```
**********************************************************
   35641 Fri May  8 18:04:44 2015
new/usr/src/uts/common/vm/seg_kp.c
use NULL getmemid segop as a shorthand for ENODEV
Instead of forcing every segment driver to implement a dummy function to
return (hopefully) ENODEV, handle NULL getmemid segop function pointer as
"return ENODEV" shorthand.
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
  23  */

  25 /* Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
  26 /*       All Rights Reserved   */

  28 /*
  29  * Portions of this source code were derived from Berkeley 4.3 BSD
  30  * under license from the Regents of the University of California.
  31  */

  33 /*
  34  * segkp is a segment driver that administers the allocation and deallocation
  35  * of pageable variable size chunks of kernel virtual address space. Each
  36  * allocated resource is page-aligned.
  37  *
  38  * The user may specify whether the resource should be initialized to 0,
  39  * include a redzone, or locked in memory.
  40  */

  42 #include <sys/types.h>
  43 #include <sys/t_lock.h>
  44 #include <sys/thread.h>
  45 #include <sys/param.h>
  46 #include <sys/errno.h>
  47 #include <sys/sysmacros.h>
  48 #include <sys/systm.h>
  49 #include <sys/buf.h>
  50 #include <sys/mman.h>
  51 #include <sys/vnode.h>
  52 #include <sys/cmn_err.h>
  53 #include <sys/swap.h>
  54 #include <sys/tuneable.h>
  55 #include <sys/kmem.h>
  56 #include <sys/vmem.h>
  57 #include <sys/cred.h>
  58 #include <sys/dumphdr.h>
```



```
  59 #include <sys/debug.h>
  60 #include <sys/vtrace.h>
  61 #include <sys/stack.h>
  62 #include <sys/atomic.h>
  63 #include <sys/archsystm.h>
  64 #include <sys/lgrp.h>

  66 #include <vm/as.h>
  67 #include <vm/seg.h>
  68 #include <vm/seg_kp.h>
  69 #include <vm/seg_kmem.h>
  70 #include <vm/anon.h>
  71 #include <vm/page.h>
  72 #include <vm/hat.h>
  73 #include <sys/bitmap.h>

  75 /*
  76  * Private seg op routines
  77  */
  78 static void     segkp_dump(struct seg *seg);
  79 static int      segkp_checkprot(struct seg *seg, caddr_t addr, size_t len,
  80                         uint_t prot);
  81 static int      segkp_kluster(struct seg *seg, caddr_t addr, ssize_t delta);
  82 static int      segkp_pagelock(struct seg *seg, caddr_t addr, size_t len,
  83                         struct page ***page, enum lock_type type,
  84                         enum seg_rw rw);
  85 static void     segkp_insert(struct seg *seg, struct segkp_data *kpd);
  86 static void     segkp_delete(struct seg *seg, struct segkp_data *kpd);
  87 static caddr_t  segkp_get_internal(struct seg *seg, size_t len, uint_t flags,
  88                         struct segkp_data **tkpd, struct anon_map *amp);
  89 static void     segkp_release_internal(struct seg *seg,
  90                         struct segkp_data *kpd, size_t len);
  91 static int      segkp_unlock(struct hat *hat, struct seg *seg, caddr_t vaddr,
  92                         size_t len, struct segkp_data *kpd, uint_t flags);
  93 static int      segkp_load(struct hat *hat, struct seg *seg, caddr_t vaddr,
  94                         size_t len, struct segkp_data *kpd, uint_t flags);
  95 static struct   segkp_data *segkp_find(struct seg *seg, caddr_t vaddr);
  96 static int      segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);

  97 /*
  98  * Lock used to protect the hash table(s) and caches.
  99  */
 100 static kmutex_t segkp_lock;

 102 /*
 103  * The segkp caches
 104  */
 105 static struct segkp_cache segkp_cache[SEGKP_MAX_CACHE];

 107 /*
 108  * When there are fewer than red_minavail bytes left on the stack,
 109  * segkp_map_red() will map in the redzone (if called).  5000 seems
 110  * to work reasonably well...
 111  */
 112 long            red_minavail = 5000;

 114 /*
 115  * will be set to 1 for 32 bit x86 systems only, in startup.c
 116  */
 117 int     segkp_fromheap = 0;
 118 ulong_t *segkp_bitmap;

 120 /*
 121  * If segkp_map_red() is called with the redzone already mapped and
 122  * with less than RED_DEEP_THRESHOLD bytes available on the stack,
 123  * then the stack situation has become quite serious;  if much more stack
```

```
 124  * is consumed, we have the potential of scrogging the next thread/LWP
 125  * structure.  To help debug the "can't happen" panics which may
 126  * result from this condition, we record hrestime and the calling thread
 127  * in red_deep_hires and red_deep_thread respectively.
 128  */
 129 #define RED_DEEP_THRESHOLD      2000

 131 hrtime_t        red_deep_hires;
 132 kthread_t       *red_deep_thread;

 134 uint32_t        red_nmapped;
 135 uint32_t        red_closest = UINT_MAX;
 136 uint32_t        red_ndoubles;

 138 pgcnt_t anon_segkp_pages_locked;        /* See vm/anon.h */
 139 pgcnt_t anon_segkp_pages_resv;          /* anon reserved by seg_kp */

 141 static struct   seg_ops segkp_ops = {
 142         .fault          = segkp_fault,
 143         .checkprot      = segkp_checkprot,
 144         .kluster        = segkp_kluster,
 145         .dump           = segkp_dump,
 146         .pagelock       = segkp_pagelock,
 148         .getmemid       = segkp_getmemid,
 147 };
_____unchanged_portion_omitted_

1354 /*ARGSUSED*/
1355 static int
1356 segkp_pagelock(struct seg *seg, caddr_t addr, size_t len,
1357     struct page ***ppp, enum lock_type type, enum seg_rw rw)
1358 {
1359         return (ENOTSUP);
1362 }

1364 /*ARGSUSED*/
1365 static int
1366 segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
1367 {
1368         return (ENODEV);
1360 }
_____unchanged_portion_omitted_
```

```
**********************************************************
    9329 Fri May  8 18:04:44 2015
new/usr/src/uts/common/vm/seg_kpm.c
use NULL getmemid segop as a shorthand for ENODEV
Instead of forcing every segment driver to implement a dummy function to
return (hopefully) ENODEV, handle NULL getmemid segop function pointer as
"return ENODEV" shorthand.
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License, Version 1.0 only
   6  * (the "License").  You may not use this file except in compliance
   7  * with the License.
   8  *
   9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  10  * or http://www.opensolaris.org/os/licensing.
  11  * See the License for the specific language governing permissions
  12  * and limitations under the License.
  13  *
  14  * When distributing Covered Code, include this CDDL HEADER in each
  15  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  16  * If applicable, add the following below this CDDL HEADER, with the
  17  * fields enclosed by brackets "[]" replaced with your own identifying
  18  * information: Portions Copyright [yyyy] [name of copyright owner]
  19  *
  20  * CDDL HEADER END
  21  */
  22 /*
  23  * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
  24  * Use is subject to license terms.
  25  */

  27 /*
  28  * Kernel Physical Mapping (kpm) segment driver (segkpm).
  29  *
  30  * This driver delivers along with the hat_kpm* interfaces an alternative
  31  * mechanism for kernel mappings within the 64-bit Solaris operating system,
  32  * which allows the mapping of all physical memory into the kernel address
  33  * space at once. This is feasible in 64 bit kernels, e.g. for Ultrasparc II
  34  * and beyond processors, since the available VA range is much larger than
  35  * possible physical memory. Momentarily all physical memory is supported,
  36  * that is represented by the list of memory segments (memsegs).
  37  *
  38  * Segkpm mappings have also very low overhead and large pages are used
  39  * (when possible) to minimize the TLB and TSB footprint. It is also
  40  * extentable for other than Sparc architectures (e.g. AMD64). Main
  41  * advantage is the avoidance of the TLB-shootdown X-calls, which are
  42  * normally needed when a kernel (global) mapping has to be removed.
  43  *
  44  * First example of a kernel facility that uses the segkpm mapping scheme
  45  * is seg_map, where it is used as an alternative to hat_memload().
  46  * See also hat layer for more information about the hat_kpm* routines.
  47  * The kpm facilty can be turned off at boot time (e.g. /etc/system).
  48  */

  50 #include <sys/types.h>
  51 #include <sys/param.h>
  52 #include <sys/sysmacros.h>
  53 #include <sys/systm.h>
  54 #include <sys/vnode.h>
  55 #include <sys/cmn_err.h>
  56 #include <sys/debug.h>
  57 #include <sys/thread.h>
  58 #include <sys/cpuvar.h>
```



```
  59 #include <sys/bitmap.h>
  60 #include <sys/atomic.h>
  61 #include <sys/lgrp.h>

  63 #include <vm/seg_kmem.h>
  64 #include <vm/seg_kpm.h>
  65 #include <vm/hat.h>
  66 #include <vm/as.h>
  67 #include <vm/seg.h>
  68 #include <vm/page.h>

  70 /*
  71  * Global kpm controls.
  72  * See also platform and mmu specific controls.
  73  *
  74  * kpm_enable -- global on/off switch for segkpm.
  75  * . Set by default on 64bit platforms that have kpm support.
  76  * . Will be disabled from platform layer if not supported.
  77  * . Can be disabled via /etc/system.
  78  *
  79  * kpm_smallpages -- use only regular/system pagesize for kpm mappings.
  80  * . Can be useful for critical debugging of kpm clients.
  81  * . Set to zero by default for platforms that support kpm large pages.
  82  *   The use of kpm large pages reduces the footprint of kpm meta data
  83  *   and has all the other advantages of using large pages (e.g TLB
  84  *   miss reduction).
  85  * . Set by default for platforms that don't support kpm large pages or
  86  *   where large pages cannot be used for other reasons (e.g. there are
  87  *   only few full associative TLB entries available for large pages).
  88  *
  89  * segmap_kpm -- separate on/off switch for segmap using segkpm:
  90  * . Set by default.
  91  * . Will be disabled when kpm_enable is zero.
  92  * . Will be disabled when MAXBSIZE != PAGESIZE.
  93  * . Can be disabled via /etc/system.
  94  *
  95  */
  96 int kpm_enable = 1;
  97 int kpm_smallpages = 0;
  98 int segmap_kpm = 1;

 100 /*
 101  * Private seg op routines.
 102  */
 103 faultcode_t segkpm_fault(struct hat *hat, struct seg *seg, caddr_t addr,
 104                         size_t len, enum fault_type type, enum seg_rw rw);
 105 static void     segkpm_dump(struct seg *);
 106 static int      segkpm_pagelock(struct seg *seg, caddr_t addr, size_t len,
 107                         struct page ***page, enum lock_type type,
 108                         enum seg_rw rw);

 110 static struct seg_ops segkpm_ops = {
 111         .fault          = segkpm_fault,
 112         .dump           = segkpm_dump,
 113         .pagelock       = segkpm_pagelock,
 114 //#ifndef SEGKPM_SUPPORT
 115 #if 0
 116 #error FIXME: define nop
 117         .dup            = nop,
 118         .unmap          = nop,
 119         .free           = nop,
 120         .faulta         = nop,
 121         .setprot        = nop,
 122         .checkprot      = nop,
 123         .kluster        = nop,
 124         .sync           = nop,
```

```
 125         .incore         = nop,
 126         .lockop         = nop,
 127         .getprot        = nop,
 128         .getoffset      = nop,
 129         .gettype        = nop,
 130         .getvp          = nop,
 131         .advise         = nop,
 132         .setpagesize    = nop,
 133         .getmemid       = nop,
 133         .getpolicy      = nop,
 134 #endif
 135 };
______unchanged_portion_omitted_
```

****
90456 Fri May 8 18:04:44 2015
new/usr/src/uts/common/vm/vm_as.c
use NULL getmemid segop as a shorthand for ENODEV
Instead of forcing every segment driver to implement a dummy function to return (hopefully) ENODEV, handle NULL getmemid segop function pointer as "return ENODEV" shorthand.
****
_____***unchanged_portion_omitted_***

```
3510 /*
3511  * return memory object ID
3512  */
3513 int
3514 as_getmemid(struct as *as, caddr_t addr, memid_t *memidp)
3515 {
3516         struct seg      *seg;
3517         int             sts;

3519         AS_LOCK_ENTER(as, &as->a_lock, RW_READER);
3520         seg = as_segat(as, addr);
3521         if (seg == NULL) {
3522                 AS_LOCK_EXIT(as, &as->a_lock);
3523                 return (EFAULT);
3524         }
3525         /*
3526          * catch old drivers which may not support getmemid
3527          */
3528         if (seg->s_ops->getmemid == NULL) {
3529                 AS_LOCK_EXIT(as, &as->a_lock);
3530                 return (ENODEV);
3531         }

3526         sts = segop_getmemid(seg, addr, memidp);

3528         AS_LOCK_EXIT(as, &as->a_lock);
3529         return (sts);
3530 }
```

_____***unchanged_portion_omitted_***

****
55132 Fri May 8 18:04:45 2015
new/usr/src/uts/common/vm/vm_seg.c
use NULL getmemid segop as a shorthand for ENODEV
Instead of forcing every segment driver to implement a dummy function to return (hopefully) ENODEV, handle NULL getmemid segop function pointer as "return ENODEV" shorthand.
****

_____***unchanged_portion_omitted_***

```
2014 int
2015 segop_getmemid(struct seg *seg, caddr_t addr, memid_t *mp)
2016 {
2017         if (seg->s_ops->getmemid == NULL)
2018                 return (ENODEV);
2017         VERIFY3P(seg->s_ops->getmemid, !=, NULL);

2020         return (seg->s_ops->getmemid(seg, addr, mp));
2021 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   11889 Fri May  8 18:04:45 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
use NULL getmemid segop as a shorthand for ENODEV
Instead of forcing every segment driver to implement a dummy function to
return (hopefully) ENODEV, handle NULL getmemid segop function pointer as
"return ENODEV" shorthand.
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T */
/* All Rights Reserved */

/*
 * Portions of this source code were derived from Berkeley 4.3 BSD
 * under license from the Regents of the University of California.
 */

/*
 * VM - segment for non-faulting loads.
 */

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/param.h>
#include <sys/mman.h>
#include <sys/errno.h>
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/vnode.h>
#include <sys/proc.h>
#include <sys/conf.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/lgrp.h>

#include <vm/page.h>
#include <vm/hat.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/vpage.h>

/*
```



```
 * Private seg op routines.
 */
static int      segnf_dup(struct seg *seg, struct seg *newseg);
static int      segnf_unmap(struct seg *seg, caddr_t addr, size_t len);
static void     segnf_free(struct seg *seg);
static faultcode_t segnf_nomap(void);
static int      segnf_setprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t prot);
static int      segnf_checkprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t prot);
static int      segnf_nop(void);
static int      segnf_getprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t *protv);
static u_offset_t segnf_getoffset(struct seg *seg, caddr_t addr);
static int      segnf_gettype(struct seg *seg, caddr_t addr);
static int      segnf_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
static void     segnf_dump(struct seg *seg);
static int      segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
                    struct page ***ppp, enum lock_type type, enum seg_rw rw);
static int      segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
                    uint_t szc);
static int      segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);


struct seg_ops segnf_ops = {
        .dup            = segnf_dup,
        .unmap          = segnf_unmap,
        .free           = segnf_free,
        .fault          = (faultcode_t (*)(struct hat *, struct seg *, caddr_t,
            size_t, enum fault_type, enum seg_rw))segnf_nomap,
        .faulta         = (faultcode_t (*)(struct seg *, caddr_t)) segnf_nomap,
        .setprot        = segnf_setprot,
        .checkprot      = segnf_checkprot,
        .sync           = (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
                segnf_nop,
        .incore         = (size_t (*)(struct seg *, caddr_t, size_t, char *))
                segnf_nop,
        .lockop         = (int (*)(struct seg *, caddr_t, size_t, int, int,
            ulong_t *, size_t))segnf_nop,
        .getprot        = segnf_getprot,
        .getoffset      = segnf_getoffset,
        .gettype        = segnf_gettype,
        .getvp          = segnf_getvp,
        .advise         = (int (*)(struct seg *, caddr_t, size_t, uint_t))
                segnf_nop,
        .dump           = segnf_dump,
        .pagelock       = segnf_pagelock,
        .setpagesize    = segnf_setpagesize,
        .getmemid       = segnf_getmemid,
};
```

```
_____unchanged_portion_omitted_

/*ARGSUSED*/
static int
segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
    uint_t szc)
{
        return (ENOTSUP);
}

/*ARGSUSED*/
static int
segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
{
        return (ENODEV);
}
```