polygonal data sets for modelling areas of interest like roads and the habitat of species. Unique challenges arise when performing joint analyses on these types of data.

Volume: A global raster with a cell size of 100 $\times$ 100 meters exceeds 100 gigabytes of data. Current satellite instruments like MODIS[3] send 70 gigabytes of data per day, with the next generation of satellites increasing this by an order of magnitude.

Velocity: In order to meet the expectations of interactive users, operations need to be processed in near real-time. This is especially important for exploratory analysis and refinement of results.

Traditional tools for the processing of spatial data confine scientists to a limited set of questions that can be answered. Thus, several approaches have been proposed to extend the expressiveness and efficiency of the tools by applying Big Data principles to spatial data. In contrast, we present our architecture of an integrated system that follows a holistic approach to vector and raster data. In particular, we employ tiled processing, low-resolution previews, and the efficient usage of modern hardware architectures. Our focus is the provision of high-performance building blocks while we will address scalability and distributed processing in our future work.

The rest of this paper is structured as follows. It first explains our notion of exploratory workflows and how they support scientists in their data-driven research. Section 3 describes the fundamental requirements we identified for an interactive exploratory research system. The next section addresses related technologies and describes their shortcomings with respect to these requirements. Section 5 introduces our own approach for overcoming current limitations. The evaluation in Section 6 shows the validity of our approach by experimentally comparing it to competing systems. It also demonstrates the opportunities of using heterogeneous hardware architectures.

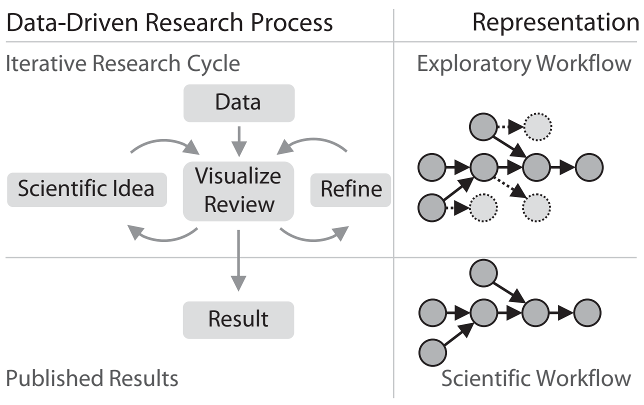

Fig. 1 Relationship between the steps of a data-driven research process and their representation as workflows

2 Exploratory Workflows

Scientific research involves multiple steps from an initial idea to the publication of results. Usually, scientists formulate a hypothesis first. Then, in order to test it, they gather appropriate data and afterwards publish the verified results. The Big Data era enables an alternative approach termed data-driven research [14] that is facilitated by shared data, shared infrastructures and new analysis tools. Scientists can search for ideas or hypotheses by reviewing available data and immediately test and verify them.

Fig. 1 illustrates an abstract data-driven research process cycle. By starting with available data, it is often necessary to involve multiple iterations where new data and intermediate results are visualized and reviewed. In many cases users are not fully satisfied with current results, though. Thus, they would like to refine the processing by changing parameters and introducing new data sets. It may even result in new scientific ideas, e.g. a new hypothesis, which users would like to examine. This approach also includes branching of research processes. It occurs when different ideas are pursued. Scientists may use different parameter settings for the same processing step. They also may incorporate new data sets to validate their hypotheses.

The right-hand side of Fig. 1 shows the corresponding workflows of a research task. The exploratory workflows represent the iterative research cycle of data-driven research processes, including all used data sets, branches and executed steps. It is possible to build an exploratory workflow in parallel to the interactive research process. If a user classifies the result of an exploratory workflow as scientifically meaningful, it is then possible to extract the relevant processing steps and create a scientific workflow.

[3]modis.gsfc.nasa.gov

Cuevas-Vicenttín et al. define a scientific workflow as "a description of a process for accomplishing a scientific objective" [10]. It is expressed as a directed graph where its nodes represent basic operators for data import, processing and data export. The directed links between the nodes refer to the data flow. Scientific workflows provide lineage information and facilitate reproducibility for other researchers.

In the context of data-driven geo-science the concept of exploratory workflows reflects the typical usage of a GIS-application or working with spatial data in general. Users start with available data layers and create new layers by applying operators to them. The results are visualized and reviewed, data or ideas are refined and new ideas are inspired starting a new iteration of the exploratory research cycle.

3 Requirements

Section 2 presented an overview of modern data-driven biodiversity research and our understanding of exploratory workflows. In the following, we identify ten criteria as mandatory for a workflow system that effectively supports such an exploratory approach.

R1: **Heterogeneous Data Support** A support for both raster images and vector data is obligatory. Additionally, temporal information is an integral part of nearly all data items. A system must be aware of the temporal validity to compute meaningful results.

R2: **Visualization** Providing interactive visualization of data is essential in facilitating exploratory research. A system should incorporate methods from visual analytics [2] that help users to understand their data and form new ideas for further processing.

R3: **Automatic Data Transformation** The format of input data for an operator can vary depending on the source. A system should provide data transformation techniques that allow unifying data. It should apply them automatically to take the responsibility from the user.

R4: **Low-Level Operators** Providing task-centric operators allows for performance optimizations but runs the risk of supporting only certain use cases. A general purpose system has to provide a rich set of efficient and well-defined low-level operators. Two examples are the intersection of points and polygons and the filtering of points by their attributes. Users can define specialized operators by composing multiple low-level operators.

R5: **Workflows** An exploratory research system should support workflows as first-class citizens. Workflows can again use other workflows as operators. They should have representations in a standardized format to allow exchange and reuse. The data input and data output of workflows should follow standardized formats. Systems should support a rich set of these formats.

R6: **Data Lineage** Apart from the technical aspects it is essential for science that each step of a process is denoted and the computation is comprehensible. Data lineage [7] is the composition of all processing steps from raw measurements to a scientific result. Publishing results along with the data lineage of a workflow allows other scientists to reproduce the results and verify them.

R7: **Low Latency Computation** Section 2 introduced the exploratory research approach. It greatly benefits from a rapid execution and adjustment of workflows. Consequently, it is mandatory to make the results of a computation available in near real-time. This requires efficient algorithms and the consideration of characteristics of modern computer hardware. A system must support flexible compositions and modifications of workflows with high throughput. It should allow parameter sweeps with different settings to facilitate the exploratory approach. This enables scientists to quickly explore different scenario settings.

R8: **Data Set Provision and Scalability** Growing data sizes make it infeasible for researchers to keep copies of all relevant data on their local workstations. A system should manage all data centrally, e.g. on a private cluster rather than globally distributed, and provide commonly used data sets at each point in time. The continuously increasing amount of data requires a scalable approach and a plan for timely data provision.

R9: **Custom Data** Researchers must be able to upload custom data. This includes also unpublished, temporary data for a branch of an existing workflow. Systems should ensure privacy and data security with regard to other users.

R10: **Rich User Interface** An intuitive and efficient web interface shall allow researchers to explore data and generate workflows. This avoids the expense of installations and maintenance on the user's local workstation except for a web browser.

4 Survey of Geo-Processing Systems and Related Work

There is a large variety of software for spatio-temporal processing. In this section we present a survey that aims to classify these software components and matches them with the requirements of Section 3. The systems span from low-level programming APIs to GIS with rich graphical user interfaces and also cover scientific databases, workflow systems and Big Data solutions.

The traditional way for creating scientific workflows is to use a programming language directly. This requires the user to have decent knowledge in computer science and lacks any graphical user interface (R10). Programming can either include calling command-line programs or using suitable APIs. Current scripting languages, e.g. R[4], already offer a lot of functionality to investigate and visualize spatial data (R1, R2, R4). For more sophisticated tooling, the user can include spatial data frameworks. Representatives are GDAL [24] for reading, writing as well as re-projecting spatial data and GEOS[5] for vector operations. These frameworks are well maintained and offer a lot of features. However, they do not exploit recent hardware developments sufficiently well (R7).

Geographic Information Systems (GIS) are on the other end of the spectrum. Commonly, they are desktop applications that combine spatial processing capabilities (R1, R4) with a rich user interface for importing, visualizing and processing data (R2, R10). Open source systems like QGIS[6] provide a wide range of low and high level operators. Most systems also support users in unifying data between outputs and inputs of consecutive processing steps (R3). Some GIS implementations offer basic lineage information (R6) as well as rudimentary workflows (R5). However, this is not sufficient for exploratory data analysis. Users can either work exploratory or use a model builder to create a workflow. An effective combination of both approaches is not sufficiently well supported. Another downside is the installation on the user's local machine that limits the available hardware resources (R8). Furthermore, most current GIS implementations cannot efficiently handle data that exceeds main memory.

A recent trend is the development of cloud-based GIS solutions. The scalability helps overcoming the limitations of local hardware (R8). A lot of applications are highly specialized for one particular use case and focus e.g. more on data visualization than on processing (R2, R4). A good example is Map of Life [16] which utilizes the Google Earth Engine[7]. It offers multiple tools, each of them is well fitted to a certain use case, e.g. the consensus range map for species[8]. Drawbacks are the lack of generalization and the difficulty to extend these systems for other use cases.

Scientific database systems for spatio-temporal data have a focus on data storage and processing rather than on visualization (R2, R4). On the one hand, there are specialized systems that can cope with large raster data sets. Prominent representatives are RasDaMan [5] and SciDB [8]. They provide scalability (R8) but lack support for vector data (R1). On the other hand there are general purpose systems. PostGIS[9] adds support for geographic data to the open source system PostgreSQL (R1). Being based on a classical RDBMS manifests in a lack of performance required for spatial data analysis (R7).

Popular tools in the domain of biodiversity informatics are scientific workflow systems [10, 23]. They allow specifing a data flow model from input data sets to an intended result set (R5). Most systems provide a graphical user interface to create such workflows via drag and drop (R10). Taverna[10], an Apache Incubator project, is one example. It supports the user in orchestrating web services and manages the data transmission. These workflows are re-usable e.g. as sub-workflows in a larger workflow. There are platforms like the BiodiversityCatalogue[11] that facilitate sharing of workflows with the community. Thus, a lot of functionality is already available. Another well-known scientific workflow system is Kepler[12], which DataONE[13] recommends. It is similar to Taverna in orchestrating workflows, but additionally puts more emphasis on executing local scripts and programs. In both systems, users have only little control over the availability and stability of remote web services, impeding reproducibility of computations (R6). Web services are mainly available as a black box, preventing independent verification of the data processing steps.

The aspect of data lineage and reproducibility is among other things tackled by Pegasus [21]. It is a distributed workflow tool suite including a compiler that has a sophisticated provenance model and error management. This means it keeps track of each operator, parameter and data location (R6). Pegasus maps abstract workflows expressed in XML to specialized workflows on nodes in a distributed system. It also supports optimizations of the data flow. HUBZero [21] is a simulation and modelling infrastructure. It incorporates virtual processing environments for software components and supports Pegasus workflows. HUBZero enables parameter sweeps over multiple Pegasus workflows.

A general drawback of web services besides availability is the performance. Calling a web service incurs overhead for data serialization and network traffic. This is especially noticeable when working on larger data sets. Web service orchestration does not solve scalability issues as a single overloaded service can delay the whole workflow (R8). There is no flexible mechanism to distribute the overloaded service to other machines. This causes a slow execution of larger workloads and impedes the exploratory usage of such systems (R7).

Big Data systems are currently ubiquitous. Many of them are built on Hadoop [19] and its distributed file system. Hadoop-GIS [1] extends Hadoop with GIS features. This

[4]www.r-project.org
[5]geos.osgeo.org
[6]www.qgis.org
[7]earthengine.google.org
[8]species.mol.org
[9]www.postgis.net

[10]www.taverna.org.uk
[11]www.biodiversitycatalogue.org
[12]www.kepler-project.org
[13]www.dataone.org

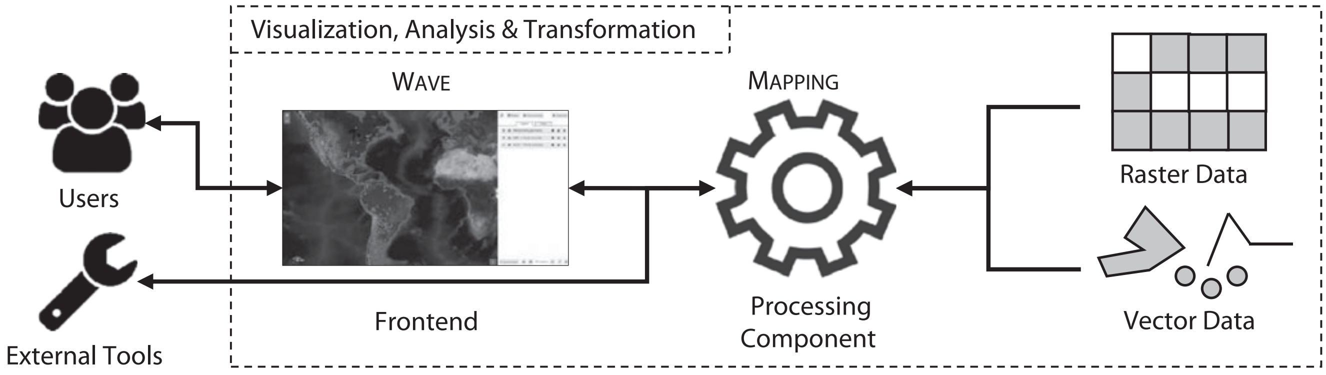

Fig. 2 Overview of the architecture of the VAT system

approach however suffers from frequent disk usage as each intermediate result has to be written to disk. Apache Spark [25] eliminates this disadvantage by using a more sophisticated process communication model. So-called Resilient Distributed Datasets (RDD) make it possible to cache intermediate results in memory without incurring huge I/O costs. GeoTrellis for Spark [17] has started developing an RDD for spatial data sets. It will offer a rich set of basic operators which makes it powerful for geographic processing (R4). A drawback for both approaches is the long start-up time of distributed systems [11] that hinders the desired low latency computation of exploratory workflows (R7).

Concluding our survey we did not find a single system that matches all of our requirements. A combination of the key features of the above mentioned approaches within a single system would be beneficial for enabling data-driven science in spatial applications.

5 The VAT System

Fig. 2 illustrates the architecture of our **V**isualization, **A**nalysis & **T**ransformation system (VAT). It consists of two main parts. The back end, called **M**arburg's **A**nalysis, **P**rocessing and **P**rovenance of **I**nformation for **N**etworked **G**eographics (MAPPING), manages spatio-temporal data of different formats and from different sources. Its main task is to execute workflows on the managed data. The front end, called **W**orkflow, **A**nalysis and **V**isualization **E**ditor (WAVE), is a web application for visualization and manipulation of data, similar to a web-based GIS application. WAVE allows the interactive creation of exploratory workflows as explained in Section 2. Users can execute workflows either using WAVE, or by an external application via standard web requests.

In the following, we introduce our core concepts and design decisions as well as the current state of the system after the first year of development. Subsection 5.1 discusses the support of spatio-temporal raster and vector data. We also give a brief summary of operators for processing spatio-temporal data. Subsection 5.2 presents first ideas of our exploratory workflow approach and how these workflows can be used in an interoperable manner. Subsection 5.3 introduces our first approaches to processing and optimizing queries. The implementation of operators on GPUs is discussed in Subsection 5.4. In Subsection 5.5, we sum up the current state of the VAT system with respect to the list of the requirements presented in Section 3.

5.1 Support of Spatio-Temporal Data

The VAT system is designed to support processing of spatio-temporal data. Fig. 3 shows the basic data types of our system.

Rasters are stored in a read-efficient way using a lightweight custom engine. To improve query performance, rasters are always tiled and stored with pyramids of different resolutions.

For vector data, we follow the semantics of the Simple Feature Access OGC[14] standard [22]. Each feature consists of one or more spatial objects and associated attributes. The supported spatial objects are points, lines and polygons. Our data types are collections of homogeneous features with matching spatial types and attributes, e.g. a political map where each feature represents a country. Here, one or more polygons per country model the state territory, while associated attributes can store information like name and population count. Splitting the *SimpleFeatureCollection* into three subclasses based on the type of the geometries allows us to retain clear semantics and to define the in- and output specification of an operator. Operators are able to load vector data from multiple sources. GFBio's biodiversity data is currently stored using PostGIS. The sensor data from IDESSA's weather stations is

[14]www.opengeospatial.org

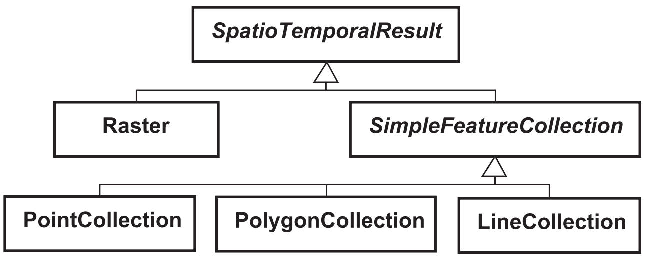

Fig. 3 Overview of the supported data types of VAT

stored in EventStore [6], a system for storing temporal data streams that is highly optimized for write throughput.

We use libraries like GDAL and GEOS for import, export and low-level operations whenever applicable. The system currently supports 36 basic operators for classifications, correlations (including correlations between data of different types), coordinate transformations, aggregations and analyses. For specialized requirements, we support the upload of R scripts which the system executes as an operator. While these operators are already sufficient to support most of our scientific use cases, we continue including more analytical operators, both for general and specialized applications.

5.2 Implementation of Exploratory Workflows

MAPPING models workflows as a rooted tree, with each node being an operator. The leafs of the tree are inputs, usually loading data from disk. Each operator requests data from its child nodes and passes the processed results to its parent. Finally, the root node outputs the result of the workflow. It is encoded in a standardized format and transferred to the client.

The result of a workflow must be reproducible in order to support the scientific method. If a workflow is executed again, it must yield the same results. Reproducibility requires three basic features. First, workflows must have a textual representation that can be permanently stored, shared and retrieved. This is also required to encourage sharing and collaboration between scientists. Second, both data and operators must be versioned so that workflows can be re-run on old data with old versions of the operators. Third, we must require each operator to produce deterministic results, based only on the input data. All three features combined allow us to re-run a workflow at any point in time and receive the same results. This property relieves us from having to store each result, since it can be recalculated on demand. It also makes aggressive caching possible, since the result of a workflow (or even a partial workflow) is not allowed to change.

Our front end, WAVE, supports scientists in creating exploratory workflows. Users may load and visualize data by creating leaf operators and workflows on top. They can iteratively add, remove or change operators until an interesting workflow is achieved. To ease this task for the user, we automate the technical bookkeeping as much as possible. For instance, when two results with different coordinate reference systems are correlated, WAVE automatically inserts a re-projection operator such that they are combinable. Workflows can be visualized, including a small preview of each intermediate result to ease traceability and debugging. They can be shared between scientists, combined and extended.

Standard compliant interfaces are important for the interoperability of the system. We support import and export using standard OGC protocols, like Web Map Service (WMS), Web Coverage Service (WCS) and Web Feature Service (WFS). This allows users to include MAPPING as part of a larger, distributed workflow, if desired. A scientist might use our system for computationally intensive tasks first and then export the result to a local machine for analysis with specialized tools.

5.3 Query Processing and Optimization

To support the exploratory approach, VAT needs to deliver results in a timely manner. Fast response times are a key aspect in getting users to accept the VAT system for their purposes.

A major performance consideration which heavily influenced our architecture is the principle of on-demand processing. For visualization, users can pan and zoom on their map. When zoomed in, we restrict the data for processing according to the visible area. Moreover, a raster intended for visualization does not need a higher resolution than the user's monitor can display. MAPPING supports both the calculation on the full resolution for producing exact scientific results and the calculation on a lower resolution for the purpose of visualization and quick approximate results.

The demand-driven paradigm also reflects in the filter push-down mechanism as it is known from database query optimization [13]. The processing of our workflows starts at the root operator, which is parametrized with a spatial region and time interval of interest. These important filter predicates are pushed down to the child operators in a recursive fashion. Thus, the leaf operators of the workflow only read the data (from disk) that is actually needed within the workflow.

A push-down of spatial and temporal filter predicates can often be done directly without any kind of modifications. However, there are also situations where a direct transfer is not possible, as the following scenarios exemplify. A raster operator averaging neighboring pixels requires a slightly enlarged region of interest, so that neighbors are available for boundary pixels, too. A re-projection operator translating between different coordinate reference systems transforms the region of interest into the child's coordinate reference system before querying it. This is a non-trivial task.

We designed our raster storage to allow such queries on smaller regions. All rasters are cut into tiles, which are stored separately. We also store pre-aggregated lower resolution versions of each raster – these are called *pyramids* in geoscience or *mip-maps* in image processing. The vector back ends (both PostGIS and EventStore) efficiently answer filter queries of our system.

5.4 Exploiting GPU Acceleration

Graphical Processing Units (GPUs) are designed for fast image manipulation, making them suitable for operations on raster data. GPUs consist of tens or hundreds of parallel computing units, allowing the concurrent calculation of multiple values.

Our system supports the implementation of operators using the Open Computing Language[15] (OpenCL). OpenCL provides an abstract model for parallel GPU and CPU programming. This allows us to incorporate both processing architectures in our computations.

Not all algorithms are a good fit for the GPU, though. Inherently sequential algorithms should be run on the CPU, which offers higher single-thread performance. The CPU is also required for interaction with the operating system, e.g. for file I/O. Small computations do not benefit from GPUs either, since all relevant data needs to be copied to GPU memory before processing. For very light workloads, this *data migration* can be more expensive than the actual computation.

We examine the recently published Heterogeneous System Architecture [18] (HSA), where CPU and GPU share a common main memory address space. This architecture offers several advantages. The cost of data migration is greatly reduced, requiring no copy. The data size is not restricted by the GPU's limited memory (today usually no more than 4GB), but only by the size of the main memory that can be extended up to 32 GB on AMD's first generation HSA systems.

We designed all our data structures to be suitable for GPU processing. Our simple feature implementation stores coordinates in contiguous memory, avoiding pointer references wherever possible. Compared to a naïve implementation of dynamically allocated objects per feature (as used by GEOS and the JTS Topology Suite[16]), this approach has less memory overhead for allocations and pointers, increases cache locality for common operations and decreases the amount of memory locations that need to be transferred to the GPU.

[15]www.khronos.org/opencl

[16]www.vividsolutions.com/jts

5.5 Current State

In this subsection we compare the current state of VAT to the ten requirements listed in Section 3. MAPPING supports heterogeneous data (R1), can automatically transform data as needed (R3) and is based on workflows (R5). VAT is already equipped with important low-level operators (R4), but new operators will be added in the future.

Our web-based frontend WAVE is a prototype of a rich user interface (R10), enabling data browsing, exploratory building of workflows and visualization of the results (R2). Due to the concept of on-demand-processing, we can provide fine-grained data lineage (R6), identifying exactly the subsets of the source data sets that were used to compute a result. Workflows can already use custom data in various formats (R9), but we have not yet addressed the rights and license management required for secure storage of custom data in a shared system.

While our current work focuses on enabling low latency computation (R7) on a single node, we plan to make our approach scalable in a private cluster (R8) based on our optimized operators.

6 Evaluation

To evaluate the validity of our overall approach, we implemented a test scenario using a plausible real-world example, based on our experience with biodiversity scientists in our research projects. Using this scenario, we compared multiple geo-processing systems and examined the performance benefits of GPU processing.

6.1 Description of the Use Case

Our use case is inspired by biodiversity research where potential habitats of species are modelled by combining known preferred environmental parameters with appropriate spatial data. The European wildcat (*Felis silvestris silvestris*) was selected as the species of interest for our use case. According to the IUCN/SSC Cat Specialist Group[17], the following environmental parameters characterize habitats:

1. Little to no human activity, preferably forests
2. Little to moderate snow cover
3. Elevation below 2250 m

To produce a map with potential habitats we made use of the *Global Landcover Map for the year 2000* (GLC2000[18]) to identify forests and low human activity. Additionally, we

[17]www.catsg.org

[18]ec.europa.eu/jrc/en/scientific-tool/global-land-cover

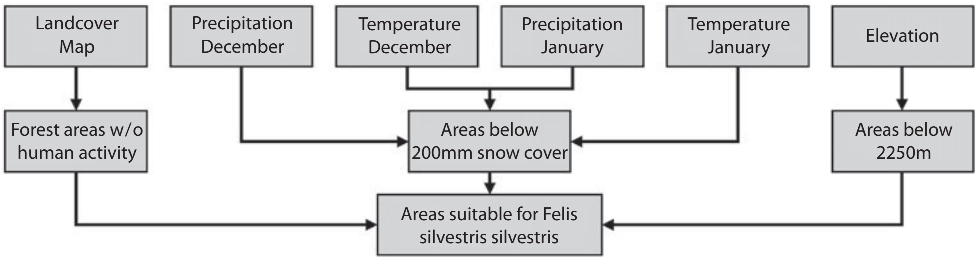

Fig. 4 Workflow to find potential habitats for European wildcat (*Felis silvestris silvestris*)

used *WorldClim* [15] as source for elevation data and to estimate snow cover. Fig. 4 shows the workflow of the use case where the base data sets are shown on top and the processing of the workflow is in top-down direction. The landcover map includes many different types of land use. We used it to process a binary habitat classification. A similar binary classification is also used to estimate pixels with heavy snow cover. The estimation of snow cover is more difficult and uses the temperature and precipitation data for December and January from *WorldClim*. Both estimations are then used to classify pixels below 200 mm as possible habitat. For the third criterion the elevation data from *WorldClim* is used to classify pixels below 2250 m as suitable. In the last processing step the raster data from the previous operators are merged into the potential habitat map. Pixels where all classifications indicate a potential habitat are marked as true.

To validate the result, we designed a second workflow. It compares the resulting raster of possible habitats with actually observed occurrences of wildcats. For this, it correlates occurrences from the GBIF[19] catalogue with our result raster, and produces a histogram over the points' new attribute.

6.2 Evaluation of Systems

In our evaluation of systems we compared the usability as well as the processing time for representative systems from different domains. For each investigated system we briefly outline how we implemented the benchmark and how well they supported this process. Since several of the compared systems do not support vector data, this comparison does not include the verification workflow.

6.2.1 Implementation

For the comparison we selected systems from different domains. Scripting languages are represented by R, GIS applications are represented by QGIS, Taverna represents scientific workflow systems and web services, and GeoTrellis on Spark represents a Big Data processing engine. To facilitate comparison, all rasters were transformed into a common, uniform grid before executing the workflow. This relieves the systems from performing the required transformations themselves. While all evaluated systems are able to handle different data types and coordinate systems, implementing the required conversions in multiple systems turned out to be too laborious.

R allows loading and processing raster data using the *raster* package. While it is possible to implement the use case as a multi-step computation, this proved to be very slow. As a consequence we implemented it as a single-step computation. The implementation of the workflow required experience in R, but was otherwise straightforward.

QGIS provides a model builder where operators from different GIS programs are available to build workflows. We implemented the use case using the `r.mapcalc` module from GRASS[20] which applies a user-defined formula to each pixel. The visual tools were easy to use and allowed a quick implementation of the use case workflow which worked out of the box.

Taverna focuses on web services. We installed pyWPS[21], a geo-processing web service, with GRASS as processing back end. To avoid being limited by network bandwidth, Taverna and pyWPS were executed on the same machine. Implementing the use case as a Taverna workflow proved to be challenging. Setting up pyWPS as web service required manual programming and configuration. We also tried using external pyWPS instances, but most of them denied processing the raster data for the use case because of its size. Implementing and running the use case in Taverna produced several difficulties, from XML encoding problems to out-of-memory errors. We managed to work around some of these problems, but were unable to run the workflow for all data set sizes.

GeoTrellis on Spark is in an early state of development. Hence, we faced several small obstacles while implementing the use case. The actual programming was straightforward but required experience in the Scala language. Using Spark,

[19]www.gbif.org

[20]grass.osgeo.org
[21]pywps.wald.intevation.org

Table 1 Processing time of the Felis silvestris silvestris workflow on different systems

	Mapping	QGIS	R	Spark	Taverna
Germany	0.7 s	3.5 s	48 s	4.9 s	53.1 s
Europe	3.8 s	35 s	3600 s	118 s	-
World	29 s	368 s	18600 s	1337 s	-

GeoTrellis[22] is currently able to process data either from HDFS or Accumolo[23], which is a distributed key-value store. We chose HDFS and thus had to ingest the rasters into the file system as a first step. This required splitting the large rasters into smaller tiles in order to avoid out of memory errors. In a second step we were able to load them from HDFS and perform `map` as well as `combine` operations on the rasters, which the engine executes on a tile by tile basis. Finally, we stored the resulting raster on HDFS again. In order to perform a sound comparison with the other systems, we did not make use of Spark's ability to distribute the computation across multiple machines. Instead, we ran the use case in standalone mode on a single virtual machine.

Mapping, our own system, uses built-in operators for all processing steps. The implementation was mainly done using the *expression* operator which works similar to GRASS' *r.mapcalc*. Using the exploratory workflow approach of our system, the use case was implemented step by step. We extracted the final workflow and used it for the experiments. Neither multi-threading nor GPU processing was exploited in the experiment.

6.2.2 Results

We ran all experiments in a virtual machine on a computer with an Intel Core i7 CPU, 16 GB of main memory and an SSD. To facilitate a fair comparison, we designed the experiment to use only one CPU-core. Table 1 shows the run time of the workflow execution on areas of different size. The individual rasters for the area of Germany are 1200×1200 pixels in size (2.8 MB). The ones for Europe are roughly 40 times larger than the German ones (113 MB) and the world rasters are again 12 times bigger than the European ones (1.34 GB). The results show a substantial improvement in single-thread performance of MAPPING in comparison to its competitors. For the Germany scenario MAPPING is between 5 and 75 times faster than the other systems. The difference is even more apparent for larger regions.

6.3 Evaluation of GPU Acceleration

To evaluate the impact of GPU computing, we executed the use case on the Europe and World data sets using MAPPING, both with and without GPU acceleration. Additionally, we incorporated the verification workflow to benchmark operations on vector data. Running the workflow against the 9009 occurrences of wild cats available in GBIF yielded unreliable benchmark times, so we duplicated the cats' occurrences until we had 1.8 million points.

Table 2 Computation time of the different workflows

	CPU	GPU	HSA
Felis Europe	0.55 s	0.37 s	0.11 s
Felis World	6.2 s	-	1.13 s
Verification	19 ms	50 ms	3 ms

All GPU tests were executed on an HSA enabled AMD A10-7850K APU, which features 4 CPU cores at 3700 MHz and 512 GPU shaders at 720 MHz. The results in Table 2 contain only the computation times of the `expression` and `point_attribute_from_raster` operators, determined using the per-operator profiling built into MAPPING. Overhead for system initialization and disk I/O is not included.

The Felis World workflow cannot be executed on the GPU, because it exceeds the maximum buffer size allowed by AMD's OpenCL drivers. In contrast, HSA does not have this buffer size limitation and is able to run all experiments. The verification workflow is slower on the GPU due to the overhead of data copies. The better performance of HSA over GPU is only due to the zero-copy data migration. In these experiments, HSA shows a $5\times$ speedup over an optimized, multi-threaded CPU implementation.

7 Conclusion and Future Work

In this paper we considered the challenges for supporting exploratory and data-driven geo-science which pose unique processing requirements. We showed that these requirements are not fully addressed in current systems. With the goal to support all the required functionality within a single system, we presented an overview of our VAT system and detailed some of its components, e.g. GPU-based query processing. Results of experiments showed that the VAT system performs geo-scientific queries substantially faster than other systems, in particular when exploiting GPU acceleration. In our future work we will address scalability in our system by distributing data and computation across multiple machines in order to match the growth in data size and in-depth analysis requirements.

Acknowledgments This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. SE 553/7-1 (GFBio) and by the Bundesministerium für Bildung und Forschung (BMBF) under grant no. 01LL1301 (IDESSA).

[22]github.com/geotrellis/geotrellis
[23]accumulo.apache.org

References

1. Aji A, Wang F, Vo H et al (2013) Hadoop-GIS: a High Performance Spatial Data Warehousing System over MapReduce. Proceedings VLDB Endowment 6(11): 1009–1020
2. Andrienko G, Andrienko N, Demsar U, Dransch D, Dykes J, Fabrikant SI, Jern M, Kraak MJ, Schumann H, Tominski C (2010) Space, time and visual analytics. Int J Geogr Inf Sci 24(10):1577–1600
3. Authmann C, Beilschmidt C, Drönner J, Mattig M, Seeger B (2015) Rethinking Spatial Processing in Data-Intensive Science. BTW 2015 Workshop
4. Bach K, Schäfer D, Enke N et al (2012) A comparative evaluation of technical solutions for long-term data repositories in integrative biodiversity research. Ecological Informatics 11:16–24
5. Baumann P, Dehmel A, Furtado P et al (1998) The Multidimensional Database System RasDaMan. In: ACM SIGMOD Conference '98, pp. 575–577. ACM
6. Baumgärtner L, Strack C, Hossbach B, Seidemann M, Seeger B, Freisleben B (2015) Complex Event Processing for Reactive Security Monitoring in Virtualized Computer Systems. The 9th ACM International Conference on Distributed Event-Based Systems
7. Bose R, Frew J (2005) Lineage retrieval for scientific data processing: a survey. ACM Computing Surveys 37 (CSUR) (1):1–28
8. Brown PG (2010) Overview of sciDB: large scale array storage, processing and analysis. In: Proc. of the ACM SIGMOD Conference, pp. 963–968
9. Corvalan C, Hales S, McMichael AJ (2005) Ecosystems and human well-being: health synthesis. World health organization
10. Cuevas-Vicenttín V, Dey S et al Scientific Workflows and Provenance: Introduction and Research Opportunities. Datenbank-Spektrum 12 (3), 193–203 (2012)
11. Dean J, Ghemawat S (2008) Mapreduce. Comm. of the ACM 51(1):107
12. Diepenbroek M, Glöckner F, Grobe P et al (2014) Towards an Integrated Biodiversity and Ecological Research Data Management and Archiving Platform: The German Federation for the Curation of Biological Data (GFBio). GI: Informatik 2014 – Big Data Komplexität meistern
13. Garcia-Molina H, Ullman JD, Widom J (2000) Database system implementation, vol. 654. Prentice Hall Upper Saddle River, NJ
14. Hey AJ, Tansley S, Tolle KM et al (2009) The Fourth Paradigm: Data-Intensive Scientific Discovery, vol. 1. Microsoft Research Redmond, WA
15. Hijmans RJ, Cameron SE, Parra JL et al (2005) Very high resolution interpolated climate surfaces for global land areas. Int J Climatol 25(15):1965–1978
16. Jetz W, McPherson JM, Guralnick RP (2012) Integrating biodiversity distribution knowledge: toward a global map of life. Trends Ecol & Evol 27(3):151–159
17. Kini A, Emanuele R (2014) Geotrellis: Adding Geospatial Capabilities to Spark. Spark Summit 2014
18. Kyriazis G (2012) Heterogeneous system architecture: a technical review. AMD Fusion Developer Summit
19. Lam C (2010) Hadoop in Action, 1st edn. Manning Pub., Greenwich, CT, USA
20. Manyika J, Chui M, Brown B et al (2011) Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute
21. McLennan M, Clark S, Deelman E, Rynge M, Vahi K, McKenna F, Kearney D, Song C (2013) Bringing scientific workflow to the masses via Pegasus and HUBzero. Proceedings of the 5th International Workshop on Science Gateways 13 p. 14
22. Open Geospatial Consortium Inc. (2011) OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture
23. Vitolo C, Elkhatib Y, Reusser D et al (2015) Web technologies for environmental Big Data. Environmental Modelling & Software 63 pp. 185–198
24. Warmerdam F (2008) The geospatial data abstraction library. Open Source Approaches in Spatial Data Handling. Springer Berlin Heidelberg, pp. 87–104
25. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I (2012) Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association