Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace
Posted by Pierre Peiffer on Fri, 23 Nov 2007 10:49:11 GMT
View Forum Message <> Reply to Message

Ok, I have the patch ready, but before sending it, I worry about the size of struct ipc_namespace if we mark struct ipc_ids as ___cacheline_aligned....

Of course, you we fall into a classical match: performance vs memory size.

As I don't think that I have the knowledge to decide what we must focus on, here after is, for info, the size reported by pahole (on x86, Intel Xeon)

With the patch sent at the beginning of this thread we have:

```
struct ipc_namespace {
        struct kref             kref;           /*     0     4 */
        struct ipc_ids          ids[3];         /*     4   156 */
        /* --- cacheline 2 boundary (128 bytes) was 32 bytes ago --- */
        int                     sem_ctls[4];    /*   160    16 */
        int                     used_sems;      /*   176     4 */
        int                     msg_ctlmax;     /*   180     4 */
        int                     msg_ctlmnb;     /*   184     4 */
        int                     msg_ctlmni;     /*   188     4 */
        /* --- cacheline 3 boundary (192 bytes) --- */
        atomic_t                msg_bytes;      /*   192     4 */
        atomic_t                msg_hdrs;       /*   196     4 */
        size_t                  shm_ctlmax;     /*   200     4 */
        size_t                  shm_ctlall;     /*   204     4 */
        int                     shm_ctlmni;     /*   208     4 */
        int                     shm_tot;        /*   212     4 */

        /* size: 216, cachelines: 4 */
        /* last cacheline: 24 bytes */
};      /* definitions: 1 */
```

With the new patch, if we mark the struct ipc_ids as ____cacheline_aligned, we have (I put kref at the end, to save one more cacheline):

```
struct ipc_namespace {
        struct ipc_ids          sem_ids;        /*     0    64 */

        /* XXX last struct has 12 bytes of padding */

        /* --- cacheline 1 boundary (64 bytes) --- */
        int                     sem_ctls[4];    /*    64    16 */
        int                     used_sems;      /*    80     4 */
```

```
        /* XXX 44 bytes hole, try to pack */

        /* --- cacheline 2 boundary (128 bytes) --- */
        struct ipc_ids          msg_ids;          /*   128    64 */

        /* XXX last struct has 12 bytes of padding */

        /* --- cacheline 3 boundary (192 bytes) --- */
        int                     msg_ctlmax;       /*   192     4 */
        int                     msg_ctlmnb;       /*   196     4 */
        int                     msg_ctlmni;       /*   200     4 */
        atomic_t                msg_bytes;        /*   204     4 */
        atomic_t                msg_hdrs;         /*   208     4 */

        /* XXX 44 bytes hole, try to pack */

        /* --- cacheline 4 boundary (256 bytes) --- */
        struct ipc_ids          shm_ids;          /*   256    64 */

        /* XXX last struct has 12 bytes of padding */

        /* --- cacheline 5 boundary (320 bytes) --- */
        size_t                  shm_ctlmax;       /*   320     4 */
        size_t                  shm_ctlall;       /*   324     4 */
        int                     shm_ctlmni;       /*   328     4 */
        int                     shm_tot;          /*   332     4 */
        struct kref             kref;             /*   336     4 */

        /* size: 384, cachelines: 6 */
        /* sum members: 252, holes: 2, sum holes: 88 */
        /* padding: 44 */
        /* paddings: 3, sum paddings: 36 */
};      /* definitions: 1 */
```

We can put all sysctl related values together, in one cacheline and keep ipc_ids cacheline aligned ? But I really wonder about the performance gain here...

Humm humm, comment ?

P.

Pavel Emelyanov wrote:
> Pierre Peiffer wrote:
>> Hi,
>>
>> Thanks for reviewing this !
>>
>> Pavel Emelyanov wrote:

>>> Pavel Emelyanov wrote:
>>>> Cedric Le Goater wrote:
>>>>> Pierre Peiffer wrote:
>>> [snip]
>>>
>>>>> Pavel, what do you think of it ?
>>>> Looks sane, good catch, Pierre.
>>>>
>>>> But I'd find out whether these three ipc_ids intersect any
>>>> cache-line. In other words I'd mark the struct ipc_ids as
>>>> ____cacheline_aligned and checked for any differences.
>>> BTW! It might be also useful to keep ipc_ids closer to their
>>> sysctl parameters.
>>>
>> It makes sense indeed.
>>
>> That would mean to have something like this, right ?
>
> Yup :)
>
>> struct ipc_namespace {
>> struct kref kref;
>>
>> struct ipc_ids sem_ids;
>> int sem_ctls[4];
>> int used_sems;
>>
>> struct ipc_ids msg_ids;
>> int msg_ctlmax;
>> int msg_ctlmnb;
>> int msg_ctlmni;
>> atomic_t msg_bytes;
>> atomic_t msg_hdrs;
>>
>> struct ipc_ids shm_ids;
>> size_t shm_ctlmax;
>> size_t shm_ctlall;
>> int shm_ctlmni;
>> int shm_tot;
>> };
>>
>> After a quick look, that implies to rework a little bit procfs... othwise, it's
>> not a big deal as I can see.
>
> Thanks!
>
>> P.
>>

>>>>> Acked-by: Cedric Le Goater <clg@fr.ibm.com>
>>>>>
>>>>> Thanks,
>>>> Thanks,
>>>> Pavel
>>>>
>>>>> C.
>>> [snip]
>>> -
>>> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
>>> the body of a message to majordomo@vger.kernel.org
>>> More majordomo info at http://vger.kernel.org/majordomo-info.html
>>> Please read the FAQ at http://www.tux.org/lkml/
>>>
>
>

--
Pierre Peiffer

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers