

# Re: Massive performance loss from OS::sleep hack

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2007-09/msg00038.html>

---

- *From:* "Kip Macy" <[kip.macy@xxxxxxxxxx](mailto:kip.macy@xxxxxxxxxx)>
  - *Date:* Sat, 15 Sep 2007 22:25:29 -0700
- 

Or more likely they'll continue to maintain a sched\_yield that isn't posix compliant. We may just want to add some sort of interface so the jvm can tell the kernel that sched\_yield should be non-compliant for the current process.

On 9/15/07, Daniel Eischen <[deischen@xxxxxxxxxx](mailto:deischen@xxxxxxxxxx)> wrote:

On Sat, 15 Sep 2007, Kurt Miller wrote:

Hello Kris,

I recall why I added the os\_sleep() call. While working on the certification

testing one of the JCK tests was deadlocking. The test itself was faulty in

that it created a high priority thread in a tight yield loop. Since pthread\_yield() on a single processor system will not yield to lower priority threads, the higher priority thread effectively blocked the lower priority thread from running and the test deadlocked.

I filed a formal appeal to have the JCK test corrected. However since the api states:

[http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Thread.html#yield\(\)](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Thread.html#yield())  
"Causes the currently executing thread object to temporarily pause and allow other threads to execute."

the appeal was denied. I further argued that many publications written by or or-authored by Sun employees state that Thread.yield() can not be relied upon in this fashion:

[ ... ]

It's odd that Sun would take this position since the POSIX behavior for `sched_yield()` is:

The `sched_yield()` function shall force the running thread to relinquish the processor until it again becomes the head of its thread list. It takes no arguments.

The "its thread list" mentioned above is the list of threads for its given priority. POSIX has a notion of a list of threads for each given priority; in `SCHED_RR` and `SCHED_FIFO` scheduling the higher priority threads will always run before lower priority threads.

Sun's defined Java behavior, or their interpretation, of `Thread.yield()` is