Subject: [PATCH v5] slab: Ignore internal flags in cache creation
Posted by Glauber Costa on Wed, 17 Oct 2012 11:36:51 GMT
View Forum Message <> Reply to Message

Some flags are used internally by the allocators for management
purposes. One example of that is the CFLGS_OFF_SLAB flag that slab uses
to mark that the metadata for that cache is stored outside of the slab.

No cache should ever pass those as a creation flags. We can just ignore
this bit if it happens to be passed (such as when duplicating a cache in
the kmem memcg patches).

Because such flags can vary from allocator to allocator, we allow them
to make their own decisions on that, defining SLAB_AVAILABLE_FLAGS with
all flags that are valid at creation time. Allocators that doesn't have
any specific flag requirement should define that to mean all flags.

Common code will mask out all flags not belonging to that set.

[v2: leave the mask out decision up to the allocators]
[v3: define flags for all allocators]
[v4: move all definitions to slab.h]

Signed-off-by: Glauber Costa <glommer@parallels.com>
Acked-by: Christoph Lameter <cl@linux.com>
CC: David Rientjes <rientjes@google.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
---
 mm/slab.c | 22 ----------------------
 mm/slab.h | 25 +++++++++++++++++++++++++
 mm/slab_common.c | 7 +++++++
 mm/slub.c | 3 ---
 4 files changed, 32 insertions(+), 25 deletions(-)

diff --git a/mm/slab.c b/mm/slab.c
index 8524923..8c1d447 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -162,23 +162,6 @@
  */
 static bool pfmemalloc_active __read_mostly;

-/* Legal flag mask for kmem_cache_create(). */
-#if DEBUG
-# define CREATE_MASK (SLAB_RED_ZONE | \
-    SLAB_POISON | SLAB_HWCACHE_ALIGN | \
-    SLAB_CACHE_DMA | \
-    SLAB_STORE_USER | \

```
-    SLAB_RECLAIM_ACCOUNT | SLAB_PANIC | \
-    SLAB_DESTROY_BY_RCU | SLAB_MEM_SPREAD | \
-    SLAB_DEBUG_OBJECTS | SLAB_NOLEAKTRACE | SLAB_NOTRACK)
-#else
-# define CREATE_MASK (SLAB_HWCACHE_ALIGN | \
-    SLAB_CACHE_DMA | \
-    SLAB_RECLAIM_ACCOUNT | SLAB_PANIC | \
-    SLAB_DESTROY_BY_RCU | SLAB_MEM_SPREAD | \
-    SLAB_DEBUG_OBJECTS | SLAB_NOLEAKTRACE | SLAB_NOTRACK)
-#endif
-
 /*
  * kmem_bufctl_t:
  *
@@ -2385,11 +2368,6 @@ __kmem_cache_create (struct kmem_cache *cachep, unsigned long
flags)
  if (flags & SLAB_DESTROY_BY_RCU)
   BUG_ON(flags & SLAB_POISON);
 #endif
- /*
-  * Always checks flags, a caller might be expecting debug support which
-  * isn't available.
-  */
- BUG_ON(flags & ~CREATE_MASK);

  /*
   * Check that size is in terms of words.  This is needed to avoid
diff --git a/mm/slab.h b/mm/slab.h
index 7deeb44..35b60b7 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -45,6 +45,31 @@ static inline struct kmem_cache *__kmem_cache_alias(const char *name,
size_t siz
 #endif


+/* Legal flag mask for kmem_cache_create(), for various configurations */
+#define SLAB_CORE_FLAGS (SLAB_HWCACHE_ALIGN | SLAB_CACHE_DMA |
SLAB_PANIC | \
+    SLAB_DESTROY_BY_RCU | SLAB_DEBUG_OBJECTS )
+
+#if defined(CONFIG_DEBUG_SLAB)
+#define SLAB_DEBUG_FLAGS (SLAB_RED_ZONE | SLAB_POISON | SLAB_STORE_USER)
+#elif defined(CONFIG_SLUB_DEBUG)
+#define SLAB_DEBUG_FLAGS (SLAB_RED_ZONE | SLAB_POISON | SLAB_STORE_USER | \
+     SLAB_TRACE | SLAB_DEBUG_FREE)
+#else
+#define SLAB_DEBUG_FLAGS (0)
```

```
+#endif
+
+#if defined(CONFIG_SLAB)
+#define SLAB_CACHE_FLAGS (SLAB_MEM_SPREAD | SLAB_NOLEAKTRACE | \
+     SLAB_RECLAIM_ACCOUNT | SLAB_TEMPORARY | SLAB_NOTRACK)
+#elif defined(CONFIG_SLUB)
+#define SLAB_CACHE_FLAGS (SLAB_NOLEAKTRACE | SLAB_RECLAIM_ACCOUNT | \
+     SLAB_TEMPORARY | SLAB_NOTRACK)
+#else
+#define SLAB_CACHE_FLAGS (0)
+#endif
+
+#define CACHE_CREATE_MASK (SLAB_CORE_FLAGS | SLAB_DEBUG_FLAGS |
SLAB_CACHE_FLAGS)
+
 int __kmem_cache_shutdown(struct kmem_cache *);

 #endif
diff --git a/mm/slab_common.c b/mm/slab_common.c
index 9c21725..0e2b8e3 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -107,6 +107,13 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t
size, size_t align
  if (!kmem_cache_sanity_check(name, size) == 0)
   goto out_locked;

+ /*
+  * Some allocators will constraint the set of valid flags to a subset
+  * of all flags. We expect them to define CACHE_CREATE_MASK in this
+  * case, and we'll just provide them with a sanitized version of the
+  * passed flags.
+  */
+ flags &= CACHE_CREATE_MASK;

  s = __kmem_cache_alias(name, size, align, flags, ctor);
  if (s)
diff --git a/mm/slub.c b/mm/slub.c
index 2258ed8..7197f141 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -112,9 +112,6 @@
  *     the fast path and disables lockless freelists.
  */

-#define SLAB_DEBUG_FLAGS (SLAB_RED_ZONE | SLAB_POISON | SLAB_STORE_USER | \
-  SLAB_TRACE | SLAB_DEBUG_FREE)
-
```

```
 static inline int kmem_cache_debug(struct kmem_cache *s)
 {
 #ifdef CONFIG_SLUB_DEBUG
--
1.7.11.7
```

Subject: Re: [PATCH v5] slab: Ignore internal flags in cache creation
Posted by David Rientjes on Wed, 17 Oct 2012 21:07:55 GMT
View Forum Message <> Reply to Message

On Wed, 17 Oct 2012, Glauber Costa wrote:

> Some flags are used internally by the allocators for management
> purposes. One example of that is the CFLGS_OFF_SLAB flag that slab uses
> to mark that the metadata for that cache is stored outside of the slab.
>
> No cache should ever pass those as a creation flags. We can just ignore
> this bit if it happens to be passed (such as when duplicating a cache in
> the kmem memcg patches).
>
> Because such flags can vary from allocator to allocator, we allow them
> to make their own decisions on that, defining SLAB_AVAILABLE_FLAGS with
> all flags that are valid at creation time. Allocators that doesn't have
> any specific flag requirement should define that to mean all flags.
>
> Common code will mask out all flags not belonging to that set.
>
> [v2: leave the mask out decision up to the allocators]
> [v3: define flags for all allocators]
> [v4: move all definitions to slab.h]
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> Acked-by: Christoph Lameter <cl@linux.com>
> CC: David Rientjes <rientjes@google.com>
> CC: Pekka Enberg <penberg@cs.helsinki.fi>

Acked-by: David Rientjes <rientjes@google.com>

Subject: Re: [PATCH v5] slab: Ignore internal flags in cache creation
Posted by akpm on Thu, 18 Oct 2012 22:42:03 GMT
View Forum Message <> Reply to Message

On Wed, 17 Oct 2012 15:36:51 +0400
Glauber Costa <glommer@parallels.com> wrote:

> Some flags are used internally by the allocators for management
> purposes. One example of that is the CFLGS_OFF_SLAB flag that slab uses
> to mark that the metadata for that cache is stored outside of the slab.
>
> No cache should ever pass those as a creation flags. We can just ignore
> this bit if it happens to be passed (such as when duplicating a cache in
> the kmem memcg patches).

I may be minunderstanding this, but...

If some caller to kmem_cache_create() is passing in bogus flags then
that's a bug, and it is undesirable to hide such a bug in this fashion?

> Because such flags can vary from allocator to allocator, we allow them
> to make their own decisions on that, defining SLAB_AVAILABLE_FLAGS with
> all flags that are valid at creation time. Allocators that doesn't have
> any specific flag requirement should define that to mean all flags.
>
> Common code will mask out all flags not belonging to that set.

Subject: Re: [PATCH v5] slab: Ignore internal flags in cache creation
Posted by Glauber Costa on Fri, 19 Oct 2012 09:32:43 GMT
View Forum Message <> Reply to Message

On 10/19/2012 02:42 AM, Andrew Morton wrote:
> On Wed, 17 Oct 2012 15:36:51 +0400
> Glauber Costa <glommer@parallels.com> wrote:
>
>> Some flags are used internally by the allocators for management
>> purposes. One example of that is the CFLGS_OFF_SLAB flag that slab uses
>> to mark that the metadata for that cache is stored outside of the slab.
>>
>> No cache should ever pass those as a creation flags. We can just ignore
>> this bit if it happens to be passed (such as when duplicating a cache in
>> the kmem memcg patches).
>
> I may be minunderstanding this, but...
>
> If some caller to kmem_cache_create() is passing in bogus flags then
> that's a bug, and it is undesirable to hide such a bug in this fashion?
>

Not necessarily.

This part is part of the kmemcg-slab series. In that use case, I copy
the flags from the original kmem cache, and create a duplicate. That
duplicate need to have the same flags, but only the creation flags.

We had many attempts to mask it out in different places, and after some discussion, it seemed best to independently do it from common code in slab_common.c at creation time. It gets quite independent from the kmemcg-slab this way, and so I posted independently to reduce my churn

---

Subject: Re: [PATCH v5] slab: Ignore internal flags in cache creation
Posted by Pekka Enberg on Wed, 31 Oct 2012 07:13:49 GMT
View Forum Message <> Reply to Message

On Thu, Oct 18, 2012 at 12:07 AM, David Rientjes <rientjes@google.com> wrote:
> On Wed, 17 Oct 2012, Glauber Costa wrote:
>
>> Some flags are used internally by the allocators for management
>> purposes. One example of that is the CFLGS_OFF_SLAB flag that slab uses
>> to mark that the metadata for that cache is stored outside of the slab.
>>
>> No cache should ever pass those as a creation flags. We can just ignore
>> this bit if it happens to be passed (such as when duplicating a cache in
>> the kmem memcg patches).
>>
>> Because such flags can vary from allocator to allocator, we allow them
>> to make their own decisions on that, defining SLAB_AVAILABLE_FLAGS with
>> all flags that are valid at creation time. Allocators that doesn't have
>> any specific flag requirement should define that to mean all flags.
>>
>> Common code will mask out all flags not belonging to that set.
>>
>> [v2: leave the mask out decision up to the allocators]
>> [v3: define flags for all allocators]
>> [v4: move all definitions to slab.h]
>>
>> Signed-off-by: Glauber Costa <glommer@parallels.com>
>> Acked-by: Christoph Lameter <cl@linux.com>
>> CC: David Rientjes <rientjes@google.com>
>> CC: Pekka Enberg <penberg@cs.helsinki.fi>
>
> Acked-by: David Rientjes <rientjes@google.com>

Applied, thanks!
