Subject: Re: [PATCH 2/6] BC: beancounters core (API)
Posted by dev on Wed, 23 Aug 2006 13:46:45 GMT
View Forum Message <> Reply to Message

Alexey Dobriyan wrote:
> On Wed, Aug 23, 2006 at 03:03:07PM +0400, Kirill Korotaev wrote:
>
>
>>--- /dev/null
>>+++ ./include/bc/beancounter.h
>
>
>>+#define BC_RESOURCES 0
>
>
> Do you want userspace to see it?
yep.

>>+struct bc_resource_parm {
>>+ unsigned long barrier; /* A barrier over which resource allocations
>>+ * are failed gracefully. e.g. if the amount
>>+ * of consumed memory is over the barrier
>>+ * further sbrk() or mmap() calls fail, the
>>+ * existing processes are not killed.
>>+ */
>>+ unsigned long limit; /* hard resource limit */
>>+ unsigned long held; /* consumed resources */
>>+ unsigned long maxheld; /* maximum amount of consumed resources */
>>+ unsigned long minheld; /* minumum amount of consumed resources */
>
>
> Stupid question: when minimum amount is useful?
to monitor usage statistics (range of used resources).
this value will be usefull when ubstat will be added.
this field probably would be more logical to add later,
but since it is part of user space interface it is left here
for not changing API later.

>>+/*
>>+ * Kernel internal part.
>>+ */
>
>
> Redundant comment, YMMV.
>
>
>>+#ifdef __KERNEL__
>>+

>>+#include <linux/config.h>
>
>
> config.h is unneeded. You can drop it from everywhere.
ok.

>>+struct beancounter
>>+{
>
>
> nit:
> "struct beancounter {"
>
>
>>+ atomic_t bc_refcount;
>>+ spinlock_t bc_lock;
>>+ uid_t   bc_id;
>>+ struct hlist_node hash;
>>+
>>+ /* resources statistics and settings */
>>+ struct bc_resource_parm bc_parms[BC_RESOURCES];
>>+};
>>+
>>+enum severity { BC_BARRIER, BC_LIMIT, BC_FORCE };
>
>
> bc_severity?
ok

>
>
>>--- /dev/null 2006-07-18 14:52:43.075228448 +0400
>>+++ ./kernel/bc/beancounter.c 2006-08-21 13:13:11.000000000 +0400
>
>
>>+#define bc_hash_fun(x) ((((x) >> 8) ^ (x)) & (BC_HASH_SIZE - 1))
>>+
>>+struct hlist_head bc_hash[BC_HASH_SIZE];
>>+spinlock_t bc_hash_lock;
>>+
>>+EXPORT_SYMBOL(bc_hash);
>>+EXPORT_SYMBOL(bc_hash_lock);
>
>
> tasklist_lock was unexported recently and this looks equally low-level.
> I couldn't find place in patchbomb where you use it in modular fashion.
> perhaps, i need more sleep.
I answered to Andi that it is left from prev patch set which used it in proc.c

will be removed
thanks for noting this!!!

>>+void __put_beancounter(struct beancounter *bc)
>>+{
>>+ unsigned long flags;
>>+
>>+ /* equivalent to atomic_dec_and_lock_irqsave() */
>>+ local_irq_save(flags);
>>+ if (likely(!atomic_dec_and_lock(&bc->bc_refcount, &bc_hash_lock))) {
>>+ local_irq_restore(flags);
>>+ if (unlikely(atomic_read(&bc->bc_refcount) < 0))
>>+   printk(KERN_ERR "BC: Bad refcount: bc=%p, "
>>+   "luid=%d, ref=%d\n",
>>+   bc, bc->bc_id,
>>+   atomic_read(&bc->bc_refcount));
>
>
> Should this BUG_ON() ?
BUG_ON doesn't print much information :)
ok, will replace

>>+ return;
>>+ }
>>+
>>+ BUG_ON(bc == &init_bc);
>>+ verify_held(bc);
>>+ hlist_del(&bc->hash);
>>+ spin_unlock_irqrestore(&bc_hash_lock, flags);
>>+ kmem_cache_free(bc_cachep, bc);
>>+}
>>+
>>+EXPORT_SYMBOL(__put_beancounter);
>
>
>>+int bc_charge_locked(struct beancounter *bc, int resource, unsigned long
>>val,
>>+ enum severity strict)
>>+{
>>+ /*
>>+ * bc_value <= BC_MAXVALUE, value <= BC_MAXVALUE, and only one
>>addition
>>+ * at the moment is possible so an overflow is impossible.
>>+ */
>>+ bc->bc_parms[resource].held += val;
>>+
>>+ switch (strict) {
>>+ case BC_BARRIER:

>
>
> nit: one tab less
>
> -
> 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/
>