Report from the first workshop on Scalable Workflow Enactment Engines and Technology (SWEET'12)

Jan Hidders
TUDelft, The Netherlands
a.j.h.hidders@tudelft.nl

Jacek Sroka
University of Warsaw, Poland
j.sroka@mimuw.edu.pl

Paolo Missier
Newcastle University, UK
paolo.missier@ncl.ac.uk

ABSTRACT

This report summarizes the presentations and discussions of SWEET 2012, the First International Workshop on Scalable Workflow Enactment Engines and Technologies. SWEET was held in conjunction with the 2012 SIGMOD conference in Scottsdale, Arizona, USA on May 20th, 2012. The goal of the workshop was to bring together researchers and practitioners to explore the state of the art in workflow-based programming for data-intensive applications, and the potential of cloud-based computing in this area. The program featured two very well attended invited talks by Pawel Garbacki from Google and Jimmy Lin from the University of Maryland, on leave at Twitter at the time, as well as a tutorial on *Oozie*, Yahoo's workflow engine based on *Hadoop*, by Mohammad Islam from Yahoo/Cloudera.

1. INTRODUCTION

Current developments in cloud computing are facilitating the convergence of workflow-based processing with traditional data management, potentially providing users with the best of both worlds. However, while it appears that workflow technology is well-positioned to benefit from the scalability of computing resources offered by a cloud infrastructure, before this workshop took place, we were aware of only few examples of cloud-based workflow systems, notably Pegasus [3] and eScience Central [4], along with experimental prototypes that show how MapReduce implementations can be exposed as workflow patterns [6]. The SWEET workshop was aimed at exploring the cross-over between languages and models for parallel data processing, and traditional workflow technology, primarily on a cloud infrastructure and for data-intensive applications. Some of the notable data points at the interface of cloud computing and databases include the well-known HadoopDB [1] and Yahoo's Pig Latin [5], as well as recent work done in the context of the Stratosphere EU project,[1] including amongst others a parallel data processor [2] built on the Nephele parallel data processing framework [7].

Somewhat to our surprise, the blend of peer-reviewed and invited contributions to the workshop revealed a natural division in terms of application domain, namely between (i) workflow systems in support of computational science, on one side, and (ii) workflows in support of large scale social media analytics, on the other. At the same time, a second distinction in terms of the *purpose* served by the workflows also emerged. In the case of computational science, the main motivation for the workflow engines is the need to provide portability across different computational environments, as well as the need to hide the complexity of computational infrastructure to users in order to facilitate their use of the available computing resources. Solutions like *Makeflow*, for example, offer a relatively simple scripting environment which only requires basic knowledge of the commonly used Make program, offering high portability in return. Other data points in this space include the *Turbine* and *DAGwoman* systems, which are briefly discussed in the next section.

The social media analytics space included two invited contributions, one from Google discussing the *FlumeJava* workflow system and one from Twitter on their data management infrastructure, as well as a peer-reviewed paper presenting *Oozie*, Yahoo's own workflow system. These contributions are discussed in Section 3. In contrast to scientific workflows, a common motivation for using workflow technology in this space is to provide coordination and orchestration capabilities across multiple and heterogeneous tools and technologies. Here workflow technology is designed to ensure data integration while leaving development groups relatively

[1] https://www.stratosphere.eu

free to use diverse technologies to solve their specific problems. Thus, while features like portability and usability are not nearly as prominent as in the computational science, this space appears to be dominated by the need to rapidly adjust to evolving business models and technology, in the presence of continuous growth in the scale of the analytics tasks. The emphasis is therefore on letting developers choose technologies they are the most comfortable and productive with.

Details of the papers, keynotes and tutorials are available on the workshop web-site[2], and the proceedings are published on the ACM DL. The rest of the report provides a summary of the contributions, and is structured along the distinction in scope and purpose introduced above. We begin by presenting the contributions in support of large-scale scientific-workflow processing, followed by those in the social media and large scale search space.

2. WORKFLOW ENGINES FOR DATA-INTENSIVE COMPUTATIONAL SCIENCE

The motivation for research on these workflow engines lies in making powerful computational resources accessible and available to non-expert users. The emphasis is therefore on ease-of-use and portability across existing environments used for computational science. This is usually accomplished by offering a simple but powerful interface based on a graphical workflow notation or a simple scripting language. At the same time the scalability of the execution engine is also researched, both in terms of the data size as well as the number of tasks that have to be performed. Four papers represented this line of research.

(Paper) Makeflow: Portable Workflow Management for Distributed Computing

Michael Albrecht from the University of Notre Dame presented this paper on behalf of co-authors Patrick Donnelly, Peter Bui and Douglas Thain. It introduces the *Makeflow* system, which features a simple scripting language *Makeflow* inspired by the Unix *Make* tool. Its goal is to provide a simple workflow interface that is portable and works across different runtimes such as dedicated clusters like the *SUN Grid Engine*, cycle scavenged grids like *Condor*, storage clouds like *Hadoop*, and combinations of the above like *Work Queue*, a master-worker framework designed to work natively with *Makeflow*. The *Makeflow* system analyses workflows to optimize parallelization and can deal with faulty execution engines by intelligently rescheduling tasks. Its effectiveness and scalability is shown with respect to a set of basic data-intensive workflow patterns, and three real-world use cases from the bioinformatics domain involving the execution of *BLAST* services, the analysis of expressed sequence tags and the exploration of interesting regions of assembled genomes through the analysis of single nucleotide polymorphisms.

[2]`http://sites.google.com/site/sweetworkshop2012`

(Paper) Turbine: A distributed-memory dataflow engine for extreme-scale many-task applications

In this paper from the Argonne National Laboratory, authors Justin Wozniak, Timothy Armstrong, Ketan Maheshwari, Ewing Lusk, Daniel Katz, Michael Wilde and Ian Foster present the *Turbine* system. *Turbine* executes workflows specified in the earlier defined *Swift* language which is specifically aimed at specifying programs on large-scale, high performance computing (HPC) systems. The *Turbine* system allows for distributed-memory evaluation of dataflow programs such that the overhead of program evaluation and task generation is spread throughout an extreme-scale computing system. This involves for example the introduction of *futures*, i.e., objects that act as proxies for results that are not yet available. Notable features are the detection of parallel loops and concurrent function invocations, which are translated into parallel executable fragments that optimally use distributed memory and message passing to synchronize. The scalability of *Turbine* was demonstrated on several use cases and in particular analyzed on separate aspects: (i) raw task distribution, (ii) data operations, (iii) distributed data structure creation, (iv) distributed data structure creation and (v) distributed iteration.

(Paper) Evaluating Parameter Sweep Workflows in High Performance Computing

Fernando Chirigati from the Federal University of Rio de Janeiro, Brazil, presented an investigation into the execution of *parameter sweep workflows*, which are workflows that mostly consist of a particular task being executed for a wide range of different input parameters. The other authors are Victor Silva, Eduardo Ogasawara, Daniel Oliveira, Jonas Dias, Fabio Porto, Patrick Valduriez and Marta Mattoso. Parameter sweep workflows are often found in, for example, scientific workflows for the purpose of exploratory analysis. There are different strategies to execute such workflows on high performance computing environments such as clusters,

grids and clouds. For example, the task dispatching strategy can be static or dynamic, i.e., the tasks are distributed in advance over processors or are allocated during the computation to idle processors. Another choice is whether the task is executed in parallel or sequentially for each input vector. This results in four different strategies, and their tradeoffs are investigated when implemented on top of the Chiron workflow engine.

(Paper) DAGwoman: enabling DAGMan-like workflows on non-Condor platforms

Computing *DAGMan* workflows normally requires support from *Condor-G*, the computation management agent for multi-institutional grids that *DAGMan* is a part of. In this paper, Heiko Schmidt and Thomas Tschager from the University of Vienna describe *DAGwoman*, a new workflow engine that is capable of executing *DAGMan* workflows without the need for Condor support. The authors tested the system on one artificial and two bioinformatics workflows, and compared it to *GridWay*'s *GWDAG* engine and to *DAGMan*, showing comparable efficiency in terms of workflow engine delay.

3. WORKFLOW AND DATA ANALYTICS INFRASTRUCTURE FOR SOCIAL MEDIA DATA PROCESSING

While in the previous section the emphasis was on workflow engines for non-programming users, the focus here is on engines and frameworks that are used by developers as back-ends for extremely data-intensive web applications such as social media. As such, they are both used for real-time data processing as well as off-line data analytics. The design goal of these frameworks is to allow for a loosely coupled integration across different and heterogeneous technologies for storing and manipulating these large data sets. The need for this comes from dynamic organizations that grow rapidly and tend to follow the push from business to quickly monetize new ways of collecting and using data. Such flexibility is achieved by letting groups of developers adopt the technologies they are most familiar and thus productive with. These include programming languages, libraries, databases, etc. While this strategy increases group productivity, it also results in multiple technology and data integration problems. The challenges faced in this scenario are the main topic of the papers and presentations in this section.

(Keynote) Data Processing Workflows @ Google

In his keynote, Google engineer Pawel Garbacki gave an overview of current developments at Google in the area of data processing workflows. Distributed large scale data-processing at Google ranges across a variety of tasks, from building indices, to computing ads placement, identifying copyrighted YouTube videos, and constructing geo maps. Whilst self-contained architectures such as *Pregel* and *FlumeJava* ara available to implement specific classes of tasks, there is a need for an overarching workflow system that integrates their capabilities. There is for instance a need to feed the output of a generic MapReduction to a *Pregel* computation, whose output is in turn processed by *Tenzing*, an SQL Implementation on top of the MapReduce framework. The talk discussed the design challenges for such a system, including for example fault-tolerance and automated rescheduling of failed tasks. At the user level, the workflow language should allow quick prototyping of workflows and make reuse of old workflows easy. The performance behavior should be understandable and predictable, and there should be control over resource use. Some solutions where discussed in the talk, but most of these issues are still largely unsolved and pose several interesting research questions.

(Tutorial and Paper) Oozie: Towards a Scalable Workflow Management System for Hadoop

Mohammad Islam from Yahoo delivered a paper talk and then a tutorial on Apache *Oozie*, a workflow management system initially developed at Yahoo and later donated to the Apache Foundation, and aimed specifically at executing workflows on a *Hadoop* platform. Mohammad's co-authors are Angelo Huang, Mohamed Battisha, Michelle Chiang, Santhosh Srinivasan, Craig Peters, Andreas Neumann and Alejandro Abdelnur. *Oozie* workflows are essentially specified by directed acyclic graphs of actions, are designed for scalability, and support security and multi-tenancy features. The system consists of a server engine, reachable through a REST API, with persistence support from both the underlying *Hadoop* cluster, and an SQL database for storing workflow execution metadata. Scalability relies on both types of storage: horizontal scalability comes with the *Hadoop* platform, and scalability in terms of workflow size is achieved by minimizing and efficiently storing the metadata associated to each workflow execution. Multi-tenancy is supported by providing a single web service to which different users can submit their workflows. The system provides security by user authentication for

workflow submitters through a pluggable authentication module. All task management, including scheduling and fault-management, is dealt with by the system, partially by *Oozie* itself and also partially by the underlying *Hadoop* platform. The system was tested in a production setting within Yahoo, and efficiency aspects were measured such as acceptance rate, scalability in terms of length of the task queue and the amount of overhead per workflow and task. Preliminary results seem good, but the scalability could still be improved and better load balancing seems necessary and possible.

(Keynote) Flexibility without Anarchy: Analytics Infrastructure at Twitter

Jimmy Lin from the University of Maryland, on leave at Twitter at the time of this talk, was the second keynote speaker. He elaborated on the needs of the different stakeholders of Twitter's data processing technology, namely the data engineers who maintain the data management infrastructure, the data scientists who create the insights from the acquired data, and the sales people who request those insights. The main challenge associated with the data-processing infrastructure is the flexible orchestration of heterogeneous technology components, including *Pig*, *Hive*, and *Oozie* (briefly described in the previous section), which must cater to each of those stakeholders.

Diversity of technology comes from the company's policy to essentially let developers choose their tools. The Analytics stack runs multiple types of code, including *Hadoop* jobs containing *Java* code, *Pig* programs calling *Java* and *JRuby* functions, and *Pig*, which itself is being called from *Python* scripts and *Ruby* programs. Different storage systems are used such as *HBase*, *MySQL* and *Vertica*. *Cascading* is used to develop data analytics workflows, and although intended for Java developers it is also used with *Python* and *Scala* bindings. Not yet in production use are also *Hive*, which is a data warehouse system on top of *Hadoop*, *Storm*, which is a real-time processing system for coordinating distributed computation that can process messages and update databases in real-time, and *Kafka*, a persistent, distributed message queue capable of loading data into *Hadoop*. Data formats also vary, ranging from *JSON*, to *protobuf*, which is Google's data interchange format, and Apache *Thrift*, all in different encodings.

The main tool for managing such heterogenous infrastructure is *ARM*, the Analytics Resource Manager. This uses as a client library Apache *ZooKeeper*, which is a centralized service for maintaining configuration information, providing distributed synchronization and group services. Unlike with Oozie, where coordination is centralized, in ARM only the state of the nodes is managed centrally, and nodes are activated as soon as they are in a *ready* state.

For internal *Pig* jobs the associated *Oink* scheduler is used. All this, however, does mean that it can be hard to get a complete overview of the workflow. Data storage is based on *HDFS*, but *Vertica* is used for data aggregation, with the results being cached in *MySQL* databases.

Many challenges still remain. Importing logs, for example, which is necessary for generating the *firehouse* service that Twitter offers, still takes about an hour and no real general-purpose solution for real-time processing exists. Currently the tools provided to data scientists are fairly primitive and they usually just have to wait for the output of their *Pig* scripts without getting much intermediate information. Finally, the democratization of resource distribution, i.e., making this fair and transparent, has not yet been fully achieved.

4. CONCLUSION

The presentations and tutorials at SWEET 2012 provided an overview of current developments and emerging issues in the area of both tightly integrated workflow engines and loosely coupled heterogenous frameworks for the execution of data-intensive workflows. These proceedings suggest that while much has been achieved In both areas, this is a still timely and active area of research.

Acknowledgements: We would like to thank the PC members, keynote speakers, authors, local workshop organizers and attendees for making SWEET 2012 a successful workshop. We also express our great appreciation for the support from Google Inc.

5. REFERENCES

[1] Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J Abadi, Alexander Rasin, and Avi Silberschatz. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. *PVLDB*, 2(1):922–933, 2009.

[2] Dominic Battré, Stephan Ewen, Fabian Hueske, Odej Kao, Volker Markl, and Daniel Warneke. Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing. In *Proceedings of the 1st ACM symposium on*

Cloud computing, SoCC '10, pages 119–130, New York, NY, USA, 2010. ACM.

[3] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, Anastasia C Laity, Joseph C Jacob, and Daniel S Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[4] Hugo Hiden, Paul Watson, Simon Woodman, and D. Leahy. e-Science Central: Cloud-based e-Science and its application to chemical property modelling. Technical report cs-tr-1227, School of Computing Science, Newcastle University, 2011.

[5] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1099–1110, New York, NY, USA, 2008. ACM.

[6] Jianwu Wang, Daniel Crawl, and Ilkay Altintas. Kepler + Hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems. In *WORKS '09: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, pages 1–8, New York, NY, USA, 2009. ACM.

[7] Daniel Warneke and Odej Kao. Nephele: efficient parallel data processing in the cloud. In *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, MTAGS '09, pages 8:1—-8:10, New York, NY, USA, 2009. ACM.