A Service for Data-Intensive Computations on Virtual Clusters

Rainer Schmidt, Christian Sadilek, and Ross King
Austrian Research Centers GmbH - ARC
Digital Memory Engineering
Donau-City-Str. 1, 1220 Vienna, Austria
rainer.schmidt@univie.ac.at

Abstract

Digital Preservation deals with the long-term storage, access, and maintenance of digital data objects. In order to prevent a loss of information, digital libraries and archives are increasingly faced with the need to electronically preserve vast amounts of data while having limited computational resources in-house. However, due to the potentially immense data sets and computationally intensive tasks involved, preservation systems have become a recognized challenge for e-science. We argue that grid and cloud technology can provide the crucial technology for building scalable preservation systems. In this paper, we present recent developments on a Job Submission Service that is based on standard grid mechanisms and capable of providing a large cluster of virtual machines. The service allows clients to specify and execute preservation tools on large data sets based on dynamically generated job descriptors. This approach allows us to utilize a cloud infrastructure that is based on platform virtualization as a scaling environment for the execution of preservation workflows. Finally, we present experimental results that have been conducted on the Amazon EC2 and S3 utility cloud infrastructure.

1. Introduction

Due to rapid changes in information technology, a significant fraction of digital data, documents, and records are doomed to become uninterpretable bit-streams within short time periods. The EU project Planets [14] aims to provide a service-based solution to ensure long-term access to the growing collections of digital scientific and cultural assets. Within this project, the Interoperability Framework (IF) provides the technical environment for integrating preservation services, meta-data, and archival storage elements. Components that perform preservation actions often rely on preinstalled tools (e.g. a file format converter) that are wrapped by a service interface on the lowest-layer. The Planets IF workflow engine implements a component-oriented enactor that governs life-cycle operation of the various preservation components, such as instantiation, communication, and data provenance. Distributed preservation workflows are conducted from high-level components that abstract the underlying protocol layers. A crucial aspect of the preservation system is the establishment of a distributed, reliable, and scalable computational tier. A typical preservation workflow may consist of a set of components for data characterization, migration, and verification and may be applied to millions of digital objects. In principle, these workflows could be easily parallelized and run in a massively parallel environment. However, the fact that preservation tools often rely on closed source, third party libraries and applications that often require a platform-dependent and non-trivial installation procedure prevents the utilization of standard high performance computing (HPC) facilities. In order to efficiently execute a preservation plan, a varying set of preservation tools would need to be available on a scalable number of computational nodes. The solution proposed in this paper tackles this problem by incorporating hardware virtualization, allowing us to instantiate sets of transient system images on demand, which are federated as a virtualized cluster. The presented Job Submission Service (JSS) is utilized as the computational tier of a digital preservation system. Jobs are capable of executing data-intensive preservation workflows by utilizing a MapReduce [9] implementation that is instantiated within a utility cloud infrastructure. The presented system is based on the Planets Interoperability Framework, Apache Hadoop [4], and a JSS prototype providing a grid middleware layer on top of the AWS cloud infrastructure. In this paper, we focus on a execution service for preservation tools which relies on standard grid mechanisms and protocols like the Job Submission Description Language [2] (JSDL) and the HPC basic web service profile (HPCBP) [11]. Finally, we present experimental results that have been conducted using the Amazon Simple Storage Service (S3) and Elastic Compute Cloud (EC2) services (AWS) [3]. The paper is organized as

follows: Section 2 provides an overview of related work in the area of cloud and virtual computing, grids, and digital preservation, section 3 presents the Job Submission Service and the prototype implementation, section 4 reports experimental results, and section 5 concludes the paper.

2 Background and Related Work

2.1 Cloud and Virtual Computing

The demand for storage and computational power of scientific computations often exceeds the resources that are available locally. Grid infrastructures, services and remote HPC facilities can provide a viable solution for scientists to overcome these limitations. However, many applications require dedicated platforms or need time-consuming adaptations in order to utilize a remote resource. Virtual machine technology provides software that virtualizes a physical host machine, allowing the deployment of platform-independent system images. The deployment of virtual computer instances is supported by a virtual machine monitor, also called a hypervisor. Cloud systems are consumable via Internet-based services offering IT-technology in the form of applications, hosting platforms, or access to computer infrastructures. Amazon's EC2 and S3 services, one of the most prominent commercial offerings, allow users to rent large computational and storage resources on-demand. EC2 is based on the Xen [25] hypervisor allowing one to prepare and deploy virtual system instances that suit individual application needs. S3 provides access to a global, distributed, and replicated storage system. A detailed evaluation of Amazon's compute, storage, and coordination (SQS) web services and their suitability for scientific computing is given in [17] [22]. Deelman et al. provides cost-based analysis of utilizing the Amazon cloud infrastructure for scientific computing [10]. A proof-of-concept study that runs a complex nuclear physics application on a set of virtual machine nodes is presented in [19]. The Nimbus workspace cloud provides a service to scientific communities allowing the provisioning of customized compute nodes in the form of Xen virtual machines that are deployed on physical nodes of a cluster [21]. A study that compares differences of grid and cloud systems that is based on Amazon's EC2 and S3 services is given in [1]. An experiment were a large set of scanned newspaper articles haven been converted to PDF documents using the Amazon cloud infrastructure has been reported in [8].

2.2 Preservation Systems in Grids

Research in fields like high-energy physics and earth science produce large amounts of irreplaceable data that must be accessed and preserved over time. For example, in earth observation, data is typically geographically dispersed over different archive and acquisition sites, using a multitude of data and meta-data formats [5]. Grid systems provide dependable access and the coordinated resource sharing across different organizational domains [15]. Data grids [26] focus on the controlled sharing and management of large data sets that is distributed over heterogeneous sites and organizations. An important aspect is the storage of data in a reliable, distributed, and replicated way. Digital libraries focus on the creation, discovery, and publication of digital collections. Digital preservation and archiving deals with the management and treatment of large data stores in order to preserve their content over time. Preservation archives are systems that implement long-term preservation managing data integrity and technological evolution. This includes migrating digital objects to new technologies, maintaining their relationships and preservation metadata. Data grids can be used as the underlying technology to implement digital libraries and distributed preservation archives [20]. Computational grid systems provide a complimentary technology and are often combined with data grids. For example, the EGEE project [13], currently the world's largest production grid, provides large quantities of distributed CPUs and petabytes of storage. A survey of initiatives that focus on the integration of emerging technologies like digital libraries, grid, and web services for distributed processing and long-term preservation of scientific knowledge is given in [16]. The Job Submission Service presented in this paper, provides a grid service for digital libraries and preservation archives that allows a client to utilize third party tools based on customized virtual clusters and data intensive computation mechanisms.

3 The Job Submission Service

3.1 Motivation

In the context of grid computing and data grids, digital preservation archives are systems that can preserve the output of computational grid processes [20]. An important issue in the context of preserving existing digital content is the process of deriving metadata from digital assets like file collections in order to extract significant semantic information for their preservation (e.g. format characterization). Decisions in preservation planning [6] rely on information that needs to be generated by algorithms and tools for feature extraction, format identification, characterization, and validation [7]. Migrating digital entities between different formats typically relies on sequential, third party libraries and tools that are not supported by scientific parallel and grid systems. Therefore, we propose a service that employs clusters of customizable virtual nodes in order to overcome these restrictions. The IF JSS implements a grid service that

provides access to a virtual cluster of large numbers of individually tailored compute nodes that can process bulk data based on data-intensive computing mechanisms and that is integratable with computational and data grid systems.

3.2 Web Service Profile

Developing an infrastructure for digital preservation involves many grid-specific aspects including the processing of large volumes of data, conducting experiments in distributed and heterogeneous environments, and executing workflows that cross administrative and institutional boundaries. The service presented in this paper focuses on the aspect of submitting and executing data-intensive jobs as part of a digital preservation infrastructure. In order to be able to take advantage of existing grid solutions and to promote interoperability and integration, the IF JSS service is based on a standard grid service profile (HPCBP) for job scheduling (called the basic HPC use case) that is being well adopted by scientific and industrial systems [23]. The OGF Basic Execution Service (BES) [18] defines Web service interfaces for starting, managing, and stopping computational processes. Clients define computational activities in a grid based on JSDL documents. The OGF HPC Basic Profile (HPCBP) specification defines how to submit, monitor, and manage jobs using standard mechanisms that are compliant across different job schedulers and grid middlewares by leveraging standards like BES, JSDL, and SOAP. Our current implementation provides interfaces that support the BES base case specification and accept JSDL documents that are compliant with the HPCBP profile.

3.3 Basic Service Components

The Job Submission Service (JSS) prototype has been implemented based on a set of exchangeable core components, which are described below. The JSS is a stand-alone Web Service deployed in a Java EE Web Container as shown in Fig. 1. It is secured using HTTPS and SSL/TLS for the transport-layer and WS-Security based on X.509 server certificates and username/password client credentials for the message-layer. In order to submit a request to the JSS, username and password have to be provided that match a previously created account for the institution that utilizes the service. The individual accounts, utilization history, and potentially billing information are maintained by the *Account Manager* component. As HPCBP is used as the web service profile, JSDL documents are used to describe the individual job requests which need to be mapped to physical resources by the resource manager. The *JSDL parser* component validates the XML document and creates an object structure that serves as input for the *Execution Manager*. A *Session Handler* maps service requests based on activity identifiers to physical jobs and keeps track of their current status (e.g. pending, running, finished, failed). The *Execution Manager* interfaces with three components *the Handle Resolver*, *Input Generator*, and *Job Manager* that depend on the resource manager implementation, which is provided by Apache Hadoop in our case. The file handle resolver is used to validate a logical file handle (a URI) and resolve the physical and accessible data reference. The next step is the generation of an input file for a bulk of data that needs to be processed by a parallel application utilizing a particular preservation tool. Finally, the Job Manager prepares a job script and schedules a job using the resource manager.

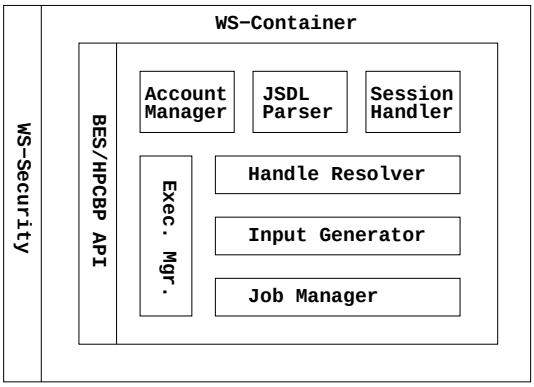

Figure 1. Job Submission Service Components

3.4 Implementation for MapReduce and Amazon's EC2, and S3 Services

The experimental results presented in section 4 have been conducted using an *Execution Manager* implementation for (1) the Hadoop resource manager, (2) Amazon's EC2 compute cloud, and (3) the S3 storage infrastructure. In principle, each of the aforementioned components could be exchanged by different implementations and be connected to different resources, for example a local (e.g Condor [24] based) workstation cluster and network file system. In the following, we describe the functionality of the "cloud-enabled" execution manager. A file handle resolver is used to map a logical handle of a data collection to physical references that are meaningful for the application that needs to access the data (e.g. a file URI, a HTTP URL). Our file handle resolver is implemented in a way that it utilizes the S3 REST-based API to simply generate a list of URIs for files that are contained within an input bucket. The *Input Generator* uses this information to create an input file for the map/reduce application that processes the input data. *MapReduce* is a framework and programming

model that has been introduced by Google to support parallel data-intensive computations. Apache Hadoop is an open source MapReduce implementation that can be used to cluster commodity computers. Also, Hadoop provides built-in support for EC2 and S3. We use Hadoop's own distributed file system to store input files across the computing nodes. The *Job Manager* component passes the input file together with an map/reduce application (the *CommandExecuter*) and information extracted from the JSDL object to the Hadoop job scheduler. The *CommandExecuter* is responsible for handling the S3 bulk data i/o, processing the *input splits* based on pre-installed applications as specified by the user, and for output generation. Finally, the outputs produced by each node are merged to form the output data collection.

3.5 Client Infrastructure and Integration

In the following, we sketch how the execution service is integrated with the existing client infrastructure of the Planets preservation system. However, a detailed discussion of the Planets workflow environment and infrastructure is beyond the scope of this paper. In general, Planets preservation workflows are build from Java components and serialized as XML workflow documents. The client API defines an object model which allows a workflow developer to assemble typical preservation cases from atomic services. Proxy classes that implement the service interfaces are dynamically loaded and provide the required glue code for the invocation of a particular preservation tool or service. Planets services operate upon the concept of digital objects. A digital object holds metadata like descriptive, technical, or preservation information about a digital resource including a handle to the actual data. Digital objects can be passed between different preservation services and point to different types of digital resources (e.g. files, collections, archives). The preservation metadata of processed digital objects needs to be handled on the workflow level and is managed by trusted Java components. For the JSS, a client component abstracts the complexity of interacting with the execution service allowing a client to utilize the JSS within the Planets workflow environment. At runtime, the required information for executing a particular preservation tool on a data resource within the cluster is extracted from the client request allowing a proxy object to invoke the JSS based on a dynamically generated a JSDL document. The Planets registry foundation provides directory services for preservation tools, services, and understood metadata. Therefore, a preservation tool and corresponding services that provide this tool can be discovered at runtime. The JSS provides a generic interface for the parallel processing of bulk data based on an extensible set of different preservation tools. We argue that building such compute services based on virtual images can provide a viable technology for the provisioning of domain-specific applications on a larger scale.

4 Experimental Results

4.1 Preliminary Considerations

The experiments were carried out as a quantitative evaluation of utilizing a virtual, cloud-based infrastructures for executing digital preservation tools. For all experiments, a simple workflow was implemented that migrates one file collection into a new collection of a different format using the `ps2pdf` command-line tool. It is important to note that the selected tool is replaceable and not relevant for the presented experiments. Four dimensions have been analyzed and compared to sequential executions on local execution environments: the execution time, the number of tasks, the number of computing nodes, the physical size of the digital collections to migrate. As performance metrics we calculate Speedup and Efficiency [12] as formally described in equations $S_{s,n}$ (1) and E_p (2).

$$S_{s,n} = Tseq_{s,n}/Tp_{s,n} \tag{1}$$

$$E_p = S_{s,n}/p \tag{2}$$

where:
s - is the physical object size,
n - is the number of tasks,
p - is the number of computing nodes.
$Tseq$ - is the sequential execution time,
Tp - is the execution time with p computing nodes.

4.2 Experiment Setup

For the experiments, we utilized the Amazon Elastic Compute Cloud (EC2) as a cloud infrastructure, leasing up to 150 cluster nodes, each running a custom virtual images based on RedHat Fedora 8 i386, Apache Hadoop 0.18.0, and a set of preinstalled the migration tools. The used default system instances provide one virtual core with one *EC2 Compute Unit*, which is equivalent to the capacity of a 1.0-1.2 GHz 2007 Opteron or a 2007 Xeon processor. Bulk data was stored outside the compute nodes using Amazon's Simple Storage System (S3) due to scale and persistence considerations. We experienced an average download speed from S3 to EC2 of 32.5 MByte/s and an average upload speed from EC2 to S3 of 13.8 MByte/s at the Java level. At the time conducting the presented experiments, the per hour price for an EC2 default instance was $0.10.

4.3 Measurements and Results

For the experiments shown in Fig. 2 we executed all computations on a constant number of five virtual nodes. The number of migration tasks was increased using different sized digital collections to compare the execution time within EC2 to a sequential local execution (SLE) on a single node with identical hardware characteristics. Fig. 2 focuses on the intersection points of the corresponding curves for SLE and EC2 identifying the critical job size for which the parallel execution within EC2 is faster than the sequential execution on a local machine. The results including Speedup and Efficiency for jobs with a large task sizes outside the bounding box of Fig. 2 are shown in table 1. For the experiments shown in Fig. 3 we held the number of tasks constant (migration of a set of one thousand 70kB files) and increased the number of computing nodes form 1 to 150 to evaluate scalability. The values for Speedup, Efficiency and execution time were calculated based on the sequential local execution time for a given parallel job. As shown in table 2, Speedup increases significantly with an increasing number of nodes due to relatively small overheads of the data parallel application model (see 4.4).

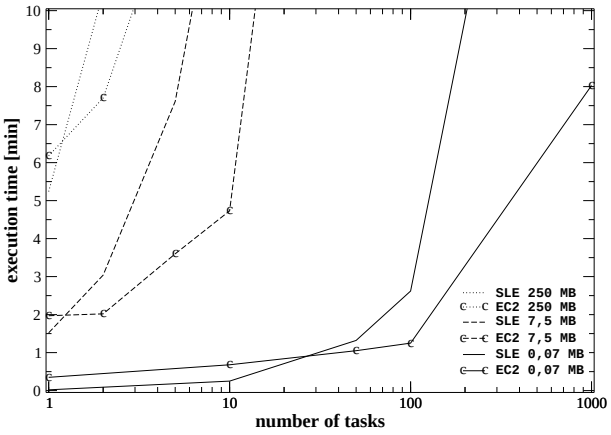

Figure 2. Execution time for an increasing number of migrations tasks and a constant number of computing nodes. The execution on five (EC2) nodes is compared to a sequential local execution (SLE) of the same task.

4.4 Interpretation of Results

Already for a small number of migration tasks the parallel execution within EC2 proved to be faster than the sequential execution on a single node (see Fig. 2). A Speedup

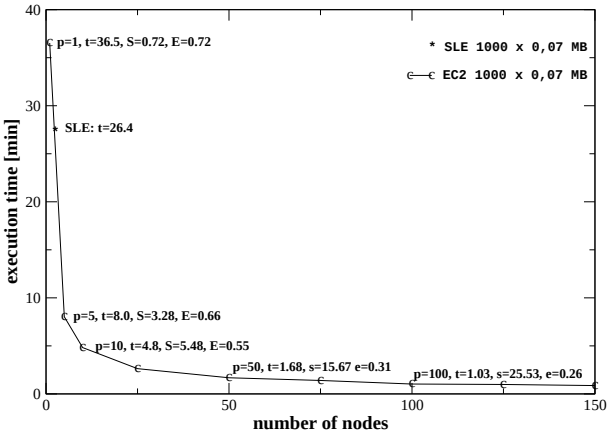

Figure 3. Execution time for 1000 constant migration tasks using an increasing number of computing nodes.

Tasks (n)	Size (s) [MB]	SLE exec. time [min]	EC2 exec. time [min]	$S_{s,n}$	E_p
1000	0.07	26.38	8.03	3.28	0.67
100	7.5	152.17	42.27	3.60	0.72
1000	7.5	1521.67	342.70	4.44	0.88
100	250	523.83	156.27	3.36	0.67
1000	250	5326.63	1572.73	3.37	0.68

Table 1. Results outside the bounding box of Fig. 2 including Speedup and Efficiency

of 4.4 was achieved for 5 nodes with n=1000 and s=7.5 MB (see table 1) proving the suitability and potential of employing (even small) clusters of virtual nodes for digital preservation of large data amounts. Results in Fig. 2 show that the system achieves good scalability when significantly increasing the number of utilized cluster nodes. However, following overheads which affect the efficiency of the described experiments have been identified: (1) Local execution (SLE) vs. cloud-based execution (p=1, n=1000). The master server for the Hadoop distributed file system which is running on a single worker node added 30% (8min) overhead on that node compared to an SLE (26min). We experienced less than 10% overhead introduced by S3 (compared to a local file system). (2) For a larger number of nodes ($p > 50$, n=1000) efficiency decreases for various reasons, e.g. coordination. As all nodes are considered blocked until a job is processed, a large fraction of nodes are idle until the last process has finished. Also for short execution times per node, relatively small overheads like network delays and

Number of nodes (p)	EC2 exec. time [min]	$S_{s,n}$	E_p
1	36.53	0.72	0.72
5	8.03	3.28	0.66
10	4.82	5.48	0.55
25	2.63	10.02	0.40
50	1.68	15.67	0.31
75	1.40	18.84	0.25
100	1.03	25.53	0.26
125	0.98	26.83	0.21
150	0.87	30.44	0.20

Table 2. Results shown in Fig. 3 compared to the sequential local execution of a given job (n=1000, s=0.07 MB) of 26.38 min.

startup time have considerable impact on efficiency.

5 Conclusions

We motivate the integration of on-demand virtual clusters as one convergence path for grid and cloud computing technology. In this paper, we presented an grid execution service that provides data-intensive computing based on customizable virtual nodes as part of a digital preservation infrastructure. We have discussed the main components, the service interface, and it's utilization. Furthermore, the service has been evaluated using Amazon's utility cloud infrastructure. We feel that in the area of digital archives in particular, legal concerns, security policies, and SLAs will require extensive consideration. For future work, we plan to experiment with a cloud-based research hosting infrastructure and distributed resource management.

Acknowledgments

Work presented in this paper is partially supported by European Community under the IST 6th Framework Programme for RTD - Project Planets (IST-033789).

References

[1] An EGEE Comparative Study: Grids and Clouds Evolution or Revolution, 2008. `https://edms.cern.ch/file/925013/3/EGEE-Grid-Cloud.pdf`.

[2] A. Savva et. al. Job Submission Description Language (JSDL) Specification, Version 1.0. Technical report, 2005.

[3] Amazon Web Services. `http://aws.amazon.com`.

[4] Apache Hadoop. `http://hadoop.apache.org/`.

[5] J. V. Bemmelen, L. Fusco, and V. Guidetti. Access to Distributed Earth Science Data Supported by Emerging Technologies. In *EnviroInfo 2005*, September 2005.

[6] CCSDS - Consultative Committee for Space Data Systems. *Reference Model for an Open Archival Information System (OAIS), Blue Book, Issue 1*, January 2002.

[7] C. Chou. Format Identification, Validation, Characterization and Transformation in DAITSS. In *Proc. of IS&T Archiving 2007*, pages 151–156, May 2007.

[8] D. Gottfrid. `http://open.blogs.nytimes.com/2007/11/01`.

[9] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *Proc. of OSDI'04*, 2004.

[10] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: the Montage example. In *In Proceedings of SC'08*, pages 1–12, 2008.

[11] B. Dillaway, M. Humphrey, C. Smith, M. Theimer, and G. Wasson. HPC Basic Profile, v. 1.0. GFD-R-P.114. Technical report, 2007.

[12] D. Eager, J. Zahorjan, and E. D. Lozowska. Speedup Versus Efficiency in Parallel Systems. *IEEE Trans. Comput.*, 38(3):408–423, 1989.

[13] EGEE. `http://project.eu-egee.org`.

[14] A. Farquhar and H. Hockx-Yu. Planets: Integrated Services for Digital Preservation. *International Journal of Digital Curation*, 2(2), 2007.

[15] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[16] L. Fusco, J. van Bemmelen, and V. Guidetti. Emerging technologies in support of long-term data and knowledge preservation for the earth science community. In *PV 2005*.

[17] S. Garfinkel. An Evaluation of Amazons Grid Computing Services: EC2, S3 and SQS. Technical report, 2007.

[18] I. Foster et al. OGSA Basic Execution Service Version 1.0. OGF, GFD-R-P.108, August 2007.

[19] K. Keahey, T. Freeman, J. Lauret, and D. Olson. Virtual Workspaces for Scientific Applications. In *SciDAC 2007 Conference*, June 2007.

[20] R. Moore, A. Rajasekar, and M. Wan. Data Grids, Digital Libraries, and Persistent Archives: An Integrated Approach to Sharing, Publishing, and Archiving Data. *Proceedings of the IEEE*, 93(3):578–588, March 2005.

[21] Nimbus Cloud. `http://workspace.globus.org/clouds/nimbus.html`.

[22] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for science grids: a viable solution? In *DADC '08*, pages 55–64, 2008.

[23] C. Smith, T. Kielmann, S. Newhouse, and M. Humphrey. The HPC Basic Profile and SAGA: Standardizing Compute Grid Access in the Open Grid Forum. `http://www.ogf.org/OGF_Special_Issue/smith-hpc_profile-saga.pdf`.

[24] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[25] The Xen Project. `http://xen.org/`.

[26] S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of Data Grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1):3, 2006.