Computers & Graphics 49 (2015) 69–81

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on Processing Large Geospatial Data

A modular software architecture for processing of big geospatial data in the cloud

Michel Krämer [a,b,*], Ivo Senner [a]

[a] Fraunhofer Institute for Computer Graphics Research IGD, 64283 Darmstadt, Germany
[b] Technische Universität Darmstadt, 64283 Darmstadt, Germany

ARTICLE INFO

Article history:
Received 1 September 2014
Received in revised form
16 February 2015
Accepted 18 February 2015
Available online 26 February 2015

Keywords:
Cloud computing
Big Data
Geoprocessing
Distributed systems
Software architectures
Domain-specific languages

ABSTRACT

In this paper we propose a software architecture that allows for processing of large geospatial data sets in the cloud. Our system is modular and flexible and supports multiple algorithm design paradigms such as MapReduce, in-memory computing or agent-based programming. It contains a web-based user interface where domain experts (e.g. GIS analysts or urban planners) can define high-level processing workflows using a domain-specific language (DSL). The workflows are passed through a number of components including a parser, interpreter, and a service called job manager. These components use declarative and procedural knowledge encoded in rules to generate a processing chain specifying the execution of the workflows on a given cloud infrastructure according to the constraints defined by the user. The job manager evaluates this chain, spawns processing services in the cloud and monitors them. The services communicate with each other through a distributed file system that is scalable and fault-tolerant. Compared to previous work describing cloud infrastructures and architectures we focus on the processing of big heterogeneous geospatial data. In addition to that, we do not rely on only one specific programming model or a certain cloud infrastructure but support several ones. Combined with the possibility to control the processing through DSL-based workflows, this makes our architecture very flexible and configurable. We do not only see the cloud as a means to store and distribute large data sets but also as a way to harness the processing power of distributed computing environments for large-volume geospatial data sets. The proposed architecture design has been developed for the IQmulus research project funded by the European Commission. The paper concludes with the evaluation results from applying our solution to two example workflows from this project.

1. Introduction

With the availability of modern sensors and LiDAR scanners (*L*ight *D*etection *A*nd *R*anging) that deliver hundreds of GiB up to several TiB per hour, the world-wide volume of geospatial data grows exponentially. While processing spatial information has always been a complex and time-consuming task, this new kind of large-volume data (*Big Data* or *Big Geo Data*) requires new techniques that make use of modern technology such as multi-core programming and GPGPU programming [1]. However, such large data typically does not fit into the memory of one computer. It is often stored in a distributed manner on multiple computer systems. These computers make up the nodes of a distributed infrastructure typically referred to as a *cloud*. There is an ongoing effort to utilise the cloud for the processing of Big Geo Data and to make it available for a wide range of applications such as earth observation [2–4], environmental protection [5,6], or urban planning for future smart cities [7–9].

An important property of clouds is their scalability which is facilitated by the fact that a cloud infrastructure offers virtually unlimited resources in terms of processing power and memory. Large data centres contain hundreds of computer systems with inexpensive, medium-sized hardware—often referred to as commodity hardware—that can work in concert to store large-volume data and to perform complex computations. Should resources become low, new nodes can be added to the cloud, typically without any system downtime. At the same time, clouds provide broad network access which makes the stored data available from virtually anywhere through thick and thin clients [10]. This is an important fact to be considered if up-to-date geospatial data is required in the field.

In this work we present a software architecture for processing of large-volume geospatial data in the cloud. Our design consists of a web-based user interface where expert users can define a processing workflow using a domain-specific language (DSL). This workflow is evaluated by an interpreter and a software component

* Corresponding author.
E-mail address: michel.kraemer@igd.fraunhofer.de (M. Krämer).

http://dx.doi.org/10.1016/j.cag.2015.02.005
0097-8493/© 2015 Elsevier Ltd. All rights reserved.

called job manager subsequently. Both components use predefined rules to create a processing chain that specifies execution of the workflow on a given infrastructure according to user-defined constraints. While previous work focuses on spatial data storage and provisioning [5] as well as on specific programming paradigms [3,11,12] we present an architecture that is modular and flexible. It supports multiple algorithm design paradigms such as MapReduce [13], actor-based programming [14] or in-memory computing [15]. Which kind of programming paradigm is used for a certain algorithm depends on the specific use case and requirements. In addition to that, our architecture does not rely on a specific cloud infrastructure but can be deployed to multiple ones.

The rest of this paper is organised as follows. We start with an analysis of related work and describe how our approach differentiates from other cloud architectures for geospatial processing. We also describe functional and technical requirements and how our design meets them. In the main part of this paper we present the overall architecture of our system and then describe the individual components in detail. In the following we discuss implementation challenges and present evaluation results from applying our solution to two example workflows from the IQmulus international research project funded by the European Commission. We finish the paper with a summary and conclusion.

2. Related work

While there has been a lot of work on cloud computing and cloud architectures as well as on geospatial data processing and large spatial databases in the past, the combination of the two, cloud architectures for spatial processing, has only become subject to research in the last couple of years [2,11]. The availability of commercial cloud solutions such as Amazon EC2 or Microsoft Azure has facilitated applications in this area. For example, Qazi et al. describe a software architecture for Modelling Domestic Wastewater Treatment Solutions in Ireland [5]. Their solution is based on the Amazon cloud services on which they install the commercial tool ArcGIS Server via special Amazon Machine Images (AMIs) provided by ESRI. Qazi et al. make use of ArcGIS Server's REST interface to deploy web services providing the spatial data sets. Additionally, they implement a web application that can be used for decision support. While the focus of their work is on deploying a highly available data storage and a decision support tool, they do not cover the issue of very large geospatial data sets and how the capabilities of cloud computing can be exploited to process them. In addition to that, their work depends on the commercial ArcGIS Server and the respective Amazon Machine Images. Our architecture, on the other hand, makes use of freely available open-source software and can be configured to run on different infrastructures.

Li et al. on the other hand leverage the Microsoft Azure infrastructure to process large volume data sets of satellite imagery in a short amount of time [3]. Their solution consists of a cluster of 150 virtual machine instances which they claim to be almost 90 times faster than a conventional application on a high-end desktop machine. They achieve this performance gain by implementing an algorithm based on reprojecting and reducing. This approach can be compared to MapReduce [13]. However, it was explicitly developed for the Azure API which provides a queue-based task model that is quite different to MapReduce. Compared to their approach our work does not focus on one specific processing model. Instead, we describe an architecture that is flexible and facilitates a number of different approaches to distributed algorithm design.

Since a growing number of cloud infrastructure providers support MapReduce—in particular its open-source implementation Apache Hadoop—the geospatial community has started developing solutions specifically targeted at this. ESRI GIS Tools for Hadoop, for example, provides a number of libraries that allow Big Geo Data to be analysed in the cloud. The libraries are released as open source. They offer a wide range of functionality including analytical functions, geometry types and operations based on the ESRI Geometry API. While there has been work leveraging this framework [16,17] the MapReduce paradigm implies fundamental changes to geospatial algorithm design as it has been done before. The effort of migrating an existing algorithm to MapReduce often outweighs its advantages. MapReduce is not the only solution to exploit cloud computing infrastructures. Other approaches such as actor-based programming or in-memory computing are often more appropriate for certain algorithms and in some cases even a lot faster [18]. Our architecture enables arbitrary algorithms to be executed in the cloud which allows developers to select the most appropriate programming paradigm for a specific purpose.

Since geospatial applications in the cloud are quite new, the community is still looking for best practices. Agarwal and Prasad report from their experience with implementing a cloud-based system called Crayons that facilitates high-performance spatial processing on the Microsoft Azure infrastructure [11]. They present several lessons learnt ranging from data storage to system design. In particular, they state that a large system should be designed with an open architecture so individual components can be replaced by others without affecting the overall system's functionality. Our architecture is service-oriented and consists of loosely coupled components that can be exchanged quite easily. That way, a wide range of spatial processing services are supported and can be extended later without requiring fundamental changes to the architecture. Additionally, our approach allows individual components to be replaced if requirements should change in the future.

The OGC (Open Geospatial Consortium) has recently set up a new domain working group for Big Data dealing with service-oriented architectures for distributed processing of spatial data. Our architecture is very flexible and allows for various kinds of processing services. This also includes web processing services such as the OGC WPS (Web Processing Service). The OGC is of major importance for the geospatial community and we consider the possibility of integrating OGC services into our system an advantage. However, OGC services are web-based and have an HTTP interface. This means data has to be transferred through HTTP before it can be processed. This imposes a major performance hit. In our architecture we deploy a distributed file system (see Section 7) to which processing service are directly connected. This allows for faster data access. Nevertheless, including services such as the OGC WMS (Web Map Service) or WFS (Web Feature Service) as an external data sources can be very beneficial if this improves the quality of processing results.

Our system employs a workflow editor (Section 5) and a component called job manager (Section 9.3) that is able to manage the cloud infrastructure, to deploy services to computing nodes and to execute distributed workflows. There are other approaches that also support cloud-based workflow management. Malawski et al. describe a model for the execution of scientific applications [12]. Based on their experience with grid-based component models they show how to dynamically couple so-called workers to create a processing workflow. These workers are stored as a ready to deploy virtual appliance and provide a standardised interface to describe how to connect it to other components. A Ruby-based API allows developers and scientists to create workflows by specifying all involved workers and their order of execution.

The Pegasus Workflow Management System [19] is similar to the approach of Malawski et al. in the sense that it also provides a programming interface for developers and scientists using Java,