DstatPackage

DStat Release 3

I've done some fairly major internal re-working of DStat which has provided some extra functionality, a large step on the road towards having a real, useful API and object model, and spent some time getting a tentative dynamic integration into the DSpace UI.

The code for this version of dstat is located here: http://www.thesesalive.ac.uk/download/dstat3.tar.gz

To install this version follow the instructions in the INSTALL document, and to customise the report scripts have a look at SCRIPT_INFO. Otherwise you should use:

```
[dspace]/bin/dsrun ac.ed.dspace.stats.LogAnalyser -help
```

and

```
[dspace]/bin/dsrun ac.ed.dspace.stats.ReportGenerator -help
```

For information on how to use these two processes.

DStat works by analysing reports using LogAnalyser and creating aggregation files in its own (very basic) internal format. The ReportGenerator then knows how to conver this aggregation file into a human readable report (currently only supports HTML). So an example usage of these two processes might be:

```
% [dspace]/bin/dsrun ac.ed.dspace.stats.LogAnalyser -start 2005-01-01 -end 2005-01-31 -out january-aggregation.
dat

% [dspace]/bin/dsrun ac.ed.dspace.stats.ReportGenerator -format html -in january-aggregation.dat -out january-
report.html
```

This would then produce an HTML file with a summary of system activity in january 2005

Unfortunately, due to certain constraints, most expecially time, the UI for this system is sensitive to the names of report files, and monthly reports should be of the form "report-2005-1.html" whilst general reports should be of the form "report-general-2005-3-1.html". This is one of the many things which could do with improvement. Thinking of which, today's wishlist is:

1. Improve the API still further and create a genuine object model
2. Extend the functionality to complete aggregation of all actions and include consideration of stack traces
3. Increase configurability to allow for turning report sections on and off and defining the order in which they are displayed
4. Produce, initially as a test of the object model, a plain text report generating class; this should then lead on to having XML and other format generation.
5. Attempt tracking of user actions and item usages and so forth (could require some thought); i.e. go beyond basic aggregating

I would like to make this code available as part of the forthcoming 1.3 release, but it may be insufficiently mature.

Java DStat

DStat has been recently transcoded into java, bringing with it nearly two orders of magnitude improvement in execution time. Functionally there's not a tremendous difference from the version previously posted here, but the production of HTML reports has been significantly improved and have been integrated, sort of, into the DSpace UI.

The code to use DStat is located here: http://www.thesesalive.ac.uk/download/dstat-2.tar.gz

I've bundled a jar file with the package, but if you want to install from source you should use

```
ant -D[dspace lib directory] -D[dspace config file] -D[dspace install source lib]
```

This will build the jar and put it in the installation source lib directory for DSpace. Otherwise just drop the supplied jar into the dspace source lib directory.

Next, drop the dstat/jsp contents into dspace-source/jsp/local directory. Add the lines in dstat/etc/dstat-web.xml into the file dspace-source/etc/dspace-web.xml to provide the UI servlet mappings then put dstat/config/dstat.cfg into your dspace/config directory (the live dspace directory, *not* the source directory).

Finally, re-build your DSpace as usual and restart the web server.

To check that DStat is installed in the front end, go to: http://www.mydspaceinstance.edu/statistics and see if there's a page there with some introductory text in it.

To use DStat you need to run 2 command line utilities:

1. ```
   [dspace]/bin/dsrun ac.ed.dspace.stats.LogAnalyser [options] > data.file
   ```

   This will use the options you give it to parse your log files and produce an aggregation file which contains the results of the analysis in a semi-human readable form. For information on how to use LogAnalyser execute it with the -help flag and no "> date.file" on the end.
2. ```
   [dspace]/bin/dsrun ac.ed.dspace.stats.ReportGenerator -format html -in data.file > report.html
   ```

   This will generate an HTML body which contains the results of the analysis (from data.file) in a fully formatted, human readable form, for inclusion into the DSpace UI. Again, use the -help flag for more information on command usage.

Since the UI end of this is still *very* basic, you need the results.html file to be placed in tomcat/webapps/dspace/jsp/statistics/. I'm working on this, and in the end a better solution will be available, but for the time being it will do the job.

There are a number of features which I'm still looking at implementing:

1. A proper API for calling DStat from within DSpace, mostly to aid integration
2. A proper implementation of a Strategy pattern for report format generation
3. Plain text and XML report generation implementations
4. Multiple report handing (e.g. monthly reports and so forth)
5. Improved log file parsing algorithm to make date restrained analyses faster

I'm going to look into putting this code in a CVS somewhere, and then development can hopefully happen in a collaborative way. Feedback welcome (r.d. jones (at) ed.ac.uk).

Old DStat Instructions

The DStat package is at the moment of half-implemented perl scripts and compiled java. It's derived from the original dspace log-reporter package, and works in a very similar manner. It's main differences are increased configurability via the dstat.cfg file, more organised layout, as well as the basics of two different forms of layout: plain text and HTML, and it also interfaces with the database to obtain some archive statistics too.

Attached to this page is the file dstat.tar.gz, and when you unzip and untar this you should find a bunch of perl scripts, a dstat.cfg file, a jar file and a java directory. The java directory contains the source code, but I've bundled the pre-compiled jar too to save you some time.

Drop the dstat.jar file into your dspace lib directory, put the perl scripts into your dspace bin directory and the dstat.cfg file into your dspace config directory (although you can specify a cfg file at run time). There are a few ways of running the software:

1. Use ./dstat to produce results for the whole system and the previous 3 months all in separate output files. This is not recommended just now as it's a bit of a mess and I need to look at the controlling behaviour again before I will be satisfied it's any use.
2. Use ./log-analyser to produce a basic plain text summary of the log files. It dumps to the standard out.
3. Use ./log-analysis to produce a results file which can then be read by log-plain-output or log-html-output. Stuff the output into a file, then run log-plain-output or log-html-output on it. This is my recommended method just now as it represents the most recent code. log-analyser will eventually disappear, since all its functionality is included across the log-analysis and output files.

So an example might be:

```
./log-analysis > output.dat
./log-html-output --file output.dat > stats.html
```

at the moment the date stuff is working for log-analysis but the output files have no way of printing the info. This is high on my list of priorities for this bit of code.

I am open to the idea of re-coding this entirely in java, depending on what people think the performance advantages might be. At the moment this code takes around 3 minutes to analyse all of my logs, which adds up to about 5 months worth. I hope to alleviate this problem in the long term by using the output from ./log-analysis as a kind of long term store of aggregation data and using this instead of trawling through logs that we've already trawled once before.

This is a list of things that I would like to add, or am thinking about adding, to the package:

1. Decent HTML output (rather than the somewhat ropey output that I put together in a hurry for the current pacakge)
2. More careful organisation or the executive summary, especially when producing multiple reports for several months
3. Better date handling, and faster parsing algorithm (based on assumption of sequential log records)
4. Consider port to java
5. Plan out more carefully how to use aggregation files to improve parsing times, and to remove the need to scan old log files
6. Propagate date handling to display mechanism

Feedback welcome

The files for this software are available here: http://www.thesesalive.ac.uk/download/dstat.tar.gz