Live Geoinformation with Standardized Geoprocessing Services

Theodor Foerster, Bastian Baranski & Harald Borsutzky

Institute for Geoinformatics, University of Muenster, Germany
{theodor.foerster, bastian.baranski, h.b}@uni-muenster.de

Abstract. To realize live geoinformation, which is about providing information as soon as it is available, new approaches for instant geoprocessing and efficient resource utilization are required. Currently, such geoprocessing on the web is handled sequentially instead. This article describes a new approach by processing geodata streams and thereby enabling a continuous processing for improved resource utilization rates. In particular, this work applies HTTP Live Streaming for the example of standardized geoprocessing services. The approach is evaluated for processing large volume datasets of OpenStreetMap data. The presented implementation is based on Free and Open Source software.

Keywords
Live Geoinformation, Geoprocessing, HTTP Live Streaming, Web Processing Service.

1 Introduction

Live geoinformation is considered to be crucial for applications in which decisions a) are based on massive volume of data and b) need to be carried out near real-time (as soon as the data is available). For instance in risk management scenarios, live geoinformation can directly support time critical decision making for saving human lives and infrastructure. Other examples are near real-time analysis of crowd-sourced geodata. All these applications are framed by the idea of the Digital Earth (Gore, 1998) which provides an integrated platform for accessing different kinds of distributed data in near-real time. We envision that live geoinformation will be an integral part of Digital Earth in the future (Craglia et al., 2008).

Providing such information and transforming raw data into value-added information is supported by geoprocessing. Currently, these processes as well as the data are available on the web through web service interfaces. Web service interfaces are currently designed along a sequential request-response mechanism,

in which the data is sent to the service, processed and then sent back. These different phases are handled sequentially, which means, that the service and the client remain idle in the meantime and wait for the other party to complete. This is not sufficient for live geoinformation which is defined as providing information with geographic context to the user as soon as it is available.

To realize live geoinformation, different aspects need to be addressed. One of them is developing an efficient approach for processing geodata streams and thereby improving the latency of the service (initial response time) and resource utilization. Several approaches for improving the scalability and performance of geoprocessing services have been described (e.g. applying Cloud and Grid Computing infrastructures (Baranski, Foerster, Schäffer, & Lange, 2011; Baranski, 2008; Di, Chen, Yang, & Zhao, 2003; Lanig, Schilling, Stollberg, & Zipf, 2008) or the mobile code paradigm (Müller, Bernard, & Brauner, 2010)). Scholten, Klamma, & Kiehle (2006) identify caching, network adaptation, data granularity and communication modes (synchronous vs. asynchronous) as performance criteria. However, an approach for processing geodata streams to realize live geoinformation has not been investigated yet. This article presents an approach for processing geodata streams over the web using standardized web service interfaces and protocols. In particular, the approach is based on HTTP Live Streaming as a loss-less format for real-time data streaming and the OGC Web Processing Service, which is an established web service interface and de-facto standard for processing geodata on the web. The presented approach is applied to OpenStreetMap data, as an example of a massive volume dataset.

Section 2 defines live geoinformation and describes relevant concepts such as geoprocessing services and web-based media streaming. Section 3 presents the proposed approach for processing real-time geodata streams with standardized geoprocessing services and Section 4 presents a proof-of-concept implementation of the presented concept. The presented approach is evaluated against the classic sequential WPS communication pattern in Section 5. Finally, in Section 6 the advantages and disadvantages of the presented concept are discussed. Furthermore, directions for future research are outlined.

2 Related Concepts

This section describes related concepts, such as live geoinformation, geoprocessing services and web-based media streaming. Throughout this section, it will become evident, that efficient processing of real-time geodata streams for live geoinformation has not been achieved yet.

Live geoinformation is about providing information as soon as it is available to the user. This is extremely important for Digital Earth, in which several resources are accessible through a common interoperability layer (Grossner, Goodchild, & Clarke, 2008). To draw appropriate conclusions in time-critical scenarios (e.g. crisis management) from the available data, most up-to-date geoinformation needs to be available. Consequently, one of the backbones of live geoinformation is the

excessive use of web technologies to provide the user instantly with information (anywhere, anytime). Therefore, live geoinformation needs to be developed based on current web technologies.

Overall, live geoinformation imposes requirements to data collection, data communication and data integration. These key requirements are high resource utilization rates, simplicity, interoperability and usability. For this article, data integration and data communication are considered from a computational perspective.

Building blocks of live geoinformation are efficiently creating and handling live geodata streams, as applied in this paper. In the context of geoprocessing services, the real-time processing of live geodata streams and publishing such streams is required. Moreover, detecting and extracting events from such geodata streams is highly interesting in the context of Complex Event Processing (Everding, Echterhoff, & Jirka, 2009). Finally, live geoinformation requires a scalable event- and streaming-based architecture for supporting Digital Earth in the future. Regarding the communication within the architecture, we envision a fully push-based architecture, in which the processes are triggered from the sources (e.g. sensors or created by events). This will limit the communication overhead to a minimum. Technically, this is realized through notification and call-back methods.

2.1 Geoprocessing Services

Geoprocessing services are considered to be one of the building blocks for transforming raw data into valuable information on the web (Foerster, Schaeffer, Baranski, & Brauner, 2011; Friis-Christensen, Ostlander, Lutz, & Bernard, 2007) and are therefore essential for live geoinformation. Geoprocessing services are of interest to academia as well as to industry. In Brauner, Foerster, Schaeffer, & Baranski (2009) the authors describe related challenges, where among others performance is one of them. Currently, geoprocessing services are mainly available through the Web Processing Service (WPS) interface as specified by the Open Geospatial Consortium (OGC).

The WPS interface specification defines a standardized way for publishing and executing web-based (geo-) processes (OGC, 2007). According to the WPS interface specification a process is defined as any calculation operating on spatially referenced data. The WPS interface specification describes three operations, which are all handled in a stateless manner: GetCapabilities, DescribeProcess and Execute. The GetCapabilities operation (Figure 1a) is common to any type of OGC Web Service and returns service metadata. In case of WPS interface, it also returns a brief description of the processes offered by the specific service instance. To get more information about the hosted processes, the WPS interface provides detailed process metadata through the DescribeProcess operation (Figure 1b). This operation returns a description of all parameters,

which are required to run the designated process. Finally, the client sends the Execute request to the WPS and triggers the desired process (Figure 1c). The described communication with the WPS interface is based on HTTP-GET and HTTP-POST using an OGC-specific XML message encoding.

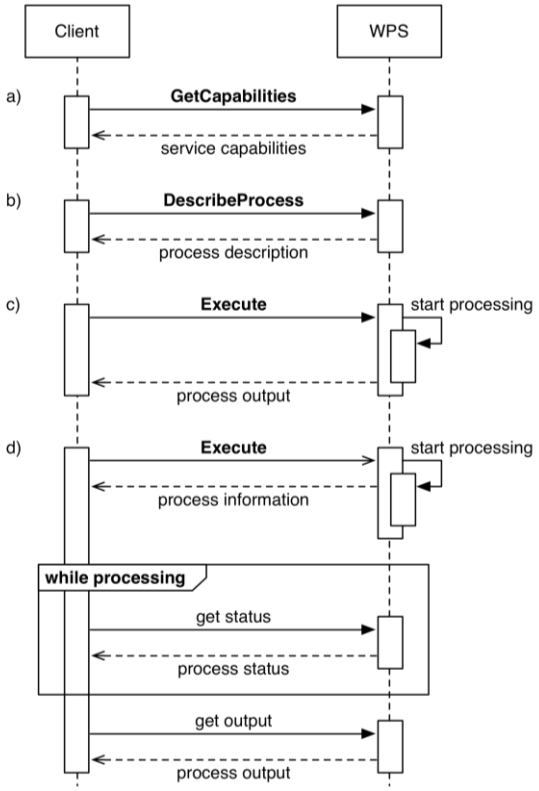

Figure 1. The Web Processing Service (WPS) offers service (a) and process metadata (b). By default, synchronous (c) and asynchronous (d) process execution is supported.

Besides this synchronous communication pattern, the WPS interface provides functionality for scalable processing such as asynchronous processing implemented using a pull model (Figure 1d) and/or storing of process results at the service. First, when executing a process (synchronously or asynchronously) the whole input data is sent to the service. Either directly included in the XML–based request or via a reference (e.g. URL) to the origin of the data; for instance by specifying a GetFeature query to an external OGC Web Feature Service (WFS) (OGC, 2005). Second, the client receives (Figure 1c) or downloads (Figure 1d) the result - typically complex and large data sets - as a whole bunch as soon as a process is finished. The WPS interface specification allows clients to receive basic process status information (e.g. the degree of completeness) but no intermediate results (e.g. a list of feature objects). Therefore, a WPS temporarily stores the

results before the process is completed and is then submitted to or downloaded by the client.

However, two features are missing: continuously processing of real-time data streams and publishing of intermediate process results. Implementing these features will improve the initial service response time as well as the service round-trip performance and allows the processing of (potentially continuous) streams of geodata with standardized geoprocessing services for the first time.

2.2 Media Streaming

Streaming of data is mostly applied for the case of multimedia applications. An overview of the foundation for handling data streams is presented in Muthukrishnan (2005). Streams can be defined as "…a sequence of digitally encoded signals used to represent information in transmission". A common definition of media streaming is not available, but a set of core features of media streaming can be deducted. An overview of the history of media streaming and of the different features such as compression and variable bit rates is described by Conklin, Greenbaum, Lillevold, Lippman, & Reznik (2001). For this article, media streaming enables the parallelization of data transfer and portrayal. This implies that the data stream is available in an appropriate format, ready to be decoded and portrayed. Media streaming reduces the latency, but also reduces the volume of data to be available at a certain time, which is important for continuous data streams.

There are specific requirements for media streaming protocols such as continuous and reliable data delivery (ensured bit rate) and dynamic compression (Conklin et al., 2001). In the context of web-based Geographic Information Systems (GIS), the live provision of data needs to be supported by the protocols. Several of these requirements are implemented by different protocols for media streaming (such as Real-time Transport Protocol (Schulzrinne, Casner, Frederick, & Jacobson, 1996)). A comparison of these Media format is described in Li, Claypool, Kinicki, & Nichols (2005). This article describes HTTP Live Streaming in more detail, as it has been applied in this work.

The Internet Engineering Task Force (IETF) describes a simple protocol for transferring continuous streams of multimedia data over the web, called HTTP Live Streaming (May & Pantos, 2011). HTTP Live Streaming was originally developed by Apple Inc. to distribute live or pre-recorded audio and video in near-real time over the web. The core idea of HTTP Live Streaming for offering media streams over the web is the so-called playlist file, containing an "ordered list of media URIs and informational tags. Each media URI refers to a media file that is a segment of a single contiguous stream. To play the stream, the client first obtains the playlist file and then obtains and plays each media file in the playlist. It reloads the playlist file (...) to discover additional segments" (May & Pantos, 2011). The

format of the playlist file is aligned to the simple but common M3U Playlist file format, used in and developed for current MP3-player software (e.g. Winamp). Since HTTP Live Streaming is completely based on HTTP, simple and well established mechanisms can be applied for realizing reliable and high performance content delivery such as HTTP Caching (Fielding et al., 1999) and for load balancing.

3 Approach for Streaming-based Processing

This section describes the proposed approach for processing real-time geodata streams by the example of the OGC WPS interface. The approach is designed regarding the requirements described in Section 3.1. The benefits of streaming-based processing are highlighted in Section 3.2. The approach is described along a walkthrough (Section 3.3 and Section 3.4).

3.1 Requirements

To enable live geoinformation, processing of real-time geodata streams is essential. The presented approach has been developed along the following requirements.

Loss-less encoding and transfer. GIS analysis relies on complete and accurate datasets, to support decision making sufficiently. Consequently, loosing data artifacts for continuous data delivery is not desirable. In media streaming however, this is a consequence, which is taken most of the time into account for ensuring constant data delivery (for instance by dropping video frames, decreasing bit rate). For this work we thereby chose a protocol for loss-less encoding and data transfer to ensure the delivery of complete datasets.

Interoperability. One of the key concerns of Web Services is interoperability (Alonso, Casati, Kuno, & Machiraju, 2004). It ensures seamless integration into new or existing applications. The proposed approach has to be compliant regarding existing specifications. This has two consequences, reuse as many existing approaches as possible and do not change existing specifications. Meeting this requirement will enable to adopt the proposed approach into existing architectures with little effort required.

Handling, processing, creating of geodata streams. The approach needs to support decoding of incoming data streams and encoding of process results as a data stream according to the specific protocol. Moreover, the process incorporated in the service needs to cope with the incoming data streams. It has to be noted, that not all kinds of algorithms can be performed efficiently on data streams. Especially, algorithms requiring a global knowledge of the data are not suitable for continuous data streams.

3.2 Benefits of Streaming-Based Geoprocessing

Streaming-based processing enables live geoinformation and thereby supports decision makers especially in the context of Digital Earth. This overall benefit can be divided into more specific aspects, which are described in this section.

Processing of geodata streams over the web has several benefits over existing approaches of transferring and processing data sequentially, as designed in the WPS interface specification. Sending data to a geoprocessing service can be realized through a (potentially continuous) stream of data chunks, instead of sending the complete data as a whole at one time to the server. Therefore, a geoprocessing service starts a process immediately after receiving the first data chunk instead of waiting for the whole dataset to be transferred. Second, delivering the process output data from the server to the client can also be realized via streaming to realize the on-demand delivery of intermediate process results. Third, both mentioned aspects improve the initial service response time and the overall service round-trip performance.

Figure 2 depicts the benefits of streaming-based processing over sequential processing regarding the initial service response time and service round-trip performance. In the sequential client-service interaction (Figure 2a), only one party can be active at a time by sending input, processing the input or sending the result. In the streaming case (Figure 2b), the tasks can be performed in parallel. The process starts directly after the first piece of input data is transferred (t2). The service permanently receives further input data and processes the incoming input data at the same time. As soon as the process of a single data chunk is completed, the service starts returning the intermediate result immediately to the client (t3), while the service receives additional data chunks for processing simultaneously. Therefore, the client receives intermediate results shortly after the input data is available for processing. Depending on the volume of the data chunks and the complexity of the underlying algorithm, the final results are much earlier available (t5) than in the sequential case (t6). This is especially true for processing of continuous (endless) data streams, where step t4 is never reached. Consequently, the idle time for the client is reduced to a minimum and the latency is improved.

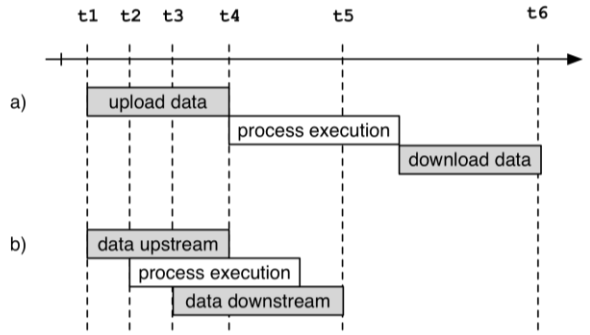

Figure 2. Comparison of sequential communication (a) and streaming-based processing (b).

It has to be noted, that in real-world scenarios the overall service round-trip performance also heavily depends on limiting factors such as the general availability of the input data (network upload speed), the computational capacity of the server (number of CPU cores, memory size), the general ability to process incoming input data in parallel and step-by-step (depends on the type of the algorithm) and the ability to offer instantly intermediate process results to the executing client (disk read and write speed, network download speed).

Streaming can be applied at different parts of the communication between client and server (Figure 3). In the common case the input data and the output data is provided sequentially by sending/downloading the data as a whole at one time to/from the server (Figure 3a). In the second case, either sending data to the service or retrieving data from the service can be implemented using a streaming-based approach (Figure 3b and Figure 3c). Both cases cannot realize the processing of continuous data streams. However, retrieving the data as a stream allows the client to provide intermediate process results to the user. The fully streaming-based approach is depicted in Figure 3d, where request and response are both handled as streams. This allows the architecture to handle real-time and continuous data streams.

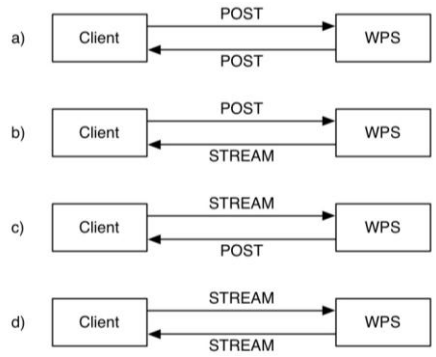

Figure 3. Different applications of streaming between client and WPS.

Comparing the sequential and the streaming-based approach, it becomes clear that streaming-based processing has benefits in cases, where only parts of the data need to be available to perform the process (instead of all the data). Consequently and with regard to the data structure, streaming-based processes can be applied to processes with a local focus, which are applied to single features, or parts of the data (such as buffer, simplification). Other data, where the topological context of the data is required or where a global optimization has to be achieved, streaming-based processing is not beneficial over conventional mechanisms.

3.3 Walkthrough

This section presents a walkthrough of the streaming-based processing using the WPS interface and the HTTP Live Streaming protocol. The walkthrough shows how a client can invoke a streaming-based process and how the service creates the resulting data stream. For simplicity, Figure 4 depicts the sequence of interaction with a WPS that supports the creation of output data streams (Figure 3b). In principle, the same interaction pattern can be extended to loading streaming-based data (e.g. through WFS interface) for supporting also the comprehensive approach (Figure 3d).

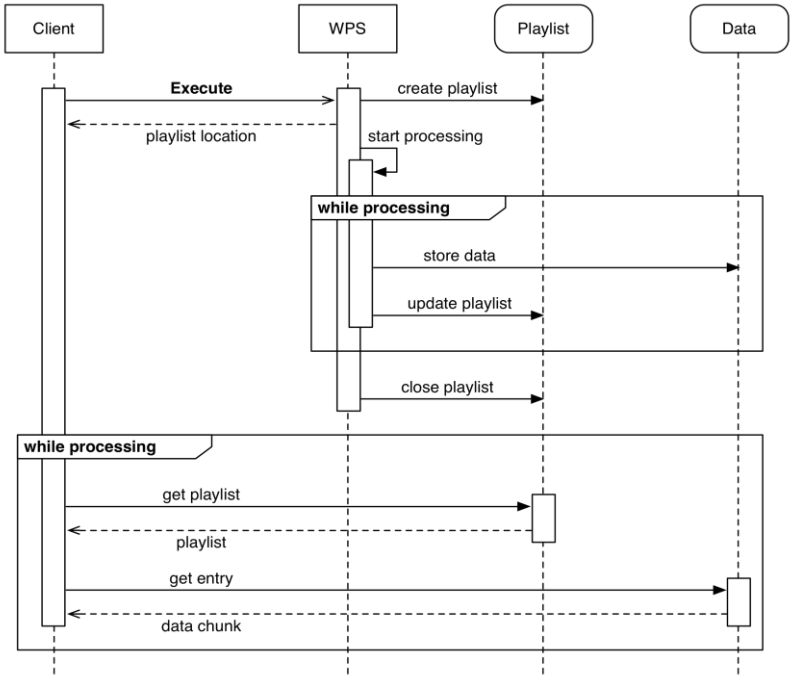

Figure 4. The communication pattern of the streaming-enabled WPS (instant output of intermediate process results).

In the presented walkthrough, the client and the service are interoperable and the client recognizes the streaming capabilities of the service based on specific metadata elements (e.g. mimetype). After retrieving service and process metadata (Figure 1a, Figure 1b), the client builds an Execute request and triggers a specific process via the Execute operation. Since the service offers intermediate process results instantly to the client, the client initiates an asynchronous communication with the server through an Execute request (Figure 4).

When the WPS receives an asynchronous Execute request, an Execute response is instantly returned to the client and the process execution is scheduled in the background (Figure 1d). The Execute response includes a 'Status' element that contains information about the overall status of the process ('accepted', 'started', 'paused', 'succeeded' or 'failed') and an (optional) progress indicator showing the percentage rate of process completion. Furthermore, the Execute response includes a 'statusLocation' element that links another Execute response, which always contains the latest status information about a process. As soon as a process has completed, this Execute response contains the process result(s). The client can constantly pull this Execute response until the final result is available.

In the proposed approach, the body of the 'Status' element includes an URL to a playlist file as specified by the HTTP Live Streaming draft specification instead of

indicating detailed information about the progress of the process as in current WPS implementations (e.g. the amount of features that have been processed). Listing 1 demonstrates an example of an Execute response containing a reference to a playlist file. The format of the playlist file is described further in Section 3.4.

```
<ExecuteResponse service="WPS" version="1.0.0" statusLocation="...">
 <Process ns:processVersion="1.0.0">
  <Identifier>StreamDouglasPeuckerAlgorithm</Identifier>
 </Process>
 <Status creationTime="...">
  <ProcessStarted>
    http://host:port/wps/playlist?id=123&pollingRate=1
  </ProcessStarted>
</Status>
</ExecuteResponse>
```

Listing 1. Exemplary Execute response with an URL of a playlist that contains real-time intermediate results.

The playlist file contains a sorted list of URLs that represents previous and current intermediate results. When an intermediate result is created and stored by the service, the service also updates the playlist file (an URL returning the latest intermediate result is attached). Therefore, by frequently calling the playlist file URL the client receives the latest intermediate results. As soon as a process is completed, the service adds a special tag to the playlist file accordingly. By not adding such a tag, the client knows that the process might run continuously. The format of the playlist is described further in Section 3.4.

3.4 Playlist Format

As presented in the walkthrough (Section 3.3), the playlist plays an important role. The actual playlist of an output stream is provided as an URL in an Execute response (Listing 1) and it provides access to the intermediate results. To have more control over the playlist and the streaming process, the URL returning the playlist file has a specific format as exemplified in Listing 2.

```
http://host:port/wps/playlist?id=123&pollingRate=10
```

Listing 2. An example of the URL returning a playlist file.

The mandatory parameter `id` is unique for each process and allows the client to retrieve the processed results stored by the service. The optional parameter `pollingRate` allows the client to control the size of each entry in the playlist (e.g. the number of feature objects referenced by one entry in the playlist) dynamically and thereby to avoid communication bottlenecks.

The format of the playlist is aligned to the extended M3U playlist format that is used by HTTP Live Streaming (Section 2.2). An example playlist file is shown in Listing 3.

```
#EXTM3U
http://host:port/wps/output?id=123&start=01&stop=10
http://host:port/wps/output?id=123&start=11&stop=20
http://host:port/wps/output?id=123&start=21&stop=30

(...)

#EXT-X-ENDLIST
```

Listing 3. An example playlist document containing a list of URLs referencing intermediate process results.

Each entry in the playlist is again a URL referencing another file stored by the service that contains an intermediate result. The format of that URL is not specified further, but has been enhanced to deliver an intermediate result. In particular, the URLs (as shown in Listing 3) contain the parameter `id` for identifying the intermediate result of a specific process, and the parameter `start` and `stop` reflecting the size of each entry in the playlist as indicated by the `pollingRate` parameter within the playlist URL.

4 Implementation

This section presents a proof-of-concept implementation of the presented approach for processing real-time data streams with standardized geoprocessing services (Section 3.2 and Section 3.3). The implementation is demonstrated for a real-world scenario that incorporates OpenStreetMap data (Haklay & Weber, 2008) and the Douglas Peucker algorithm for line simplification (Douglas & Peucker, 1973). Generalizing OpenStreetMap data has been investigated by Ying, Mooney, Padraig, & Winstanley (2011) for the case of progressive transfer. For this article, we use the generalization of OpenStreetMap data to demonstrate our approach. Generalizing OpenStreetMap data can be interesting to deliver customized data products to specific customers especially for mobile devices with low network bandwidth and the intrinsic requirement for instant data display (e.g. as in crisis management).

The extended WPS interface is realized based on the 52°North WPS implementation[3] which provides a pluggable framework for data and processes. Consequently, only a new data handler had to be implemented and a new type of algorithm, which supports the creation of data streams. Further modifications of the framework were not necessary. The client for handling streaming-based processing has been implemented based on OpenLayers and Google Web Toolkit.

[3] 52°North Geoprocessing community website: http://www.52north.org/wps.

The client triggers the generalization functionality available on the service through the Execute request of the WPS interface. In particular, the Execute request includes a parameter referring the data to be processed and a parameter describing the tolerance value, which is required by the Douglas Peucker algorithm. Based on the parameter the WPS starts processing the data stream and creates and updates the data stream (e.g. the playlist) with the intermediate results continuously (see Section 3.3). The client constantly observes the playlist, pulls the results and visualizes them. The visualized sequence (Figure 5) shows the different intermediate results of generalizing a stream of OpenStreetMap data including the final result.

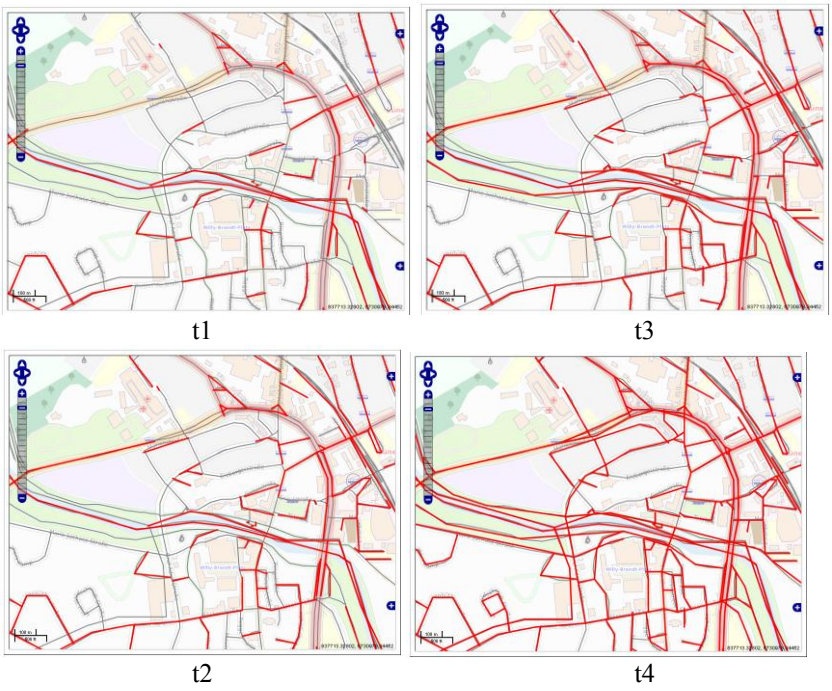

Figure 5. An exemplary sequence of retrieving generalized OpenStreetMap data as a processed geodata stream at different timepoints (t1 < t2 < t3 < t4).

5 Evaluation

This section presents an in-depth performance evaluation of the presented conceptual approach based on the proof-of-concept implementation (Section 4). The overall process runtime and the time for receiving intermediate results are analyzed regarding different amounts of input data.

To evaluate the proposed approach, the response behavior of a classical 52°North WPS implementation (synchronous process execution) and the described

streaming-enabled extension of the 52°North WPS implementation (posting input data and streaming output data; Figure 3b) are analyzed and compared. The overall service round-trip performance of the two approaches (sequential and streaming-based) is measured by sending Execute requests to the service, each with a different amount of input data. Therefore, a sequential and a streaming-based Douglas Peucker algorithm are performed several times with 100, 1000 and 10000 features as input data (served through WFS interface).

Table 1a. The overall response time of a classic WPS implementation mainly depends on the time required to process the actual data.

Number of features (file size)	Initial response time (ms)	Total response time (ms)	Input data (ms)	Process data (ms)	Output data (ms)
100 (0.1 MB)	1214	**1408**	415	**755**	46
1000 (1MB)	7534	**8143**	1145	**6292**	482
10000 (10MB)	66470	**71409**	4975	**61382**	4902

Table 1b. A streaming-enabled WPS implementation produces small overhead for managing data streams, but provides intermediate process results shortly after the input data is available.

Number of features (file size)	Intermediate Output (ms)	Input Data (ms)	Process Data (ms)	Output Data (ms)
100 (0.1 MB)	**515**	408	**896**	12
1000 (1MB)	**1159**	1118	**6677**	9
10000 (10 MB)	**5755**	5689	**65256**	10

NOTE 1: Each use case of the performance evaluation is repeated multiple times and the presented measurements are average values. Therefore, the sum of the times for reading input data, processing and delivering output data might differ from the total response time. The evaluation has been performed on a machine with 2.4 GHz dual core CPU and with 4 GB RAM installed. The data sent to the WPS is stored on geoserver (www.geoserver.org).

In this article, we chose a tabular view, as the different performance indicators cannot be easily accumulated, as depicted in Figure 2. Especially, in the streaming-enabled processing, several tasks are performed concurrently (e.g. data streaming, processing). Table 1a shows the results of the performance evaluation for the sequential approach. The first column (number of features) indicates the number of geometric objects that are processed in the specific test case. The second column (initial response time) indicates the latency of the approach, which is the elapsed time from sending the request to the WPS until receiving the first byte of the response. The overall round trip performance is depicted in the third column (total response time). As the performance, also depends on the time required to fetch the data by the service from the source, this is depicted in the fourth column (input data). The fifth column (process data) indicates the total time

required to process all features by the service. The sixth column (output data) indicates the time required to deliver the process output over the network to the requesting client.

Table 1b shows the results of the performance evaluation of the streaming-based approach, as presented in this article. The second column (intermediate output) depicts the latency of the different configurations, which is the time elapsed until the first intermediate process result is available. In the sequential approach (Table 1a), the most relevant indicator is the total service response time which covers the time for sending the XML request document, fetching the geometry from the source (WFS), processing the actual data and receiving the complete output. The streaming-based WPS is based on asynchronous process execution and the execute response document (which contains the URL of the playlist) is immediately available at the client after the WPS receives the request. Therefore, there are no valuable measurements for the initial and total response time of the streaming-based WPS. Furthermore, the time for sending the Execute response over the network to the client is nearly constant independent of the amount of data that should be processed.

The performance evaluation shows that the overall processing time is nearly equal in both approaches. The streaming-based WPS produces a small overhead for managing the output data stream (approximately 10% of the plain processing time). This management overhead could be significantly reduced (down to 0%) for production on multi-core systems, if the service delegates the management of the playlist file to a separate thread. However, the significant advantage of the streaming-enabled WPS is the instant availability of intermediate results (reduced latency) due to the utilized streaming protocol. Directly after fetching the input data over the network (that takes the same time in both use cases), the streaming-enabled WPS offers the first intermediate result to the client (e.g. after 5.7 seconds for 10000 features). The sequential WPS only provides results after the complete process is finished (e.g. after 61.3 seconds for 10000 features).

This benefit of the streaming-based over the sequential WPS can be augmented by implementing the most advanced streaming pattern, in which the input to the process as well as the output of the process is constantly streamed (Figure 3d). Consequently, the required time for providing the first intermediate process results is expected to be constant independently of the amount of input data (even in cases of continuous input data streams).

6 Conclusion

Enabling live geoinformation on the web is an important aspect to improve decision making for applications such as disaster management. Moreover, live geoinformation is promising to overcome the data-focused approach of existing initiatives and can support the vision of Digital Earth, in which different information sources are integrated in near real-time. In this article we identified that processing of geodata streams is important to realize live geoinformation.

Processing of geodata streams through standardized web service interfaces such as OGC WPS has not been proposed yet.

In particular, we review existing approaches such as geoprocessing services and media streaming (Section 2). It becomes clear, that processing of geodata streams has not been considered yet, but is promising to improve the initial service response time and the overall round-trip performance of geoprocessing services (Section 3.2). Based on the requirements (Section 3.1), we describe an approach for enabling geodata streams based on HTTP Live Streaming and the WPS interface specification. For simplicity reasons and to demonstrate especially the streaming-based processing (handling and creation of data streams), the described walkthrough (Section 3.2) excludes the retrieval of streaming-based resources (Figure 2b). The other described scenarios (Figure 2c and Figure 2d) could be implemented on a conceptual level in a similar way. The proposed approach is implemented with Free and Open Source Software and is demonstrated for the use case of generalizing OSM data. The presented approach is successfully evaluated over the sequential approach demonstrating a significant improvement regarding the latency of the service (Section 5). This is an important step to achieve the processing of continuous data and thereby enable live geoinformation.

Overall, the WPS interface specification has proven to be a suitable candidate to support streaming-based processing. The combination of asynchronous requests and client-based pulling is sufficient to realize an efficient streaming-based approach regarding client and service.

The requirements (Section 3.1) are met regarding several aspects. The presented approach uses a loss-less encoding scheme (HTTP Live Streaming). Based on the playlists created by the streaming source and HTTP as transportation protocol, it is ensured that clients retrieve all the processed data and that no artifact is lost. The presented approach is interoperable, as it does not require changes to the OGC WPS interface specification, but rather defines a WPS application profile for processing and offering (geo-) data streams. Therefore, we propose to include a new mime type parameter in the process description to reflect the streaming capability of a specific process (e.g. the already existing `application/x-winamp-playlist` or the `audio/x-mpegurl` mime types which are related to the HTTP Live Streaming protocol). Furthermore, the implementation shows that the requirement of handling input as data streams, as well as processing the input stream (e.g. generalization) and creating new data streams as output is possible.

Future research needs to focus on streaming-based protocols for data services such as feature services and sensor data services. This would then fully enable a streaming-based architecture and provide live geoinformation as a holistic approach to Spatial Data Infrastructures (SDIs). Regarding WPS interface, the presented approach shows that advanced process management is required to for instance terminate continuous processes, which is an anticipated functionality for the new version of the WPS interface specification. This will improve the flexibility of the framework, as clients can free computational resources on the

service, as for instance a process is misconfigured and not used anymore. Further, existing approaches for progressive transfer (Bertolotto & Egenhofer, 2001; van Oosterom, 2005) should be applied to order the sequence of data chunks being included in the data stream and thereby to improve the user experience with such geo data streams. Finally, the performance in production environments (concurrent requests/massive data sets) and the usability of the proposed approach need to be evaluated thoroughly to achieve live geoinformation in real-world applications.

Acknowledgements

The presented work has been supported by Raphael Rupprecht from the Institute for Geoinformatics. We acknowledge the various comments from Bastian Schäffer and the input from the Geoprocessing Community of 52°North Open Source initiative. Finally, we are thankful for the valuable comments of the anonymous reviewers.

References

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web Services* (1st ed.). Springer Verlag.

Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (1996). *RTP: A Transport Protocol for Real-Time Applications* (Standards track No. RFC 1889) (p. 74). IETF.

Baranski, B. (2008). Grid Computing Enabled Web Processing Service. In E. Pebesma, M. Bishr, & T. Bartoschek (Eds.), *Proceedings of the 6th Geographic Information Days*, IfGI prints (Vol. 32, pp. 243-256). Presented at the GI-days 2008, Muenster, Germany: Institute for Geoinformatics. Retrieved from http://www.gi-tage.de/archive/2008/downloads/acceptedPapers/Papers/Baranski.pdf

Baranski, B., Foerster, T., Schäffer, B., & Lange, K. (2011). Matching INSPIRE Quality of Service Requirements with Hybrid Clouds. *Transactions in GIS*, *15*(s1), 125-142. doi:10.1111/j.1467-9671.2011.01265.x

Bertolotto, M., & Egenhofer, M. J. (2001). Progressive Transmission of Vector Map Data over the World Wide Web. *Geoinformatica*, *5*(4), 345-373.

Brauner, J., Foerster, T., Schaeffer, B., & Baranski, B. (2009). Towards a Research Agenda for Geoprocessing Services. In J. Haunert, B. Kieler, & J. Milde (Eds.), *12th AGILE International Conference on Geographic Information Science*. Presented at the AGILE 2009, Hanover, Germany: IKG, Leibniz University of Hanover. Retrieved from http://www.ikg.uni-hannover.de/agile/fileadmin/agile/paper/124.pdf

Conklin, G. J., Greenbaum, G. S., Lillevold, K. O., Lippman, A. F., & Reznik, Y. A. (2001). Video coding for streaming media delivery on the Internet. *IEEE Transactions on Circuits and Systems for Video Technology*, *11*(3), 269-281. doi:10.1109/76.911155

Craglia, M., Goodchild, M., Annoni, A., Camara, G., Gould, M., Kuhn, W., Mark, D. M., et al. (2008). Next-generation Digital Earth. *International Journal of Spatial Data Infrastructure Research*, *3*, 146-167. doi:10.2902/1725-0463.2008.03.art9

Di, L., Chen, A., Yang, W., & Zhao, P. (2003). The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data (pp. 24-27). Presented at the GGF8 & HPDC12 2003, Seattle, WA, USA: Science Press.

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, *10*(2), 112-122.

Everding, T., Echterhoff, J., & Jirka, S. (2009). Event Processing in Sensor Webs. *Geoinformatik 2009*, ifgiPrints (Vol. 35, pp. 11-19). University of Münster.

Fielding, R. T., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). *Hypertext Transfer Protocol* (Standards track No. RFC 2616) (p. 176). IETF.

Foerster, T., Schaeffer, B., Baranski, B., & Brauner, J. (2011). Geospatial Web Services for Distributed Processing - Applications and Scenarios. In P. Zhao & L. Di (Eds.), *Geospatial Web Services: Advances in Information Interoperability* (pp. 245-286). Hershey, PA: IGI Global.

Friis-Christensen, A., Ostlander, N., Lutz, M., & Bernard, L. (2007). Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions. *Transactions in GIS*, *11*(6), 799-818. doi:10.1111/j.1467-9671.2007.01075.x

Gore, A. (1998). The digital earth: Understanding our planet in the 21st century. *Australian surveyor*, *43*(2), 89–91.

Grossner, K. E., Goodchild, M. F., & Clarke, K. C. (2008). Defining a Digital Earth System. *Transactions in GIS*, *12*(1), 145-160. doi:10.1111/j.1467-9671.2008.01090.x

Haklay, M. (Muki), & Weber, P. (2008). OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, *7*(4), 12-18. doi:10.1109/MPRV.2008.80

Lanig, S., Schilling, A., Stollberg, B., & Zipf, A. (2008). Towards Standards-based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS). *ICCSA*, Lecture Notes in Computer Science (Vol. 5073, pp. 191-203). Presented at the Computational Science and Its Applications - ICCSA 2008, Perugia, Italy: Springer Verlag. doi:http://dx.doi.org/10.1007/978-3-540-69848-7_17

Li, M., Claypool, M., Kinicki, R., & Nichols, J. (2005). Characteristics of streaming media stored on the Web. *ACM Trans. Internet Technol.*, *5*(4), 601–626. doi:http://doi.acm.org/10.1145/1111627.1111629

May, W., & Pantos, R. (2011). *HTTP Live Streaming* (Internet Draft No. draft-pantos-http-live-streaming-06) (p. 24). Cupertino, CA: IETF.

Müller, M., Bernard, L., & Brauner, J. (2010). Moving Code in Spatial Data Infrastructures - Web Service Based Deployment of Geoprocessing Algorithms. *Transactions in GIS*, *14*, 101-118. doi:10.1111/j.1467-9671.2010.01205.x

Muthukrishnan, S. (2005). *Data streams: Algorithms and applications*. Now Publishers Inc.

OGC. (2005). *Web Feature Service Implementation Specification* (Implementation specification No. OGC 04-094). Retrieved from http://www.opengeospatial.org/standards/wfs

OGC. (2007). *OpenGIS Web Processing Service* (OGC implementation specification No. OGC 05-007r7). Open Geospatial Consortium. Retrieved from http://www.opengeospatial.org/standards/wps

Scholten, M., Klamma, R., & Kiehle, C. (2006). Evaluating performance in spatial data infrastructures for geoprocessing. *IEEE Internet Computing*, *10*(5), 34-41.

van Oosterom, P. (2005). Variable-scale Topological Data Structures Suitable for Progressive Data Transfer: The GAP-face Tree and GAP-edge Forest. *Cartography and Geographic Information Science*, *32*(4), 331-346.

Ying, F., Mooney, P., Padraig, C., & Winstanley, A. (2011). Selective progressive transmission of vector data. Presented at the GeoComputation 2011, London, UK.