

```

*****
113669 Tue Nov 24 09:35:00 2015
new/usr/src/uts/common/vm/seg_dev.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
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 */

22 /*
23  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

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

30 /*
31  * University Copyright- Copyright (c) 1982, 1986, 1988
32  * The Regents of the University of California
33  * All Rights Reserved
34  *
35  * University Acknowledgment- Portions of this document are derived from
36  * software developed by the University of California, Berkeley, and its
37  * contributors.
38  */

40 /*
41  * VM - segment of a mapped device.
42  *
43  * This segment driver is used when mapping character special devices.
44  */

46 #include <sys/types.h>
47 #include <sys/t_lock.h>
48 #include <sys/sysmacros.h>
49 #include <sys/vtrace.h>
50 #include <sys/system.h>
51 #include <sys/vmsystem.h>
52 #include <sys/mman.h>
53 #include <sys/errno.h>
54 #include <sys/kmem.h>
55 #include <sys/cmn_err.h>
56 #include <sys/vnode.h>
57 #include <sys/proc.h>
58 #include <sys/conf.h>
59 #include <sys/debug.h>
60 #include <sys/ddidevmap.h>
61 #include <sys/ddi_implfuncs.h>

```

```

62 #include <sys/lgrp.h>

64 #include <vm/page.h>
65 #include <vm/hat.h>
66 #include <vm/as.h>
67 #include <vm/seg.h>
68 #include <vm/seg_dev.h>
69 #include <vm/seg_kp.h>
70 #include <vm/seg_kmem.h>
71 #include <vm/vpage.h>

73 #include <sys/sunddi.h>
74 #include <sys/esunddi.h>
75 #include <sys/fs/snodel.h>

78 #if DEBUG
79 int segdev_debug;
80 #define DEBUGF(level, args) { if (segdev_debug >= (level)) cmn_err args; }
81 #else
82 #define DEBUGF(level, args)
83 #endif

85 /* Default timeout for devmap context management */
86 #define CTX_TIMEOUT_VALUE 0

88 #define HOLD_DHP_LOCK(dhp) if (dhp->dh_flags & DEVMAP_ALLOW_REMAP) \
89     { mutex_enter(&dhp->dh_lock); }

91 #define RELE_DHP_LOCK(dhp) if (dhp->dh_flags & DEVMAP_ALLOW_REMAP) \
92     { mutex_exit(&dhp->dh_lock); }

94 #define round_down_p2(a, s) ((a) & ~((s) - 1))
95 #define round_up_p2(a, s) (((a) + (s) - 1) & ~((s) - 1))

97 /*
98  * VA_PA_ALIGNED checks to see if both VA and PA are on pgsz boundary
99  * VA_PA_PGSIZE_ALIGNED check to see if VA is aligned with PA w.r.t. pgsz
100 */
101 #define VA_PA_ALIGNED(uvaddr, paddr, pgsz) \
102     (((uvaddr | paddr) & (pgsz - 1)) == 0)
103 #define VA_PA_PGSIZE_ALIGNED(uvaddr, paddr, pgsz) \
104     (((uvaddr ^ paddr) & (pgsz - 1)) == 0)

106 #define vpgtob(n) ((n) * sizeof(struct vpage)) /* For brevity */

108 #define VTOCVP(vp) (VTOS(vp)->s_commonvp) /* we "know" it's an snode */

110 static struct devmap_ctx *devmapctx_list = NULL;
111 static struct devmap_softlock *devmap_slist = NULL;

113 /*
114  * mutex, vnode and page for the page of zeros we use for the trash mappings.
115  * One trash page is allocated on the first ddi_umem_setup call that uses it
116  * XXX Eventually, we may want to combine this with what segnf does when all
117  * hat layers implement HAT_NOFAULT.
118  *
119  * The trash page is used when the backing store for a userland mapping is
120  * removed but the application semantics do not take kindly to a SIGBUS.
121  * In that scenario, the applications pages are mapped to some dummy page
122  * which returns garbage on read and writes go into a common place.
123  * (Perfect for NO_FAULT semantics)
124  * The device driver is responsible to communicating to the app with some
125  * other mechanism that such remapping has happened and the app should take
126  * corrective action.
127  * We can also use an anonymous memory page as there is no requirement to

```

```

128 * keep the page locked, however this complicates the fault code. RFE.
129 */
130 static struct vnode trashvp;
131 static struct page *trashpp;

133 /* Non-pageable kernel memory is allocated from the umem_np_arena. */
134 static vmem_t *umem_np_arena;

136 /* Set the cookie to a value we know will never be a valid umem_cookie */
137 #define DEVMAP_DEVMEM_COOKIE ((ddi_umem_cookie_t)0x1)

139 /*
140 * Macros to check if type of devmap handle
141 */
142 #define cookie_is_devmem(c) \
143     ((c) == (struct ddi_umem_cookie *)DEVMAP_DEVMEM_COOKIE)

145 #define cookie_is_pmem(c) \
146     ((c) == (struct ddi_umem_cookie *)DEVMAP_PMEM_COOKIE)

148 #define cookie_is_kpmem(c) (!cookie_is_devmem(c) && !cookie_is_pmem(c) && \
149     ((c)->type == KMEM_PAGEABLE))

151 #define dhp_is_devmem(dhp) \
152     (cookie_is_devmem((struct ddi_umem_cookie *)((dhp)->dh_cookie)))

154 #define dhp_is_pmem(dhp) \
155     (cookie_is_pmem((struct ddi_umem_cookie *)((dhp)->dh_cookie)))

157 #define dhp_is_kpmem(dhp) \
158     (cookie_is_kpmem((struct ddi_umem_cookie *)((dhp)->dh_cookie)))

160 /*
161 * Private seg op routines.
162 */
163 static int segdev_dup(struct seg *, struct seg *);
164 static int segdev_unmap(struct seg *, caddr_t, size_t);
165 static void segdev_free(struct seg *);
166 static faultcode_t segdev_fault(struct hat *, struct seg *, caddr_t, size_t,
167     enum fault_type, enum seg_rw);
168 static faultcode_t segdev_faulta(struct seg *, caddr_t);
169 static int segdev_setprot(struct seg *, caddr_t, size_t, uint_t);
170 static int segdev_checkprot(struct seg *, caddr_t, size_t, uint_t);
171 static void segdev_badop(void);
172 static int segdev_sync(struct seg *, caddr_t, size_t, int, uint_t);
173 static size_t segdev_incore(struct seg *, caddr_t, size_t, char *);
174 static int segdev_lockop(struct seg *, caddr_t, size_t, int, int,
175     ulong_t *, size_t);
176 static int segdev_getprot(struct seg *, caddr_t, size_t, uint_t *);
177 static u_offset_t segdev_getoffset(struct seg *, caddr_t);
178 static int segdev_gettype(struct seg *, caddr_t);
179 static int segdev_getvp(struct seg *, caddr_t, struct vnode **);
180 static int segdev_advise(struct seg *, caddr_t, size_t, uint_t);
181 static void segdev_dump(struct seg *);
182 static int segdev_pagelock(struct seg *, caddr_t, size_t,
183     struct page ***, enum lock_type, enum seg_rw);
184 static int segdev_setpagesize(struct seg *, caddr_t, size_t, uint_t);
185 static int segdev_getmemid(struct seg *, caddr_t, memid_t *);
186 static int segdev_capable(struct seg *, segcapability_t);

187 /*
188 * XXX this struct is used by rootnex_map_fault to identify
189 * the segment it has been passed. So if you make it
190 * "static" you'll need to fix rootnex_map_fault.
191 */
192 struct seg_ops segdev_ops = {

```

```

193     .dup = segdev_dup,
194     .unmap = segdev_unmap,
195     .free = segdev_free,
196     .fault = segdev_fault,
197     .faulta = segdev_faulta,
198     .setprot = segdev_setprot,
199     .checkprot = segdev_checkprot,
200     .kluster = (int (*)( ))segdev_badop,
201     .sync = segdev_sync,
202     .incore = segdev_incore,
203     .lockop = segdev_lockop,
204     .getprot = segdev_getprot,
205     .getoffset = segdev_getoffset,
206     .gettype = segdev_gettype,
207     .getvp = segdev_getvp,
208     .advise = segdev_advise,
209     .dump = segdev_dump,
210     .pagelock = segdev_pagelock,
211     .setpagesize = segdev_setpagesize,
212     .getmemid = segdev_getmemid,
213     .capable = segdev_capable,
213 };
    unchanged portion omitted_

4014 static int
4015 segdev_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
4016 {
4017     struct segdev_data *sdp = (struct segdev_data *)seg->s_data;

4019     /*
4020      * It looks as if it is always mapped shared
4021      */
4022     TRACE_0(TR_FAC_DEVMAP, TR_DEVMAP_GETMEMID,
4023         "segdev_getmemid:start");
4024     memidp->val[0] = (uintptr_t)VTOCVP(sdp->vp);
4025     memidp->val[1] = sdp->offset + (uintptr_t)(addr - seg->s_base);
4028     return (0);
4029 }

4031 /*ARGSUSED*/
4032 static int
4033 segdev_capable(struct seg *seg, segcapability_t capability)
4034 {
4026     return (0);
4027 }
    unchanged portion omitted_

```

new/usr/src/uts/common/vm/seg\_kp.c

1

```
*****
36709 Tue Nov 24 09:35:00 2015
new/usr/src/uts/common/vm/seg_kp.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
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 */
24
25 /* Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
26 /* All Rights Reserved */
27
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 */
32
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 */
41
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/system.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>
```

new/usr/src/uts/common/vm/seg\_kp.c

2

```
62 #include <sys/atomic.h>
63 #include <sys/archsystem.h>
64 #include <sys/lgrp.h>
65
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>
74
75 /*
76 * Private seg op routines
77 */
78 static void segkp_badop(void);
79 static void segkp_dump(struct seg *seg);
80 static int segkp_checkprot(struct seg *seg, caddr_t addr, size_t len,
81                          uint_t prot);
82 static int segkp_kluster(struct seg *seg, caddr_t addr, ssize_t delta);
83 static int segkp_pagelock(struct seg *seg, caddr_t addr, size_t len,
84                          struct page ***page, enum lock_type type,
85                          enum seg_rw rw);
86 static void segkp_insert(struct seg *seg, struct segkp_data *kpd);
87 static void segkp_delete(struct seg *seg, struct segkp_data *kpd);
88 static caddr_t segkp_get_internal(struct seg *seg, size_t len, uint_t flags,
89                                  struct segkp_data **tkpd, struct anon_map *amp);
90 static void segkp_release_internal(struct seg *seg,
91                                   struct segkp_data *kpd, size_t len);
92 static int segkp_unlock(struct hat *hat, struct seg *seg, caddr_t vaddr,
93                       size_t len, struct segkp_data *kpd, uint_t flags);
94 static int segkp_load(struct hat *hat, struct seg *seg, caddr_t vaddr,
95                      size_t len, struct segkp_data *kpd, uint_t flags);
96 static struct segkp_data *segkp_find(struct seg *seg, caddr_t vaddr);
97 static int segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
98 static int segkp_capable(struct seg *seg, segcapability_t capability);
99
100 /*
101 * Lock used to protect the hash table(s) and caches.
102 */
103
104 static kmutex_t segkp_lock;
105
106 /*
107 * The segkp caches
108 */
109 static struct segkp_cache segkp_cache[SEGKP_MAX_CACHE];
110
111 #define SEGKP_BADOP(t) (t(*)())segkp_badop
112
113 /*
114 * When there are fewer than red_minavail bytes left on the stack,
115 * segkp_map_red() will map in the redzone (if called). 5000 seems
116 * to work reasonably well...
117 */
118 long red_minavail = 5000;
119
120 /*
121 * will be set to 1 for 32 bit x86 systems only, in startup.c
122 */
123 int segkp_fromheap = 0;
124 ulong_t *segkp_bitmap;
125
126 /*
127 * If segkp_map_red() is called with the redzone already mapped and
128 * with less than RED_DEEP_THRESHOLD bytes available on the stack,
```

```
127 * then the stack situation has become quite serious; if much more stack
128 * is consumed, we have the potential of scrogging the next thread/LWP
129 * structure. To help debug the "can't happen" panics which may
130 * result from this condition, we record hrestime and the calling thread
131 * in red_deep_hires and red_deep_thread respectively.
132 */
133 #define RED_DEEP_THRESHOLD      2000

135 hrttime_t      red_deep_hires;
136 kthread_t     *red_deep_thread;

138 uint32_t       red_nmapped;
139 uint32_t       red_closest = UINT_MAX;
140 uint32_t       red_ndoubles;

142 pgcnt_t anon_segkp_pages_locked;      /* See vm/anon.h */
143 pgcnt_t anon_segkp_pages_resv;       /* anon reserved by seg_kp */

145 static struct seg_ops segkp_ops = {
146     .dup          = SEGKP_BADOP(int),
147     .unmap        = SEGKP_BADOP(int),
148     .free         = SEGKP_BADOP(void),
149     .fault        = segkp_fault,
150     .faulta       = SEGKP_BADOP(faultcode_t),
151     .setprot      = SEGKP_BADOP(int),
152     .checkprot    = segkp_checkprot,
153     .kluster      = segkp_kluster,
154     .swapout      = SEGKP_BADOP(size_t),
155     .sync         = SEGKP_BADOP(int),
156     .incore       = SEGKP_BADOP(size_t),
157     .lockop       = SEGKP_BADOP(int),
158     .getprot      = SEGKP_BADOP(int),
159     .getoffset    = SEGKP_BADOP(u_offset_t),
160     .gettype      = SEGKP_BADOP(int),
161     .getvp        = SEGKP_BADOP(int),
162     .advise       = SEGKP_BADOP(int),
163     .dump         = segkp_dump,
164     .pagelock     = segkp_pagelock,
165     .setpagesize  = SEGKP_BADOP(int),
166     .getmemid    = segkp_getmemid,
167     .capable      = segkp_capable,
168 };
169 unchanged_portion_omitted_

1396 /*ARGSUSED*/
1397 static int
1398 segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
1399 {
1400     return (ENODEV);
1401 }
1402 unchanged_portion_omitted_

1405 /*ARGSUSED*/
1406 static int
1407 segkp_capable(struct seg *seg, segcapability_t capability)
1408 {
1409     return (0);
1410 }
1411 unchanged_portion_omitted_
```

```

*****
9600 Tue Nov 24 09:35:00 2015
new/usr/src/uts/common/vm/seg_kpm.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
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 * extensible for other than Sparc architectures (e.g. AMD64). Main
41 * advantage is the avoidance of the TLB-shutdown 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 facility 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/system.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 void segkpm_badop(void);
107 static int segkpm_notsup(void);
108 static int segkpm_capable(struct seg *, segcapability_t);

109 #define SEGKPM_BADOP(t) (t(*)())segkpm_badop
110 #define SEGKPM_NOTSUP (int(*)())segkpm_notsup

112 static struct seg_ops segkpm_ops = {
113     .dup = SEGKPM_BADOP(int),
114     .unmap = SEGKPM_BADOP(int),
115     .free = SEGKPM_BADOP(void),
116     .fault = segkpm_fault,
117     .faulta = SEGKPM_BADOP(int),
118     .setprot = SEGKPM_BADOP(int),
119     .checkprot = SEGKPM_BADOP(int),
120     .kluster = SEGKPM_BADOP(int),
121     .swapout = SEGKPM_BADOP(size_t),
122     .sync = SEGKPM_BADOP(int),
123     .incore = SEGKPM_BADOP(size_t),
124     .lockop = SEGKPM_BADOP(int),
125     .getprot = SEGKPM_BADOP(int),
126     .getoffset = SEGKPM_BADOP(u_offset_t),

```

```
127     .gettype      = SEGKPM_BADOP(int),
128     .getvp       = SEGKPM_BADOP(int),
129     .advise      = SEGKPM_BADOP(int),
130     .dump        = segkpm_dump,
131     .pagelock    = SEGKPM_NOTSUP,
132     .setpagesize = SEGKPM_BADOP(int),
133     .getmemid    = SEGKPM_BADOP(int),
134     .getpolicy   = SEGKPM_BADOP(lgrp_mem_policy_info_t *),
136     .capable     = segkpm_capable,
135 };
```

unchanged\_portion\_omitted

```
322 /*
323  * segkpm pages are not dumped, so we just return
324  */
325 /*ARGSUSED*/
326 static void
327 segkpm_dump(struct seg *seg)
328 {}
```

```
332 /*
333  * We claim to have no special capabilities.
334  */
335 /*ARGSUSED*/
336 static int
337 segkpm_capable(struct seg *seg, segcapability_t capability)
338 {
339     return (0);
340 }
```

```

*****
57698 Tue Nov 24 09:35:00 2015
new/usr/src/uts/common/vm/seg_map.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
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 2009 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */

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

29 /*
30 * Portions of this source code were derived from Berkeley 4.3 BSD
31 * under license from the Regents of the University of California.
32 */

34 /*
35 * VM - generic vnode mapping segment.
36 *
37 * The segmap driver is used only by the kernel to get faster (than seg_vn)
38 * mappings [lower routine overhead; more persistent cache] to random
39 * vnode/offsets. Note than the kernel may (and does) use seg_vn as well.
40 */

42 #include <sys/types.h>
43 #include <sys/t_lock.h>
44 #include <sys/param.h>
45 #include <sys/sysmacros.h>
46 #include <sys/buf.h>
47 #include <sys/system.h>
48 #include <sys/vnode.h>
49 #include <sys/mman.h>
50 #include <sys/errno.h>
51 #include <sys/cred.h>
52 #include <sys/kmem.h>
53 #include <sys/vtrace.h>
54 #include <sys/cmn_err.h>
55 #include <sys/debug.h>
56 #include <sys/thread.h>
57 #include <sys/dumphdr.h>
58 #include <sys/bitmap.h>
59 #include <sys/lgrp.h>

61 #include <vm/seg_kmem.h>

```

```

62 #include <vm/hat.h>
63 #include <vm/as.h>
64 #include <vm/seg.h>
65 #include <vm/seg_kpm.h>
66 #include <vm/seg_map.h>
67 #include <vm/page.h>
68 #include <vm/pvn.h>
69 #include <vm/rm.h>

71 /*
72 * Private seg op routines.
73 */
74 static void      segmap_free(struct seg *seg);
75 faultcode_t segmap_fault(struct hat *hat, struct seg *seg, caddr_t addr,
76                          size_t len, enum fault_type type, enum seg_rw rw);
77 static faultcode_t segmap_faulta(struct seg *seg, caddr_t addr);
78 static int      segmap_checkprot(struct seg *seg, caddr_t addr, size_t len,
79                                  uint_t prot);
80 static int      segmap_kluster(struct seg *seg, caddr_t addr, ssize_t);
81 static int      segmap_getprot(struct seg *seg, caddr_t addr, size_t len,
82                                uint_t *protv);
83 static u_offset_t segmap_getoffset(struct seg *seg, caddr_t addr);
84 static int      segmap_gettype(struct seg *seg, caddr_t addr);
85 static int      segmap_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
86 static void      segmap_dump(struct seg *seg);
87 static int      segmap_pagelock(struct seg *seg, caddr_t addr, size_t len,
88                                 struct page ***ppp, enum lock_type type,
89                                 enum seg_rw rw);
90 static void      segmap_badop(void);
91 static int      segmap_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
92 static int      segmap_capable(struct seg *seg, segcapability_t capability);

93 /* segkpm support */
94 static caddr_t segmap_pagecreate_kpm(struct seg *, vnode_t *, u_offset_t,
95                                     struct smap *, enum seg_rw);
96 struct smap      *get_smap_kpm(caddr_t, page_t **);

98 #define SEGMAP_BADOP(t) (t(*)())segmap_badop

100 static struct seg_ops segmap_ops = {
101     .dup          = SEGMAP_BADOP(int),
102     .unmap        = SEGMAP_BADOP(int),
103     .free         = segmap_free,
104     .fault        = segmap_fault,
105     .faulta       = segmap_faulta,
106     .setprot      = SEGMAP_BADOP(int),
107     .checkprot    = segmap_checkprot,
108     .kluster      = segmap_kluster,
109     .swapout      = SEGMAP_BADOP(size_t),
110     .sync         = SEGMAP_BADOP(int),
111     .incore       = SEGMAP_BADOP(size_t),
112     .lockop       = SEGMAP_BADOP(int),
113     .getprot      = segmap_getprot,
114     .getoffset    = segmap_getoffset,
115     .gettype      = segmap_gettype,
116     .getvp        = segmap_getvp,
117     .advise       = SEGMAP_BADOP(int),
118     .dump         = segmap_dump,
119     .pagelock     = segmap_pagelock,
120     .setpagesize  = SEGMAP_BADOP(int),
121     .getmemid     = segmap_getmemid,
122     .capable      = segmap_capable,
122 };
    unchanged_portion_omitted

2178 static int

```

```
2179 segmap_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
2180 {
2181     struct segmap_data *smd = (struct segmap_data *)seg->s_data;

2183     memidp->val[0] = (uintptr_t)smd->smd_sm->sm_vp;
2184     memidp->val[1] = smd->smd_sm->sm_off + (uintptr_t)(addr - seg->s_base);
2187     return (0);
2188 }

2190 /*ARGSUSED*/
2191 static int
2192 segmap_capable(struct seg *seg, segcapability_t capability)
2193 {
2185     return (0);
2186 }
_____unchanged_portion_omitted_
```



```

*****
83919 Tue Nov 24 09:35:00 2015
new/usr/src/uts/common/vm/seg_spt.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
_____unchanged_portion_omitted_____

```

```

114 static int segspt_shmdup(struct seg *seg, struct seg *newseg);
115 static int segspt_shmunmap(struct seg *seg, caddr_t raddr, size_t ssize);
116 static void segspt_shmfree(struct seg *seg);
117 static faultcode_t segspt_shmfault(struct hat *hat, struct seg *seg,
118     caddr_t addr, size_t len, enum fault_type type, enum seg_rw rw);
119 static faultcode_t segspt_shmfaultra(struct seg *seg, caddr_t addr);
120 static int segspt_shmsetprot(register struct seg *seg, register caddr_t addr,
121     register size_t len, register uint_t prot);
122 static int segspt_shmcheckprot(struct seg *seg, caddr_t addr, size_t size,
123     uint_t prot);
124 static int segspt_shmkluster(struct seg *seg, caddr_t addr, ssize_t delta);
125 static size_t segspt_shmswapout(struct seg *seg);
126 static size_t segspt_shmincore(struct seg *seg, caddr_t addr, size_t len,
127     register char *vec);
128 static int segspt_shmsync(struct seg *seg, register caddr_t addr, size_t len,
129     int attr, uint_t flags);
130 static int segspt_shmlockop(struct seg *seg, caddr_t addr, size_t len,
131     int attr, int op, ulong_t *lockmap, size_t pos);
132 static int segspt_shmgetprot(struct seg *seg, caddr_t addr, size_t len,
133     uint_t *protv);
134 static u_offset_t segspt_shmgetoffset(struct seg *seg, caddr_t addr);
135 static int segspt_shmgettype(struct seg *seg, caddr_t addr);
136 static int segspt_shmgetvvp(struct seg *seg, caddr_t addr, struct vnode **vvp);
137 static int segspt_shmadvise(struct seg *seg, caddr_t addr, size_t len,
138     uint_t behav);
139 static void segspt_shmdump(struct seg *seg);
140 static int segspt_shmpagelock(struct seg *, caddr_t, size_t,
141     struct page ***, enum lock_type, enum seg_rw);
142 static int segspt_shmsetpgsz(struct seg *, caddr_t, size_t, uint_t);
143 static int segspt_shmgetmemid(struct seg *, caddr_t, memid_t *);
144 static lgrp_mem_policy_info_t *segspt_shmgetpolicy(struct seg *, caddr_t);
145 static int segspt_shmcapable(struct seg *, segcapability_t);

```

```

146 struct seg_ops segspt_shmops = {
147     .dup           = segspt_shmdup,
148     .unmap        = segspt_shmunmap,
149     .free         = segspt_shmfree,
150     .fault        = segspt_shmfault,
151     .faultra      = segspt_shmfaultra,
152     .setprot      = segspt_shmsetprot,
153     .checkprot    = segspt_shmcheckprot,
154     .kluster      = segspt_shmkluster,
155     .swapout      = segspt_shmswapout,
156     .sync         = segspt_shmsync,
157     .incore       = segspt_shmincore,
158     .lockop       = segspt_shmlockop,
159     .getprot      = segspt_shmgetprot,
160     .getoffset    = segspt_shmgetoffset,
161     .gettype      = segspt_shmgettype,
162     .getvvp       = segspt_shmgetvvp,
163     .advise       = segspt_shmadvise,
164     .dump         = segspt_shmdump,
165     .pagelock     = segspt_shmpagelock,
166     .setpagesize  = segspt_shmsetpgsz,
167     .getmemid     = segspt_shmgetmemid,
168     .getpolicy    = segspt_shmgetpolicy,
169     .capable      = segspt_shmcapable,
170 };

```

\_\_\_\_\_unchanged\_portion\_omitted\_\_\_\_\_

```

3070 /*
3071  * Get memory allocation policy info for specified address in given segment
3072  */
3073 static lgrp_mem_policy_info_t *
3074 segspt_shmgetpolicy(struct seg *seg, caddr_t addr)
3075 {
3076     struct anon_map      *amp;
3077     ulong_t              anon_index;
3078     lgrp_mem_policy_info_t *policy_info;
3079     struct shm_data      *shm_data;
3081     ASSERT(seg != NULL);
3083     /*
3084      * Get anon_map from segshm
3085      *
3086      * Assume that no lock needs to be held on anon_map, since
3087      * it should be protected by its reference count which must be
3088      * nonzero for an existing segment
3089      * Need to grab readers lock on policy tree though
3090      */
3091     shm_data = (struct shm_data *)seg->s_data;
3092     if (shm_data == NULL)
3093         return (NULL);
3094     amp = shm_data->shm_amp;
3095     ASSERT(amp->refcnt != 0);
3097     /*
3098      * Get policy info
3099      *
3100      * Assume starting anon index of 0
3101      */
3102     anon_index = seg_page(seg, addr);
3103     policy_info = lgrp_shm_policy_get(amp, anon_index, NULL, 0);
3105     return (policy_info);
3108 }
3110 /*ARGSUSED*/
3111 static int
3112 segspt_shmcapable(struct seg *seg, segcapability_t capability)
3113 {
3114     return (0);
3106 }

```

\_\_\_\_\_unchanged\_portion\_omitted\_\_\_\_\_

```

*****
285806 Tue Nov 24 09:35:01 2015
new/usr/src/uts/common/vm/seg_vn.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
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) 1986, 2010, Oracle and/or its affiliates. All rights reserved.
23 * Copyright 2015, Joyent, Inc. All rights reserved.
24 * Copyright 2015 Nexenta Systems, Inc. All rights reserved.
25 */
27 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 /*      All Rights Reserved */

30 /*
31 * University Copyright- Copyright (c) 1982, 1986, 1988
32 * The Regents of the University of California
33 * All Rights Reserved
34 *
35 * University Acknowledgment- Portions of this document are derived from
36 * software developed by the University of California, Berkeley, and its
37 * contributors.
38 */

40 /*
41 * VM - shared or copy-on-write from a vnode/anonymous memory.
42 */

44 #include <sys/types.h>
45 #include <sys/param.h>
46 #include <sys/t_lock.h>
47 #include <sys/errno.h>
48 #include <sys/system.h>
49 #include <sys/mman.h>
50 #include <sys/debug.h>
51 #include <sys/cred.h>
52 #include <sys/vmsystem.h>
53 #include <sys/tuneable.h>
54 #include <sys/bitmap.h>
55 #include <sys/swap.h>
56 #include <sys/kmem.h>
57 #include <sys/sysmacros.h>
58 #include <sys/vtrace.h>
59 #include <sys/cmn_err.h>
60 #include <sys/callb.h>
61 #include <sys/vm.h>

```

```

62 #include <sys/dumphdr.h>
63 #include <sys/lgrp.h>

65 #include <vm/hat.h>
66 #include <vm/as.h>
67 #include <vm/seg.h>
68 #include <vm/seg_vn.h>
69 #include <vm/pvn.h>
70 #include <vm/anon.h>
71 #include <vm/page.h>
72 #include <vm/vpage.h>
73 #include <sys/proc.h>
74 #include <sys/task.h>
75 #include <sys/project.h>
76 #include <sys/zone.h>
77 #include <sys/shm_impl.h>

79 /*
80 * segvn_fault needs a temporary page list array. To avoid calling kmem all
81 * the time, it creates a small (PVN_GETPAGE_NUM entry) array and uses it if
82 * it can. In the rare case when this page list is not large enough, it
83 * goes and gets a large enough array from kmem.
84 *
85 * This small page list array covers either 8 pages or 64kB worth of pages -
86 * whichever is smaller.
87 */
88 #define PVN_MAX_GETPAGE_SZ      0x10000
89 #define PVN_MAX_GETPAGE_NUM    0x8

91 #if PVN_MAX_GETPAGE_SZ > PVN_MAX_GETPAGE_NUM * PAGESIZE
92 #define PVN_GETPAGE_SZ      ptob(PVN_MAX_GETPAGE_NUM)
93 #define PVN_GETPAGE_NUM    PVN_MAX_GETPAGE_NUM
94 #else
95 #define PVN_GETPAGE_SZ      PVN_MAX_GETPAGE_SZ
96 #define PVN_GETPAGE_NUM    btop(PVN_MAX_GETPAGE_SZ)
97 #endif

99 /*
100 * Private seg op routines.
101 */
102 static int      segvn_dup(struct seg *seg, struct seg *newseg);
103 static int      segvn_unmap(struct seg *seg, caddr_t addr, size_t len);
104 static void      segvn_free(struct seg *seg);
105 static faultcode_t segvn_fault(struct hat *hat, struct seg *seg,
106                                caddr_t addr, size_t len, enum fault_type type,
107                                enum seg_rw rw);
108 static faultcode_t segvn_faulta(struct seg *seg, caddr_t addr);
109 static int      segvn_setprot(struct seg *seg, caddr_t addr,
110                                size_t len, uint_t prot);
111 static int      segvn_checkprot(struct seg *seg, caddr_t addr,
112                                size_t len, uint_t prot);
113 static int      segvn_kluster(struct seg *seg, caddr_t addr, ssize_t delta);
114 static size_t   segvn_swapout(struct seg *seg);
115 static int      segvn_sync(struct seg *seg, caddr_t addr, size_t len,
116                                int attr, uint_t flags);
117 static size_t   segvn_incore(struct seg *seg, caddr_t addr, size_t len,
118                                char *vec);
119 static int      segvn_lockop(struct seg *seg, caddr_t addr, size_t len,
120                                int attr, int op, ulong_t *lockmap, size_t pos);
121 static int      segvn_getprot(struct seg *seg, caddr_t addr, size_t len,
122                                uint_t *protv);
123 static u_offset_t segvn_getoffset(struct seg *seg, caddr_t addr);
124 static int      segvn_gettype(struct seg *seg, caddr_t addr);
125 static int      segvn_getvp(struct seg *seg, caddr_t addr,
126                                struct vnode **vpp);
127 static int      segvn_advise(struct seg *seg, caddr_t addr, size_t len,

```

```

128         uint_t behav);
129 static void      segvn_dump(struct seg *seg);
130 static int       segvn_pagelock(struct seg *seg, caddr_t addr, size_t len,
131         struct page ***ppp, enum lock_type type, enum seg_rw rw);
132 static int       segvn_setpagesize(struct seg *seg, caddr_t addr, size_t len,
133         uint_t szc);
134 static int       segvn_getmemid(struct seg *seg, caddr_t addr,
135         memid_t *memidp);
136 static lgrp_mem_policy_info_t *segvn_getpolicy(struct seg *, caddr_t);
137 static int       segvn_capable(struct seg *seg, segcapability_t capable);
137 static int       segvn_inherit(struct seg *, caddr_t, size_t, uint_t);

139 struct seg_ops segvn_ops = {
140     .dup          = segvn_dup,
141     .unmap        = segvn_unmap,
142     .free         = segvn_free,
143     .fault        = segvn_fault,
144     .faulta       = segvn_faulta,
145     .setprot      = segvn_setprot,
146     .checkprot    = segvn_checkprot,
147     .kluster      = segvn_kluster,
148     .swapout      = segvn_swapout,
149     .sync         = segvn_sync,
150     .incore       = segvn_inc core,
151     .lockop       = segvn_lockop,
152     .getprot      = segvn_getprot,
153     .getoffset    = segvn_getoffset,
154     .gettype      = segvn_gettype,
155     .getvp        = segvn_getvp,
156     .advise       = segvn_advise,
157     .dump         = segvn_dump,
158     .pagelock     = segvn_pagelock,
159     .setpagesize  = segvn_setpagesize,
160     .getmemid     = segvn_getmemid,
161     .getpolicy    = segvn_getpolicy,
163     .capable      = segvn_capable,
162     .inherit      = segvn_inherit,
163 };
_____ unchanged_portion_omitted_

```

```

9651 /*
9652  * Get memory allocation policy info for specified address in given segment
9653  */
9654 static lgrp_mem_policy_info_t *
9655 segvn_getpolicy(struct seg *seg, caddr_t addr)
9656 {
9657     struct anon_map      *amp;
9658     ulong_t              anon_index;
9659     lgrp_mem_policy_info_t *policy_info;
9660     struct segvn_data    *svn_data;
9661     u_offset_t           vn_off;
9662     vnode_t              *vp;

9664     ASSERT(seg != NULL);

9666     svn_data = (struct segvn_data *)seg->s_data;
9667     if (svn_data == NULL)
9668         return (NULL);

9670     /*
9671      * Get policy info for private or shared memory
9672      */
9673     if (svn_data->type != MAP_SHARED) {
9674         if (svn_data->tr_state != SEGVN_TR_ON) {
9675             policy_info = &svn_data->policy_info;
9676         } else {

```

```

9677         policy_info = &svn_data->tr_policy_info;
9678         ASSERT(policy_info->mem_policy ==
9679             LGRP_MEM_POLICY_NEXT_SEG);
9680     } else {
9681         amp = svn_data->amp;
9682         anon_index = svn_data->anon_index + seg_page(seg, addr);
9683         vp = svn_data->vp;
9684         vn_off = svn_data->offset + (uintptr_t)(addr - seg->s_base);
9685         policy_info = lgrp_shm_policy_get(amp, anon_index, vp, vn_off);
9686     }

9689     return (policy_info);
9692 }

9694 /*ARGSUSED*/
9695 static int
9696 segvn_capable(struct seg *seg, segcapability_t capability)
9697 {
9698     return (0);
9699 }
_____ unchanged_portion_omitted_

```

new/usr/src/uts/common/vm/vm\_seg.c

1

```
*****
54422 Tue Nov 24 09:35:01 2015
new/usr/src/uts/common/vm/vm_seg.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
_____unchanged_portion_omitted_____
```

```
1998 int
1999 segop_capable(struct seg *seg, segcapability_t cap)
2000 {
2001     if (seg->s_ops->capable == NULL)
2002         return (0);
2004 #endif /* ! codereview */
2005     return (seg->s_ops->capable(seg, cap));
2006 }
2008 int
2009 segop_inherit(struct seg *seg, caddr_t addr, size_t len, uint_t op)
2010 {
2011     if (seg->s_ops->inherit == NULL)
2012         return (ENOTSUP);
2014     return (seg->s_ops->inherit(seg, addr, len, op));
2015 }
```

```

*****
16692 Tue Nov 24 09:35:01 2015
new/usr/src/uts/i86xpv/vm/seg_mf.c
6149 use NULL capable segop as a shorthand for no-capabilities
*****
_____unchanged_portion_omitted_____

502 /*ARGSUSED*/
503 static int
504 segmf_capable(struct seg *seg, segcapability_t capability)
505 {
506     return (0);
507 }

502 /*
503  * Add a set of contiguous foreign MFNs to the segment. soft-locking them. The
504  * pre-faulting is necessary due to live migration; in particular we must
505  * return an error in response to IOCTL_PRIVCMD_MMAPPATCH rather than faulting
506  * later on a bad MFN. Whilst this isn't necessary for the other MMAP
507  * ioctl(s), we lock them too, as they should be transitory.
508  */
509 int
510 segmf_add_mfns(struct seg *seg, caddr_t addr, mfn_t mfn,
511               pgcnt_t pgcnt, domid_t domid)
512 {
513     struct segmf_data *data = seg->s_data;
514     pgcnt_t base;
515     faultcode_t fc;
516     pgcnt_t i;
517     int error = 0;

519     if (seg->s_ops != &segmf_ops)
520         return (EINVAL);

522     /*
523      * Don't mess with dom0.
524      *
525      * Only allow the domid to be set once for the segment.
526      * After that attempts to add mappings to this segment for
527      * other domains explicitly fails.
528      */

530     if (domid == 0 || domid == DOMID_SELF)
531         return (EACCES);

533     mutex_enter(&data->lock);

535     if (data->domid == 0)
536         data->domid = domid;

538     if (data->domid != domid) {
539         error = EINVAL;
540         goto out;
541     }

543     base = seg_page(seg, addr);

545     for (i = 0; i < pgcnt; i++) {
546         data->map[base + i].t_type = SEGMF_MAP_MFN;
547         data->map[base + i].u.m.m_mfn = mfn++;
548     }

550     fc = segmf_fault_range(seg->s_as->a_hat, seg, addr,
551                          pgcnt * MMU_PAGESIZE, F_SOFTLOCK, S_OTHER);

553     if (fc != 0) {

```

```

554         error = fc_decode(fc);
555         for (i = 0; i < pgcnt; i++) {
556             data->map[base + i].t_type = SEGMF_MAP_EMPTY;
557         }
558     }

560 out:
561     mutex_exit(&data->lock);
562     return (error);
563 }
_____unchanged_portion_omitted_____

746 static struct seg_ops segmf_ops = {
747     .dup           = segmf_dup,
748     .unmap        = segmf_unmap,
749     .free         = segmf_free,
750     .fault        = segmf_fault,
751     .faulta       = segmf_faulta,
752     .setprot      = segmf_setprot,
753     .checkprot    = segmf_checkprot,
754     .kluster      = segmf_kluster,
755     .sync         = segmf_sync,
756     .incore       = segmf_inc core,
757     .lockop       = segmf_lockop,
758     .getprot      = segmf_getprot,
759     .getoffset    = segmf_getoffset,
760     .gettype      = segmf_gettype,
761     .getvp        = segmf_getvp,
762     .advise       = segmf_advise,
763     .dump         = segmf_dump,
764     .pagelock     = segmf_pagelock,
765     .setpagesize  = segmf_setpagesize,
766     .getmemid     = segmf_getmemid,
774     .capable      = segmf_capable,
767 };
_____unchanged_portion_omitted_____

```