European Journal of Molecular & Clinical Medicine
ISSN 2515-8260 Volume 07, Issue 02, 2020

IOT BASED DATA STREAMING IN DISTRIBUTED HYBRID CLOUD USING SPARKFLOW

[1]M P Rajakumar, [2] S.Ganesan, [3]Dr. N. Sankar Ram, [4]Dr. D. Hemanand, [5]Dr Rajesh Thumma

[1]Associate Professor, Department of Computer Science and Engineering, St.Joseph's College of Engineering, OMR, Chennai - 600 119, Tamilnadu, India.

[2]Professor and Head, Department of Electronics and Communication Engineering, Mailam Engineering College, Mailam-604304, Tindivanam, Villupuram District, Tamilnadu, India.

[3]Professor, Department of Computer Science and Engineering, Dr. NGP Institute of Technology, Dr N. G. P nagar, Kalapatti road ,Coimbatore, Tamilnadu, India.

[4]Assistant Professor, Department of Computer Science and Engineering, Sriram Engineering College, Perumalpattu, Veppampattu (R.S.), Tiruvallur District -602024, Tamilnadu, India.

[5]Associate Professor, Department of Electronics and Communication Engineering, Anurag Group of Institutions, Venkatapur (v), Hyderabad, Telangana-500088, India.

Abstract - Data services provided by many public clouds - files, queues, tables - high performance, streaming friendly cardio services not yet included. Considering the data constantly coming in from the 1.4 million smart meters in homes, there is an urgent need to constantly analyze it to identify the maximum power consumption that is coming in the smart power grid and to inform the utility to respond by increasing or decreasing the additional power sources. Load. Measures to reduce demand. The IoT Workflow Framework, which supports this data model variation, including streaming data, structured archives and files, currently lacks the ability to perform reliable and measurable work on elastic computing platforms such as the cloud. In this paper, we address this by proposing a scientific workflow framework that supports the various data models required for these emerging scientific applications and assesses its performance and reliability on desktop and cloud platforms.

Key Terms: IoT, Spark Flow, Data Streaming, Stream Manager, Cloud Platforms

1. Introduction

Scientific workflows are common in e-science applications. However, the lack of integrated support for data models, including streaming data, structured archives and files, limits the workflow capability that supports emerging applications in stream-oriented energy informatics. This increases the shortage of cloud data services that support reliable and efficient streams. In this paper, we propose and demonstrate a framework in the scientific workflow that supports streams as first-class data and optimizes it for performance and reliable management on desktop and cloud platforms [1].

Introduces Workflow Framework features and empirical evaluation of private eucalyptus cloud [1]. Due to advances in workflow systems, the diversity of data models supported in the workflow is insufficient. The elastic resources available in the cloud meet the unbalanced resource requirements of these applications and the minimum latency requirements required by the cloud [2].

However, the local data services provided by many public clouds - files, queues and tables - are not yet included in the high performance and streaming friendly heart services. However, the lack of integrated support for data models, including streaming data, structured archives and files, limits the workflow capability that supports emerging applications in stream-oriented energy informatics. This increases the shortage of cloud data services that support reliable and efficient streams [3].

2. Related work

T. M. McPhillips et s. Bowers also proposed and introduced the Framework, a scientific workflow that supports streams as first-class data, and has been adapted for performance and reliable implementation on desktop and cloud platforms. Provides workflow framework features and empirical evaluation of the private eucalyptus cloud [4]. Here we showcase the workflow architecture that supports three common data models found in science and engineering applications - files, structured archives and data streams - the ability to move data smoothly from one data model to another; We evaluate the technologies in the workflow framework to ensure the high performance of streaming applications on desktop and cloud platforms; Describe the structural features that increase the reliability of distribution and data flows in cloud environments for streaming applications [5].

In this paper, we address this lacuna by proposing a scientific workflow framework that supports the different data models required for these emerging scientific applications and assesses its performance and reliability on desktop and cloud platforms. In particular, we make the following contributions:

1) We trigger and demonstrate a workflow structure that locally supports three common data models found in science and engineering applications - files, structured archives and data streams - the ability to move data smoothly from one data model to another;

2) We integrate and evaluate methods in the workflow framework to ensure high performance of streaming applications on desktop and cloud platforms; And

3) We describe structural features that increase the reliability of data streams in distributed cloud environments for streaming applications.

The rest of this paper is structured as follows: Application diversity from the eight information enhances the need for partitioning with IV partition, Section 5, Section 6, discussing the discussed conclusion and future improvements.

3. OBJECTIVES & OVERVIEW OF THE PROPOSED MECHANISM

In our dedicated IoT workflow, the framework supports three inspired data models: files, archives, and streams. Workflow Tasks Supports these models as first class input or output ports or parameters. We describe these models in this section with specific data flow prototypes to work in a transparent manner.

A. Data model properties

Files: Files are a popular data model for scientific applications and generally support workflow. They are limited data on the local disk or shared file system, and can be accessed through workflow tasks through standard file system primitives. The contents of the file may change over time, although the scientific data will remain the same, as the workflow will create updated copies instead of tracking the

files. File storage media is also the default. Files can expose the welded structure (e.g. HDF or XML) or use opaque binary format with random access.

Structured Collections: Collections have an order element with a well-defined structure. They are usually limited in size. Disclosure of their structure allows for interesting questions and access patterns. The item-wise nature of the collections makes them well suited to assign them. Collections can be grouped and elements can represent two other data models, files and streams. Collections may contain data for object mapping in advanced languages for access through workflow tasks. Workflow systems such as Kepler / Commod and Taverna support collections. Our proposed IoT workflow framework supports the three data models that were motivated, viz., files, collections and streams. Tasks in a workflow support these models as first class input or output ports or parameters. We describe these models in this section along with specialized dataflow primitives to operate on them in a transparent manner.

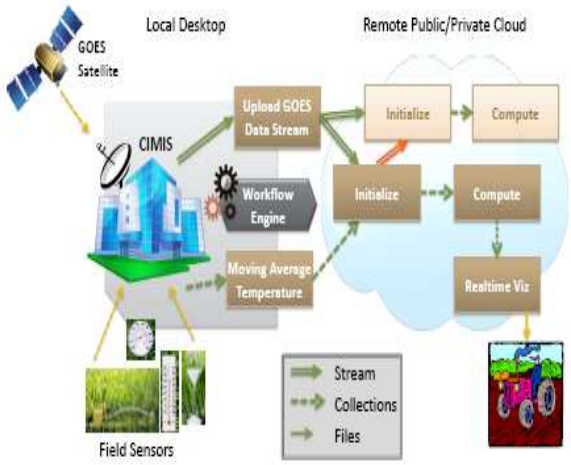

Figure 1. IoT based Sparkflow model in Hybrid Cloud space

Figure 1 shows an ideal workflow that uses streams, files, and archives to create Rns and ET0 maps. Local computers retrieve and re-format special satellite and sensor stream data formats. Provides standard streams for cloud computing systems for image processing, parameter creation and compilation of various datasets. The results are high environmental indicators [5] [6]. It can initiate various trust-management events, including authentication and recognition. Virtual storage supports color generation, embedding and extraction.

B. Workflow Primitives for Streaming

File Management: File management is an unrestricted resource in our system. When we open a file to read or write, the flow of sent and received data becomes available and it becomes a stream. A transmission is a series of bytes that travel from source to destination via a communication path. Files are a popular data model for scientific applications and generally support workflow. They are limited data on a local disk or shared file system, and can be accessed through workflow tasks through standard file system

primitives. The contents of the file may change over time, although the scientific data will remain the same, as the workflow will create updated copies instead of tracking the files. File storage media is also the default. Files can reveal a well-defined structure or use an opaque binary format with random access.

Data Streams: Streams are a continuous series of binary data. They are often unlimited in size - the main difference - and access as a logical byte stream. The continuous behavior of the streams invites them if they are not mapped to another data model. Streams often need to be maintained at high flow rates, but these rates may vary. Streams may have milestones on them [8] which serve as a reference point and help define them. The device that starts and stops the milestones for the stream from the device. A simple and indirect data model supported by workflows are value parameter types such as strings, numbers and booleans. These are well understood and generally supportive. Their discussion was briefly deleted.

a) Stream Manager

Stream administrators act as stream providers. The output is to create an output stream and, if the packet is not present, the stream operators to communicate with the source, recovering the physical flow indefinitely from the damaged location and identifying the total number of packets received and the number of lost packets. This is the main task of the stream manager rather than preventing permanent failure.

B) Transmission

Packets lost during transmission are identified by the stream manager. The stream manager contacts the source and returns the missing packets to the source. So the lost packet is delivered in less time, thereby increasing the capacity.

.

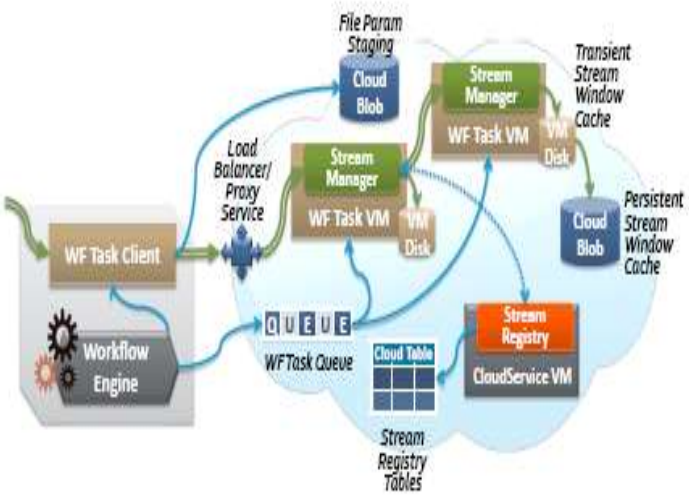

Figure 2. Data Stream and IoT workflow

Figure 2. The structure of the workflow framework on cloud platforms is shown below. Green arrows indicate high-performance streaming data flow; Blue arrows indicate small control data flow.

Data parallelism: Data parallelism often utilizes scientific workflows [and it is an important way to achieve efficient management of distributed resources. The workflow framework should provide the logic needed to distribute streams of work to non-static tasks that can prevent streams of data from being sorted and run in parallel for some time. With the concept of milestones, there are two ways to achieve data parallelism for static tasks running in streams. First, streams can be mapped to border collections (or files) using the conversions previously discussed, with each collection limited by the number of items or the time window, and the elements in the collection executing data in parallel with examples of the same work.

This is the obvious materialization of the stream, and the time it takes to buffer the stream if it is not piped overhead. Alternatively, the stream can be duplicated and transferred to multiple instances of a task, each of which is responsible for processing beyond the itemmark. If things are ideally done on a single cloud VM, or areas between uninteresting milestones can be tracked or avoided, or the streaming framework can be clearly filtered by milestones for optimization. However, some jobs need to be maintained between invoices. These tasks range from simple averages, maximum or minimum calculations, to complex stream analysis such as determining simple item sets. Here, data units (e.g. single data elements, archives or files) need to be processed continuously, which completely eliminates the use of data-parallel approaches.

5. Performance characteristics in the streaming cloud

Unlike regular cloud data services such as files, blogs, tables and queues, streams in the cloud perform poorly. We demonstrate the best data transfer bandwidth using streams as a way to transfer files to the Azure public cloud regarding BLOB file transfers. In addition to those transfer optimizations, we have added several new features on top of the framework in our workflow to further enhance the streaming data model spread across desktop and cloud platforms..

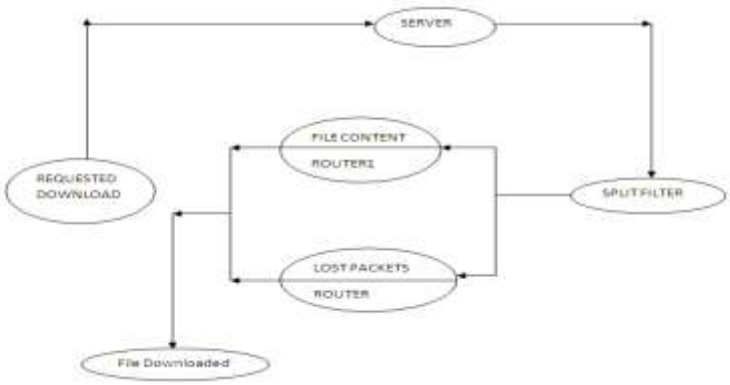

Figure 3. Data flow diagram for responding from server

The use of named streams and landmarks allow streams to be shared with multiple destinations. This is of prime importance when a stream source at a task running in the desktop is shared by several workflow

tasks in the Cloud. Duplicating this stream transfer will use up bandwidth and be punitive in terms of cost. It may also affect the latency of task execution since some Cloud vendors throttle the cumulative bandwidth for a single user account into their public Cloud. The peering stream managers we support in combination with the stream registry addresses this by sharing the stream within the Cloud while passing just one stream from desktop to Cloud. The ability of the stream manager to cache the streams locally on VM disk ensures that the performance benefits of shared streams will outlast the memory available in the VM.

Additionally, the use of Cloud persistent storage to cache some of the streams will help them be reused within the Cloud beyond the lifetime of the VM, and also offload bandwidth or computation overhead on a VM caused by its sharing a stream and evaluated by using figure 3. Currently, our stream managers do not coordinate access to replicas of streams and it is likely under certain cases for a particular VM hosting a stream to be overloaded by requests. We are working on more intelligent and fair stream sharing.The use of named streams and milestones allows streams to be shared with multiple destinations. This is especially important when sharing tasks in multiple workflows in the cloud streaming source of the task running on the desktop. Duplicating this stream transfer uses bandwidth and is penalized based on cost. This can cause delays in task execution as some cloud sellers drag cumulative bandwidth of a single user account into their public cloud. Together with the stream registry we support peering stream administrators fix this by sharing the stream in the cloud when a stream goes from desktop to cloud.

The ability of the stream manager to cache streams locally on a VM disk ensures that the performance of shared streams exceeds the available memory on the VM. Additionally, using cloud persistent storage to cache certain streams can be used again in the cloud over the life of the VM, as well as sharing the stream with offload bandwidth or calculated overhead on the VM. 3. Currently, our stream administrators do not coordinate access to duplicates of streams and in some special cases may be overloaded with requests for a stream hosted by a specific VM. We will work for a more intelligent and affordable stream sharing.

6. Measuring reliability in streaming applications

The ever-running nature of our applications and the fact that they are used by a large user community means that the workflow must withstand errors. Our previous work identified false recovery models for a file-based workflow. Here, we limit our focus to the reliability of the workflow used by the streaming model.

There are two sides to error prevention:

(1) temporary or permanent damage to the physical network;

(2) Damage to virtual machines in the cloud or services running on them.

Transmitting claims and TCP through sockets on the desktop can lead to error, especially the longevity of logical streams. The network connection between the desktop and the cloud is intended to be configured on the desktop workstation, to convert the laptop to another wireless network, or to restart the desktop server stream source after installing the patches. The use of the exposed Logical Stream model as a Java class that executes an interface similar to ByteStream hides damage from the underlying network traffic and workflow application.

Reconnecting to the same source may result in a disconnect of the TCP socket due to an inconsistent network error. If available, permanent network failure can be overcome by finding and connecting a duplicate stream source or optimizing to reconnect with the original source. The protocols used by stream administrators to communicate with each other are recovered from the physical stream crash site and the delay to the recipient of the stream is slightly increased instead of a permanent failure.

The loss of network connectivity between VMs in the Cloud is less frequent, but can be handled in the same manner as above. More of a concern is the loss of a VM instance due to, say, the loss of the physical host or a rolling upgrade to the Cloud fabric. One casuality in such a case could be the stream cached in the memory or local disk of the VM that was lost. We address this by trickling the stream from the VM memory/local disk to Cloud persistent store in a background thread. This ensures persistence of the stream window even if the VM is lost and limits the extent to which the stream has to be retransmitted from desktop client to the recovered VM instance or other VM instances accessing that stream.

7. CONCLUSION

Here the need for streaming support in scientific workflows to support the next generation of scientific and engineering applications that respond to events in the environment at real time. We propose a data model for streams that can coexist with collections and files that are currently supported by workflows. Our implementation of this abstraction for the Rest Flow workflow system shows it to be performant and reliable for operating across desktop and the Cloud. We plan to further build on this initial framework to implement the energy informatics applications we motivated and address novel data optimization challenges that emerge.

REFERENCES

1. E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," Future Generation Computer Systems, vol. 25, no. 5, pp. 528–540, 2018.

2. T. M. McPhillips and S. Bowers, "An approach for pipelining nested collections in scientific workflows," SIGMOD Record, vol. 34, no. 3, pp. 12–17, 2017.

3. T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: Lessons in creating a workflow environment for the life sciences," Concurrency and Computation: Practice & Experience, pp. 1067–1100.2010

4. L. Ramakrishnan, Y. Simmhan, and B. Plale, "Realization of dynamically adaptive weather analysis and forecasting in lead: Four years down the road," in Computational Science – ICCS 2007, ser. Lecture Notes in Computer Science, 2019, vol. 4487, pp. 1122–1129.

5. Y. Simmhan, S. Aman, B. Cao, M. Giakkoupis, A. Kumbhare, Q. Zhou, D. Paul, C. Fern, A. Sharma, and V. Prasanna, "An informatics approach to demand response optimization in smart grids," Computer Science Department, University of Southern California, Tech. Rep., 2015.

6. Q. Hart, M. Brugnach, and S. Ustin, "Calculation of daily reference evapotranspiration for California using GOES satellite measurements and CIMIS weather station interpolation,"California Department of Water Resources, Tech. Rep., 2019.

7. C. Rigollier, O. Bauer, and L. Wald, "On the clear sky model of ESRA(Euorpean solar radiation atlas)with respect to the Heliosat method," Solar Energy, vol. 68, no. 1, pp. 33–48, 2015.

8. [8] B. Temesgen, "CIMIS - past, present, and future," Water Conservation News, October 2003, California Department of Water Resources. [Online]. Available: http://www.owue.water.ca.gov/news/news.cfm.

9. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in Symposium on Operating System Design and Implementation (OSDI), 2016.

10. D. Zinn, Q. Hart, B. Lud¨ascher, and Y. Simmhan, "Streaming satellite data to cloud workflows for on-demand computing of environmental data products," in 5th Workshop on Workflows in Support of Large-Scale Science (WORKS), 2010.