```
**********************************************************
  181699 Wed May 27 16:12:11 2015
new/usr/src/uts/common/os/kmem.c
XXXX kmem: remove a check that's always false
It's been a very long time since this code ran on systems with less than
24MB RAM.
**********************************************************
______unchanged_portion_omitted_

4327 void
4328 kmem_init(void)
4329 {
4330         kmem_cache_t *cp;
4331         int old_kmem_flags = kmem_flags;
4332         int use_large_pages = 0;
4333         size_t maxverify, minfirewall;

4335         kstat_init();

4337         /*
4338          * Small-memory systems (< 24 MB) can't handle kmem_flags overhead.
4339          */
4340         if (physmem < btop(24 << 20) && !(old_kmem_flags & KMF_STICKY))
4341                 kmem_flags = 0;

4343         /*
4338          * Don't do firewalled allocations if the heap is less than 1TB
4339          * (i.e. on a 32-bit kernel)
4340          * The resulting VM_NEXTFIT allocations would create too much
4341          * fragmentation in a small heap.
4342          */
4343 #if defined(_LP64)
4344         maxverify = minfirewall = PAGESIZE / 2;
4345 #else
4346         maxverify = minfirewall = ULONG_MAX;
4347 #endif

4349         /* LINTED */
4350         ASSERT(sizeof (kmem_cpu_cache_t) == KMEM_CPU_CACHE_SIZE);

4352         list_create(&kmem_caches, sizeof (kmem_cache_t),
4353             offsetof(kmem_cache_t, cache_link));

4355         kmem_metadata_arena = vmem_create("kmem_metadata", NULL, 0, PAGESIZE,
4356             vmem_alloc, vmem_free, heap_arena, 8 * PAGESIZE,
4357             VM_SLEEP | VMC_NO_QCACHE);

4359         kmem_msb_arena = vmem_create("kmem_msb", NULL, 0,
4360             PAGESIZE, segkmem_alloc, segkmem_free, kmem_metadata_arena, 0,
4361             VMC_DUMPSAFE | VM_SLEEP);

4363         kmem_cache_arena = vmem_create("kmem_cache", NULL, 0, KMEM_ALIGN,
4364             segkmem_alloc, segkmem_free, kmem_metadata_arena, 0, VM_SLEEP);

4366         kmem_hash_arena = vmem_create("kmem_hash", NULL, 0, KMEM_ALIGN,
4367             segkmem_alloc, segkmem_free, kmem_metadata_arena, 0, VM_SLEEP);

4369         kmem_log_arena = vmem_create("kmem_log", NULL, 0, KMEM_ALIGN,
4370             segkmem_alloc, segkmem_free, heap_arena, 0, VM_SLEEP);

4372         kmem_firewall_va_arena = vmem_create("kmem_firewall_va",
4373             NULL, 0, PAGESIZE,
4374             kmem_firewall_va_alloc, kmem_firewall_va_free, heap_arena,
4375             0, VM_SLEEP);

4377         kmem_firewall_arena = vmem_create("kmem_firewall", NULL, 0, PAGESIZE,
```



```
4378             segkmem_alloc, segkmem_free, kmem_firewall_va_arena, 0,
4379             VMC_DUMPSAFE | VM_SLEEP);

4381         /* temporary oversize arena for mod_read_system_file */
4382         kmem_oversize_arena = vmem_create("kmem_oversize", NULL, 0, PAGESIZE,
4383             segkmem_alloc, segkmem_free, heap_arena, 0, VM_SLEEP);

4385         kmem_reap_interval = 15 * hz;

4387         /*
4388          * Read /etc/system.  This is a chicken-and-egg problem because
4389          * kmem_flags may be set in /etc/system, but mod_read_system_file()
4390          * needs to use the allocator.  The simplest solution is to create
4391          * all the standard kmem caches, read /etc/system, destroy all the
4392          * caches we just created, and then create them all again in light
4393          * of the (possibly) new kmem_flags and other kmem tunables.
4394          */
4395         kmem_cache_init(1, 0);

4397         mod_read_system_file(boothowto & RB_ASKNAME);

4399         while ((cp = list_tail(&kmem_caches)) != NULL)
4400                 kmem_cache_destroy(cp);

4402         vmem_destroy(kmem_oversize_arena);

4404         if (old_kmem_flags & KMF_STICKY)
4405                 kmem_flags = old_kmem_flags;

4407         if (!(kmem_flags & KMF_AUDIT))
4408                 vmem_seg_size = offsetof(vmem_seg_t, vs_thread);

4410         if (kmem_maxverify == 0)
4411                 kmem_maxverify = maxverify;

4413         if (kmem_minfirewall == 0)
4414                 kmem_minfirewall = minfirewall;

4416         /*
4417          * give segkmem a chance to figure out if we are using large pages
4418          * for the kernel heap
4419          */
4420         use_large_pages = segkmem_lpsetup();

4422         /*
4423          * To protect against corruption, we keep the actual number of callers
4424          * KMF_LITE records seperate from the tunable.  We arbitrarily clamp
4425          * to 16, since the overhead for small buffers quickly gets out of
4426          * hand.
4427          *
4428          * The real limit would depend on the needs of the largest KMC_NOHASH
4429          * cache.
4430          */
4431         kmem_lite_count = MIN(MAX(0, kmem_lite_pcs), 16);
4432         kmem_lite_pcs = kmem_lite_count;

4434         /*
4435          * Normally, we firewall oversized allocations when possible, but
4436          * if we are using large pages for kernel memory, and we don't have
4437          * any non-LITE debugging flags set, we want to allocate oversized
4438          * buffers from large pages, and so skip the firewalling.
4439          */
4440         if (use_large_pages &&
4441             ((kmem_flags & KMF_LITE) || !(kmem_flags & KMF_DEBUG))) {
4442                 kmem_oversize_arena = vmem_xcreate("kmem_oversize", NULL, 0,
4443                     PAGESIZE, segkmem_alloc_lp, segkmem_free_lp, heap_arena,
```

```
4444                        0, VMC_DUMPSAFE | VM_SLEEP);
4445        } else {
4446                kmem_oversize_arena = vmem_create("kmem_oversize",
4447                    NULL, 0, PAGESIZE,
4448                    segkmem_alloc, segkmem_free, kmem_minfirewall < ULONG_MAX?
4449                    kmem_firewall_va_arena : heap_arena, 0, VMC_DUMPSAFE |
4450                    VM_SLEEP);
4451        }

4453        kmem_cache_init(2, use_large_pages);

4455        if (kmem_flags & (KMF_AUDIT | KMF_RANDOMIZE)) {
4456                if (kmem_transaction_log_size == 0)
4457                        kmem_transaction_log_size = kmem_maxavail() / 50;
4458                kmem_transaction_log = kmem_log_init(kmem_transaction_log_size);
4459        }

4461        if (kmem_flags & (KMF_CONTENTS | KMF_RANDOMIZE)) {
4462                if (kmem_content_log_size == 0)
4463                        kmem_content_log_size = kmem_maxavail() / 50;
4464                kmem_content_log = kmem_log_init(kmem_content_log_size);
4465        }

4467        kmem_failure_log = kmem_log_init(kmem_failure_log_size);

4469        kmem_slab_log = kmem_log_init(kmem_slab_log_size);

4471        /*
4472         * Initialize STREAMS message caches so allocb() is available.
4473         * This allows us to initialize the logging framework (cmn_err(9F),
4474         * strlog(9F), etc) so we can start recording messages.
4475         */
4476        streams_msg_init();

4478        /*
4479         * Initialize the ZSD framework in Zones so modules loaded henceforth
4480         * can register their callbacks.
4481         */
4482        zone_zsd_init();

4484        log_init();
4485        taskq_init();

4487        /*
4488         * Warn about invalid or dangerous values of kmem_flags.
4489         * Always warn about unsupported values.
4490         */
4491        if (((kmem_flags & ~(KMF_AUDIT | KMF_DEADBEEF | KMF_REDZONE |
4492            KMF_CONTENTS | KMF_LITE)) != 0) ||
4493            ((kmem_flags & KMF_LITE) && kmem_flags != KMF_LITE))
4494                cmn_err(CE_WARN, "kmem_flags set to unsupported value 0x%x. "
4495                    "See the Solaris Tunable Parameters Reference Manual.",
4496                    kmem_flags);

4498 #ifdef DEBUG
4499        if ((kmem_flags & KMF_DEBUG) == 0)
4500                cmn_err(CE_NOTE, "kmem debugging disabled.");
4501 #else
4502        /*
4503         * For non-debug kernels, the only "normal" flags are 0, KMF_LITE,
4504         * KMF_REDZONE, and KMF_CONTENTS (the last because it is only enabled
4505         * if KMF_AUDIT is set). We should warn the user about the performance
4506         * penalty of KMF_AUDIT or KMF_DEADBEEF if they are set and KMF_LITE
4507         * isn't set (since that disables AUDIT).
4508         */
4509        if (!(kmem_flags & KMF_LITE) &&
```



```
4510            (kmem_flags & (KMF_AUDIT | KMF_DEADBEEF)) != 0)
4511                cmn_err(CE_WARN, "High-overhead kmem debugging features "
4512                    "enabled (kmem_flags = 0x%x).  Performance degradation "
4513                    "and large memory overhead possible. See the Solaris "
4514                    "Tunable Parameters Reference Manual.", kmem_flags);
4515 #endif /* not DEBUG */

4517        kmem_cache_applyall(kmem_cache_magazine_enable, NULL, TQ_SLEEP);

4519        kmem_ready = 1;

4521        /*
4522         * Initialize the platform-specific aligned/DMA memory allocator.
4523         */
4524        ka_init();

4526        /*
4527         * Initialize 32-bit ID cache.
4528         */
4529        id32_init();

4531        /*
4532         * Initialize the networking stack so modules loaded can
4533         * register their callbacks.
4534         */
4535        netstack_init();
4536 }
```

_____***unchanged_portion_omitted_***