Why Use Java 32-bit on Ubuntu 64-bit?

> As of Nov 2016 I have am no longer endorsing or providing support for 32-bit Java on 64-bit Ubuntu. Though I have not had a chance to do performance testing, with Cloud computing, lack of documention on 32-bit Java, and hopefully Oracle has continued to tune for 64-bit, it's time to move on.

One problem with 64-bit Java is the amount of Memory it occupies:

1. As explained by Oracle, "... every native pointer in the system takes up 8 bytes instead of 4".
2. Also, we have observed that when a Java Application has access to a large amount of memory, the application gets bogged down when garbage collection occurs.

As of 2011 we feel that the majority of application profiles we have encountered do not require 64-bit Java as they often do not take advantage of using more than than 1-2GB of memory.

> This has improved due to newer algorithms such as concurrency, parallelization and generational collection. However we observe there is still significant impact.

Generally, it is also better to scale horizontally (running multiple JVMs) as opposed to using one giant JVM. This point is true for both 32-bit and 64-bit JVMs.

At least in the cloud based environments we use, where memory is a premium, 32-bit Java makes more sense.

Just to be clear, you still should use a 64-bit **Operating Systems**. A 64-bit Operating System will allow a 32-bit JVM to come closer to the theoretical 4GB memory limit and of course take full advantage of the system memory.

> We have not encountered in our own benchmarking, but here is a an example of 64-bit Java performing better than 32-bit Java on a 64-bit OS. The accepted theory here is that this is due to the emulation mode required by the 64-bit OS to run 32-bit Java. Again, we have yet to have encountered this in our usage.

JVM	Operating System	Maximum Heap Size	Reference
32-bit	32-bit Windows	1.4GB to 1.6GB	Oracle
32-bit	32-bit Solaris	2GB	Oracle
32-bit	64-bit Solaris	Approaching 4GB	Oracle
32-bit	64-bit Ubuntu		

* Due to various additional constraints such as available swap, kernel address space usage, memory fragmentation, and VM overhead, in practice the limits will vary.

So I should never use 64-bit Java?

No, the point is to closely look at your application profile, user base and hardware and determine if 32-bit or 64-bit makes sense. In some cases, 64-bit is actually faster (depending on the hardware and OS).

It has just been in our experience that often our profiles fall in,

- Application profiles use less than 4GB of memory and if they used more we use multiple JVM instances
- There is usually a 10%-15% hit in performance
- Large customer base
- Hardware with multiple CPUs handles multiple JVMs better

And let's not forget that if we want to switch to 64-bit it is simply a matter of recompiling.

Research

Overall there is a hit to performance and when using a 64-bit JVM. Here are the top level reasons,

1. Native pointer in the system takes up 8 bytes instead of 4 (per Oracle)
2. Garbage Collection Pause (per Oracle)

According to, Software - Practice & Experience archive Volume 36 Issue 1, January 2006,

> *We conclude that the space an object takes in the heap in 64-bit mode is 39.3% larger on average than in 32-bit mode. We identify three reasons for this: (i) the larger pointer size, (ii) the increased header and (iii) the increased alignment. The minimally required heap size is 51.1% larger on average in 64-bit than in 32-bit mode. From our experimental setup using hardware*

> *performance monitors, we observe that 64-bit computing typically results in a significantly larger number of data cache misses at all levels of the memory hierarchy. In addition, we observe that when a sufficiently large heap is available, the IBM JDK 1.4.0 VM is 1.7% slower on average in 64-bit mode than in 32-bit mode.*

Garbage collection may result in pauses in the application,

> *Remember, however, that this additional heap must be garbage collected at various points in your application's life span. This additional garbage collection can cause large pauses in your Java application if you do not take this into consideration.*

More details explaining the mechanics can be put here or linked in another document.

Good review of differences and pros and cons - http://java.dzone.com/articles/should-i-use-32-or-64-bit-jvm