

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

1

```
*****
113437 Fri May 8 18:04:49 2015
new/usr/src/uts/common/vm/seg_dev.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummpp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21
22 /*
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>
```

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

2

```
59 #include <sys/debug.h>
60 #include <sys/ddidevmap.h>
61 #include <sys/ddi_implfuncs.h>
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 VTOS(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);
184 static int segdev_getmemid(struct seg *, caddr_t, memid_t *);

186 /*
187 * XXX this struct is used by rootnex_map_fault to identify
188 * the segment it has been passed. So if you make it
189 * "static" you'll need to fix rootnex_map_fault.

```

```

190 */
191 struct seg_ops segdev_ops = {
192     .dup = segdev_dup,
193     .unmap = segdev_unmap,
194     .free = segdev_free,
195     .fault = segdev_fault,
196     .faulta = segdev_faulta,
197     .setprot = segdev_setprot,
198     .checkprot = segdev_checkprot,
199     .kluster = (int (*)( ))segdev_badop,
200     .sync = segdev_sync,
201     .incore = segdev_incore,
202     .lockop = segdev_lockop,
203     .getprot = segdev_getprot,
204     .getoffset = segdev_getoffset,
205     .gettype = segdev_gettype,
206     .getvp = segdev_getvp,
207     .advise = segdev_advise,
208     .dump = segdev_dump,
209     .pagelock = segdev_pagelock,
210     .setpagesize = segdev_setpagesize,
211     .getmemid = segdev_getmemid,
212 };
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

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

1

```
*****
9308 Fri May 8 18:04:49 2015
new/usr/src/uts/common/vm/seg_kpm.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummpp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License, Version 1.0 only
6  * (the "License"). You may not use this file except in compliance
7  * with the License.
8  *
9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 * or http://www.opensolaris.org/os/licensing.
11 * See the License for the specific language governing permissions
12 * and limitations under the License.
13 *
14 * When distributing Covered Code, include this CDDL HEADER in each
15 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 * If applicable, add the following below this CDDL HEADER, with the
17 * fields enclosed by brackets "[]" replaced with your own identifying
18 * information: Portions Copyright [yyyy] [name of copyright owner]
19 *
20 * CDDL HEADER END
21 */
22 /*
23  * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */
27 /*
28  * Kernel Physical Mapping (kpm) segment driver (segkpm).
29  *
30  * This driver delivers along with the hat_kpm* interfaces an alternative
31  * mechanism for kernel mappings within the 64-bit Solaris operating system,
32  * which allows the mapping of all physical memory into the kernel address
33  * space at once. This is feasible in 64 bit kernels, e.g. for Ultrasparc II
34  * and beyond processors, since the available VA range is much larger than
35  * possible physical memory. Momentarily all physical memory is supported,
36  * that is represented by the list of memory segments (memsegs).
37  *
38  * Segkpm mappings have also very low overhead and large pages are used
39  * (when possible) to minimize the TLB and TSB footprint. It is also
40  * extensible for other than Sparc architectures (e.g. AMD64). Main
41  * advantage is the avoidance of the TLB-shootdown X-calls, which are
42  * normally needed when a kernel (global) mapping has to be removed.
43  *
44  * First example of a kernel facility that uses the segkpm mapping scheme
45  * is seg_map, where it is used as an alternative to hat_memload().
46  * See also hat layer for more information about the hat_kpm* routines.
47  * The kpm 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>
```

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

2

```
59 #include <sys/bitmap.h>
60 #include <sys/atomic.h>
61 #include <sys/lgrp.h>
63 #include <vm/seg_kmem.h>
64 #include <vm/seg_kpm.h>
65 #include <vm/hat.h>
66 #include <vm/as.h>
67 #include <vm/seg.h>
68 #include <vm/page.h>
70 /*
71  * Global kpm controls.
72  * See also platform and mmu specific controls.
73  *
74  * kpm_enable -- global on/off switch for segkpm.
75  * . Set by default on 64bit platforms that have kpm support.
76  * . Will be disabled from platform layer if not supported.
77  * . Can be disabled via /etc/system.
78  *
79  * kpm_smallpages -- use only regular/system pagesize for kpm mappings.
80  * . Can be useful for critical debugging of kpm clients.
81  * . Set to zero by default for platforms that support kpm large pages.
82  * . The use of kpm large pages reduces the footprint of kpm meta data
83  * . and has all the other advantages of using large pages (e.g TLB
84  * . miss reduction).
85  * . Set by default for platforms that don't support kpm large pages or
86  * . where large pages cannot be used for other reasons (e.g. there are
87  * . only few full associative TLB entries available for large pages).
88  *
89  * segmap_kpm -- separate on/off switch for segmap using segkpm:
90  * . Set by default.
91  * . Will be disabled when kpm_enable is zero.
92  * . Will be disabled when MAXBSIZE != PAGESIZE.
93  * . Can be disabled via /etc/system.
94  *
95  */
96 int kpm_enable = 1;
97 int kpm_smallpages = 0;
98 int segmap_kpm = 1;
100 /*
101  * Private seg op routines.
102  */
103 faultcode_t segkpm_fault(struct hat *hat, struct seg *seg, caddr_t addr,
104                          size_t len, enum fault_type type, enum seg_rw rw);
105 static void segkpm_dump(struct seg *);
106 static int segkpm_pagelock(struct seg *seg, caddr_t addr, size_t len,
107                            struct page ***page, enum lock_type type,
108                            enum seg_rw rw);
110 static struct seg_ops segkpm_ops = {
111     .fault = segkpm_fault,
112     .dump = segkpm_dump,
113     .pagelock = segkpm_pagelock,
114     /*#ifndef SEGKPM_SUPPORT
115     #if 0
116     #error FIXME: define nop
117     .dup = nop,
118     .unmap = nop,
119     .free = nop,
120     .faulta = nop,
121     .setprot = nop,
122     .checkprot = nop,
123     .kluster = nop,
124     .sync = nop,
125     */
126     /*#endif
127     */
128 }
```

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

```

*****
82299 Fri May 8 18:04:49 2015
new/usr/src/uts/common/vm/seg_spt.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummpp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
*****
_____unchanged_portion_omitted_____

85 static int segspt_shmdup(struct seg *seg, struct seg *newseg);
86 static int segspt_shmunmap(struct seg *seg, caddr_t raddr, size_t ssize);
87 static void segspt_shmfree(struct seg *seg);
88 static faultcode_t segspt_shmfault(struct hat *hat, struct seg *seg,
89     caddr_t addr, size_t len, enum fault_type type, enum seg_rw rw);
90 static faultcode_t segspt_shmfaultra(struct seg *seg, caddr_t addr);
91 static int segspt_shmsetprot(register struct seg *seg, register caddr_t addr,
92     register size_t len, register uint_t prot);
93 static int segspt_shmcheckprot(struct seg *seg, caddr_t addr, size_t size,
94     uint_t prot);
95 static int segspt_shmkluster(struct seg *seg, caddr_t addr, ssize_t delta);
96 static size_t segspt_shmincore(struct seg *seg, caddr_t addr, size_t len,
97     register char *vec);
98 static int segspt_shmsync(struct seg *seg, register caddr_t addr, size_t len,
99     int attr, uint_t flags);
100 static int segspt_shmlockop(struct seg *seg, caddr_t addr, size_t len,
101     int attr, int op, ulong_t *lockmap, size_t pos);
102 static int segspt_shmgetprot(struct seg *seg, caddr_t addr, size_t len,
103     uint_t *protv);
104 static u_offset_t segspt_shmgetoffset(struct seg *seg, caddr_t addr);
105 static int segspt_shmgettype(struct seg *seg, caddr_t addr);
106 static int segspt_shmgetvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
107 static int segspt_shmadvise(struct seg *seg, caddr_t addr, size_t len,
108     uint_t behav);
109 static void segspt_shmdump(struct seg *seg);
110 static int segspt_shmpagelock(struct seg *, caddr_t, size_t,
111     struct page ***, enum lock_type, enum seg_rw);
112 static int segspt_shmsetpgsz(struct seg *, caddr_t, size_t, uint_t);
112 static int segspt_shmgetmemid(struct seg *, caddr_t, memid_t *);
113 static lgrp_mem_policy_info_t *segspt_shmgetpolicy(struct seg *, caddr_t);

115 struct seg_ops segspt_shmops = {
116     .dup = segspt_shmdup,
117     .unmap = segspt_shmunmap,
118     .free = segspt_shmfree,
119     .fault = segspt_shmfault,
120     .faultra = segspt_shmfaultra,
121     .setprot = segspt_shmsetprot,
122     .checkprot = segspt_shmcheckprot,
123     .kluster = segspt_shmkluster,
124     .sync = segspt_shmsync,
125     .incore = segspt_shmincore,
126     .lockop = segspt_shmlockop,
127     .getprot = segspt_shmgetprot,
128     .getoffset = segspt_shmgetoffset,
129     .gettype = segspt_shmgettype,
130     .getvp = segspt_shmgetvp,
131     .advise = segspt_shmadvise,
132     .dump = segspt_shmdump,
133     .pagelock = segspt_shmpagelock,
135     .setpagesize = segspt_shmsetpgsz,
134     .getmemid = segspt_shmgetmemid,
135     .getpolicy = segspt_shmgetpolicy,
136 };
_____unchanged_portion_omitted_____

```

```

2952 /*ARGSUSED*/
2953 void
2954 segspt_shmdump(struct seg *seg)
2955 {
2956     /* no-op for ISM segment */
2959 }

2961 /*ARGSUSED*/
2962 static faultcode_t
2963 segspt_shmsetpgsz(struct seg *seg, caddr_t addr, size_t len, uint_t szc)
2964 {
2965     return (ENOTSUP);
2957 }
_____unchanged_portion_omitted_____

```

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

1

```
*****
55144 Fri May 8 18:04:49 2015
new/usr/src/uts/common/vm/vm_seg.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummpp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
*****
_____unchanged_portion_omitted_

2006 int
2007 segop_setpagesize(struct seg *seg, caddr_t addr, size_t len, uint_t szc)
2008 {
2009     if (seg->s_ops->setpagesize == NULL)
2010         return (ENOTSUP);
2009     VERIFY3P(seg->s_ops->setpagesize, !=, NULL);

2012     return (seg->s_ops->setpagesize(seg, addr, len, szc));
2013 }
_____unchanged_portion_omitted_
```

```
*****
16536 Fri May 8 18:04:50 2015
new/usr/src/uts/i86xpv/vm/seg_mf.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummpp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
*****
```

```
_____unchanged_portion_omitted_
485 /*ARGSUSED*/
486 static int
487 segmf_setpagesize(struct seg *seg, caddr_t addr, size_t len, uint_t szc)
488 {
489     return (ENOTSUP);
490 }

485 static int
486 segmf_getmemid(struct seg *seg, caddr_t addr, memid_t *memid)
487 {
488     struct segmf_data *data = seg->s_data;

490     memid->val[0] = (uintptr_t)VTOCVP(data->vp);
491     memid->val[1] = (uintptr_t)seg_page(seg, addr);
492     return (0);
493 }
_____unchanged_portion_omitted_
```

```
739 static struct seg_ops segmf_ops = {
740     .dup           = segmf_dup,
741     .unmap        = segmf_unmap,
742     .free         = segmf_free,
743     .fault        = segmf_fault,
744     .faulta       = segmf_faulta,
745     .setprot      = segmf_setprot,
746     .checkprot    = segmf_checkprot,
747     .kluster      = segmf_kluster,
748     .sync         = segmf_sync,
749     .incore       = segmf_inc core,
750     .lockop       = segmf_lockop,
751     .getprot      = segmf_getprot,
752     .getoffset    = segmf_getoffset,
753     .gettype      = segmf_gettype,
754     .getvp        = segmf_getvp,
755     .advise       = segmf_advise,
756     .dump         = segmf_dump,
757     .pagelock     = segmf_pagelock,
765     .setpagesize = segmf_setpagesize,
758     .getmemid     = segmf_getmemid,
759 };
_____unchanged_portion_omitted_
```

```

*****
11638 Fri May 8 18:04:50 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummpp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22 * Copyright 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);
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 int     segnf_nop(void);
70 static int     segnf_getprot(struct seg *seg, caddr_t addr,
71                              size_t len, uint_t *protv);
72 static u_offset_t segnf_getoffset(struct seg *seg, caddr_t addr);
73 static int     segnf_gettype(struct seg *seg, caddr_t addr);
74 static int     segnf_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
75 static void    segnf_dump(struct seg *seg);
76 static int     segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
77                              struct page ***ppp, enum lock_type type, enum seg_rw rw);
78 static int     segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
79                                 uint_t szc);

80 struct seg_ops segnf_ops = {
81     .dup           = segnf_dup,
82     .unmap        = segnf_unmap,
83     .free         = segnf_free,
84     .fault        = (faultcode_t (*)(struct hat *, struct seg *, caddr_t,
85                                     size_t, enum fault_type, enum seg_rw)) segnf_nomap,
86     .faulta       = (faultcode_t (*)(struct seg *, caddr_t)) segnf_nomap,
87     .setprot      = segnf_setprot,
88     .checkprot    = segnf_checkprot,
89     .sync         = (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
90                     segnf_nop,
91     .incore       = (size_t (*)(struct seg *, caddr_t, size_t, char *))
92                     segnf_nop,
93     .lockop       = (int (*)(struct seg *, caddr_t, size_t, int, int,
94                             ulong_t *, size_t)) segnf_nop,
95     .getprot      = segnf_getprot,
96     .getoffset    = segnf_getoffset,
97     .gettype      = segnf_gettype,
98     .getvp        = segnf_getvp,
99     .advise       = (int (*)(struct seg *, caddr_t, size_t, uint_t))
100                    segnf_nop,
101     .dump         = segnf_dump,
102     .pagelock     = segnf_pagelock,
103     .setpagesize  = segnf_setpagesize,
104 };
105
106 unchanged_portion_omitted

440 /*
441  * segnf pages are not dumped, so we just return
442  */
443 /* ARGSUSED */
444 static void
445 segnf_dump(struct seg *seg)
446 {}

448 /*ARGSUSED*/
449 static int
450 segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
451               struct page ***ppp, enum lock_type type, enum seg_rw rw)
452 {
453     return (ENOTSUP);
454 }

```



```
459 /*ARGSUSED*/
460 static int
461 segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
462                 uint_t szc)
452 {
453     return (ENOTSUP);
454 }
unchanged_portion_omitted
```