FOPTuningGuide

This guide is written by a semi competent sys admin who has tested FOP and read around the mailing lists to discover what to do. You may be able to achieve much more ... if so please add to this guide.

Tuning to exploit your hardware

Having played around with FOP transformations I have found that large xml/xhtml to PDF transformations are very memory hungry. On a default tomcat install the transformations don't have to get very large before the java virtual machine (JVM) runs out of memory (I think JVMs default to a max heap of 256m, I may be wrong).

To tune up a Tomcat install in order to exploit all the memory your machine has you are first going to have to edit the catalina.sh file in the tomcat bin directory. You want to tune the CATALINA_OPTS variable so it sets up your JVM with as much memory as you can spare (remember to leave at least 20% for the operating system). To my catalina.sh on tomcat-4.1.24-LE-jdk1.4 i added the line

```
CATALINA_OPTS="$CATALINA_OPTS -server -Xms256m -Xmx1800m -Xincgc"
echo "Using CATALINA_OPTS:   $CATALINA_OPTS"
```

Just before it starts echoing your CATALINA_BASE, CATALINA_HOME etc variables (catalina.sh has changed between versions of tomcat so your mileage may vary).

This line tells Tomcat to fire up in a JVM with 256meg of memory (-Xms) and that that JVM can grow upto 1800meg (-Xmx). It also tells it to incrementally garbage collect (-Xincgc) so as not to slow down too much when it decides it needs to look through all its memory heap for stuff it can throw away. An Xmx of 1800m is about the maximum heap size you can have on a windows or linux box as the jvm can't grow any larger Java bug report. You may be able to get up close to 4 gig on a 32 bit sparc box. I have no idea about 64 bit systems. I have tested this with Suns JDK, Bea's IBM and Blackdown and they all break close to 1900m max heap size.

Predicting FOP transformation load

Unfortunately there seems to be no way of predicting how much memory a FOP transformation will need. It can vary depending on what your stylesheet looks like. For instance it is advised on the mailing lists not to nest tables as this greatly increases the size of the DOM tree needed to represent your FO transformation.

Test scenario

An example of FOPs thirst for memory can be seen by an example transformation we did. We have a project to convert properly formatted XHTML articles to PDF. An export of all articles produced a 3meg XML file. When we did a simple transformation on this to PDF it peaked at 530meg of memory and took 600 seconds of processor time (it was a 4 processor machine so actually only took about 3 minutes). This produced a 920page 1.6meg PDF file. Obviously this is a stupid example as there would rarely be a need to produce a 920 page PDF file. However this was a transformation as simple as we could make, a more complex transformation including tables, complex formatting etc etc on a smaller document may require just as much memory.

I plan on producing a demo app that uses XSP to dynamically generate xml of various lengths (see note below), this will then be transformed to PDF. This way you will be able to see how large a file your server is capable of transforming.

Reader's notes

- The PDF caching sample (at http://localhost:8888/samples/fop/ in 2.1) generates any number of PDF pages based on a request parameter, it might be useful for your tests. The pages are nearly empty though, so it might not be representative of your actual app – BertrandDelacretaz