An On-Demand Processing Framework for Faster Remote Sensing Big Data Analysis

Zhenchun Huang
Department of Computer Science and Technology, Tsinghua University, Beijing huangzc@tsinghua.edu.cn

Abstract. Remote sensing data are considered as “big geo data” because of their huge data volume, significant heterogeneity and challenge of fast analysis. The traditional remote sensing analysis workflows make earth scientists to download raw data files to local workstations before processing them for science discoveries. The data transfer often costs a lot of time and slows down the analysis workflows. Due to results of remote sensing data analysis models are usually much smaller than raw data to be processed in most cases, “on-demand processing”, which tries to upload and run data analysis models near the remote sensing data, can make the remote sensing analysis workflows faster. In this paper, an on-demand remote sensing data processing framework is proposed based on a three-layered architecture for faster remote sensing data analysis workflows. The evaluation on a prototype system shows that the on-demand processing framework can accelerate the execution of analysis models significantly by reducing data transfers, especially for those analysis workflows which transfer data through low bandwidth Internet.

1. Introduction
In the last decades, the volume of remote sensing data continues to multiply exponentially due to the launch of new flying platforms, hosting more powerful, multispectral, and accurate sensors. For example, the data amount managed by NASA’s Earth Observing System Data and Information System (EOSDIS) [1] is more than 9 Peta-Bytes in the beginning of 2018, and still grows at 6.4 TB/day. At the same time, the increase of computing power enables simulations at a global scale with unprecedented accuracy. For many scientists, the ever-growing observation capability and computing power enable a completely new approach for data intensive scientific discovery known as Fourth Paradigm.

Besides its volume, remote sensing data is also characterized by a significant heterogeneity due to historical and technological reasons, such as different acquisition sensors, diverse methodology to describe the real world phenomena, etc. [2] Meanwhile, There are still some challenges for earth scientists to develop a fast and accurate remote sensing data analysis model. One of the challenges is remote sensing data analysis models needs too much time to prepare and process the big data set. Today, Earth science data users are required to download files over low-bandwidth networks to local workstations and process data before science questions can be addressed. In this traditional working model, users access data discovery tools such as Earth Online [3] to download data download. Users must download data over low-bandwidth wide area networks to local system for further processing. It may require users to download thousands of granules before any data sub setting can occur, because they are performed locally. The traditional data processing workflow could consume days, if not weeks or more, of elapsed time.

In most cases, results of remote sensing data analysis models are much smaller than raw data to be processed. If raw data can be on-demand processed “near” where they are stored, the time cost for data
transfer will be reduced because only smaller analysis results are need to be transferred through low-bandwidth WAN. It is called “on-demand processing”. To process raw data “near” on-demand where they are stored, the remote sensing data analysis models need to be uploaded to processing nodes “near” raw data and dynamically deployed for execution.

There are already some efforts at on-demand processing in both service computing and remote sensing data processing. For example, On-demand service in grid tries to speed up mass data processing service by “provide right service for user according to his requirement” in a SOA based grid-computing environment [4]. Furthermore, Web Coverage Processing Service (WCPS) [5] by the Open Geospatial Consortium (OGC), defines a protocol-independent language for the extraction, processing, and analysis of multi-dimensional coverages representing sensor, image, or statistics data. Davis et al. also tries to “move the tools required for processing data to computer system of data providers, and away from the data consumers” to “improve turnaround times for data processing workflows” in LP DAAC Satellite Data Archives [6].

But, there is still a lot of work for on-demand processing of remote sensing data analysis models, one of them is an extensible framework for on-demand processing environment, which makes them possible to build right runtime environments for data analysis models, deploy and invoke the models on given data sets, and control the input/output of runtime model instances.

In this paper, after the introduction about architecture design of the on-demand processing framework in section 2, Section 3 discusses some implementation details such as environment description and on-demand model deployment. Then, section 4 builds a prototype system based on the framework and tested for performance evaluation. At last, section 5 concludes the paper.

2. Architecture
The on-demand remote sensing data processing framework takes a layered architecture shown in figure 1.

There are three layers in the architecture. The bottom one is “resource layer”, with all kinds of resources, e.g., computing nodes, storage devices, archived data, and software components. Resources are organized as a “resource pool” for capabilities of on-demand processing system, such as data storage and processing.

![Figure 1. Architecture](image)

In the engine layer, there are three major function modules: Environment Builder, Deployment Module, and On-Demand Processing Adapter/Monitor, which serves the three major steps for on-demand processing. Environment Builder creates a runtime environment for the remote sensing data analysis models to execute in. Model Deployment Module deploys the analysis models in the runtime environment founded by Environment Builder, and On-Demand Processing Adapter/Monitor invokes the deployed data analysis models and monitors their execution.
The top layer is collaboration layer. In this layer, Programming Interface or User Interface receives On-Demand Processing Descriptions first. Then, On-Demand Processing Description Interpreter parses and translates them as runtime environment descriptions, analysis model bodies, and parameters for models. At last, under the guidance of translated description, the On-Demand Processing Scheduler creates a runtime environment by Environment Builder, deploy the analysis model in the runtime environment by Deployment Module, invoke and monitor the analysis model by On-Demand Processing Adapter/Monitor. Figure 2 shows the workflow.

To launch an on-demand processing workflow, user or client application must create a document named “On-Demand Processing Description”, which should think of everything about the on-demand data analysis processing jobs: what to do (Model Body), how to do (Launching Parameters), and what depends on (Runtime Environment Description). When the On-Demand Processing Description is submitted to the framework through the User Interface or Programming Interface (step (1)), Description Interpreter parses and interprets the description into three parts (Runtime Environment Description, Model Body Description, and Launching Parameters), and send them to On-Demand Processing Dispatcher (step (2)).

![Figure 2. Workflow for on-demand processing](image)

Under the guide of the three parts, the on-demand processing workflow starts. First, Environment Builder creates an appropriate runtime environment described by Runtime Environment Description based on resources provided by resource layer (step (3)). Then, Model Deployer deploys the remote sensing data analysis model into the runtime environment under the guide of Model Body Description (step (4)). At last, On-Demand Processing Adapter/Monitor starts the deployed analysis model by the Launching Parameters and monitors them in the runtime environment (step (5)). When the execution of analysis model accomplishes, results are collected and sent back to users or client applications through programming/user interface (step (6)), and the workflow finishes.

3. Implementation Details
For the successful deployment of analysis models, it is very important to describe the runtime environment fully detailed, including the hardware platform, operating system, compilers, static/dynamic libraries, APIs, etc. It depends on Runtime Environment Description, which often adopts XML format. For example, figure 3 describes a runtime environment with “Windows 10 64bits OS and JDK 1.8.x on a simple computing node”.

In figure 3, hardware platform is selected from a pre-defined list by attribute “type”; operating system is pointed out by element <OS>; and all required components such as compilers and libraries are listed by list of elements <tool>. With a detailed list of frequently used platforms, operation systems, libraries, and toolkits, Runtime Environment Description helps Environment Builder to find
or create a runtime environment for the requirements in the document, and to finish step (3) in figure 2.

Deployment of models is another problem for on-demand processing. There are several ways to describe bodies of remote sensing data analysis models, for example, source codes written by programming language such as C/C++ or Java, executable program files on target OS and hardware platform, scripts interpreted by Python or Unix shell, functional components embedded in well-known software products such as MATLAB or ENVI, etc. Therefore, the Model Body has some different approaches, e.g. a runnable script, an executable file, or a pack of source codes.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<environment type="simple">
  <OS name="Windows">
    <version>10</version>
    <distribution>
      Windows 10 64bits
    </distribution>
  </OS>
  <tool name="JDK">
    <version>1.8</version>
  </tool>
</environment>
```

**Figure 3. Runtime Environment Description**

According to the difference of implementation methods, the models can be deployed in different ways. The simplest one is homogeneous platform based on-demand processing, in which all computing nodes are homogeneous with the same runtime environment, and analysis models are submitted as an executable program files with native codes. In step (4), the deployment accomplishes successfully by simply copying the executable program files into the homogeneous environments. It is simple to be implemented, but not compatible enough for different requirements of different remote sensing data analysis models.

Another possible solution is based on the virtual machine (VM) technology. In this solution, remote sensing data analysis models are pre-bound with runtime environment needed in a VM image, and deployed as a VM instance for execution in step (4). Because a VM image can pack everything needed for the execution of an analysis model, it can be deployed everywhere. The major downside of this solution is the extra time cost caused by uploading VM image with large volume of data. Container technologies with better performance, such as Docker, are also available for this solution.

Furthermore, script language is also possible for the description of remote sensing data analysis models. There are many alternative script languages, e.g., JavaScript, Python, function languages such as R, shell scripts such as bash, and embedded script languages such as IDL in ENVI. Script languages are very flexible; they supply rich and powerful capabilities for analysis models, and simple for deployment in step (4). However, the low execution performance and difficulty of parallelization are obvious shortcomings of script language based model description.

At last, if runtime environment for compiling and building source codes can be created on-demand, it is possible to deploy and execute remote sensing data analysis models written by programming languages such as C/C++, Java, or FORTRAN. In step (3) of this scenario, Environment Builder installs toolkits and libraries described by Runtime Environment Description in the runtime environment first. Then, in step (4), Model Deployer copies the source codes of analysis model into the runtime environment, and compiles them for their further execution. Although it is complex to describe all details about required toolkits and libraries correctly, analysis models described by source codes are much flexible and high-performance because they executes on hardware platform directly and can be well optimized by compilers.
4. Performance Evaluation

To evaluate the performance of on-demand processing framework, a prototype system is built for test. The prototype system has 2 Gigabits Ethernet connected computers, with one Intel i5-4570@3.20GHz CPU, 16GB memory and 4TB hard disk per server. One of them is used as on-demand processing server with Apache Tomcat, and the other one plays the role of client. In the prototype system, analysis models are described by java source codes, and some toolkits and libraries such as JDK and HDF library for Java are ready for on-demand deployment in Windows 64-bits based runtime environment. As an example, a global drought detection analysis model process the MODIS Surface-Reflectance Product (MOD 09) for Normal Differential Water Index (shortly NDWI) brought up by Gao in 1996 [7]. NDWI is:

\[
NDWI = \frac{(\rho_2 - \rho_5)}{(\rho_2 + \rho_5)}
\]

\[
AWI = NDWI - \text{avgNDWI}
\]

**Figure 4.** Algorithm for drought detection

In figure 4, NDWI is the difference between two bands in 8-day composite MODIS surface reflectivity data. The bands are green (\(\rho_2\), 0.86-\(\mu\)m) and near-infra-red (NIR, \(\rho_5\), 1.24-\(\mu\)m). It is an important index for remote sensing of vegetation liquid water from space, which is sensitive to changes in liquid water content of vegetation canopies. avgNDWI is the average value of NDWIs with the same tile in given time scope. As the result, AWI is short for Anomaly Water Index, which points out how drought the vegetation canopies are in a given time. The test data set is MOD09 data on tiles h27v05 and h27v06 for 4 days each year in 2000-2010. The data amount of 88 MOD09 files is about 6.42GB, and the data amount of AWI result files is about 483MB. The bandwidth limitation is set individually at 1MB/s, 4MB/s, 10MB/s, and unlimited (Gbps) by software when performance test runs.

The analysis model executes twice with classical and on-demand processing (shortly ODP) workflows for each bandwidth limitation, and time costs for data transfer, on-demand model deployment and model execution are shown in figure 5.

![Figure 5. Result of Performance Test](image1)

![Figure 6. Speedup](image2)
Figure 6 shows the speedup ratios between on-demand processing and classical workflow in different bandwidth limitations. The ratio increases from 2.75 in Gigabits Ethernet to 12.71 in 1MB/s low bandwidth network, and the time cost decreases for 63.6% ~ 92.1%. It also implies that the improvement of on-demand processing is much more obvious in low bandwidth network such as wide area Internet, which is most common in remote sensing data analysis workflows.

5. Conclusion
In the traditional remote sensing analysis workflows, scientists must make the raw image data ready to be processed on their workstations. It may cost a lot of time to download the archived raw image data because the data are often transferred through low bandwidth Internet. “on-demand processing”, a better choice to move remote sensing data analysis model to where data stores, is implemented based on on-demand processing framework which architecture is discussed in section 2.

To evaluate the performance of on-demand processing framework, a prototype system is built for test. The result shows that on-demand processing framework can accelerate the execution of analysis models in 2.8 ~ 12.7 times by reducing data transfers, especially for those analysis workflows which transfer data through low bandwidth Internet.

Besides accelerates the remote sensing data analysis workflows, on-demand processing also enables the sharing of remote sensing data analysis models. With continuously growing data, tools and libraries in the remote sensing data infrastructures, users can create more analysis models to process remote sensing big data according to their requirements, create new applications based on these models, and exchange their knowledge each other by sharing models.

6. Acknowledgement
The research was sponsored in part by National Key Research and Development Program of China from MOST (2016YFB0501504).

References
[1] Brennan, J. 2006. “NASA’s EOSDIS Data Centers Offer Something for Everyone.” In Earth Imaging Journal, September/October 2006, Vol. 3, No. 5, pp. 30-35.
[2] Ma, Yan, et al. “Remote sensing big data computing: challenges and opportunities.” Future Generation Computer Systems 51 (2015): 47-60.
[3] ESA, “Earth Online,” European Space Agency, 2018. [Online]. Available: https://earth.esa.int/web/guest/home. [Accessed Aug 2018].
[4] Huang, Z. C., He, C., Gu, L., & Wu, J. F. (2005, July). On-demand service in grid: Architecture, design and implementation. In 11th International Conference on Parallel and Distributed Systems (ICPADS'05) (Vol. 2, pp. 674-678). IEEE.
[5] Baumann, P. (2010). The OGC web coverage processing service (WCPS) standard. Geoinformatica, 14(4), 447-479.
[6] Davis, B. N., Werpy, J., Friesz, A., Impecoven, K., Quenzer, R. L., Maiersperger, T., & Meyer, D. J. (2015). Interactive Access to LP DAAC Satellite Data Archives Through a Combination of Open-Source and Custom Middleware Web Services. IEEE Geoscience and Remote Sensing Magazine, 3(4), 8-20.
[7] Gao B C., NDWI—a normalized difference water index for remote sensing of vegetation liquid water from space, Remote sensing of environment, 1996, 58(3): 257-266.