TCTool
for
Java Advanced Imaging

Version 1.0

Tile Cache Tool
for
Java Advanced Imaging

The Tile Cache Tool (TCTool) is a diagnostics program and event viewer for Sun Microsystem's implementation of tile caching in the Java Advanced Imaging extension. Its purpose is to help JAI software developers understand and improve their use of tile caching. The tile cache in JAI may act in non-intuitive ways which can cause performance bottlenecks or unpredictable behavior. Programs may allocate insufficient tile cache memory or inefficiently allocate too much memory which is wasted. Use of iterators in JAI can cause pathological situations with tile caching, impacting performance in a negative way. Programs may mistakenly hold on to object references unnecessarily, preventing garbage collection of unneeded objects, "the proverbial memory leak". Program performance may be affected by the garbage collector and whether incremental garbage collection is enabled or disabled. Programs may take insufficient advantage of tile caching, resulting in less than optimal performance with JAI. These are some of the issues that TCTool tries to identify.

Tile Cache Basics:

Some key things to know about the Sun Tile Cache: The JAI tile cache consumes almost no memory at all! It is basically a hash table that holds references to tiles. This locks them into memory as long as certain conditions are met. The default amount of memory that the cache can hold onto is 16MB and the memory control threshold is set to 75%. As each tile enters the cache it is given a simple time stamp. When the cache memory fills up to the 100% full level, tiles which are the least frequently accessed have their references removed from the tile cache and become subject to garbage collection. Tiles are "freed" until the memory level falls to the memory control threshold. If a request for a tile is made that has been removed, the tile must be pulled through the JAI operation chain again (recomputed). This has a subtle effect on performance measurements, if you measure the time it takes JAI to compute a tile, the first time the computation occurs will show the time it takes JAI to compute this tile plus any associated time required to load new Java classes. Class loading can be shown by using the java -verbose flag when starting up the application. If the tile is requested again and it is already in the tile cache, no compute time is required to retrieve the tile. Performance measurements are also tainted (possibly) by garbage collection. Basically, you can't just put a JAI imaging operation in a loop and measure its performance. It takes a very controlled experiment to measure performance and usually involves turning off the tile cache completely.

> TIP: To help prevent "OutOfMemory" exceptions start your java application with
> java -Xms64m -Xmx128m your_app (see Sun Java docs)

Although this version of TCTool does not support custom implementations of tile caching, it can be used with a custom implementation if that implementation is a subclass of SunTileCache and also implements Sun's CachedTile. These classes contain hooks to allow event monitoring.

> NOTE: There is some interaction with TCTool, the garbage collector and overall performance. Objects are created and removed, so observing the tile cache alters the use of memory in the JVM. This is primarily due to adding information to the event logger in TCTool.

TCTool Key Features:

Monitor tile cache statistics:

- cache hits
- cache misses
- number of tiles cached
- percent of tile cache memory used
- event counting

Real time monitoring of JVM heap usage:

- shows allocated memory
- shows used memory
- shows used memory in percent of allocated memory

JAI tile cache event logger:

- shows which JAI operation triggered a cache event
- shows what the event was
- shows how big the tile was
- shows the tile's time stamp

Tile Cache controller:

- force a garbage collection
- flush the tile cache
- reset all counters
- turn on/off diagnostics in JAI
- enable or disable JAI's use of the tile cache

Options:

- auto-resize the window to its original size
- increase/decrease cache memory size scales
- set the memory control threshold percentage
- adjust the time delay (update rate) for the memory monitor
- change the number of rows displayed by the event logger
- change the user interface look and feel (if supported)
- quit TCTool without quitting the application

System Requirements:

TCTool requires the following installations:

1. Java SDK j2sdk1.4.0
   available from http://www.javasoft.com
2. Java Advanced Imaging 1.1.1
   available from http://www.javasoft.com/products/java-media/jai
3. tctool.zip
   available from http://www.vvsi-imaging.com/downloads
4. color monitor with a resolution of 1024x768 or greater.

Installation Instructions:

The tctool.zip file contains the source code, software license and documentation for TCTool. This product is being released in source code form so that users who want to modify the code may do so. The current Java package name for tctool is "tilecachetool", if you want to change the installation location or package name, you may do so. Make sure that the path environmental variable for your system points to the "bin" directories in both the JRE and JDK.

1. place the tctool.zip file in the same directory/folder as your application
2. unzip tctool.zip
3. you may remove or archive the tctool.zip file
4. change directories so that you are in the tilecachetool directory
5. for systems supporting the "make" utility, just type make
   or, double-click on the build.bat file (for windows)
   or, at the command line type "javac *.java"

Developer Instructions:

In order to use the TCTool, you need to add two lines of code to your software, one to import the tilecachetool package and one to attach TCTool to your program, so you need to add

```
import tilecachetool.*;
```

then, somewhere in your code add (usually before the first call to any JAI method)

```
TCTool tctool = new TCTool( );
```

That's it! TCTool will start up with your application. TCTool is an instance of JFrame and thus has all the characteristics associated with the JFrame. For example, if you want to move TCTool's window to sit on top of your application, you can call tctool.toFront( ) to bring the window forward. For more information, see Sun's Java and Swing documentation.

Support:

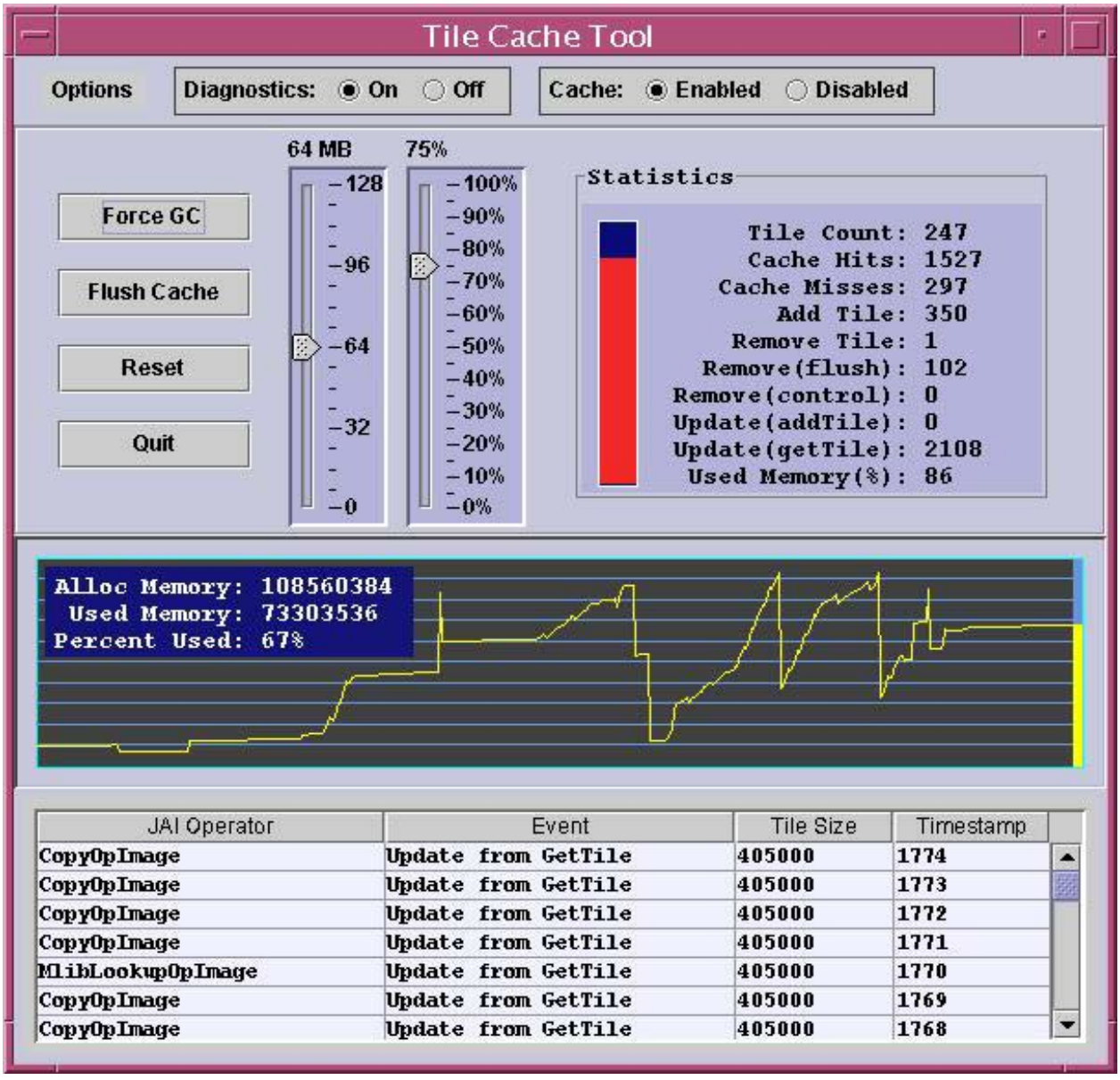
Although this product is unsupported, questions, comments, bug reports, requests for enhancement and suggestions can be e-mailed to

`support@vvsi-imaging.com`

Please do not expect support from Sun Microsystems or the jai-interest list. This product is being released to benefit the Java Advanced Imaging development community. Please see the copyright and license agreement for further information.

User Guide:

The following section describes the use and interpretation of information presented by TCTool. The user interface selected here is referred to as “Metal”, or the Java Look and Feel. Other interfaces include Windows, Motif and Mac, if supported on your platform. The following figure shows TCTool running in a JAI application.



In general, TCTool consists of four sections: The top menu bar, a control section with statistics, a JVM memory monitor graph and a JAI operator event logger. The description of TCTool will start at the upper left hand corner and flow left to right, top to bottom.

So, starting with the Options menu, users can select from the following:

Refit Window - if the window is manually resized, this option will restore the window to its original size.

Max Memory - this option controls the maximum range for the cache memory slider. The value can range from 32MB to 512MB

Time Delay - specifies the update rate for the JVM memory monitor. The default is 500ms. Values can range from 10ms to 5000ms.

Event Rows - specifies the number of rows to display in the JAI event logger. the default is 7 rows, selectable from 1, 4, 7, 10, 15 and 20

Look & Feel - specifies the user interface style. The default is Java Metal. options include Windows, Motif and Mac. Some of these may not be available on a particular platform.

Moving to the right, Diagnostics can be toggled on and off. The tile cache can be enabled or disabled.

In the next section, buttons are provided to force a garbage collection, flush (empty) the JAI tile cache, reset all counters or quit the TCTool. A slider is provided to increase/decrease the maximum range of selectable memory for the tile cache. This slider is used to adjust the current tile cache memory capacity. The next slider is used to set the memory threshold. When the cache fills up beyond this level, tiles are subject to removal via the memory controller. The statistics section contains a vertical bar showing the percentage of the tile cache memory being used. Dark blue indicates empty or unused space, while red indicates used memory. This display is dynamically updated and shows the activity of the tile cache. If the memory controller is activated because the tile cache is filling up, it may indicate that increasing the tile cache size will improve performance. A number of items are kept as a running tally, these include:

Tile Count: - the total number of tiles in the cache at a given time
Cache Hits - requests for tiles that are already cached (good)
Cache Misses - tiles not cached which must be computed (costly)
Add Tile - a request was made to add a tile to the cache
Remove Tile - a request was made to remove a tile from the cache
Remove (flush) - the tile cache removed tiles because a flush was performed
Remove (control) - memory exceeded the memory threshold, so tiles were dereferenced
Update (addTile) - a request was made to add a tile that is already in the cache (good)
Update (getTile) - a request for a tile was made that was already in the cache (good)
Used Memory (%) - percentage of tile cache memory currently in use

The next section consists of a strip chart recorder style graph of the runtime JVM heap memory used by the application and TCTool. The horizontal axis represents time and the vertical axis represents percentage of currently used memory which is the ratio of used memory to allocated memory. Since the amount of allocated memory can change, the graph will rescale itself to maintain its range of 0 to 100%. Each graph grid represents increments of 10%. The total allocated memory, used memory and percentage are provided in text form and updated along with the graph. A vertical bar graph is provided along the right side of the plot which also shows percentage of memory used. The rate at which samples are updated can be adjusted from 10ms to 5000ms. The default update rate is set to 500ms.

The bottom section of TCTool consists of the JAI operator event logger. This scrollable table shows which JAI operator triggered an event, what the event was, how big the tile is and the tile's time stamp. The most recent entries are inserted at the top of the table. Larger values of the time stamp represent tiles which have been most recently accessed. This list can become quite long, so the Reset button will empty the table.

Again, the purpose of TCTool is to help JAI developers identify and optimize their program's use of Sun's tile caching features.

TIP: In general, applications that use very short op-chains may be able to use a small tile cache size. If an application never fills the tile cache, reducing the cache memory capacity may be of benefit.

Release Notes:

It is also possible to instantiate TCTool by just using
TCTool tctool = new tilecachetool.TCTool();

The event logging list is inverted, new entries are inserted into the beginning of the list instead of appended to the end. This allows the list to grow without requiring the user to scroll the list to see the latest entry.

New features and enhancements need to be added with care so that the tool does not impact the monitoring of an application. At some point, the tool itself would dominate the behavior of the JVM obscuring the information pertinent to the application.

NOTE: If the TCTool cache enable/disable toggles are used, it is very important to understand what these options do. The JAI static methods JAI.enableDefaultTileCache() and JAI.disableDefaultTileCache() both are "from now on" state controllers. This means that these methods only apply to the tile cache at the point (and forward) that they are invoked. So whatever state the tile cache was in before one of these calls, will remain "as is" after the method invocation. The JAI.disableDefaultTileCache() will also flush (clear) the tile cache but pre-existing cached ops will be re-cached if their tiles are requested again. If the TCTool has been running with the cache enabled and tiles exist in the cache, toggling the cache to disabled mode will initially clear/flush the cache, but it will still monitor and reload tiles that already existed. While the cache is disabled, no new tiles will be added.