

# Re: Multitasking and containers

---

*Source:* <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2006-11/msg00431.html>

---

- *From:* "Dmitry A. Kazakov" <[mailbox@xxxxxxxxxxxxxxxxxxxxx](mailto:mailbox@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 28 Nov 2006 19:21:42 +0100
- 

On 28 Nov 2006 09:12:27 -0800, Matthew Heaney wrote:

Dmitry A. Kazakov wrote:

2. Using a protected object's procedure/entry would kill concurrency by serialization of the action to undertake.

There is a difference between "synchronizing access to a shared resource" and "waiting for a resource to become available".

Calling a protected function or procedure is an example of the former. Calling a protected procedure would hardly "kill concurrency". In a monitor there is only synchronization. (I think it's the case that the task stays in a running state.)

Ah, you mean: in absence of preemptive scheduling.

Calling a protected entry whose barrier condition is false is an example of the latter. If the barrier condition were false this would mean the task waits (it transitions to a blocked state). I would be loathe to say that that would "kill" concurrency since in typical designs that's exactly what the task is supposed to do.

No difference if the above premise holds, i.e. no task switches as long as the barrier is closed.

But, the above is true *\*only\** for a single-CPU system. So for a truly parallel system it could become a problem. Dual-cores aren't that expensive in these days.

Further, even on a single CPU, where protected functions and procedures are equivalent, the requirement "no task switches while lock held" might be unacceptable if you hold it for too long. Searching a container within a protected action ... well, one should be a quite strong believer for this.

Re: Multitasking and containers

I wouldn't dismiss it completely, but I definitely don't like it. For hashes I would at least take one with an external hash function computed outside the protected action.

--

Regards,  
Dmitry A. Kazakov  
<http://www.dmitry-kazakov.de>

.