

Re: [parisc-linux] [patch 15/23] Add cmpxchg_local to parisc

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-08/msg11777.html>

- *From:* Grant Grundler <grundler@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 28 Aug 2007 11:27:46 -0600
-

On Tue, Aug 28, 2007 at 07:50:18AM -0400, Mathieu Desnoyers wrote:

...

A few questions/nits:

- o Did you attempt quantify how many places in the kernel could use this? I'm just trying to get a feel for how useful this really is vs just using existing mechanisms (that people understand) to implement a non-SMP-safe counter that protects updates (writes) against interrupts. If you did, adding some references to local_ops.txt would be helpful so folks could look for examples of "correct usage".

Good question. Since it is useful to implement fast, interrupt reentrant, counters of any kind without disabling interrupts, I think it could be vastly used in the kernel. I also use it in my LTTng kernel tracer implementation to provide very fast buffer management. It is used in LTTng, but could be used for most kind of buffering management too; meaning that we could manage buffers without disabling interrupts.

So I don't expect to come with an "upper bound" about where it can be used...

Ok...so I'll try to find one in 2.6.22.5:

```
grundler <1855>find -name \*.c | xargs fgrep DEFINE_PER_CPU | fgrep atomic_t
./arch/s390/kernel/time.c:static DEFINE_PER_CPU(atomic_t, etr_sync_word);
grundler <1856>find -name \*.c | xargs fgrep DEFINE_PER_CPU | fgrep local_t
./arch/x86_64/kernel/nmi.c:static DEFINE_PER_CPU(local_t, alert_counter);
```

uhm, I was expecting more than that. Maybe there is some other systemic problem with how PER_CPU stuff is used/declared?

In any case, some references to LTT usage would be quite helpful. E.g. a list of file and variable names at the end of local_ops.txt file.

Re: [parisc-linux] [patch 15/23] Add cmpxchg_local to parisc

o How can a local_t counter protect updates (writes) against interrupts but not preemption?

I always thought preemption required some sort of interrupt or trap. Maybe the local_ops.txt explains that and I just missed it.

"Local atomic operations only guarantee variable modification atomicity wrt the CPU which owns the data. Therefore, care must be taken to make sure that only one CPU writes to the local_t data. This is done by using per cpu data and making sure that we modify it from within a preemption safe context." -> therefore, preemption must be disabled around local ops usage. This is required to be pinned to one CPU anyway.

Sorry...the quoted text doesn't answer my question. It's a definition of semantics, not an explanation of the "mechanics".

I want to know what happens when (if?) an interrupt occurs in the middle of a read/modify/write sequence that isn't prefixed with LOCK (or something similar for other arches like "store locked conditional" ops).

Stating the semantics is a good thing – but not a substitution for describing how it works for a given architecture. Either in the code or in local_ops.txt. Otherwise people like me won't use it because we don't believe that (or understand how) it really works.

DaveM explained updates "in flight" would not be visible to interrupts and I suspect that's the answer to my question....but then I don't "feel good" the local_ops are safe to update in interrupts _and_ the process context kernel. Maybe the relationship between local_ops, preemption, and interrupts could be explained more carefully in local_ops.txt.

Does the paragraph above explain it enough or should I add some more explanation ?

Please add a bit more detail. If DaveM is correct (he normally is), then there must be limits on how the local_t can be used in the kernel process and interrupt contexts. I'd like those rules spelled out very clearly since it's easy to get wrong and tracking down such a bug is quite painful.

Note: I already missed the one critical sentence about only the "owning" CPU can write the value....there seem to be other limitations as well with respect to interrupts.

Re: [parisc-linux] [patch 15/23] Add cmpxchg_local to parisc

Re: [parisc-linux] [patch 15/23] Add cmpxchg_local to parisc

o OK to add a reference for local_ops.txt to atomic_ops.txt?
They are obviously related and anyone "discovering" one of the docs should be made aware of the other.
Patch+log entry appended below. Please sign-off if that's ok with you.

I'm perfectly ok with the idea, but suggest a small modification. See below.

Looks fine to me. Add your "Signed-off-by" and submit to DaveM since he seems to be the maintainer of atomic_ops.txt.

cheers,
grant

thanks,
grant

Diff+Commit entry against 2.6.22.5:

local_t is a variant of atomic_t and has related ops to match.
Add reference for local_t documentation to atomic_ops.txt.

Signed-off-by: Grant Grundler <grundler@xxxxxxxxxxxxxxxxxxx>

```
--- 2.6.22.5-ORIG/Documentation/atomic_ops.txt 2007-08-27
22:50:27.000000000 -0700
+++ 2.6.22.5-ggg/Documentation/atomic_ops.txt 2007-08-27
22:54:44.000000000 -0700
@@ -14,6 +14,10 @@
```

```
typedef struct { volatile int counter; } atomic_t;
```

+local_t is very similar to atomic_t. If the counter is per CPU and only
+updated by one CPU, local_t is probably more appropriate. Please see
+Documentation/local_ops.txt for the semantics of local_t.

+
The first operations to implement for atomic_t's are the
initializers and plain reads.

The text snippet is good, but I am not sure it belongs between the
description of atomic_t type and its initializers. What if we do

Re: [parisc-linux] [patch 15/23] Add cmpxchg_local to parisc

something like: (with context, I tried to explain the distinction between atomic_t and local_t some more)

Semantics and Behavior of Atomic and Bitmask Operations

David S. Miller

This document is intended to serve as a guide to Linux port maintainers on how to implement atomic counter, bitops, and spinlock interfaces properly.

atomic_t should be used to provide a type with update primitives executed atomically from any CPU. If the counter is per CPU and only updated by one CPU, local_t is probably more appropriate. Please see Documentation/local_ops.txt for the semantics of local_t.

The atomic_t type should be defined as a signed integer. Also, it should be made opaque such that any kind of cast to a normal C integer type will fail. Something like the following should suffice:

Mathieu

--

Mathieu Desnoyers

Computer Engineering Ph.D. Student, Ecole Polytechnique de Montreal

OpenPGP key fingerprint: 8CD5 52C3 8E3C 4140 715F BA06 3F25 A8FE 3BAE 9A68

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>