

- [Home](#)



Stuart Lewis' Blog

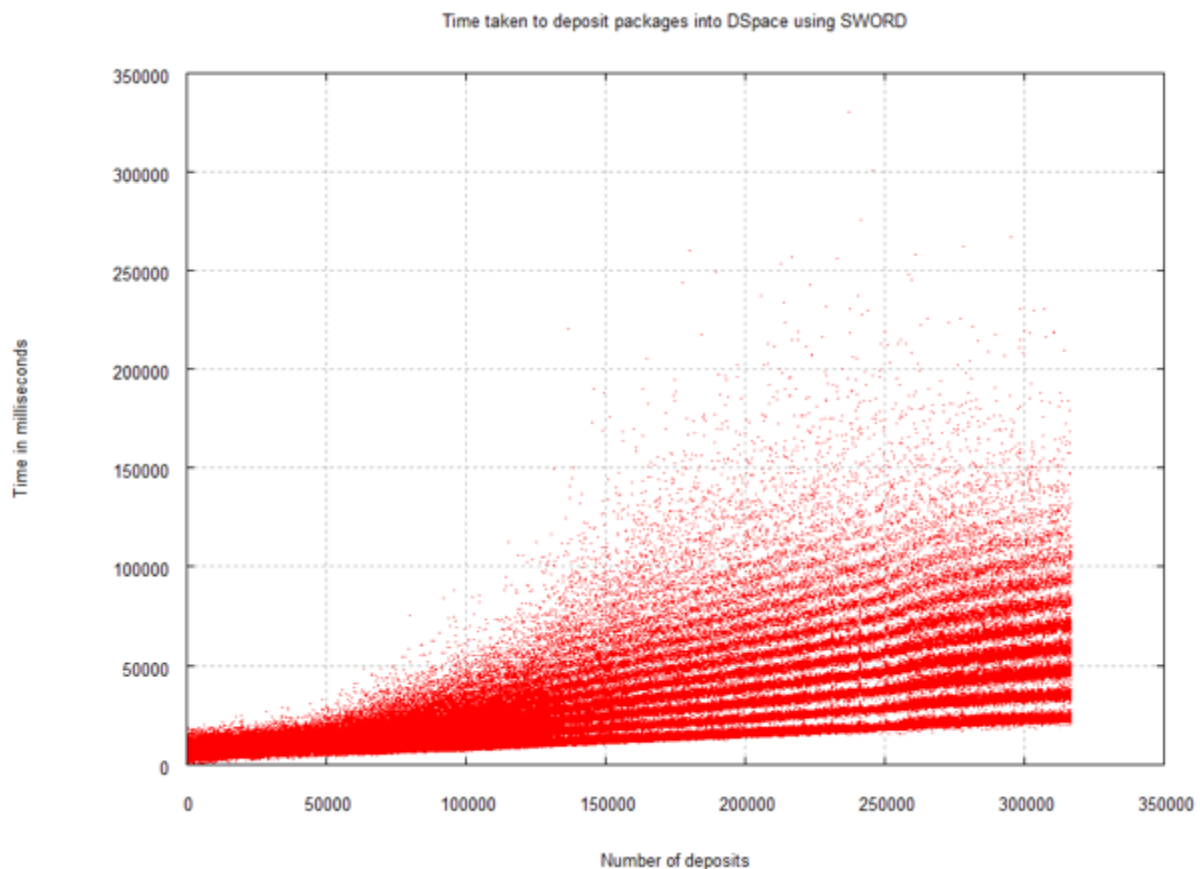
DSpace at a third of a million items

As part of the JISC-funded ROAD ([Robot-generated Open Access Data](#)) project we are load testing DSpace EPrints and Fedora to see how they cope with holding large numbers of items. For a bit of background, see an earlier blog post: '[About to load test DEF repositories](#)'

The project programmer Antony Corfield has created a [SWORD](#) deposit tool for this purpose. It is a configurable tool that allow you to define a set of SWORD packages to deposit, how fast you want them to be deposited, how many you want to deposit at a time etc. We decided to deposit using SWORD so that we can deposit a common SIP into each repository without too much extra work.

Early tests with this software depositing into DSpace on our server (8 processors, 16GB RAM) suggested that the optimal rate of deposit is 4 concurrent deposits. (This may make be due to having 8 processors, and each deposit optimally requires two processors - one for the database and one for the web application, but further tests would be required to confirm this).

The tool was left running over Christmas depositing 9Mb (approx) SWORD packages, each containing the results of an experiment. It is now almost up to 1/3rd of a million deposits. Whilst we would have liked to keep running the experiment to take the deposits to maybe 1 million, it would take more time than we have. Below is a graph of the time it took to deposit each item:



Some observations from this chart:

- As expected, the more items that were in the repository, the longer an average deposit took to complete.
- On average deposits into an empty repository took about one and a half seconds
- On average deposits into a repository with three hundred thousand items took about seven seconds
- If this linear looking relationship between number of deposits and speed of deposit were to continue at the same rate, an average deposit into a repository containing one million items would take about 19 to 20 seconds.
- Extrapolate this to work out throughput per day, and that is about 10MB deposited every 20 seconds, 30MB per minute, or 43GB of data per day.
- The ROAD project proposal suggested we wanted to deposit about 2Gb of data per day, which is therefore easily possible.
- If we extrapolate this further, then DSpace could theoretically hold 4 to 5 million items, and still accept 2B of data per day deposited via SWORD.

Some notes on how the experiment was undertaken:

- The deposits took place in one go, with no break.
- Therefore no maintenance took place during the test which might have done normally, and which might be expected to improve performance (e.g. vacuuming the database).
- The network between the machine performing the deposits and the machine running the repository was a dedicated 1Gbit/s (crossover cable).

What is most interesting in the chart is the 'banding' that seems to be taking place. The time taken to perform a deposit seems to fall into 12 or so 'bands'. For example at three hundred thousand deposits, deposits either take 2.5 OR 3.5 OR 4.5 OR 5.5 OR ... seconds to deposit, but NOT 3 OR 4 OR 5 OR... seconds to deposit.

Why is this? One hypothesis might be there when a deposit starts, the other three deposits that are also taking place can be in one of a number of 'states'. Depending upon the combination of states that the other deposits are in, it affect the time it takes for the deposit. For example if the other deposits are all in a disk intensive state, then initiating another deposit where the SIP is being uploaded (requiring even more disk activity) would be slow, leading to a slow deposit time. However if the other three happen to be in a processor intensive state, the deposit may be faster.

But... with three other deposits going on at once, if each of those had two states that they could be in, you might reasonably expect there to be only eight bands (2^3). Or if there were three states, you might reasonably expect there to be 27 states (3^3). But the graph suggests there are more than 8 states, but a lot less than 27.

Of course it may be an artifact of the way the deposit tool is working that causes this. Hopefully it isn't, but until we run the same test depositing into EPrints and Fedora, we'll not know.


It is useful to note the overhead that SWORD puts on the deposit process. Those of you who have run normal imports into repositories such as DSpace will know that they zip along quite fast, probably several times (if not more) faster than the deposit by SWORD. The reasons for this are:

- With batch import, the files are already on the repository server. With SWORD, they need to be transferred onto the server using a HTTP POST.
- With batch import the files are ready to be imported into the asset store. With SWORD the files are zipped up in a package. First the package needs to be read in order for its MD5 checksum to be computed and compared to the checksum sent by the server to ensure the package has been transmitted without and errors. Then the package has to be unzipped.
- With batch import, the metadata is held in an XML file which is parsed and read. With SWORD, the metadata is also held in an XML file, but is first transformed using an XML stylesheet which takes a bit more time.
- With batch import the results are displayed on the command line. With SWORD, and Atompub response has to be sent back over the network and then processed by the client.

(Over the next few months we'll run similar tests with EPrints and Fedora, and depending on time can try other tests such as performing the same tests but on a server which is under user-load (we'll run load testing software that simulates user load such as viewing and downloading items, performing searches, accessing the OAI-PMH interface etc) to see how that compares. Before we simulate user load, we'll have to consider what the expected load on a data repository might be. My initial guess is that it would be perhaps lower than a general institutional repository, but when a user does download data, they will probably want to download many hundreds of experiments and several gigabytes of data. So the general use would be lower, but each use would have a higher impact on the repository.)

[January 19, 2009](#) • Tags: [dspace](#), [repositories](#), [roadproject](#) • Posted in: [Uncategorized](#)

10 Responses to “DSpace at a third of a million items”

1.  [Stevan Harnad](#) - January 19th, 2009


Bravo for benchmarking the load capacity of Institutional Repositories (IRs) (and I look forward to seeing your results for EPrints).

But why the concern with how much one can cram into *one* IR, and how fast and how long? Why not just branch to another IR, and another, as each gets topped up? CalTech has at least 26 EPrints IRs! The whole point of the OAI-PMH was to make multiple distributed IRs interoperable, as if they were one big virtual (harvested) repository.


2.  [Paul Hartr](#) - January 20th, 2009

Hi Stuart,


Very interesting research and glad to see you took the time to try this. We use Intralibrary and the forthcoming version 3 supports SWORD, I must try something similar.

3.  Les Carr - January 21st, 2009

Can you compile DSpace with profiling information to get some accurate data about what activities are taking place? For example, how much of the ingest process is spent on communicating with the database, communicating with the storage server, creating thumbnails, indexing the fulltext, making a checksum etc.


4.  [stuart](#) - January 21st, 2009

Hi Les: Yes. There are different logging 'levels' that can be used. A normal DSpace instance would use the INFO log level which just logs high level information, errors and warnings. If we set this to DEBUG, we get very detailed logging information, right down to the individual SQL statements being executed. This log file would need analysing to derive the actual profiling information though.

5.  [stuart](#) - January 21st, 2009

Hi Stevan: Thanks for your comments. My take on this is that i) It is good to load test repositories. Whilst we could use multiple repositories, we need to make sure we are getting the best use out of our current software and hardware, and that we are not being stifled by bad software. Unless we run load tests, we'll never know if that is the case.


ii) Yes, OAI-PMH can be used to pull together multiple repositories into a common search system, but somewhere (such as a common search system) all the data needs to be help in one place, so we might as well see if the repositories can handle this themselves without the complexity and cost of having to run multiple repositories and a search service.

6.  [Jim Downing](#) - January 27th, 2009

Hi Stuart, good work!

Can the banding be explained by GC activity?


Did you try dropping all the indices before depositing, then rebuilding them afterwards? A commonly employed trick if the game is bulk ingest...

7.  [stuart](#) - January 29th, 2009

Hi Jim. Thanks for your comment.

I suppose garbage collection is likely to play a role here, but to what extent it contributes towards the banding I don't know.


We didn't drop the indexes as we felt that would have been cheating a bit. Our aim was to simulate normal deposit over time, rather than just seeing how quickly we can get the deposits into the repository.

8.  Debashree Pati - February 10th, 2009

Hi Stuart,

What do you mean by batch import? Does SWORD provide for bulk posting similar to the itemImport of DSpace. From the SWORD source code and sword client options, all I see is method to post a single item at a time.

Thanks,
Deb

9.  [stuart](#) - February 12th, 2009


Hi Deb,

Where the post mentions batch import, I was comparing the performance of using SWORD, to using DSpace's (or EPrint's, or Fedora's) command line batch import tool.

What the project has been using is a tool we've written to mimic that sort of bath behaviour, not only in depositing multiple items serially (one at a time) but also in parallel (many at a time).

Thanks,

Stuart

10.  [Kai](#) - February 18th, 2009

Hi Stuart,

we've also seen the banding with Fedora batch ingests. I believe these bands emerge mostly due to the hard disk(s) being under heavy load and having to operate in a predominantly non-sequential way. It got significantly better (less bands with narrower gaps) once we began distributing different parts of the system, most notably the database, to separate machines or at least separate disks.

Thanks,
Kai

Leave a Reply

Name, required

Email, will not be published, required

Website

Submit Comment

- [« Previous post](#)
- [Next post »](#)

•

• Tags

[analytics](#) [blackboard](#) [blogs](#) [citations](#) [depositplait](#) [dspace](#) [eprints](#) [fedora](#) [google scholar](#) [jedemonstrator](#) [interoperability](#) [jisc](#) [jisc-crig](#) [ldap](#)
[mashups](#) [oai-ore](#) [open access](#) [preservation](#) [repositories](#) [repository66](#) [repository](#) [mashup](#) [map](#) [roadproject](#) [rsp](#) [rspsummerschool2008](#)
[search](#) [servers](#) [sharepoint](#) [shibboleth](#) [sword](#) [training](#) [voip](#) [web](#) [windows](#) [youtube](#)

•

January 2009

M T W T F S S

1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 [17](#) 18

[19](#) 20 21 22 23 24 25

26 27 28 29 30 31

[« Dec](#) [Feb »](#)

• Archives

- [February 2009](#)
- [January 2009](#)
- [December 2008](#)
- [November 2008](#)
- [October 2008](#)
- [September 2008](#)

- [August 2008](#)
- [July 2008](#)
- [June 2008](#)
- [May 2008](#)

• Recent Comments

- [Kai](#) on [DSpace at a third of a million items](#)
- [stuart](#) on [DSpace at a third of a million items](#)
- [stuart](#) on [Test LDAP service upgraded - now with branches](#)
- [Debashree Pati](#) on [DSpace at a third of a million items](#)
- [Aaron Hossain](#) on [Test LDAP service upgraded - now with branches](#)

© [Stuart Lewis' Blog](#).

Powered by [WordPress](#), styled by [Grey Matter](#).

Ⓜ