Right tools for the job?

Interesting discoveries when processing large volume of data

Michael.Czapski@sun.com

May 2009

Just now I had an occasion to work with an integration solution intended to process lots of records. By lots I mean over 1 million smallish records. My customary platform to experiment on is Windows XP. Lots of reasons for that, most of them historical – I have tools I know and like and so on. While trying to work with such a volume of data I noticed a number of "interesting" things, which I thought I should share. These things are related to both the platforms (Windows vs. Linux), the tools and the architectural decisions.

I needed lots off data to test the solution I was contemplating, which involved XML processing, to see how constructing and parsing XML affects solution performance. To make it easier to compare timing differences I though I should use lots of records.

An Oracle 9i sample database schema SH (presumably Sales History) has over 1 million records so I thought I will extract that data to use. Here came my first surprise. DbVisualizer tool, which I typically use to work with databases, could not deal with over 1 million records. Being Java-based, it blew JVM memory allocation well before it got to 1 million records. I next tried the Oracle's own SQL*Plus Worksheet. No joy. It too could not cope with extracting over 1 million records from the database. Next I tried Java CAPS Repository-based solution. No joy there either. The result set produced by the select statement exhausted JVM memory as well. Finally, I turned to the Oracle's command line tool, plain old SQLPlus. With it I managed to extract all 1,016,271 records into a CSV file. So, the right tool for the job is required.

The output file is 166,519Kb in size - ~160Mb.

Having a file with lots of records I next set off to see how long it would take to read the file, break it up into records and write the records to another file. I used Java CAPS for this task as I know enough of Java CAPS to make my job quick and easy.

First experiment involved reading the file using the BatchLocalFile eWay, breaking it up using the BatchRecord eWay and writing records out using the BatchLocalFile eWay again. I used the BatchLocalFile and BatchRecord eWays in streaming modeto avoid reading the entire file into memory, which is what the BatchLocalFile eWay user would typically do. On my Toshiba Tecra M5, with 3.2Gb of memory and a very full and fairly fragmented disk, the process took several hours. I have not kept a record but recollection is that the time was on the order of 16 hours. I them moved the project to a Windows 2003 Server VMware Guest, configured to use 8Gb of Memory, 4 cores in a 32-way Sun x5140 Server running VMware ESX 3.5 Server, with 100Gb of half empty disk. The test repeated in that environment completed in 40 minutes, giving me 415 TPS. Again, the right tool for the job will make a great deal of difference. I stuck to the Windows 2003 for the rest of the tests involving record transfers.

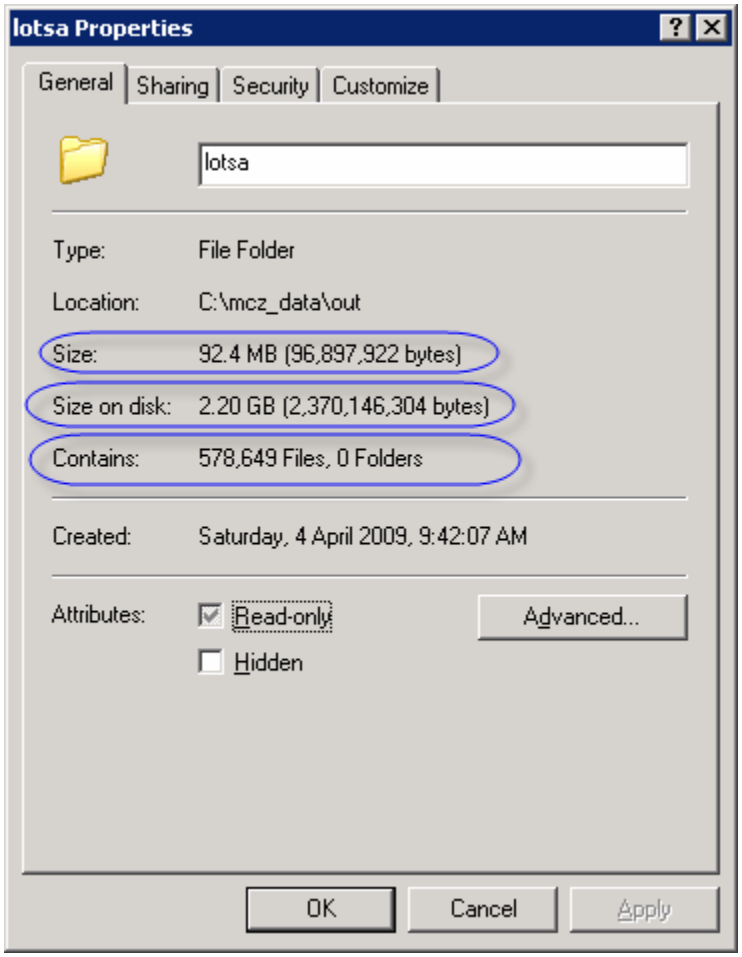
The next experiment involved transforming CSV records into equivalent XML records. I expected that the process will take longer due to assumed XML processing overhead. I also expected the output file to be larger. The process took 44 minutes, only marginally longer then the previous test. The overhead of XML construction was not very large. The output file, however, grew to 963,086 Kb - ~950Mb – that's nearly 6 times larger then the original for absolutely no added data. The processing overhead test is not very conclusive principally because the record is so small and so so simple. Here is an example:

```
<SH_SALES_VIEW xmlns="sh_sales_extract/viewSH_SALES_VIEW/XMLSchema"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation="sh_sales_extra
ct/viewSH_SALES_VIEW/XMLSchema C:/JCAPS6U1/appserver/domains/domain1/jbi/service-assem
blies/JBIPassThrough_Delim2XMLDecoding/JBIPassThrough_Delim2XMLDecoding-sun-file-bindi
ng/sun-file-binding/SH_SALES_VIEW.xsd">
<TIMESTAMP>1238641858972</TIMESTAMP>
<SEQ_NUM>33</SEQ_NUM>
<PROD_ID>36795</PROD_ID>
<PROD_NAME>Susane Street Girls' Panties</PROD_NAME>
<CUST_ID>56100</CUST_ID>
<CUST_LAST_NAME>Aubrey</CUST_LAST_NAME>
<CUST_FIRST_NAME>Anne</CUST_FIRST_NAME>
<CUST_STREET_ADDRESS>27  Cupertino Boulevard</CUST_STREET_ADDRESS>
<CUST_POSTAL_CODE>58488</CUST_POSTAL_CODE>
<CUST_CITY>Dolores</CUST_CITY>
<CUST_STATE_PROVINCE>CO</CUST_STATE_PROVINCE>
<COUNTRY_ID>US</COUNTRY_ID>
<TIME_ID>01/JAN/98</TIME_ID>
<CHANNEL_ID>S</CHANNEL_ID>
<CHANNEL_DESC>Direct Sales</CHANNEL_DESC>
<CHANNEL_CLASS>Direct</CHANNEL_CLASS>
<PROMO_ID>9999</PROMO_ID>
<QUANTITY_SOLD>2</QUANTITY_SOLD>
<AMOUNT_SOLD>8</AMOUNT_SOLD>
</SH_SALES_VIEW>
```

A larger and more complex record would most likely be more expensive to process. Perhaps I will test that at some point in the future.

Another test involved breaking the file into records and writing records out as separate files. That has proved to be much more surprising then I imagined. The process started rapidly enough but within a short while, a few minutes, it grew noticeably slower. After a few hours it would take several seconds to write a new file. I gave up after over 30 hours and only 578,649 files written. The Windows 2003 Server and Windows XP file systems (FAT32 and/or NTFS) seems to have a major issue with large numbers of small files in a single directory.

The 578,649 files are, according to Windows, 92.4Mb in total size. Yet, Windows reports that they occupy a staggering 2.2Gb of disk space! See below.



It took a few minutes to produce a 7-zip archive, a ZIP archive and a tar archive containing all the files. Not bad at all.

The 7-zip archiver's archive is 11,818Kb in size – just over 11Mb. To extract all the files on a Windows machine takes _hours_!

The ZIP archive is 169,267 Kb in size – just over 169 Mb. To extract all the files on a Windows machine takes _hours_!

The gzip compressed tar archive is 47,825 Kb in size – just over 47 Mb. To uncompress the archive takes a couple of minutes. To extract all the files from the tar archive on a Windows machine takes _hours_!

I can't process the 7-zip archive on Linux so I can't say.

A tar archive with the same files takes on the order of 10 minutes to extract all files on the Red Hat Enterprise Linux 4 on the same kind of machine as the Windows 2003 machine already mentioned. Compare Windows and hours to Linux and minutes. The differences in the file system structure and implementation are what makes all the difference. When required to write lots of small files use a Unix-like file system.

A ZIP archive with the same files takes on the order of 10 minutes to extract all files on the Red Hat Enterprise Linux 4 on the same machine as before. Note that the size of the index is around 35 Mb!

```
drwxrwsr-x   2 jcaps jcaps  36265984 May  8 13:46 lotsa
drwxrwsr-x   2 jcaps jcaps  36265984 Apr  6 07:51 lotsa_tar
```

To test a theory I modified the logic of the Java collaboration such that it would write files into multiple directories, 1000 files per directory. The process took just over 1 hour – 69 minutes to be exact, generating 1016 directories with 1000 files each, except the last which had 271 files. Way less then the time it took to write 578 thousand files into a single directory.

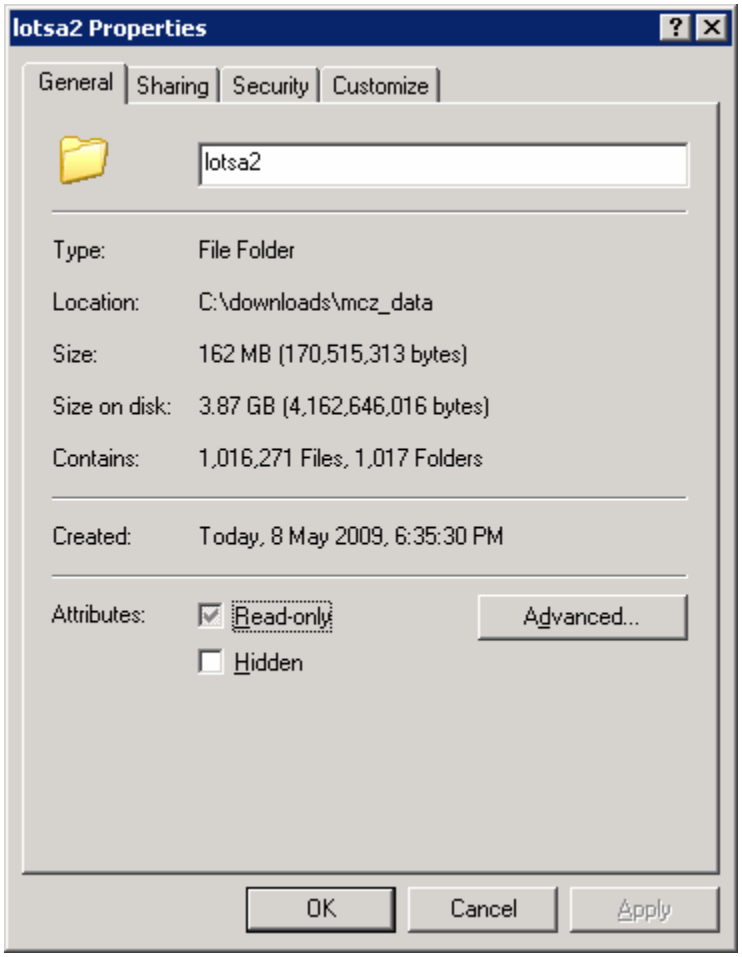
Not surprisingly producing a tar archives took only a few minutes. Since the time it took to produce a tar archive in the previous case was comparable to the time it took a zip ands 7-zip archive I did not bother repeating the experiment. The resulting tar archive was 1,016,781 Kb in size. Gzipped archive was 82,168Kb in size.

Given how much less time it took to write files out to many directories with relatively few files per directory I expected that extracting a million files from an archive to a file system on Windows will similarly take much less time then extracting just over half that number of files to a single directory.

It took just under 5 minutes on a Linux platform to extract the 1016 directories with 1000 files each from the tar archive.

It took just under 15 minutes to extract the same set of files on Windows.

The disk space usage, reported by Windows, is well in excess of 3Gb with the actual data being only on the order of 162 Mb!



Learning:

Lots of small files wastes lots of disk space.

When extracting lots of small files from an archive to a single file system directory, do the extraction on a Unix-like platform.

When writing lots of small files on Windows write them to multiple directories with relatively few files per directory, and keep the number of directories reasonable as well. Around 1000 directories with around 1000 files each seems OK.