Testing the Scalability of a DSpace-based Archive

Dharitri Misra, James Seamans, George R. Thoma, National Library of Medicine, Bethesda, Maryland, USA

Abstract

The implementation of production-level large scale archives is often based on research prototypes that possess essential functions and characteristics, e.g., storage capacity, ingest, metadata recording, ability to migrate to newer formats, etc. However, a key characteristic that is often overlooked is scalability, i.e., the ability of the system to accommodate large numbers of items without compromising performance - while ingesting, indexing or access.

*Here we describe an investigation of archive scalability in a Java-based system (*System for the Preservation of Electronic Resources *or SPER) which was built by an R&D team at the U.S. National Library of Medicine to investigate various aspects of digital preservation. SPER uses DSpace as the underlying infrastructure for building and managing the digital archive. To confirm the capability of SPER/DSpace to serve as a large archive, we conducted scalability tests by generating and ingesting data for more than a million items, and studied ingest behavior as a function of the archive size.*

This paper describes the test procedure and environment, the software developed to measure performance during ingest, and the characteristics of the ingested data. We present the ensuing results, which confirm the scalability of SPER/DSpace with acceptable ingest performance as the archive is expanded to a million items.

Introduction

SPER is an evolving Java-based system to research digital preservation functions and capabilities, including automated metadata extraction (AME) for documents, retrieval of available metadata from external databases, document archiving, and ensuring long term use through bulk file format migration [1]. SPER is built upon MIT's DSpace software (Version 1.4) [2], with some modifications and enhancements to better facilitate batch-based processing. It uses a Java RMI-based Client-Server model and runs on Windows platforms, using MySQL (Version 5.02 or higher) database, but may also be run on Solaris systems. The architecture of SPER is shown in Figure 1.

As SPER relies upon the DSpace infrastructure to build and manage its archive, to determine the scalability of SPER, it was imperative to examine the scalability of DSpace itself. DSpace is an open-source, OAIS-compliant digital repository system used by about than 300 universities and other organizations worldwide [3]. However, no actual data has been found on the archive size of these installations, neither is any benchmark available on DSpace performance. (One installation at Cambridge University had reported that ingesting new items to DSpace was too slow beyond one hundred thousand, but no details were given on the operational environment or other conditions.)

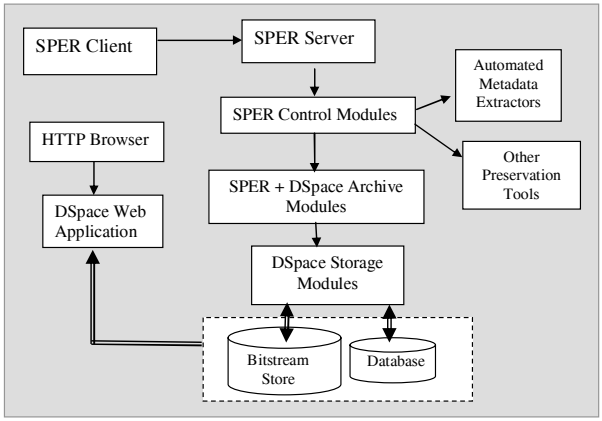

Figure 1 - SPER Architecture with DSpace Archiving Layer

So for our test, we built a DSpace archive with more than one million items (individual retrievable objects) and measured the scalability of the system in terms of the ingest rate and other parameters as the archive grew in size. The details of the experiment and ensuing results are presented in the following sections. (Note that MySQL database system used in our DSpace installation does not inherently limit scalability, as MySQL table size is limited only by the host platform/operating system.)

Test Components

The components relevant to our scalability test include the ingest data, archival storage, databases, special test software, as well as the test platform where the tests were performed. These components are described below:

Ingest Data

For practicality, the ingest data, comprising more than a million individual documents and descriptive metadata, was simulated by replicating the data we had previously archived using SPER [4] , as follows:

- The input data for simulation were from two FDA Notice of Judgment collections [5], named FDNJ (Food and Drug-related Notice of Judgment) and DDNJ (Drug-related Notice of Judgment) which were already stored in an operational SPER system. The number of items (called NJs) in these two collections was approximately 17,700 and 220 respectively.
- New collections were created by copying all source data (TIFF images and OCR text) and metadata associated with each NJ of an input collection.
- Unique identifiers (UI) and titles were created for the replicated items by prefixing the original item's UI and title (stored in its metadata file) with the new collection name, and updating the metadata file. This helped later in the visual verification of ingest during item retrieval.

Data Characteristics

Each ingested item represents one NJ record, which consists of three or more associated bitstreams, as follows:

- Monochrome TIFF image(s) of the original published page(s) containing the NJ description text
- Dublin Core metadata file
- SPER-generated preservation metadata file
- OCR textline file of the NJ

Each TIFF is a CCITT Group 4 Fax compressed image, and averages 100 KB in size. Approximately 50% of items have only one associated TIFF image, 25% have two images, and 2% have 10 images. In the DDNJ collection three items also have 47, 57 and 100 associated images. Each OCR textline file contains text for the entire NJ as converted by the OCR engine in SPER. About 80% of items have this associated text file.

The combined size of the last three text files, on the average, is 10 KB, and they are stored as three separate bitstreams for each item in the archive.

Archive Contents

The two FDA collections were replicated to create and populate the DSpace archive as follows:

- Number of DSpace Communities: 32
- Number of Collections: 109
- Number of Collections per Community: varied between 1 to 5
- Number of Ingested Items: 1,041,790
- Number of Items with 4, 5 and 10 associated bitstreams: 541442, 267104 and 1166 respectively
- Number of Items with 50, 60 and 103 bitstreams: 58 each
- Number of searchable fields in Dublin Core file of an item: 12

The contents of the searchable metadata fields are indexed using the Apache Lucene software during ingest.

Databases

The two databases, used in the test, are:

- ***Ingest Database:*** This is the standard DSpace database used for recording information related to each item in the archive. In SPER, it is implemented as a MySQL database.
- ***Ingest Performance Database:*** This is another MySQL database, distinct from the above, which is implemented specifically for our scalability test. Detailed information on each ingested item is extracted from the DSpace log file and loaded into this database as a table.

Operations Platform and Software

The Ingest performance measurement task was conducted on a dedicated virtual machine on a Sun Microsystems X4500 server. The X4500 features dual core 2.8GHz AMD Opteron processors, with 16GB of memory, 24 TB of disk space and four 1 Gb/s network interfaces. The virtual partition was allocated 4.5TB of dedicated disk space. The Java application performing ingest used 3 GB of pre-allocated memory. The other main tasks running on this virtual machine are the MySQL database server, and Apache Tomcat Web services needed for accessing the ingested items through a Web browser. Note that the directory with ingest input, and the archive upload area reside on the same disk as the archive, making faster data transfer during ingest.

The test was conducted using the following versions of the underlying operating system/software:

- Sun Solaris 10
- DSpace: V1.4.1
- MySQL: V5.01 with Connector-j 5.0.7
- Java: V1.4.2

Test Procedure and Software

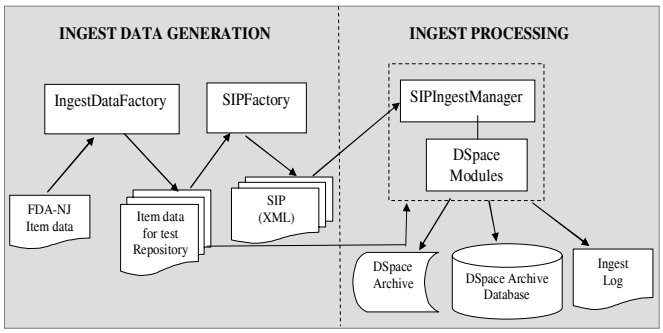

Figure 2 – SIP generation and Ingest of simulated FDA-NJ collections

The steps used to generate and ingest the data, and the software modules to perform them are shown in Figure 2. Retrieval of performance information and generation of corresponding charts are shown in Figure 3. These steps are explained below:

1. In "Ingest Data Generation" shown in Figure 2, the module *IngestDataFactory* uses the FDNJ and DDNJ collection items (images, metadata, and OCR text) stored in specified input directories, and generates replicated data with unique identifiers into output directories, corresponding to new DSpace Communities and Collections. The module *SIPFactory* then creates OAIS-type Submission Information Packages (SIPs) with these items, in the form of several XML files, each SIP specifying all items to be ingested in one batch.
2. During "Ingest Processing", The *SIPIngestManager* ingests a batch of items specified in a SIP to the DSpace archive, using the DSpace library modules, and automatically creating new communities and collections in the database, as required. It accesses the resource files indicated for each item in the SIP, and ingests the items individually to the DSpace archive. (Note that this is somewhat different from the DSpace command-line batch ingest operation, where all the items to be ingested are assumed to be in a single top level directory, and all resources for an item must be in a single directory.

   The SIPIngestManager records necessary information such as the data upload time, ingest time, number of ingested files, size of ingested data etc. for each archived item into a log file (the DSpace log file) for later analysis.
3. This performance-related data in the log file is then analyzed, parsed and formatted by a set of *Perl scripts,* and loaded to tables in the Ingest Performance database, as shown in Figure 3.

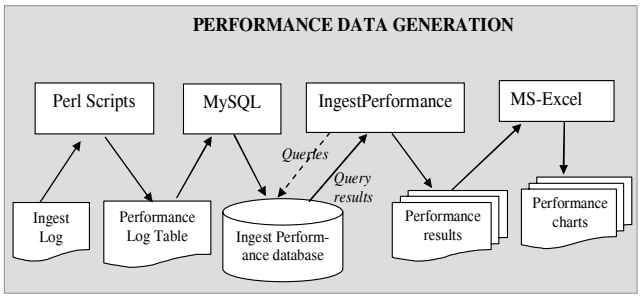

Figure 3 – Generation of performance charts from Ingest log

4. Finally, the *IngestPerformance* module accesses the Ingest Performance database, and gathers necessary performance-related data as a function of the archive size (represented as the number of items already ingested to the archive), through SQL queries. This data is then output to formatted text files, which are input to an Excel plotting package.

Operations

To assure meaningful test results, especially under varying operational conditions of the host system, ingest of a million data items was conducted twice, in two different cycles, using the same input data. In the first iteration (Test 1), the one million items were ingested to the archive in several batches, by repeatedly invoking the SIPIngestManager from Unix Shell procedures. In each session, several SIPs amounting to roughly 100,000 to 200,000 items were ingested, with each SIP containing a maximum of 36,000 items. The gap between two successive batches spanned a few hours to several days. In the second iteration (Test 2), all the SIPs were ingested in a contiguous manner in two batches (with a gap of a few hours), containing around 100,000 and 900,000 items respectively. This procedure was conducted as a stress test for the system

Results

In running the tests, we limited the size of each SIP to 36,000 items, as it was discovered that the total number of items that may be ingested in a single invocation of the SIPIngestManager with a 3 GB pre-allocated memory is approximately 40,000 (due to the heap space requirements of in-memory caches, XML trees, etc.). Several SIPs may be ingested serially, with no adverse effect, by repeatedly executing the SIPIngestManager from a Java command within a UNIX batch file.

The scalability of the system was measured and plotted as the time taken to ingest an item as a function of the size of the archive, represented by the number of items already ingested to the archive. The data points along the X axis (or the abscissa, in the related graphs) were established at increments of ten thousand – yielding more that 100 measurement points.

The factors that were measured are the following:

- Average time required to ingest an item, computed over every set of ten thousand items. This helped to smooth out the time variations of individual items due to differences in bitstream numbers and data sizes.
- Minimum time taken to ingest an item within each group of 10,000 items.
- Total number of items that were ingested within +/- 25 percent of the average ingest time within any 10,000 item group. This gives a measure of the spread in the ingest time of items (due to variation in bitstreams, etc.) as the archive size increases.
- Average time taken to ingest items with 4, 6, 8 and more bitstreams as a function of archive size.

Test 2 was conducted under a more stable system environment, and the results of this test are presented in Figure 4 through 6. Overall results from Test 1, which were performed over a longer time span with less uniform system conditions, are shown in Figure 7 for comparison purpose. It should be noted that the ingest times presented here include the data upload time to the DSpace temporary storage area, which was observed to be less than 10 percent of the ingest time in the worst case scenario, and therefore, not discussed further.

Average and Minimum Ingest Times

Figure 4 shows these times as a function of increasing archive size. The dotted curves show the average and minimum ingest times while the two straight lines represent their trends. It may be noted that the ingest times closely follow these linear trends, with the exception of items in the range 430,000 through 680,000. The ingest times are higher in this range, indicating lower performance of ≥ 20 percent for the average and ≥ 10 percent for the minimum.

The anomaly shown occurred during a 30 hour period over a week-end, as seen from the actual times (indicated through the dotted arrows) associated with these events. Other systems in our facility reported performance degradations during this time period as well, with the cause unknown at this time.

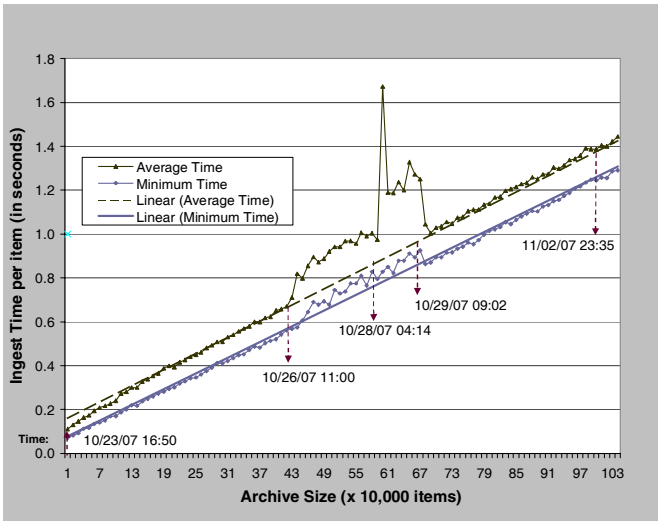

Figure 4 - Average and minimum ingest times as a function of archive size in Test 2

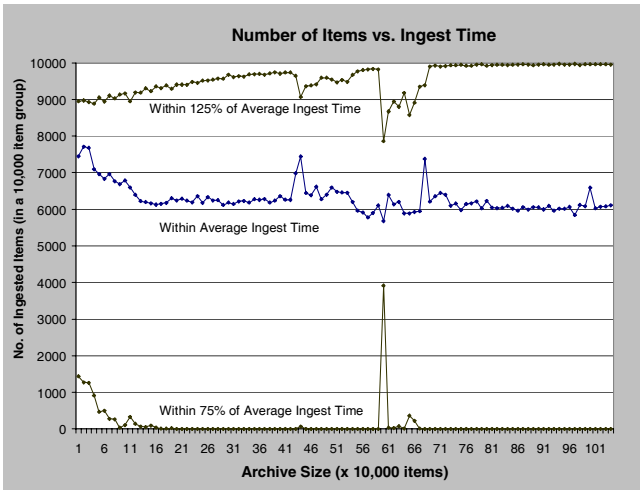

Figure 5 -Number of items ingested within different time ranges as a function of archive size

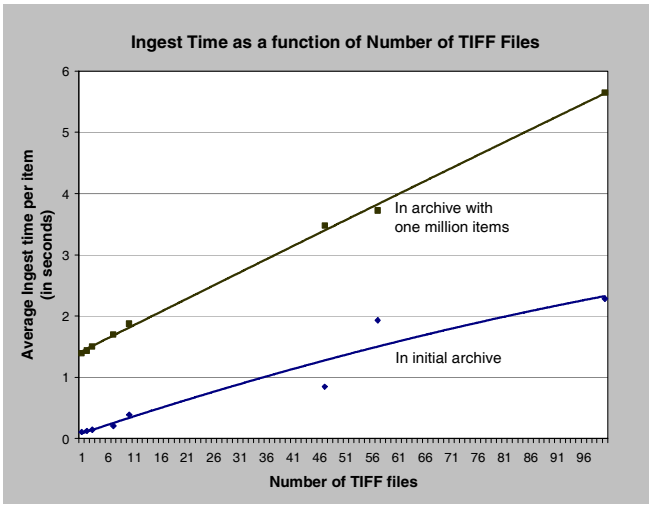

Figure 6 – Change in ingest time with increasing number of TIFF files

Spread in Ingest Time

Figure 5 provides a synopsis of the variation in the ingest times of items from the average ingest time of that group. The three curves from top to bottom indicate the number of items that were ingested within 125%, 100% and 75% of the average ingest time of each group of 10,000 items respectively.

Variation in Ingest Time due to Number of Associated Bitstreams

The time to ingest an item depends upon the number and size of its associated bitstreams, as large amounts of data require more time to upload and store them, and number of bitstreams influence the time needed to create/store the files and corresponding database records.

Table 1: Ingest time vs. number of bitstreams

No. of bitstreams	No. of TIFF files	Avg. ingest time at start (in sec)	Avg. ingest time at one million items (in sec)	Factor w.r.t. one TIFF file at one million items	Percent Increase per TIFF file
4	1	0.106	1.390	1.0	-
5	2	0.125	1.435	1.032	3.2
6	3	0.144	1.500	1.079	3.95
10	7	0.205	1.698	1.221	3.66
13	10	0.389	1.874	1.348	3.88
50	47	0.845	3.475	2.500	3.26
60	57	1.929	3.723	2.680	3.00
103	100	2.281	5.651	4.065	3.10

Table 1 shows the change in ingest time of items due to increasing number of bitstreams, in the initial stage of the archive as well as at one million items. The number of TIFF files of an item is lower by three from its number of bitstreams, due to the exclusion of the two metadata files and the OCR text file.

Figure 6 graphically represents the ingest time as a function of the number of TIFF files of the archived items from Table 1.

Results from the First Test

For the sake of comparison, we present in Figure 7 the average ingest time of items in the first test (Test 1), which was performed in a more realistic scenario involving smaller SIP batches, and periodic ingesting. During the interim period between two ingest sessions, the system was occasionally rebooted and the MySQL server was restarted.

The two dotted curves in Figure 7, representing the average and minimum ingest times, show recurring periodic peaks and troughs in ingest times, which are similar in nature to the deviations found in Figure 4 for Test 2. The sudden change in slopes in this test are found to occur mostly around new ingest sessions.

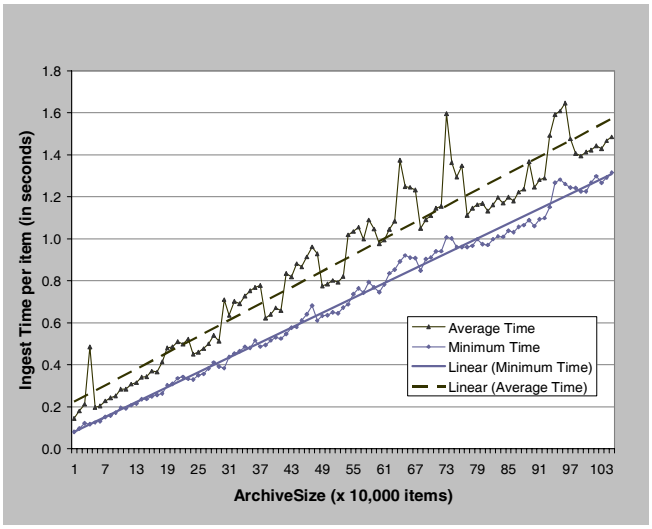

Figure 7 - Average and minimum ingest times in Test 1

Analysis

The steep rises and falls in the average ingest times occur in both Figure 4 and 7, although it occurs only once during Test 2 and recurs in an almost cyclic fashion in Test 1. The reason for the sudden change in slope in either of these charts is not immediately clear, but may be attributed to some other activities, external to ingest, affecting the system performance. These may include network activities occurring in the system, which might have affected the MySQL transaction times as well as available CPU time for the SIPIngestManager, in addition to the overheads caused by the initialization of various system components and caches after a system reboot or MySQL server restart.

From results in Figure 4, the minimum time taken to ingest an item shows a linear trend, varying from ~ 0.075 sec initially to ~ 1.3 sec at one million items. The average ingest time, on the other hand, varies from ~ 0.11 sec to 1.4 sec over the same period. The divergence between the minimum and average trends remains almost steady (changing from 0.05 sec to 0.15 sec over the entire interval). Even with the increase in the archive size, most of the items are ingested within a comfortable range around the average time, as seen in Figure 5. In spite of the differences in the actual ingest curves in Figure 4 and Figure 7, it may be noted that their trend lines are very similar in nature and match in value, yielding the average ingest time of 0.10 to 0.15 sec initially and 1.4 sec at the target size of the archive. This affirms the reliability of the overall performance test results.

From Figure 5, we see that 60% to 70% of items in any group were ingested at or below the average ingest time and more than 90% items were ingested within an additional 25% time. Also, with increasing archive size, the number of items requiring ingest time less than 75% of the average time, is minimal.

From Table 1 we see that the average ingest time of an item with 100 compressed monochrome images (total of 103 bitstreams) increases from 2.281 seconds to 5.651 seconds (a factor of approximately 2.5), in going from an empty archive to the target archive with one million items. On the other hand, the performance cost of ingesting an item with one image vs. an item with 100 images in the final archive changes by a factor of four: from approximately 1.4 sec to 5.6 sec. The last column of Table 1 shows that the time increase in adding a bitstream to an item is around three to four percent as one goes up to 100 additional bitstreams.

Conclusion

We conclude that the version of DSpace used in SPER (with MySQL database) shows acceptable ingest performance for a million-item archive. For larger archives, further benchmarks should be conducted, possibly using higher performing hardware platforms, distributed systems and data grids, as well as more diverse item types.

The experimental results shown here pertain to items with mostly one or two monochrome TIFF images, though a few items have up to 100 images. However, a number of inferences may be derived from these results.

- No real problems were found in ingesting a million items to the archive, using a Sun X4500 server machine, in terms of either performance or reliability of the SPER/DSpace software architecture and implementation.
- The test results are reliable because of the overall matching trends of the two sets (Test 1 and Test 2).
- With the increase in archive size, the average ingest time of an item increases in a smooth and predictable way.
- With increasing number of TIFF images, the ingest time (per item) increases by three to four percent for each additional image.
- If color TIFF images were used, the ingest times would increase slightly due to the overhead of copying additional data to the upload area, and to the archive's asset storage. However, other archival overheads should not change.

Acknowledgment

This research was supported by the Intramural Research Program of the National Institutes of Health (NIH), National Library of Medicine (NLM), and Lister Hill National Center for Biomedical Communications (LHNCBC).

References

[1] Mao S, Misra D, Seamans J, Thoma, G. R.: Design Strategies for a Prototype Electronic Preservation System for Biomedical Documents, Proc. IS&T Archiving Conference, Washington DC, pg 48-53. (2005).

[2] DSpace at MIT, http://www.dspace.org.

[3] http://wiki.dspace.org/index.php/DspaceInstances

[4] Misra D, Mao S, Rees J, Thoma, G.R.: Archiving a Historic Medico-legal Collection: Automation and Workflow Customization, Proc. IS&T Archiving Conference, Washington DC, pg 157-161. (2007).

[5] Public Law 59-384, repealed in 1938 by 21 U.S.C. Sec 329 (a). And U.S Food and Drug Administration, "Federal Food and Drugs Act of 1906 (The "Wiley Act")," http://www.fda.gov/opacom/laws/wileyact.htm (3 Feb. 2006).

Author Biography

Dharitri Misra is a Lead Consultant at Aquilent, Inc., and is a researcher at the U.S. National Library of Medicine working on digital preservation topics. Her work involves developing experiments and tools to help in the long term preservation of digital resources and in automated extraction of metadata from text documents. She earned her M.S. and Ph.D. degrees in Physics from the University of Maryland.

James Seamans is a Senior System Scientist with Lockheed Martin, Inc. Mr. Seamans has worked on many medical research and development computer imaging projects. He received his B.S. degree in mathematics from Ricker College and A.S. degree in Electronics and Computer Technology from DeVry University.

George R. Thoma is a Branch Chief at an R&D division of the U.S. National Library of Medicine. He directs R&D programs in document image analysis, biomedical image processing, animated virtual books, and related areas. He earned a B.S. from Swarthmore College, and the M.S. and Ph.D. from the University of Pennsylvania, all in Electrical Engineering. Dr. Thoma is a Fellow of the SPIE, the International Society for Optical Engineering.