Subject: Re: [PATCH v6 2/2] decrement static keys on real destroy time
Posted by akpm on Tue, 22 May 2012 23:11:07 GMT
View Forum Message <> Reply to Message

On Tue, 22 May 2012 15:46:10 -0700
Andrew Morton <akpm@linux-foundation.org> wrote:

> > +static inline bool memcg_proto_active(struct cg_proto *cg_proto)
> > +{
> > + return cg_proto->flags & (1 << MEMCG_SOCK_ACTIVE);
> > +}
> > +
> > +static inline bool memcg_proto_activated(struct cg_proto *cg_proto)
> > +{
> > + return cg_proto->flags & (1 << MEMCG_SOCK_ACTIVATED);
> > +}
>
> Here, we're open-coding kinda-test_bit(). Why do that? These flags are
> modified with set_bit() and friends, so we should read them with the
> matching test_bit()?
>
> Also, these bool-returning functions will return values other than 0
> and 1. That probably works OK and I don't know what the C standards
> and implementations do about this. But it seems unclean and slightly
> risky to have a "bool" value of 32! Converting these functions to use
> test_bit() fixes this - test_bit() returns only 0 or 1.
>
> test_bit() is slightly more expensive than the above. If this is
> considered to be an issue then I guess we could continue to use this
> approach. But I do think a code comment is needed, explaining and
> justifying the unusual decision to bypass the bitops API. Also these
> functions should tell the truth and return an "int" type.

Joe corrected (and informed) me:

: 6.3.1.2p1:
:
: "When any scalar value is converted to _Bool, the result is 0
: if the value compares equal to 0; otherwise, the result is 1."

So the above functions will be given compiler-generated scalar-to-boolean
conversion.

test_bit() already does internal scalar-to-boolean conversion. The
compiler doesn't know that, so if we convert the above functions to use
test_bit(), we'll end up performing scalar-to-boolean-to-boolean
conversion, which is dumb.

I assume that a way of fixing this is to change test_bit() to return bool type.  That's a bit scary.

A less scary way would be to add a new

 bool test_bit_bool(int nr, const unsigned long *addr);

which internally calls test_bit() but somehow avoids the compiler-generated conversion of the test_bit() return value into a bool.  I haven't actually thought of a way of doing this ;)

---