Discussion on CFEXECUTE on Linux/Unix

Posted At : December 9, 2004 7:07 PM | Posted By : Steven Erat
Related Categories: Java, Linux, Quality Assurance, ColdFusion, Travel

Here are some excerpts from a discussion on CFTalk-Linux regarding memory utilization when using the CFEXECUTE tag on Linux/Unix. Many reported to have observed very high memory usage whlie using CFEXECUTE, and it suspected that ColdFusion (via the JVM and the System) calls fork().

This got me thinking, so I carried out some quick, but careful tests on Linux and Solaris while monitoring memory. I did not observe any significant impact on memory utilization, but that is not to say that the reports are wrong. The key is finding out what is different between the server behavior reported in the field and the server behavior observed in these tests.

> I was wondering if anyone (Steve) could give me some more information about the performance of
> > cfexecute on Linux with MX? I was told that the cfexecute tag is implemented using fork() and
> > that each call can consume 100's of MB of memory? Is there any truth to this claim?

CFEXECUTE uses the Runtime.exec(String) method in the underlying JVM's rt.jar, on all platforms: http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Runtime.html

That in turn makes system calls I presume. I haven't yet looked at the implementation of exec().

Here's a test I just performed where I exercised CFEXECUTE by tar zipping the coldfusionmx directory, a task requiring about 20-30 seconds on my system. [CFMX 6.1, Linux]

```
[cfexecute name="/bin/tar" arguments="-czf /tmp/mycfmx.tar.gz /opt/coldfusionmx/"
timeout="600" />
```

1) Before:
System (top) - Mem: 775588k total, 538752k used
System Monitor - Used: 218m of 757m
cfusion - 8.4% MEM
screenshot

2) During:
System (top) - Mem: 775588k total, 668568k used (+129816k)
System Monitor - Used: 221m of 757m (+3m)
cfusion - 8.7% MEM (+0.3%)

screenshot

3) After:
System (top) - Mem: 775588k total, 722096k used (+183344k)
System Monitor - Used: 230m of 757m (+12m)
cfusion - 10.1% MEM (+2.7%)
screenshot

4) After 15min:
System (top) - Mem: 775588k total, 5165144k used (+183344k)
System Monitor - Used: 219m of 757m (+1m)
cfusion - 8.8% MEM (+0.4%)
screenshot

Based on this test, I don't see a problem with memory usage.

> Is there some difference between execution on Linux and Solaris?

Let's see...

Configuration

```
bash-2.03# uname -a
SunOS ps-dark 5.8 Generic_108528-19 sun4u sparc SUNW,UltraAX-i2
```

```
bash-2.03# /opt/apache/2.0.47/bin/httpd -v
Server version: Apache/2.0.47
Server built: Aug 7 2003 13:54:49
```

```
bash-2.03# ps -ef | grep cold
nobody 2941 2940 0 18:07:05 pts/3 0:48 /opt/coldfusionmx/bin/cfusion
-start default
nobody 2940 1 0 18:07:05 pts/3 0:00 /opt/coldfusionmx/bin/cfusion
-autorestart -start default
```

```
bash-2.03# /opt/coldfusionmx/runtime/bin/jrun -info
63824
4
```

```
bash-2.03# /opt/coldfusionmx/runtime/jre/bin/java -version
java version "1.4.2"
Java(TM) 2 Runtime Environment, Sta
Java HotSpot(TM) Client VM
```

Test Data

1) Before

```
load averages: 0.03, 0.16, 0.14
58 processes: 53 sleeping, 4 running, 1 on cpu
CPU states: 99.0% idle, 0.0% user, 1.0% kernel, 0.0% iowait, 0.0% swap
Memory: 1024M real, 514M free, 347M swap in use, 2416M swap free

PID USERNAME THR PRI NICE SIZE RES STATE TIME CPU COMMAND
2941 nobody 44 0 10 136M 70M sleep 0:48 1.17% cfusion
2940 nobody 4 42 0 1536K 1232K sleep 0:00 0.00% cfusion
```

2) During

```
load averages: 0.56, 0.22, 0.16
67 processes: 65 sleeping, 1 running, 1 on cpu
CPU states: 0.0% idle, 89.3% user, 9.7% kernel, 1.0% iowait, 0.0% swa
Memory: 1024M real, 502M free, 361M swap in use, 2401M swap free

PID USERNAME THR PRI NICE SIZE RES STATE TIME CPU COMMAND
2941 nobody 52 3 10 140M 76M run 1:01 42.95% cfusion
2940 nobody 4 42 0 1536K 1232K sleep 0:00 0.00% cfusion
```

3) After

```
load averages: 0.36, 0.21, 0.16
67 processes: 66 sleeping, 1 on cpu
CPU states: 99.2% idle, 0.2% user, 0.6% kernel, 0.0% iowait, 0.0% swap
Memory: 1024M real, 500M free, 361M swap in use, 2401M swap free

PID USERNAME THR PRI NICE SIZE RES STATE TIME CPU COMMAND
2941 nobody 52 29 10 139M 77M sleep 1:04 7.06% cfusion
```

Summary

CFMX start: SIZE 136M RES 70M
CFMX during: SIZE 140M RES 76M
CFMX after: SIZE 139M RES 77M

OK, this is very odd. The behavior I've observed on Solaris has been that the

> memory consumption
> during CFEXECUTE has been much higher than that. After looking into fork behavior, I concluded
> that CFMX was using fork. Your numbers don't demonstrate that. So, do you think there's some
> difference between JVMs or something else that would explain this?

I don't know what would cause a difference. If you'd like to provide a functional reproducible case and a description of the environment/configuration, then I'd be happy to carry out further experiments in order to make comparisons.

Here's what learned from a conversation with QA. On *nix, the java.lang.Runtime.exec() method in turn calls fork() on the system. The parent process then makes a duplicate copy of itself in order to create a child process. The child process inherits most of the parent's properties, including memory profile. However Unix and Linux systems do some tricks such that total system memory does not increase by a value equal to the parent process memory, but rather the OS has a single memory space (like shared memory) that is mapped to both the parent process and the child process. If the child process ever makes references a variable or other memory that is shared by the parent process then the OS will give the child process its own copy of that memory. So if the parent was at 100MB and a child was spawned, the child would show in the process list also as 100MB, however the total system memory used for both processes wouldn't necessarily be 200MB, but just the 100MB of shared memory.

Since the child process in cfexecute's case needs to run some specific binary such as /bin/tar in my example, the cfusion child process then calls the shell's exec command such that the cfusion process then becomes a tar process instead. You would see the extra cfusion process disappear and a tar process appear. The memory profile for that tar process would then go from the memory profile of cfusion process to the memory profile for a typical tar process with its own memory. If you were watching the child process you would be expected to see the process memory go from say 100MB after it forked to perhaps just 3 or 4 MB after the exec command, or however much is required to run tar. The total system memory would then show an increase of that 3 or 4 MB correlating to tar's operation, which was in fact what I saw (iirc).

For more, I think you want to lookup copy-on-write for fork, as well as the shell command exec.

See also:
Lecture notes -- Fork
man fork
Virtual Memory I, p.38 [PDF]