Unlocking the Secrets of Tuning Java for OpenVMS

Powell Hazzard
Java Engineering
HPWorld 2003 August

HP WORLD 2003
Solutions and Technology Conference & Expo

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- General Performance Tuning
- File Tuning
- Logical Names
- Tricks
- Questions

Java on OpenVMS

- Java
  - Sun's Java SDK is not an application but the environment for Java Applications and Applets
  - "Write Once Run Everywhere." ™ Sun
  - Java Development Kit (JDK) for OpenVMS Alpha
    - Includes Java Virtual Machine (JVM)
    - J2SDK V1.3.1 for OpenVMS Alpha V7.2-2 or later: Available now
    - J2SDK V1.4.1 for OpenVMS Alpha V7.3-1 or later: Available now
  - Additional features:
    - A POSIX threads implementation that will provide increased performance on multi-processor systems and increased robust interoperability with standards-conforming facilities such as DCE

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- General Performance Tuning
- File Tuning
- Logical Names
- Tricks
- Questions

Requirements – Hardware

Hardware	Performance		
	Best	Average	Worst
Processor Speed	Above 500MHz	500MHz	Below 500MHz
Physical Memory	1GB or higher	512MB	Minimum 256MB

More Important Prerequisites

- OpenVMS - version 7.2-2 or later
- C run-time library – must have latest ECO kit
  - For OpenVMS 7.2-2 DEC-AXPVMS-VMS722_ACRTL-V0100--4
  - For OpenVMS 7.3 DEC-AXPVMS-VMS73_ACRTL-V0200--4
  - For OpenVMS 7.3-1: DEC-AXPVMS-VMS731_SYS-V0200--1
- TCP/IP - version 5.0A or later plus mandatory patches
- Operating system – required ECO's as listed on the Java website

Agenda

- Java on OpenVMS – Overview
- Requirements
- **Managing Memory**
- General Performance Tuning
- File Tuning
- Logical Names
- Tricks
- Questions

Memory Memory Memory

- Managing available system memory is the most effective means for maximizing java application performance!

- Having enough physical memory is key because of the high burden the JVM puts on system resources.

Memory Usage and the FastVM

- Use the FastVM!

```
@sys$common:[JAVA$141.COM]JAVA$141_setup FAST
```

- Optimized for large memory systems – uses as much as 50% more memory than classic VM
- Use explicit heap initial and maximum sizes
  - Experiment with –Xmx and –Xms values for your application
  - Increase process quotas if necessary
  - If you are short on physical memory, use a smaller –Xmx value
  - Or let the FastVM calculate the heap values dynamically
  - Compute max memory
  - Initial heap = 10% of max memory
  - Max heap = 60% of max memory

Memory Usage and the FastVM (Cont.)

- For client-side applications use the –client switch

  –Xmx64m –Xglobal128m –Xgc:compacting

- Minimize paging – use MONITOR SYSTEM to check the page fault rate. Use the following :

  (1.2 * MaxJavaHeapSize) Divided By 8KB Pages to set WSMAX, WSEXTENT.

SYSGEN's WSMAX & UAF's WSEXTENT

- Programmer is freed from having to worry about deallocating memory
- Buying on credit – you will pay later…
- The Garbage Collector will come and visit ;-)
- Small working sets will cause extremely slow garbage collections. The standard Mark & Sweep technique will walk the entire heap search for objects that can be freed.
- For the best performance, WSMax & WSExtent should be large enough to hold the entire heap size (plus)
- -Xms???m

Setting Process Quotas

- Default quotas defined by OpenVMS are too small
- Web site Recommends:

Quota	Value
FILLM (UAF)	4096
CHANNELCNT (SYSGEN)	4096
WSDEF	2048
WSQUOTA	4096
WSEXT, WSMAX	16384
PGFLQUO	2097152
BYTLM	400000
BIOLM, DIOLM	150
TQELM	100

Picture is worth a thousand words

- Close your eyes – now think of an elephant driving a compact car ;-)
- Programmer is freed from having to worry about deallocating memory
- However, garbage collection can significantly slow down the execution of a program
- Most people attempt to execute a Java application using 512MB of heap in 130MB of working set
- Then, they email me “Why is my Java so slow?” ;-)

Don’t forget about PGFLQUO

- 2 X heap size is a good estimate
- Example, 128MB heap:

  (2*128*1024*1024)/512 = 524288

- Don’t forget to increase the system’s page file size accordingly

Know your objects memory usage


```
StringBuffer b = new StringBuffer();
    b.append(foo); b.append(bar); b.append(boo);
```

- What happens behind the scene
  - 4 locks
  - 3 allocations (if appending string is larger than the initial allocation)
  - 3 memory copies
- If you know the size… combine or sync

```
StringBuffer b = new StringBuffer(16*1024);
    synchronized (b) {
      b.append(foo); b.append(bar); b.append(boo); }
```

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- **General Performance Tuning**
- File Tuning
- Optimizing Parent & Child Process Communication
- Logical Names
- Questions

General Performance Tuning

- Reduce Exec/Kernel modes
  - High “Exec Mode” ? `$MONITOR SYS/ALL`
    Use `JAVA$CACHING_INTERNAL` to reduce stat() call
  - High Logical Name Translation Rate ? `$MONITOR IO`
    Use DECC$ENABLE_GETENV_CACHE to reduce `SYS$TRNLMN()`
- Limit sub-process creation
  - Expensive on OpenVMS

General Performance Tuning Cont.

- FastVM Garbage Collection optimization
  - Basic **copying** GC concepts
    - JVM heap is divided into two regions; Only one is used at any time
    - Once one region is full, all live objects are COPIED side by side into the other region
  - Basic **mark/sweep** GC concepts
    - Starting from the JVM root set, recursively MARK all live objects
    - During SWEEPING, release all dead objects leaving live objects in place
    - Compacts the gaps left by dead objects, when necessary
- **JDK 1.4.X FastVM uses *compacting GC* by default**
  - Xgc:copying
  - Xgc:compacting

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- General Performance Tuning
- File Tuning
- Logical Names
- Tricks
- Questions

File Tuning

- Reduce file name mapping
  - JAVA supports both ODS-2 and ODS-5
  - Turn on only the necessary `JAVA$FILENAME_CONTROLS` bits; less bits equals more performance
- Use ODS-5 disk
  - Install latest C RTL ECO kits and turn on the following logicals:
    - `DECC$ARGV_PARSE_STYLE`
    - `DECC$EFS_CASE_ PRESERVE`
    - `DECC$EFS_CASE_SPECIAL`
    - `DECC$EFS_CHARSET`
  - Use only UNIX-style file names

File Tuning Cont.

- File name mapping table
  - By default, all are enabled
  - To disable all, `DEFINE JAVA$FILENAME_MAPPING 0`
  - Limit use for best performance
  - Defined in `SYS$COMMON:[JAVA$141.COM]JAVA$FILENAME_CONTROLS.COM`
    - `JAVA$M_UNIX_AND_VMS`
    - `JAVA$M_HIDDEN_WITH_UNDERSCORE`
    - `JAVA$M_39_CHAR_TRUNCATE`
    - `JAVA$M_VALID_CHARS`
    - `JAVA$M_DIR_IN_NAME`
    - `JAVA$M_HIDDEN_REMOVE_DOT`
    - Several others

File Tuning Cont.

- Increase the caching interval
  - CRTL stat() call is expensive on OpenVMS
    - `File file = new File(name)`
    - `file.exists()`
    - `file.isDirectory()`
- Define `JAVA$CACHING_INTERVAL NNN`, **reduces I/O**
  - Cache is invalidated when :
    - Cache interval expires
    - Explicit action within the current application such as file open, file creation, and sub process creation
    - Actions taken on a file outside the current application will not invalidate the cache
    - Use it with `JAVA$CACHING_DIRECTORY` for additional caching

File Tuning Cont.

- Restrict file sharing
  - **JAVA$FILE_OPEN_MODE 0 (default)**
    - No file sharing, not recommended
  - **DECC$FILE_SHARING ENABLED**
    - All files are opened with full sharing enabled
    - (FAB$M_DEL | FAB$M_GET | FAB$M_PUT | FAB$M_UPD)
  - **JAVA$FILE_OPEN_MODE 3**
    - File are open for sharing with RMS buffer flushing overhead
  - **JAVA$FILE_OPEN_MODE 2**
    - Every read/write is synchronized with the disk

File Tuning Cont.

- Disable case logical
  - Java is case-sensitive, and supports ODS-2 and ODS-5
  - File.list() opens every file with .java and .class extension trying to match up the “real” java or class name!
  - Use ODS-5 disk, and set `JAVA$READDIR_CASE_DISABLE` to true (logical is unsupported)
- Limit the number of versions of files
  - For application with frequent directory look up, eliminating the old file versions will improve performance.

Rules of Thumb

- If you can cache it, cache it
- If you open it, close it
- If it is not STREAM-LF format, it will be slow
- Order Java$classpath with the most used Jar files first

  $define java$classpath small.jar, large.jar

  For faster startup and class searches:

  $define java$classpath large.jar,small.jar

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- General Performance Tuning
- File Tuning
- **Logical Names**
- Tricks
- Questions

Process creation

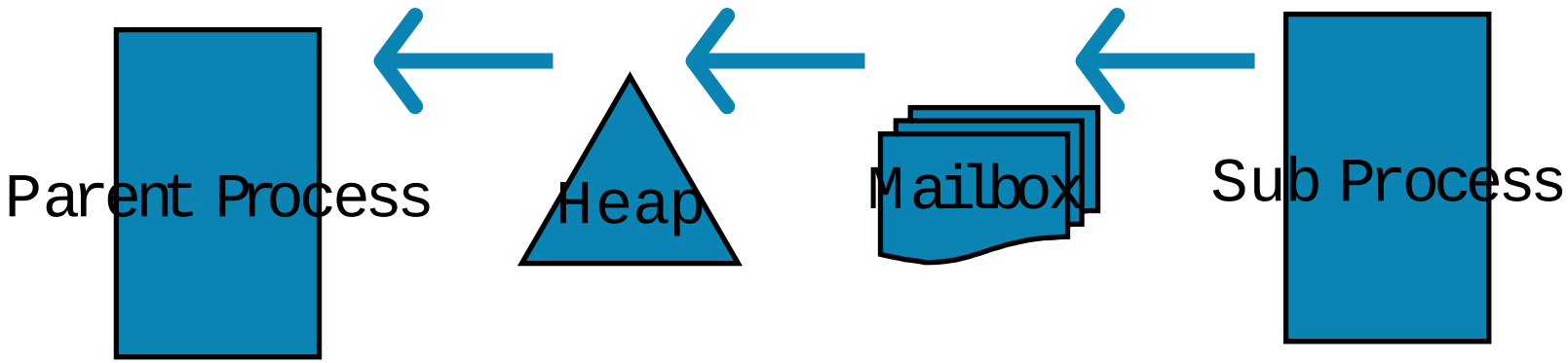
- "Write once run everywhere"
  - Nice phrase, but children do not count
- Unix Fork Vs OpenVMS subprocess creation
  - Light weight vs. heavy weight
  - Best performance - Don't fork if you don't have to
  - Tomcat would like to create a subprocess for each compile
  - Instead, do it inline
    - Save on process creation
    - Save on Java startup

Optimizing Parent Sub Processes

- IPC model between parent and child process

  Process prc = Runtime.getRuntime().exec("/dir$/child.com")
  prc.getInputStream()

- Mailbox/Heap is used for Parent Sub process IPC.
- Flow Control
  - Stops mailbox write when too many messages written
  - Bigger JAVA$FORK_MAILBOX_MESSAGES, efficient
  - But could cause RWMBX state



Optimizing Parent Sub Processes Cont.

- To disable mailbox buffering
  - `JAVA$EXEC_USE_PIPES = 1`
- Alternate Non-mailbox buffering scheme : TCP/IP Sockets
  - Define `JAVA$FORK_PIPE_STYLE` with 2
  - Uses Socket, requires a TCP/IP stack
  - Better performance, closet to UNIX pipe simulation

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- General Performance Tuning
- File Tuning
- Optimizing Parent & Child Process Communication
- Logical Names
- Questions

Java Logicals for Debugging Use

- Debugging Runtime.exec() calls
  - To show C RTL exec variant and arguments
    `DEFINE/JOB JAVA$EXEC_TRACE true`
  - To show JavaJVM arguments
    `_JAVA_LAUNCHER_DEBUG ==1`
  - `$ java –cp /mydir$/java_class spawn_sub_java 1`

```
…
option[ 0] = '-Djava.class.path=.'
option[ 1] = '-Djava.class.path=/mydir$/java_class
argv[0] = `1'
cmdv args:
0: /mydir$/curdir/subjava
1: arg
```

Java Logicals for Debugging Cont.

- Debugging File Mappings
  - Useful to debug “File not found” type message or unusual file mapping such as directory name with “.”
    `DEFINE JAVA$SHOW_FILENAME_MAPPING 1`
- `JAVA$CACHING_ATEXIT_PRINT_STAT`
  - Dumping stat() cache index after the program exits
  - Use it to troubleshoot `JAVA$CACHING_INTERVAL`

Other Useful Java Logicals

- JAVA$DIRECTORY_MAPPING_NN
  - For ODS-2 only, expand directory level
  - DEFINE JAVA$DIRECTORY_MAPPING_01 "/dir/a/b/c=/dir/abc"
  - Turn on JAVA$FILENAME_CONTROL and JAVA$M_DIRECTORY_MAP bits
- Using DEC keyboard (PF1, PF2, Find, Select…)
  - DEFINE JAVA$KEYBOARD_TYPE_DEC true
- JAVAC and JAR
  - The case of file operands is determined automatically
  - JAR cvf test.jar "TestPlot.class"
  - JAR cvf test.jar "TESTPLOT.CLASS"
  - To turn off, DEFINE JAVA$OMIT_CASE_CHECK true

Other Useful Java Logicals Cont.

- **Undocumented logicalnames**
  - JAVA$CREATE_DIR_WITH_OWNER_DELETE
  - JAVA$DELETE_ALL_VERSIONS
  - JAVA$CREATE_ONE_VERSION
  - JAVA$RENAME_ALL_VERSIONS
  - JAVA$WAIT_FOR_CHILDREN NN

Other Useful Java Logicals Cont.

- VAXC$PATH
  - OpenVMS search path for locating .EXE or .COM files
  - Runtime.getRuntime().exec("chmod") will look for chmod., chmod.com, and chmod.exe in the directories defined by VAX$PATH
- JDK 1.4.0,1.4.1 too slow on GUI applications on remote host display
  - Use –Dsun.java2d.pmoffscreen=false
- Heavy use of ASTs could starve the main Java thread
  - Set `JAVA$DAEMONIZE_MAIN_THREAD` to true
  - BridgeWorks uses JNI calls with QIO with ASTs

Other Useful Java Logicals Cont.

- JAVA$TIMED_READ_USE_QIO
  - New in 1.4.1
  - Some application use select() in a polling fashion

    Select(socket,,,timeout=10ms)

    if (timedout) do something

    if data available, read the socket
  - Now think of 100 threads doing the same logic at the same time
  - The logicals use a mix of pthread and QIOs to lessen the demand on the TCPIP stack (reducing time spend in Kernel mode)
  - Of course it might be faster if you can disable timeout

Useful DEC C-RTL Logicals

- C-RTL introduces *Feature Logicals*. Exposes JVM implementation detail (unfortunately)
- Some are in new ECO kits - get the latest C RTL ECO kit
- Use `ENABLE` or `DISABLE` to set/unset. By Default, all are disabled or set with default numeric value
- May conflict with some of the JVM logicals!
- Performance Optimizations
  - `DECC$ENABLE_GETENV_CACHE` - caches the value from getenv() call

Useful DEC C-RTL Logicals Cont.

- General UNIX Enhancements
  - DECC$ARGV_PARSE_STYLE ENABLE/DISABLE
    Preserve command line arg case when SET PROCESS/PARSE_STYLE=EXTENDED is set
  - DECC$PIPE_BUFFER_SIZE NNN
    Increase mailbox size. Default is 512 chars.
- Enhancements for UNIX-Style File Names
  - DECC$DISABLE_TO_VMS_LOGNAME_TRANSLATION
    decc$to_vms() will only treat the first element of a UNIX style name as a logical name if there is a leading slash "/"
  - DECC$EFS_CHARSET
    JAVA applications can contain ODS-5 extended characters
    e.g. /dir$/s[b/f txt ⇔ dir$:[s^[b]f^ .txt

Useful DEC C-RTL Logicals Cont.

- DECC$READDIR_DROPDOTNOTYPE
- DECC$RENAME_NO_INHERIT
  - When disabled, new file name inherits missing components from the old file such as device, file type, and version like DCL RENAME command
- Enhancements for UNIX-Style File Attributes
  - DECC$EFS_FILE_TIMESTAMPS
  - DECC$FILE_OWNER_UNIX
  - DECC$FILE_PERMISSION_UNIX
  - DECC$UMASK
  - DECC$FILE_SHARING

Useful DEC C-RTL Logicals Cont.

- UNIX Compliance Mode
  - DECC$FILENAME_UNIX_ONLY
  - DECC$DETACHED_CHILD_PROCESS
- File Name Handling
  - DECC$READDIR_KEEPDOTDIR
  - DECC$EFS_CASE_PRESERVE
    Case is preserved for file names on ODS-5 disk
  - DECC$EFS_CASE_SPECIAL
    Overrides DECC$EFS_CASE_PRESERVE. Case is preserved for the file names with lower case elements. All upper case file names are lower cased
- More logicals available. Refer to the C RTL Reference

Tricks

- Need to create a really long JAVA$CLASSPATH?
  - Use JAVA$LOGICAL.exe in CSWS_JAVA

```
$ mcr DKA200:[APACHE.JAKARTA.TOMCAT.bin]JAVA$LOGICAL -h
usage: java$logical [-options] logicalname [newvalue]

Options:
-c          Create a new logical with only the newvalue
-a          Append newvalue to logical name list (default)
-p          Prepend newvalue to logical name list
-d          Delete logical name
-P          Use LNM$PROCESS table
-J          Use LNM$JOB table
-G          Use LNM$GROUP table
-S          Use LNM$SYSTEM table
            (Default is LNM$FILE_DEV)
```

Tricks Cont.

- Logical name translation is normal
  - Files are going to be opened and closed
  - Search for a file down the classpath

    Example: Has a jsp file been compiled
- Use SDA to monitor logical name translations
  - Available in 7.3-1

  SDA> lnm load

  SDA> lnm start trace

  SDA> lnm start collection

  SDA> lnm show collection
- Remember, logical name translation is normal!!!!

Tricks Cont.

- Benchmark output
  - If you are doing performance testing
    - First attempt to limit output

      Disable debug statements

      Information messages
    - Don't send output to a terminal

      The system usually can output more information than the terminal can handle <XON><XOFF>, this will slow the benchmark results
- Log files not flushing in a timely manner?
  - Use the logical JAVA$FSYNC_INTERVAL
  - RMS buffers are not usually flushed to disk
  - This logical allows you to define an interval after which all pending output is flushed to the disk

Useful Websites

- http://www.hp.com/java
- http://www.openvms.compaq.com/openvms/products/ips/netbeans/

Agenda

- Java on OpenVMS – Overview
- Requirements
- Managing Memory
- General Performance Tuning
- File Tuning
- Logical Names
- Tricks
- Questions


invent

Use this sample for bullet slides that require a subtitle

Subtitles are Arial italic 28 point

- Copy and paste the above subtitle text box as needed

Interex, Encompass and HP bring you a powerful new HP World.