

new/usr/src/uts/common/vm/seg_dev.c

1

```
*****
114109 Fri May 8 18:03:43 2015
new/usr/src/uts/common/vm/seg_dev.c
use C99 initializers in segment ops structures
*****
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 /*
23  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */
26
27 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 /*      All Rights Reserved      */
29
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  */
39
40 /*
41  * VM - segment of a mapped device.
42  *
43  * This segment driver is used when mapping character special devices.
44  */
45
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>
```

new/usr/src/uts/common/vm/seg_dev.c

2

```
62 #include <sys/lgrp.h>
63
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>
72
73 #include <sys/sunddi.h>
74 #include <sys/esunddi.h>
75 #include <sys/fs/snodel.h>
76
77
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
84
85 /* Default timeout for devmap context management */
86 #define CTX_TIMEOUT_VALUE 0
87
88 #define HOLD_DHP_LOCK(dhp) if (dhp->dh_flags & DEVMAP_ALLOW_REMAP) \
89     { mutex_enter(&dhp->dh_lock); }
90
91 #define RELE_DHP_LOCK(dhp) if (dhp->dh_flags & DEVMAP_ALLOW_REMAP) \
92     { mutex_exit(&dhp->dh_lock); }
93
94 #define round_down_p2(a, s) ((a) & ~((s) - 1))
95 #define round_up_p2(a, s) (((a) + (s) - 1) & ~((s) - 1))
96
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)
105
106 #define vpgtob(n) ((n) * sizeof(struct vpage)) /* For brevity */
107
108 #define VTOCVP(vp) (VTOS(vp)->s_commonvp) /* we "know" it's an snode */
109
110 static struct devmap_ctx *devmapctx_list = NULL;
111 static struct devmap_softlock *devmap_slist = NULL;
112
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 lgrp_mem_policy_info_t *segdev_getpolicy(struct seg *, caddr_t);
187 static int segdev_capable(struct seg *, segcapability_t);

189 /*
190 * XXX this struct is used by rootnex_map_fault to identify
191 * the segment it has been passed. So if you make it
192 * "static" you'll need to fix rootnex_map_fault.
193 */

```

```

194 struct seg_ops segdev_ops = {
195     .dup = segdev_dup,
196     .unmap = segdev_unmap,
197     .free = segdev_free,
198     .fault = segdev_fault,
199     .faulta = segdev_faulta,
200     .setprot = segdev_setprot,
201     .checkprot = segdev_checkprot,
202     .kluster = (int (*)(()))segdev_badop,
203     .sync = segdev_sync,
204     .incore = segdev_incore,
205     .lockop = segdev_lockop,
206     .getprot = segdev_getprot,
207     .getoffset = segdev_getoffset,
208     .gettype = segdev_gettype,
209     .getvp = segdev_getvp,
210     .advise = segdev_advise,
211     .dump = segdev_dump,
212     .pagelock = segdev_pagelock,
213     .setpagesize = segdev_setpagesize,
214     .getmemid = segdev_getmemid,
215     .getpolicy = segdev_getpolicy,
216     .capable = segdev_capable,
217     .inherit = seg_inherit_notsup,
195     segdev_dup,
196     segdev_unmap,
197     segdev_free,
198     segdev_fault,
199     segdev_faulta,
200     segdev_setprot,
201     segdev_checkprot,
202     (int (*)(()))segdev_badop, /* kluster */
203     segdev_sync, /* sync */
204     segdev_incore,
205     segdev_lockop, /* lockop */
206     segdev_getprot,
207     segdev_getoffset,
208     segdev_gettype,
209     segdev_getvp,
210     segdev_advise,
211     segdev_dump,
212     segdev_pagelock,
213     segdev_setpagesize,
214     segdev_getmemid,
215     segdev_getpolicy,
216     segdev_capable,
217     seg_inherit_notsup
218 };

```

unchanged_portion_omitted

45428 Fri May 8 18:03:43 2015

new/usr/src/uts/common/vm/seg_kmem.c

use C99 initializers in segment ops structures

unchanged_portion_omitted

```
776 static struct seg_ops segkmem_ops = {
777     .dup          = SEGKMEM_BADOP(int),
778     .unmap       = SEGKMEM_BADOP(int),
779     .free        = SEGKMEM_BADOP(void),
780     .fault       = segkmem_fault,
781     .faulta     = SEGKMEM_BADOP(faultcode_t),
782     .setprot     = segkmem_setprot,
783     .checkprot  = segkmem_checkprot,
784     .kluster    = segkmem_kluster,
785     .sync       = SEGKMEM_BADOP(int),
786     .incore     = SEGKMEM_BADOP(size_t),
787     .lockop     = SEGKMEM_BADOP(int),
788     .getprot    = SEGKMEM_BADOP(int),
789     .getoffset  = SEGKMEM_BADOP(u_offset_t),
790     .gettype    = SEGKMEM_BADOP(int),
791     .getvp     = SEGKMEM_BADOP(int),
792     .advise     = SEGKMEM_BADOP(int),
793     .dump       = segkmem_dump,
794     .pagelock   = segkmem_pagelock,
795     .setpagesize = SEGKMEM_BADOP(int),
796     .getmemid   = segkmem_getmemid,
797     .getpolicy  = segkmem_getpolicy,
798     .capable    = segkmem_capable,
799     .inherit    = seg_inherit_notsup,
777     SEGKMEM_BADOP(int),          /* dup */
778     SEGKMEM_BADOP(int),          /* unmap */
779     SEGKMEM_BADOP(void),         /* free */
780     segkmem_fault,
781     SEGKMEM_BADOP(faultcode_t), /* faulta */
782     segkmem_setprot,
783     segkmem_checkprot,
784     segkmem_kluster,
785     SEGKMEM_BADOP(int),          /* sync */
786     SEGKMEM_BADOP(size_t),       /* incore */
787     SEGKMEM_BADOP(int),          /* lockop */
788     SEGKMEM_BADOP(int),          /* getprot */
789     SEGKMEM_BADOP(u_offset_t),   /* getoffset */
790     SEGKMEM_BADOP(int),          /* gettype */
791     SEGKMEM_BADOP(int),          /* getvp */
792     SEGKMEM_BADOP(int),          /* advise */
793     segkmem_dump,
794     segkmem_pagelock,
795     SEGKMEM_BADOP(int),          /* setpgsz */
796     segkmem_getmemid,
797     segkmem_getpolicy,          /* getpolicy */
798     segkmem_capable,            /* capable */
799     seg_inherit_notsup,         /* inherit */
800 };
```

unchanged_portion_omitted

new/usr/src/uts/common/vm/seg_kp.c

1

```
*****
36898 Fri May 8 18:03:43 2015
new/usr/src/uts/common/vm/seg_kp.c
use C99 initializers in segment ops structures
*****
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 lgrp_mem_policy_info_t *segkp_getpolicy(struct seg *seg,
99                                                  caddr_t addr);
100 static int segkp_capable(struct seg *seg, segcapability_t capability);
101
102 /*
103  * Lock used to protect the hash table(s) and caches.
104 */
105 static kmutex_t segkp_lock;
106
107 /*
108  * The segkp caches
109 */
110 static struct segkp_cache segkp_cache[SEGKP_MAX_CACHE];
111
112 #define SEGKP_BADOP(t) (t*())segkp_badop
113
114 /*
115  * When there are fewer than red_minavail bytes left on the stack,
116  * segkp_map_red() will map in the redzone (if called). 5000 seems
117  * to work reasonably well...
118 */
119 long red_minavail = 5000;
120
121 /*
122  * will be set to 1 for 32 bit x86 systems only, in startup.c
123 */
124 int segkp_fromheap = 0;
125 ulong_t *segkp_bitmap;
```

127 /*

```

128 * If segkp_map_red() is called with the redzone already mapped and
129 * with less than RED_DEEP_THRESHOLD bytes available on the stack,
130 * then the stack situation has become quite serious; if much more stack
131 * is consumed, we have the potential of scrogging the next thread/LWP
132 * structure. To help debug the "can't happen" panics which may
133 * result from this condition, we record hrestime and the calling thread
134 * in red_deep_hires and red_deep_thread respectively.
135 */
136 #define RED_DEEP_THRESHOLD      2000

138 hrttime_t      red_deep_hires;
139 kthread_t      *red_deep_thread;

141 uint32_t       red_nmapped;
142 uint32_t       red_closest = UINT_MAX;
143 uint32_t       red_ndoubles;

145 pgcnt_t anon_segkp_pages_locked;      /* See vm/anon.h */
146 pgcnt_t anon_segkp_pages_resv;       /* anon reserved by seg_kp */

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

```

```

171     seg_inherit_notsup      /* inherit */
172 };
_____unchanged_portion_omitted_____

```

```

*****
9838 Fri May 8 18:03:44 2015
new/usr/src/uts/common/vm/seg_kpm.c
use C99 initializers in segment ops structures
*****
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);

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

113 static struct seg_ops segkpm_ops = {
114     .dup = SEGKPM_BADOP(int),
115     .unmap = SEGKPM_BADOP(int),
116     .free = SEGKPM_BADOP(void),
117     .fault = segkpm_fault,
118     .faulta = SEGKPM_BADOP(int),
119     .setprot = SEGKPM_BADOP(int),
120     .checkprot = SEGKPM_BADOP(int),
121     .kcluster = SEGKPM_BADOP(int),
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 *),
135 .capable    = segkpm_capable,
136 .inherit    = seg_inherit_notsup,
137
138 SEGKPM_BADOP(int), /* dup */
139 SEGKPM_BADOP(int), /* unmap */
140 SEGKPM_BADOP(void), /* free */
141 segkpm_fault,
142 SEGKPM_BADOP(int), /* faulta */
143 SEGKPM_BADOP(int), /* setprot */
144 SEGKPM_BADOP(int), /* checkprot */
145 SEGKPM_BADOP(int), /* kluster */
146 SEGKPM_BADOP(int), /* sync */
147 SEGKPM_BADOP(size_t), /* incore */
148 SEGKPM_BADOP(int), /* lockop */
149 SEGKPM_BADOP(int), /* getprot */
150 SEGKPM_BADOP(u_offset_t), /* getoffset */
151 SEGKPM_BADOP(int), /* gettype */
152 SEGKPM_BADOP(int), /* getvp */
153 SEGKPM_BADOP(int), /* advise */
154 segkpm_dump, /* dump */
155 SEGKPM_NOTSUP, /* pagelock */
156 SEGKPM_BADOP(int), /* setpgsz */
157 SEGKPM_BADOP(int), /* getmemid */
158 SEGKPM_BADOP(lgrp_mem_policy_info_t *), /* getpolicy */
159 segkpm_capable, /* capable */
160 seg_inherit_notsup /* inherit */
161 };
162
163 _____
164 unchanged portion omitted
```

```

*****
58128 Fri May 8 18:03:44 2015
new/usr/src/uts/common/vm/seg_map.c
use C99 initializers in segment ops structures
*****
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 lgrp_mem_policy_info_t *segmap_getpolicy(struct seg *seg,
93                                                  caddr_t addr);
94 static int      segmap_capable(struct seg *seg, segcapability_t capability);

96 /* segkpm support */
97 static caddr_t segmap_pagecreate_kpm(struct seg *, vnode_t *, u_offset_t,
98                                     struct smap *, enum seg_rw);
99 struct smap      *get_smap_kpm(caddr_t, page_t **);

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

103 static struct seg_ops segmap_ops = {
104     .dup          = SEGMAP_BADOP(int),
105     .unmap        = SEGMAP_BADOP(int),
106     .free         = segmap_free,
107     .fault        = segmap_fault,
108     .faulta       = segmap_faulta,
109     .setprot      = SEGMAP_BADOP(int),
110     .checkprot    = segmap_checkprot,
111     .kluster      = segmap_kluster,
112     .sync         = SEGMAP_BADOP(int),
113     .incore       = SEGMAP_BADOP(size_t),
114     .lockprot     = SEGMAP_BADOP(int),
115     .getprot      = segmap_getprot,
116     .getoffset    = segmap_getoffset,
117     .gettype      = segmap_gettype,
118     .getvp        = segmap_getvp,
119     .advise       = SEGMAP_BADOP(int),
120     .dump         = segmap_dump,
121     .pagelock     = segmap_pagelock,
122     .setpagesize  = SEGMAP_BADOP(int),
123     .getmemid     = segmap_getmemid,
124     .getpolicy    = segmap_getpolicy,
125     .capable      = segmap_capable,
126     .inherit      = seg_inherit_notsup,
104     SEGMAP_BADOP(int), /* dup */

```



```
105     SEGMAP_BADOP(int),      /* unmap */
106     segmap_free,
107     segmap_fault,
108     segmap_faulta,
109     SEGMAP_BADOP(int),      /* setprot */
110     segmap_checkprot,
111     segmap_kluster,
112     SEGMAP_BADOP(int),      /* sync */
113     SEGMAP_BADOP(size_t),   /* incore */
114     SEGMAP_BADOP(int),      /* lockop */
115     segmap_getprot,
116     segmap_getoffset,
117     segmap_gettype,
118     segmap_getvp,
119     SEGMAP_BADOP(int),      /* advise */
120     segmap_dump,
121     segmap_pagelock,        /* pagelock */
122     SEGMAP_BADOP(int),      /* setpgsz */
123     segmap_getmemid,        /* getmemid */
124     segmap_getpolicy,       /* getpolicy */
125     segmap_capable,         /* capable */
126     seg_inherit_notsup     /* inherit */
127 };
```

unchanged portion omitted

 83527 Fri May 8 18:03:44 2015

new/usr/src/uts/common/vm/seg_spt.c
 use C99 initializers in segment ops structures

_____unchanged_portion_omitted_____

```

86 #define SEGSPT_BADOP(t) (t(*)())segspt_badop

88 struct seg_ops segspt_ops = {
89     .dup          = SEGSPT_BADOP(int),
90     .unmap       = segspt_unmap,
91     .free        = segspt_free,
92     .fault       = SEGSPT_BADOP(int),
93     .faulta     = SEGSPT_BADOP(faultcode_t),
94     .setprot     = SEGSPT_BADOP(int),
95     .checkprot  = SEGSPT_BADOP(int),
96     .kluster    = SEGSPT_BADOP(int),
97     .sync       = SEGSPT_BADOP(int),
98     .incore     = SEGSPT_BADOP(size_t),
99     .lockop     = SEGSPT_BADOP(int),
100    .getprot     = SEGSPT_BADOP(int),
101    .getoffset   = SEGSPT_BADOP(u_offset_t),
102    .gettype     = SEGSPT_BADOP(int),
103    .getvp      = SEGSPT_BADOP(int),
104    .advise     = SEGSPT_BADOP(int),
105    .dump       = SEGSPT_BADOP(void),
106    .pagelock   = SEGSPT_BADOP(int),
107    .setpagesize = SEGSPT_BADOP(int),
108    .getmemid   = SEGSPT_BADOP(int),
109    .getpolicy  = segspt_getpolicy,
110    .capable    = SEGSPT_BADOP(int),
111    .inherit    = seg_inherit_notsup,
89    SEGSPT_BADOP(int),          /* dup */
90    segspt_unmap,
91    segspt_free,
92    SEGSPT_BADOP(int),          /* fault */
93    SEGSPT_BADOP(faultcode_t),  /* faulta */
94    SEGSPT_BADOP(int),          /* setprot */
95    SEGSPT_BADOP(int),          /* checkprot */
96    SEGSPT_BADOP(int),          /* kluster */
97    SEGSPT_BADOP(int),          /* sync */
98    SEGSPT_BADOP(size_t),       /* incore */
99    SEGSPT_BADOP(int),          /* lockop */
100   SEGSPT_BADOP(int),          /* getprot */
101   SEGSPT_BADOP(u_offset_t),    /* getoffset */
102   SEGSPT_BADOP(int),          /* gettype */
103   SEGSPT_BADOP(int),          /* getvp */
104   SEGSPT_BADOP(int),          /* advise */
105   SEGSPT_BADOP(void),         /* dump */
106   SEGSPT_BADOP(int),          /* pagelock */
107   SEGSPT_BADOP(int),          /* setpgsz */
108   SEGSPT_BADOP(int),          /* getmemid */
109   segspt_getpolicy,           /* getpolicy */
110   SEGSPT_BADOP(int),          /* capable */
111   seg_inherit_notsup         /* inherit */
112 };

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_shmfaulta(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_shmincore(struct seg *seg, caddr_t addr, size_t len,
126     register char *vec);
127 static int segspt_shmsync(struct seg *seg, register caddr_t addr, size_t len,
128     int attr, uint_t flags);
129 static int segspt_shmlockop(struct seg *seg, caddr_t addr, size_t len,
130     int attr, int op, ulong_t *lockmap, size_t pos);
131 static int segspt_shmgetprot(struct seg *seg, caddr_t addr, size_t len,
132     uint_t *protv);
133 static u_offset_t segspt_shmgetoffset(struct seg *seg, caddr_t addr);
134 static int segspt_shmgettype(struct seg *seg, caddr_t addr);
135 static int segspt_shmgetvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
136 static int segspt_shmadvise(struct seg *seg, caddr_t addr, size_t len,
137     uint_t *protv);
138 static void segspt_shmdump(struct seg *seg);
139 static int segspt_shmpagelock(struct seg *, caddr_t, size_t,
140     struct page ***, enum lock_type, enum seg_rw);
141 static int segspt_shmsetpgsz(struct seg *, caddr_t, size_t, uint_t);
142 static int segspt_shmgetmemid(struct seg *, caddr_t, memid_t *);
143 static lgrp_mem_policy_info_t *segspt_shmgetpolicy(struct seg *, caddr_t);
144 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     .faulta     = segspt_shmfaulta,
152     .setprot     = segspt_shmsetprot,
153     .checkprot  = segspt_shmcheckprot,
154     .kluster    = segspt_shmkluster,
155     .sync       = segspt_shmsync,
156     .incore     = segspt_shmincore,
157     .lockop     = segspt_shmlockop,
158     .getprot    = segspt_shmgetprot,
159     .getoffset   = segspt_shmgetoffset,
160     .gettype    = segspt_shmgettype,
161     .getvp      = segspt_shmgetvp,
162     .advise     = segspt_shmadvise,
163     .dump       = segspt_shmdump,
164     .pagelock   = segspt_shmpagelock,
165     .setpagesize = segspt_shmsetpgsz,
166     .getmemid   = segspt_shmgetmemid,
167     .getpolicy  = segspt_shmgetpolicy,
168     .capable    = segspt_shmcapable,
169     .inherit    = seg_inherit_notsup,
147     segspt_shmdup,
148     segspt_shmunmap,
149     segspt_shmfree,
150     segspt_shmfault,
151     segspt_shmfaulta,
152     segspt_shmsetprot,
153     segspt_shmcheckprot,
154     segspt_shmkluster,
155     segspt_shmsync,
156     segspt_shmincore,
157     segspt_shmlockop,
158     segspt_shmgetprot,
159     segspt_shmgetoffset,
160     segspt_shmgettype,
161     segspt_shmgetvp,
162     segspt_shmadvise,          /* advise */
163     segspt_shmdump,
164     segspt_shmpagelock,

```

```
165     segspt_shmsetpgsz,  
166     segspt_shmgetmemid,  
167     segspt_shmgetpolicy,  
168     segspt_shmcapable,  
169     seg_inherit_notsup  
170 };  
unchanged_portion_omitted
```

new/usr/src/uts/common/vm/seg_vn.c

1

```
*****
280654 Fri May 8 18:03:44 2015
new/usr/src/uts/common/vm/seg_vn.c
use C99 initializers in segment ops structures
*****
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>
```

new/usr/src/uts/common/vm/seg_vn.c

2

```
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 int      segvn_sync(struct seg *seg, caddr_t addr, size_t len,
115     int attr, uint_t flags);
116 static size_t   segvn_incore(struct seg *seg, caddr_t addr, size_t len,
117     char *vec);
118 static int      segvn_lockop(struct seg *seg, caddr_t addr, size_t len,
119     int attr, int op, ulong_t *lockmap, size_t pos);
120 static int      segvn_getprot(struct seg *seg, caddr_t addr, size_t len,
121     uint_t *protv);
122 static u_offset_t segvn_getoffset(struct seg *seg, caddr_t addr);
123 static int      segvn_gettype(struct seg *seg, caddr_t addr);
124 static int      segvn_getvp(struct seg *seg, caddr_t addr,
125     struct vnode **vpp);
126 static int      segvn_advise(struct seg *seg, caddr_t addr, size_t len,
127     uint_t behav);
```

```

128 static void      segvn_dump(struct seg *seg);
129 static int       segvn_pagelock(struct seg *seg, caddr_t addr, size_t len,
130                          struct page ***ppp, enum lock_type type, enum seg_rw rw);
131 static int       segvn_setpagesize(struct seg *seg, caddr_t addr, size_t len,
132                          uint_t szc);
133 static int       segvn_getmemid(struct seg *seg, caddr_t addr,
134                          memid_t *memidp);
135 static lgrp_mem_policy_info_t *segvn_getpolicy(struct seg *, caddr_t);
136 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     .sync         = segvn_sync,
149     .incore       = segvn_incore,
150     .lockop       = segvn_lockop,
151     .getprot      = segvn_getprot,
152     .getoffset    = segvn_getoffset,
153     .gettype      = segvn_gettype,
154     .getvp        = segvn_getvp,
155     .advise       = segvn_advise,
156     .dump         = segvn_dump,
157     .pagelock     = segvn_pagelock,
158     .setpagesize  = segvn_setpagesize,
159     .getmemid     = segvn_getmemid,
160     .getpolicy    = segvn_getpolicy,
161     .capable      = segvn_capable,
162     .inherit      = segvn_inherit,
140     segvn_dup,
141     segvn_unmap,
142     segvn_free,
143     segvn_fault,
144     segvn_faulta,
145     segvn_setprot,
146     segvn_checkprot,
147     segvn_kluster,
148     segvn_sync,
149     segvn_incore,
150     segvn_lockop,
151     segvn_getprot,
152     segvn_getoffset,
153     segvn_gettype,
154     segvn_getvp,
155     segvn_advise,
156     segvn_dump,
157     segvn_pagelock,
158     segvn_setpagesize,
159     segvn_getmemid,
160     segvn_getpolicy,
161     segvn_capable,
162     segvn_inherit
163 };

```

unchanged_portion_omitted

16996 Fri May 8 18:03:45 2015

new/usr/src/uts/i86xpv/vm/seg_mf.c

use C99 initializers in segment ops structures

unchanged portion omitted

```
760 static struct seg_ops segmf_ops = {
761     .dup          = segmf_dup,
762     .unmap       = segmf_unmap,
763     .free        = segmf_free,
764     .fault       = segmf_fault,
765     .faulta     = segmf_faulta,
766     .setprot     = segmf_setprot,
767     .checkprot  = segmf_checkprot,
768     .kluster    = segmf_kluster,
769     .sync       = segmf_sync,
770     .incore     = segmf_inc core,
771     .lockop     = segmf_lockop,
772     .getprot    = segmf_getprot,
773     .getoffset  = segmf_getoffset,
774     .gettype    = segmf_gettype,
775     .getvp     = segmf_getvp,
776     .advise     = segmf_advise,
777     .dump       = segmf_dump,
778     .pagelock   = segmf_pagelock,
779     .setpagesize = segmf_setpagesize,
780     .getmemid   = segmf_getmemid,
781     .getpolicy  = segmf_getpolicy,
782     .capable    = segmf_capable,
783     .inherit    = seg_inherit_notsup,
761     segmf_dup,
762     segmf_unmap,
763     segmf_free,
764     segmf_fault,
765     segmf_faulta,
766     segmf_setprot,
767     segmf_checkprot,
768     (int (*)())segmf_kluster,
769     segmf_sync,
770     segmf_inc core,
771     segmf_lockop,
772     segmf_getprot,
773     segmf_getoffset,
774     segmf_gettype,
775     segmf_getvp,
776     segmf_advise,
777     segmf_dump,
778     segmf_pagelock,
779     segmf_setpagesize,
780     segmf_getmemid,
781     segmf_getpolicy,
782     segmf_capable,
783     seg_inherit_notsup
784 };
```

unchanged portion omitted

new/usr/src/uts/sparc/v9/vm/seg_nf.c

1

```
*****
12471 Fri May 8 18:03:45 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
use C99 initializers in segment ops structures
*****
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 2006 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 - segment for non-faulting loads.
36 */

38 #include <sys/types.h>
39 #include <sys/t_lock.h>
40 #include <sys/param.h>
41 #include <sys/mman.h>
42 #include <sys/errno.h>
43 #include <sys/kmem.h>
44 #include <sys/cmn_err.h>
45 #include <sys/vnode.h>
46 #include <sys/proc.h>
47 #include <sys/conf.h>
48 #include <sys/debug.h>
49 #include <sys/archsystem.h>
50 #include <sys/lgrp.h>

52 #include <vm/page.h>
53 #include <vm/hat.h>
54 #include <vm/as.h>
55 #include <vm/seg.h>
56 #include <vm/vpage.h>

58 /*
59 * Private seg op routines.
60 */
61 static int segnf_dup(struct seg *seg, struct seg *newseg);
```

new/usr/src/uts/sparc/v9/vm/seg_nf.c

2

```
62 static int segnf_unmap(struct seg *seg, caddr_t addr, size_t len);
63 static void segnf_free(struct seg *seg);
64 static faultcode_t segnf_nomap(void);
65 static int segnf_setprot(struct seg *seg, caddr_t addr,
66 size_t len, uint_t prot);
67 static int segnf_checkprot(struct seg *seg, caddr_t addr,
68 size_t len, uint_t prot);
69 static void segnf_badop(void);
70 static int segnf_nop(void);
71 static int segnf_getprot(struct seg *seg, caddr_t addr,
72 size_t len, uint_t *protv);
73 static u_offset_t segnf_getoffset(struct seg *seg, caddr_t addr);
74 static int segnf_gettype(struct seg *seg, caddr_t addr);
75 static int segnf_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
76 static void segnf_dump(struct seg *seg);
77 static int segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
78 struct page ***ppp, enum lock_type type, enum seg_rw rw);
79 static int segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
80 uint_t szc);
81 static int segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
82 static lgrp_mem_policy_info_t *segnf_getpolicy(struct seg *seg,
83 caddr_t addr);

86 struct seg_ops segnf_ops = {
87 .dup = segnf_dup,
88 .unmap = segnf_unmap,
89 .free = segnf_free,
90 .fault = (faultcode_t (*)(struct hat *, struct seg *, caddr_t,
91 size_t, enum fault_type, enum seg_rw))segnf_nomap,
92 .faulta = (faultcode_t (*)(struct seg *, caddr_t)) segnf_nomap,
93 .setprot = segnf_setprot,
94 .checkprot = segnf_checkprot,
95 .kluster = (int (*)( ))segnf_badop,
96 .sync = (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
97 segnf_nop,
98 .incore = (size_t (*)(struct seg *, caddr_t, size_t, char *))
99 segnf_nop,
100 .lockop = (int (*)(struct seg *, caddr_t, size_t, int, int,
101 ulong_t *, size_t))segnf_nop,
102 .getprot = segnf_getprot,
103 .getoffset = segnf_getoffset,
104 .gettype = segnf_gettype,
105 .getvp = segnf_getvp,
106 .advise = (int (*)(struct seg *, caddr_t, size_t, uint_t))
107 segnf_nop,
108 .dump = segnf_dump,
109 .pagelock = segnf_pagelock,
110 .setpagesize = segnf_setpagesize,
111 .getmemid = segnf_getmemid,
112 .getpolicy = segnf_getpolicy,
87 segnf_dup,
88 segnf_unmap,
89 segnf_free,
90 (faultcode_t (*)(struct hat *, struct seg *, caddr_t, size_t,
91 enum fault_type, enum seg_rw))
92 segnf_nomap, /* fault */
93 (faultcode_t (*)(struct seg *, caddr_t))
94 segnf_nomap, /* faulta */
95 segnf_setprot,
96 segnf_checkprot,
97 (int (*)( ))segnf_badop, /* kluster */
98 (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
99 segnf_nop, /* sync */
100 (size_t (*)(struct seg *, caddr_t, size_t, char *))
101 segnf_nop, /* incore */
```

```
102     (int (*)(struct seg *, caddr_t, size_t, int, int, ulong_t *, size_t))
103         segnf_nop,          /* lockop */
104     segnf_getprot,
105     segnf_getoffset,
106     segnf_gettype,
107     segnf_getvp,
108     (int (*)(struct seg *, caddr_t, size_t, uint_t))
109         segnf_nop,          /* advise */
110     segnf_dump,
111     segnf_pagelock,
112     segnf_setpagesize,
113     segnf_getmemid,
114     segnf_getpolicy,
113 };
_____unchanged_portion_omitted_
```