Adding Quality-Awareness to Evaluate Migration Web-Services and Remote Emulation for Digital Preservation

Christoph Becker[1], Hannes Kulovits[1], Michael Kraxner[1], Riccardo Gottardi[1], Andreas Rauber[1], and Randolph Welte[2]

[1] Vienna University of Technology, Vienna, Austria
http://www.ifs.tuwien.ac.at/dp
[2] University of Freiburg, Germany
http://www.uni-freiburg.de/

Abstract. Digital libraries are increasingly relying on distributed services to support increasingly complex tasks such as retrieval or preservation. While there is a growing body of services for migrating digital objects into safer formats to ensure their long-term accessability, the quality of these services is often unknown. Moreover, emulation as the major alternative preservation strategy is often neglected due to the complex setup procedures that are necessary for testing emulation. However, thorough evaluation of the complete set of potential strategies in a quantified and repeatable way is considered of vital importance for trustworthy decision making in digital preservation planning.

This paper presents a preservation action monitoring infrastructure that combines provider-side service instrumentation and quality measurement of migration web services with remote access to emulation. Tools are monitored during execution, and both their runtime characteristics and the quality of their results are measured transparently. We present the architecture of the presented framework and discuss results from experiments on migration and emulation services.

1 Introduction

Digital library systems today rely on web services and employ distributed infrastructures for accomplishing increasingly complex tasks and providing shared access to distributed content. The amount and the heterogeneity of content that is contained in these digital libraries is increasing rapidly; the new wave of web archives being built around the world will further accelerate this development in the near future.

However, this content is inherently ephemeral, and without concrete measures it will not be accessible in the very near future, when the original hardware and software environments used to create and render the content are not available any more. Thus, digital preservation tools have become a major focus of research. The two principle approaches to preservation are migration and emulation. Migration transforms the representation of a digital object by converting it to a different format that is regarded as safer, more stable and thus better suited for long-term access. Emulation simulates the original technical environment in which the object is known to function. The optimal treatment for a specific object depends on numerous factors ranging from technical aspects such as scalability, performance and platform independence, user needs to access objects, with respect to utilizing the object and familiarity with certain software environments to the concrete qualities of migration tools and the importance of authenticity and trust in these qualities. The selection process for one particular course of action is a key question of *preservation planning.*

To enable accountable decision making in preservation planning, we need to rely not on subjective judgement of the quality and suitability of a potential preservation action, but instead on repeatable and transparent measurements that can be considered a solid evidence base. These measurements can be on the one hand obtained through emerging community efforts such as the Planets Testbed[1], which is building up an infrastructure for repeatable tests and accumulating institution-independent evidence on preservation action services. This approach can provide a fundamental body of knowledge for digital preservation research and practice. However, it will usually not take into account the specific peculiarities of a planning scenario in an organisation. Thus, measurements specific to the situation at hand are used during the preservation planning procedure by applying potential actions to the specific digital objects at hand and measuring the outcomes. Service-oriented computing as means of arranging autonomous application components into loosely coupled networked services has become one of the primary computing paradigms of our decade. Web services as the leading technology in this field are widely used in increasingly distributed systems. Their flexibility and agility enable the integration of heteregoneous systems across platforms through interoperable standards. However, the thus-created networks of dependencies also exhibit challenging problems of interdependency management. Some of the issues arising are service discovery and selection, or the question of service quality and trustworthiness of service providers. Of core interest is the problem of measuring quality-of-service (QoS) attributes and using them as means for guiding the selection of the optimal service for consumption at a given time and situation.

Migration as the conversion of content to a different representation is a straightforward in/out operation. It lends itself naturally for being made accessible as a web service at least for evaluation purpose when setting up all potential migration tools may be too costly or complex. (We focus primarily on evaluation phase. For production level deployment tools will usually be installed locally for data protection and performance reasons.) Emulation as the second major preservation strategy is not as readily convertible. While considerable work has been presented on services for migrating content, emulation is still mostly used in non-distributed environments. In many cases, the effort needed just to *evaluate* an emulation software is viewed as prohibitive due to the often complex setup procedures that are required for rendering a single file.

Even considering migration services, there is still a severe lack regarding the modelling and measurement of quality-of-service (QoS). When deciding which migration path to take and which tool to use in a repository, these quality aspects are of vital importance. Tool characteristics such as the processing time and memory needed to convert digital content to different formats are an important criterion guiding the evaluation of a migration tool; even more important is often the *quality* of transformation, i.e. a quantified answer to the question 'How much information was *lost* during this transformation?'.

To obtain answers to these questions of quality is a necessary precursor to the deployment of any tool on a digital repository system. The preservation planning process described in [12,4] relies on controlled experimentation to obtain measurements of the actual qualities of different preservation action tools. While it is possible to carry out these experiments on dedicated infrastructures, it is also very time-consuming and effort-intensive. Distributed services greatly speed up this process by providing action services that can be directly invoked inside the preservation planning application *Plato*[1].

However, measuring quality attributes of web services is inherently difficult due to the very virtues of service-oriented architectures: The late binding and flexible integration ideals ask for very loose coupling, which often implies that little is known about the actual quality of services and even less about the confidence that can be put into published service metadata, particularly QoS information. Ongoing monitoring of these quality attributes is a key enabler of reliable preservation services and a prerequisite for building confidence and trust in services to be consumed.

This paper presents a generic architecture and reference implementation for non-invasively measuring quality and performance of migration tools and instrumenting the corresponding web services on the provider side. We demonstrate how a service wrapper that makes a tool installed on a specific machine accessible as a web service can monitor the execution of this tool and provide additional benchmark information as metadata together with the actual transformation result. We investigate different profiling approaches for this purpose and analyse benefits and challenges. We further describe a web service engine consisting of a registry which facilitates service discovery, metadata schemas and interfaces, and services for migration and emulation, and demonstrate how these components can be integrated through a distributed service oriented preservation planning architecture. Remote emulation is integrated into this framework by providing transparent access to virtualised emulators on remote host machines through the standard web browser interface.

The rest of this paper is structured as follows. The next section outlines related work in the areas of web service QoS modelling, performance measurement, and distributed digital preservation services. Section 3 describes the overall architectural design, the remote emulation framework and the monitoring engines, while Section 4 analyses the results of practical applications of the implemented framework. Section 5 discusses implications and sets further directions.

[1] `http://www.ifs.tuwien.ac.at/dp/plato`

2 Related Work

The initially rather slow takeup of web service technology has been repeatedly attributed to the difficulties in evaluating the quality of services and the corresponding lack of confidence in the fulfillment of non-functional requirements. The lack of QoS attributes and their values is still one of the fundamental drawbacks of web service technology [11,10].

Web service selection and composition heavily relies on QoS computation [9,6]. A considerable amount of work has been dedicated to modelling QoS attributes and web service performance, and to ranking and selection algorithms. A second group of work is covering infrastructures for achieving trustworthiness, usually by extending existing description models for web services and introducing certification roles to the web service discovery models. Most of these approaches assume that QoS information is known and can be verified by the third-party certification instance. While this works well for static quality attributes, variable and dynamically changing attributes are hard to compute and subject to change. Platzer et al. [10] discuss four principle strategies for the continuous monitoring of web service quality: provider-side instrumentation, SOAP intermediaries, probing, and sniffing. They further separate performance into eight components such as network latency, processing and wrapping time on the server, and round-trip time. While they state the need for measuring all of these components, they focus on round-trip time and present a provider-independent bootstrapping framework for measuring performance-related QoS on the client-side.

Wickramage et al. analyse the factors that contribute to the total round trip time (RTT) of a web service request and arrive at 15 components that should ideally be measured separately to optimize bottlenecks. They focus on web service frameworks and propose a benchmark for this layer [13]. Woods presents a study on migration tools used for providing access to legacy data, analysing the question of migration-on-request versus a-priori migration from a performance perspective [14]. Clausen reports on experiments for semi-automated quality assurance of document conversions by comparing attributes such as page count and differences in rendered images of the contained pages [5].

To support the processes involved in digital preservation, current initiatives are increasingly relying on distributed service oriented architectures to handle the core tasks in a preservation system [8,7,3]. The core groups of services in such a digital preservation system are

– the identification of object types and formats,
– analysis and characterisation of digital objects,
– the transformation of content to different representations, and
– the validation of the authenticity of objects in different formats [2].

In the Planets preservation planning environment, planning decisions are taken following a systematic workflow supported by the web-based planning application *Plato* which serves as the frontend to a distributed architecture of preservation services [4]. The architecture enables flexible integration of migration services from different sources and service providers [3]. However, emulation so far had to be carried out manually in an external environment.

The work presented here builds on this architecture and takes two specific steps further. We present a migration and emulation engine for provider-side monitoring and QoS-aware provisioning of migration tools wrapped as web services. In this architecture, processing time and resource usage of migration tools are monitored transparently and non-invasively on a variety of different platforms. These runtime characteristics are combined with an assessment of migration quality using characterisation functionality of tools such as ImageMagick and emerging approaches such as the eXtensible Characterisation Languages (XCL) for comparing original and transformed objects. The quality information obtained thereby is provided to the client together with the migration result. In parallel, the engine enables users to render digital objects from their own environment transparently in their browser using the remote emulation framework GRATE.

3 A Preservation Action Monitoring Infrastructure

3.1 Measuring QoS in Web Services

Figure 1 shows a simplified abstraction of the core elements of the monitoring design and their relations. The key elements are `Services`, `Engines`, and `Evaluators`, which are all contained in a `Registry`. Each `Engine` specifies which aspects of a service it is able to measure in its `MeasurableProperties`. The property definition includes the scale and applicable metrics for a property, which are used for creating the corresponding `Measurements`.

Each `Engine` is deployed on a specific hardware `Environment` that shows a certain performance. This performance is captured by the score of a `Benchmark` which is a specific configuration of services and `Data`, aggregating measurements over these data to produce a representative *score* for an environment. The benchmark scores of the engines' environments are provided to the clients as part of the service execution metadata and can be used to normalise performance data of migration tools running on different hardware platforms.

The `Services` contained in a registry are not invoked directly, but run inside a monitoring engine to enable performance measurements. This monitoring accumulates `Experience` for each service, which is collected in each successive call to a service and used to aggregate information over time. It thus enables continuous monitoring of performance and migration quality.

`CompositeEngines` are a flexible form of aggregating measurements obtained in different monitoring environments. This type of engine dispatches the service execution dynamically to several engines to collect information. This is especially useful in cases where measuring code in real-time actually changes the behaviour of that code. For example, measuring the memory load of Java code in a profiler usually results in a much slower performance, so that simultaneous measurement of memory load *and* execution speed leads to skewed results. Fortunately, in this case there is a way around this uncertainty relation – forking and distributing the execution leads to correct results.

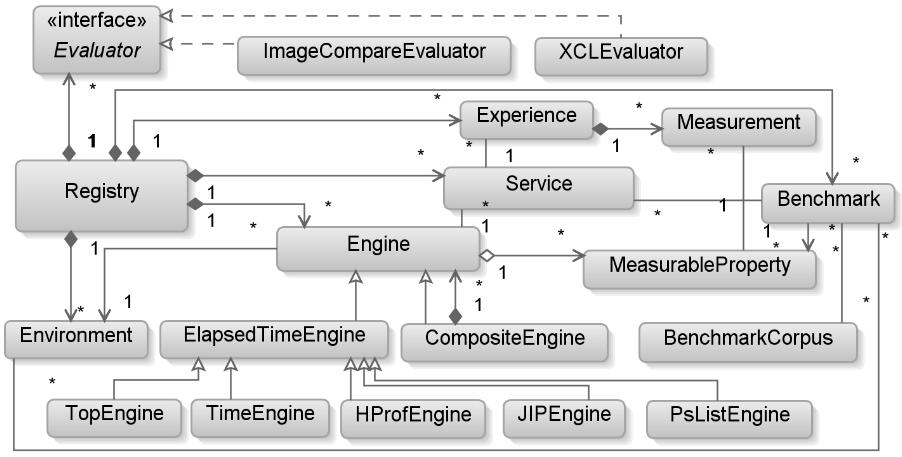


Fig. 1. Core elements of the monitoring framework

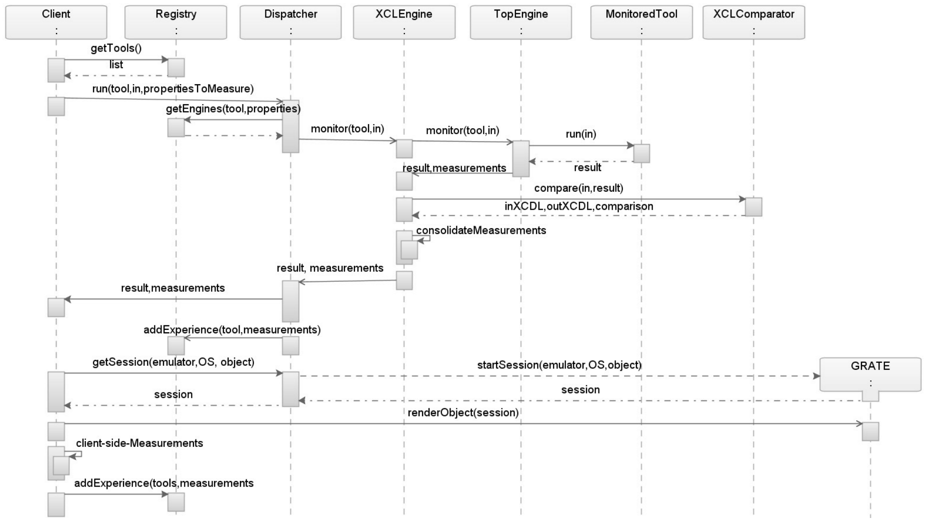


Fig. 2. Exemplary interaction in an evaluation scenario

The bottom of the diagram illustrates some of the currently deployed monitoring engines. Additional engines and composite engine configurations can be added dynamically at any time. The `ElapsedTimeEngine` is a simple default implementation measuring wall-clock time. Several engines have been implemented to support performance measurement of migration tools on different operating systems and platforms. The `TopEngine` is based on the Unix tool *top* and used for measuring memory load and execution speed of wrapped applications installed on the server. The `TimeEngine` uses the Unix call *time* to measure the

CPU time used by a process. On Windows servers, the `PsListEngine` relies on *PsList*[2]. Monitoring the performance of Java tools is accomplished in a platform-indepent manner by a combination of the `HProfEngine` and `JIPEngine`, which use the *HPROF*[3] and *JIP*[4] profiling libraries for measuring memory usage and timing characteristics, respectively.

In contrast to performance-oriented monitoring through engines, `Evaluators` are used for comparing input and output of migration tools to compute similarity measures and judge migration quality. The `ImageCompareEvaluator` relies on the *compare* utility of ImageMagick[5] to compute distance metrics for pairs of images. The more generic `XCLEvaluator` uses the eXtensible Characterisation Languages (XCL) which provide an abstract information model for digital content which is independent of the underlying file format [2], and compares different XCL documents for degrees of equality.

Figure 2 illustrates an exemplary simplified flow of interactions between service requesters, the registry, the engines, and the monitored tools, in the case of an engine measuring the execution of a migration tool through the Unix tool *top*. The dispatcher is responsible for selecting the appropriate engine for a given tool and set of properties to measure; it calls all configured evaluators after the successful migration. Both the dispatcher and the client can contribute to the accumulated experience of the registry. This allows the client to add round-trip information, which can be used to deduct network latencies, or quality measurements computed on the result of the consumed service. A single-use key prevents spamming of the experience base. The bottom of the sequence shows in a simplified form how an `EmulationService` connects the planning client to the remote emulation services provided by *GRATE*. The connector uploads the content to be rendered and provides the client with the obtained session key, i.e. a URL for gaining access to an emulator running on a remote machine via a web browser. Section 3.2 will explain this mechanism in detail.

3.2 GRATE: Global Remote Access to Emulation

GRATE is a webservice written in Java/PHP/Perl and JavaScript (Ajax) and allows for location-independent remote access to designated emulation-services over the Internet. Figure 3 shows a high-level overview of the main components of the distributed emulation service infrastructure. Not shown in this diagram are the planning tool and emulation connector services described above.

The GRATE client consists of two components, the GRATE Java applet and a Java Tight VNC client applet, embedded in PHP/JavaScript/Ajax code. Tight VNC is used for remote desktop access to the emulator[6]. Since Java applets are platform independent, every Java-enabled web-browser is suitable for running

[2] `http://technet.microsoft.com/en-us/sysinternals/bb896682.aspx`
[3] `http://java.sun.com/developer/technicalArticles/Programming/HPROF.html`
[4] `http://jiprof.sourceforge.net/`
[5] `http://www.imagemagick.org/Usage/compare/`
[6] `http://www.tightvnc.com/`

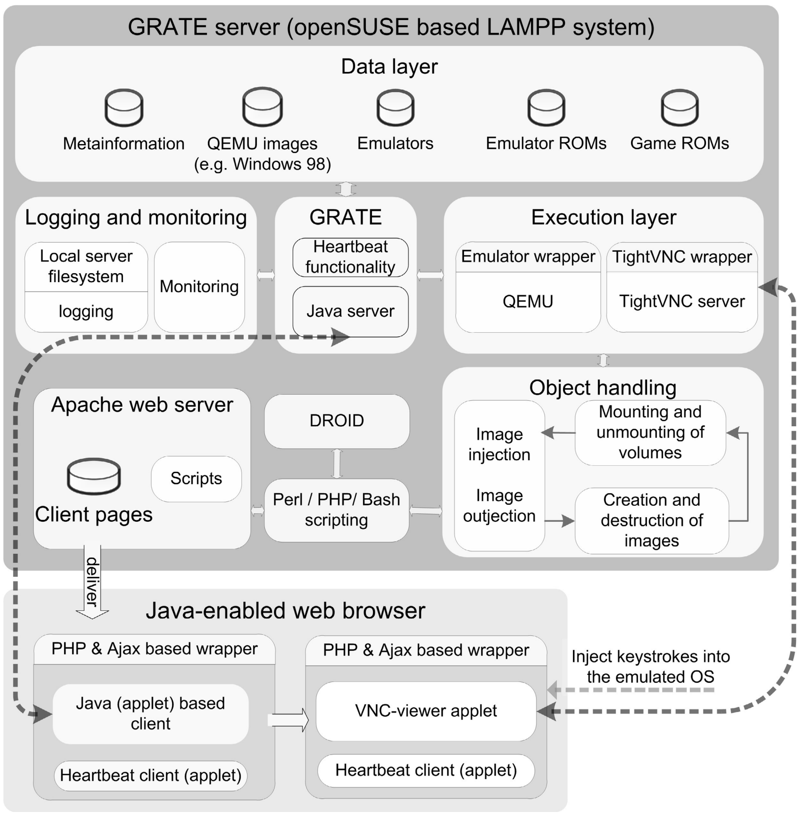


Fig. 3. GRATE architecture

the GRATE client. This client communicates with the GRATE server component, which is responsible for session management (establishing and terminating VNC sessions, executing emulators, delivering meta-information, etc.) as well as transporting uploaded digital objects into the emulated environments. GRATE is Java-based and therefore portable. It is currently running on a Linux-Apache-MySQL-Perl-PHP (LAMPP) system.

Client-server communication takes place via TCP; it is possible to input key commands into the browser, which are remotely injected into the running emulator. Injecting digital objects to be rendered is accomplished by mounting virtual drives containing the objects to be rendered. The actual emulated image, e.g. of a Windows 95 installation, then contains a listener which automatically opens the encountered object. Table 1 gives an overview of some of the formats currently supported by the Windows 98 images running on QEMU[7].

[7] Note that for some formats such as RAW, only specific camera profiles are supported, while EXE and DLL means that the contained applications can be displayed. For video formats, a number of codecs are currently installed, but not listed here.

Table 1. Formats supported by the Windows images currently deployed in GRATE

Video/Audio	Images	Documents
AIF, AU, SND, MED, MID, MP3, OGG, RA, WAV, ASF, AVI, MOV, MP4, MPG, MPEG, WMA, WMV	ANI, CUR, AWD, B3D, BMP, DIB, CAM, CLP, CPT, CRW/CR2, DCM/ACR/IMA, DCX, DDS, DJVU, IW44, DXF, DWG, HPGL, CGM, SVG, ECW, EMF, EPS, PS, PDF, EXR, FITS, FPX, FSH, G3, GIF, HDR, HDP, WDP, ICL, EXE, DLL, ICO, ICS, IFF, LBM, IMG, JP2, JPC, J2K, JPG, JPEG, JPM, KDC, LDF, LWF, Mac PICT, QTIF, MP4, MNG, JNG, MRC, MrSID, SID, DNG, EEF, NEF, MRW, ORF, RAF, DCR, SRF/ARW, PEF, X3F, NLM, NOL, NGG, PBM, PCD, PCX, PDF, PGM, PIC, PNG, PPM, PSD, PSP, PVR, RAS, SUN, RAW, YUV, RLE, SFF, SFW, SGI, RGB, SIF, SWF, FLV, TGA, TIF, TIFF, TTF, TXT, VTF, WAD, WAL, WBMP, WMF, WSQ, XBM, XPM	PDF, ODT, OTT, SXW, STW, DOC, DOCX, DOT, TXT, HTML, HTM, LWP, WPD, RTF, FODT, ODS, OTS, SXC, STC, XLS, XLW, XLT, CSV, ODP, OTP, SXI, STI, PPT, PPS, POT, SXD, ODG, OTG, SXD, STD, SGV

Table 2. Supported in/out formats of currently integrated migration tools

FLAC	LAME	OpenOffice.org	GIMP	ImageMagick	JavaIO
WAV, FLAC	WAV, MP3	PPT, SXI, SDX, ODP, ODG, ODT, SXW, DOC, RTF, TXT, ODS, SXC, XLS, SLK, CSV, ODG, SXD	BMP, JPG, PNG, TIFF	BMP, GIF, JPG, PNG, ICO, MPEG, PS, PDF, SVG, TIFF, TXT	BMP, GIF, JPG, PNG, TIFF

This combination of virtual machine allocation on a pre-configured server with remote access to the emulators reduces the total amount of time needed for evaluating a specific emulation strategy from many hours to a single click.

4 Experiment Results

We run a series of experiments to evaluate different aspects of both the tools and the engines themselves: We compare different performance measurement techniques; we demonstrate the accumulation of average experience on tool behaviour; and we compare performance and quality of image conversion tools.

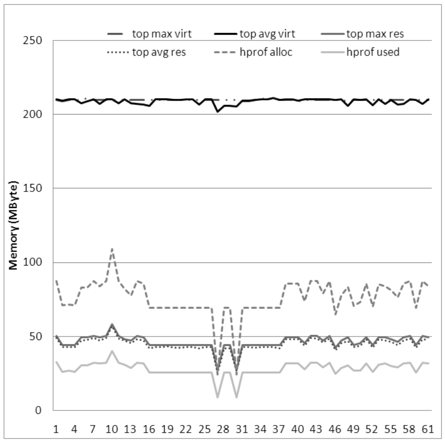
Table 2 gives an overview of currently deployed migration tools and supported formats[8].

4.1 Measurement Techniques

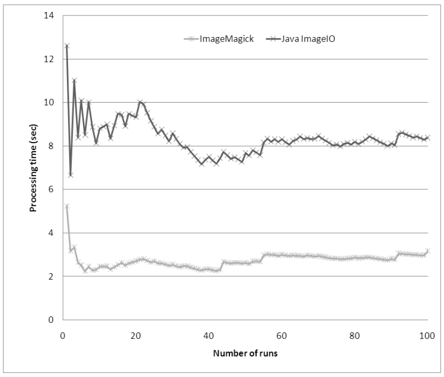
The first set of experiments compares the exactness and appropriateness of performance measurements obtained using different techniques and compares these values to check for consistency of measurements. We monitor a Java conversion tool using all available engines on a Linux machine. The processing time measured by top, time, and the JIP profiler are generally very consistent, with an empirical correlation coefficient of 0.997 and 0.979, respectively. Running HProf on the same files consistently produces much longer execution times due to the processing overhead incurred by profiling the memory usage.

Figure 4(a) shows measured memory values for a random subset of the total files to visually illustrate the variations between the engines. The virtual memory assigned to a Java tool depends mostly on the settings used to execute the JVM and thus is not very meaningful. While the resident memory measured by Top includes the VM and denotes the amount of physical memory actually used

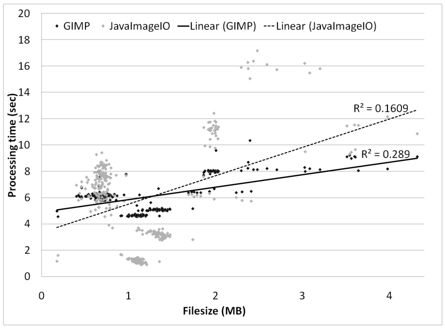
[8] To save space we list only the format's file suffix. For a detailed description of the supported format versions, please refer to the web page of the respective tool.



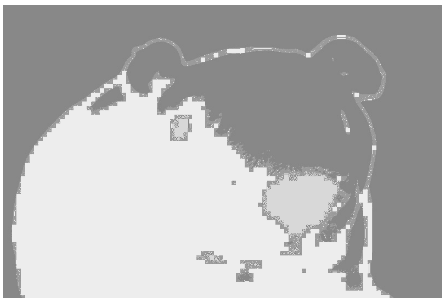
(a) Comparison of techniques for measuring memory



(b) Accumulated average processing time per MB



(c) Runtime behaviour of two conversion services



(d) Visualisation of an examplary conversion error

Fig. 4. Experiment results

during execution, HProf provides figures for memory used and allocated within the VM. Which of these measurements are of interest in a specific preservation planning scenario depends on the integration pattern. For Java systems, the actual memory within the machine will be relevant, whereas in other cases, the virtual machine overhead has to be taken into account as well.

When a tool is deployed as a service, a standard benchmark score is calculated for the server with the included sample data; furthermore, the monitoring engines report the average system load during service execution. This enables normalisation and comparison of a tool across server instances.

4.2 Accumulated Experience

An important aspect of any QoS management system is the accumulation and dissemination of experience on service quality. The described framework automatically tracks and accumulates all numeric measurements and provides aggregated averages with every service response. Figure 4(b) shows how processing time per MB quickly converges to a stable value during the initial bootstrapping sequence of service calls on benchmark content.

4.3 Tool Performance and Quality

Figure 4(c) shows the processing time of two conversion tools offered by the same service provider on 312 JPEG 1.01 image files. Simple linear regression shows the general trend of the performance relation, revealing that the Java tool is faster on smaller images but outperformed on larger images. It has to be noted that the *conversion quality* offered by GIMP is certainly higher. Figure 4(d) shows a visualisation of the conversion error introduced by the simple Java program when converting an image with transparent background from GIF to JPG. While changed pixels are shown in grey in the figure, the darker parts indicate the transparent layer that has been lost during migration. ImageMagick *compare* reports that 59.27% of the pixels are different in the migrated file, with an RMSE of 24034. In most cases, the information loss introduced by a faster tool will be considered much more important than its speed.

A comparison of the different evaluation results shows that the ImageMagick `compare` tool reports several errors when converting jpg to tiff with GIMP. The reason is that GIMP in the deployed configuration incorporates the EXIF orientation setting into the migrated picture, while ImageMagick fails to do so. While it is often possible to tweak the behaviour of tools through appropriate configuration, cases like this illustrate why there is a need for *independent* quality assurance in digital preservation: A tool must not judge its own migration quality, but instead be evaluated by an independent method. Current efforts such as the XCL languages strive to deliver this QA in the future.

5 Discussion and Conclusion

We described a preservation action monitoring infrastructure combining monitored migration web services with remote emulation. Quality measurements of migration services are transparently obtained through a flexible architecture of non-invasive monitoring engines, and the services are instrumented by the provider to deliver this quality information to the client. These quality measures include performance figures from engines as well as similarity measures obtained by evaluators by comparing input and result of migration tools. We demonstrated the performance monitoring of different categories of applications wrapped as web services and discussed different techniques and the results they yield.

The resulting measurements are of great value for transparent and reliable decision making in preservation planning. Together with the ease of access that is provided through the remote emulation system GRATE, the effort needed for planning the preservation of digital collections can be greatly reduced. Part of our current work is the introduction of flexible benchmark configurations that support the selection of specifically tailored benchmarks, e.g. to calculate scores for objects with certain characteristics; and the extension of measurements to arrive at a fully trackable set of measurements covering the entire range of factors that influence decision making in digital preservation.

Acknowledgements

Part of this work was supported by the European Union in the 6th Framework Program, IST, through the PLANETS project, contract 033789.

References

1. Aitken, B., Helwig, P., Jackson, A., Lindley, A., Nicchiarelli, E., Ross, S.: The Planets Testbed: Science for Digital Preservation. Code4Lib 1(5) (June 2008), `http://journal.code4lib.org/articles/83`
2. Becker, C., Rauber, A., Heydegger, V., Schnasse, J., Thaller, M.: Systematic characterisation of objects in digital preservation: The extensible characterisation languages. Journal of Universal Computer Science 14(18), 2936–2952 (2008)
3. Becker, C., Ferreira, M., Kraxner, M., Rauber, A., Baptista, A.A., Ramalho, J.C.: Distributed preservation services: Integrating planning and actions. In: Christensen-Dalsgaard, B., Castelli, D., Ammitzbøll Jurik, B., Lippincott, J. (eds.) ECDL 2008. LNCS, vol. 5173, pp. 25–36. Springer, Heidelberg (2008)
4. Becker, C., Kulovits, H., Rauber, A., Hofman, H.: Plato: A service oriented decision support system for preservation planning. In: Proc. JCDL 2008, Pittsburg, USA. ACM Press, New York (2008)
5. Clausen, L.R.: Opening schrödingers library: Semi-automatic QA reduces uncertainty in object transformation. In: Kovács, L., Fuhr, N., Meghini, C. (eds.) ECDL 2007. LNCS, vol. 4675, pp. 186–197. Springer, Heidelberg (2007)
6. Dustdar, S., Schreiner, W.: A survey on web services composition. International Journal of Web and Grid Services 1, 1–30 (2005)
7. Ferreira, M., Baptista, A.A., Ramalho, J.C.: An intelligent decision support system for digital preservation. International Journal on Digital Libraries 6(4), 295–304 (2007)
8. Hunter, J., Choudhury, S.: PANIC - an integrated approach to the preservation of complex digital objects using semantic web services. International Journal on Digital Libraries 6(2), 174–183 (2006)
9. Menascé, D.A.: QoS issues in web services. IEEE Internet Computing 6(6), 72–75 (2002)
10. Platzer, C., Rosenberg, F., Dustdar, S.: Enhancing Web Service Discovery and Monitoring with Quality of Service Information. In: Securing Web Services: Practical Usage of Standards and Specifications. Idea Publishing Inc. (2007)
11. Ran, S.: A model for web services discovery with QoS. SIGecom Exch. 4(1), 1–10 (2003)
12. Strodl, S., Becker, C., Neumayer, R., Rauber, A.: How to choose a digital preservation strategy: Evaluating a preservation planning procedure. In: Proc. JCDL 2007, pp. 29–38. ACM Press, New York (2007)
13. Wickramage, N., Weerawarana, S.: A benchmark for web service frameworks. In: 2005 IEEE International Conf. on Services Computing, vol. 1, pp. 233–240 (2005)
14. Woods, K., Brown, G.: Migration performance for legacy data access. International Journal of Digital Curation 3(2) (2008)