

## 第 0019 讲 2 伙伴系统算法案例分析

### 内存管理专题--2 伙伴系统算法案例分析

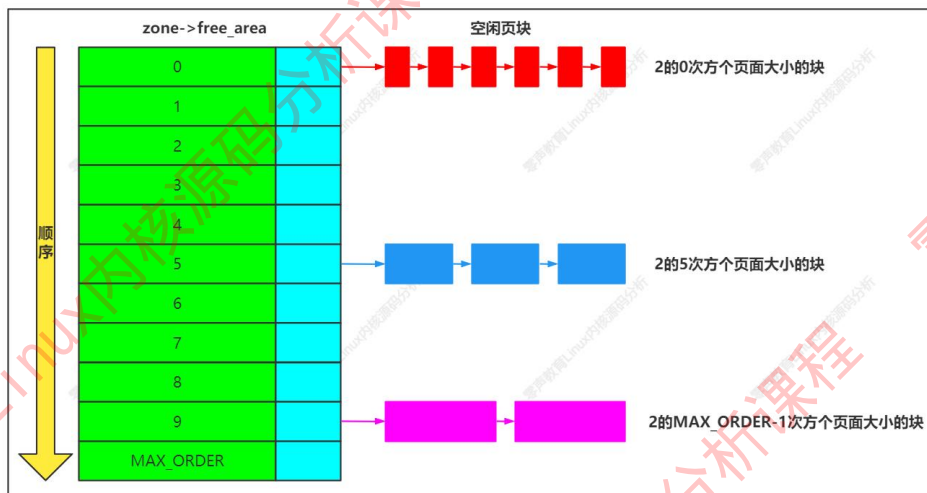
#### 一、伙伴系统算法原理

Linux 的伙伴算法把所有的空闲页面分为 11 个块组，每组中块的大小是 2 的幂次方个页面。例如，第 0 组中块的大小都为  $2^0$ （1 个页面），第 1 组中块的大小都为  $2^1$ （2 个页面），第 8 组中块的大小都为  $2^8$ （256 个页面），依次类推。也就是说，每一组中块的大小是相同的，且这同样大小的块形成一个链表。

系统内存中的每个物理内存页（页帧），都对应于一个 struct page 实例，每个内存域都关联了一个 struct zone 的实例，其中保存了用于管理伙伴数据的主要数组，具体如下：

```
include > linux > C mmzone.h > @zone_type
361
362 struct zone {
363     /* Read-mostly fields */
364
365     /* zone watermarks, access with *_wmark_pages(zone) macros */
366     unsigned long _watermark[NR_WMARK];
367     unsigned long watermark_boost;
368
369     unsigned long nr_reserved_highatomic;
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     /* Write-intensive fields used from the page allocator */
457     ZONE_PADDING(_pad1_)
458
459     /* free areas of different sizes */
460     struct free_area free_area[MAX_ORDER];
461
462     /* zone flags, see below */
463     unsigned long flags;
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
```

一般来说 MAX\_ORDER 默认定义为 11，但如果特定于体系结构的代码设置 FORCE\_MAX\_ZONEORDER 配置选项，该值也可以手工改变。

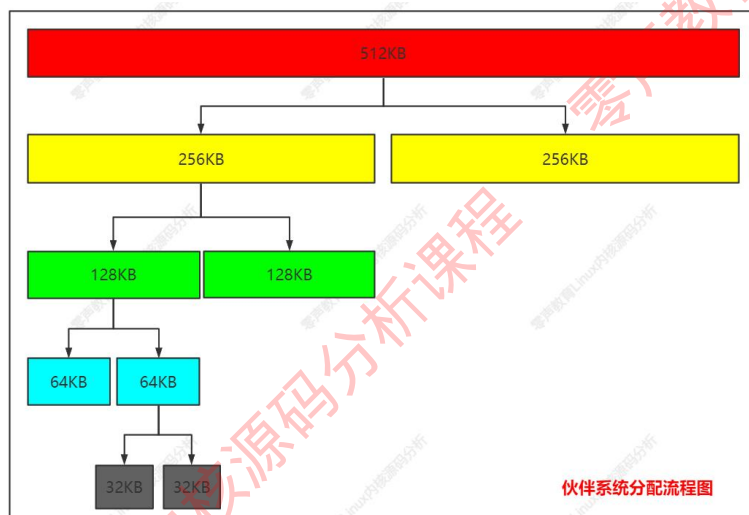


struct free\_area 是一个伙伴系统的辅助数据结构：

```
include > linux > C mmzone.h > 53 free_area
96 struct free_area {
97     struct list_head free_list[MIGRATE_TYPES];
98     unsigned long nr_free;
99 };
100
```

其中：free\_list 是用于连接空闲页的链表。页链表包含大小相同的连续内存区；nr\_free 指定了当前内存区中空闲页块的数目（对 0 阶内存区逐页计算，对 1 阶内存区计算页对的数目，对 2 阶内存区计算 4 页集合的数目，依次类推。伙伴系统的分配器维护空闲页面所组成的块，这里每一块都是 2 的方幂个页面，方幂的指数称为阶。

伙伴系统从物理连续的大小固定的段进行分配，方法如上所述，内核请求 28KB 的内存，具体案例分析如下：



## 二、分配页和释放页

1、伙伴系统设计思路和申请和释放 API 基本方式一样，Linux 内核中常用的分配物理内存页面的接口是 `alloc_pages()`，分配一个或多个连续的物理页面，分配原则个数为  $2^n$ 。相对于多次离散物理页面它可以提高系统内存的碎片化。具体源码分析如下：

```
include > linux > C gfp.h > alloc_pages(gfp_t, unsigned int)
489 /*
490  * Allocate pages, preferring the node given as nid. When nid == NUMA_NO_NODE,
491  * prefer the current CPU's closest node. Otherwise node must be valid and
492  * online.
493  */
494 static inline struct page *alloc_pages_node(int nid, gfp_t gfp_mask,
495                                             unsigned int order)
496 {
497     if (nid == NUMA_NO_NODE)
498         nid = numa_mem_id();
499     return __alloc_pages_node(nid, gfp_mask, order);
501 }
502
```

如上面函数只是简要检查防止申请内存过大，如果指定节点不存在，Linux 内核自动使用当前执行 CPU 对应的节点 ID，最后调度核心函数 `__alloc_pages_nodemask(...)`。

```
include > linux > C gfp.h > __alloc_pages_node(int, gfp_t, unsigned int)
476 /*
477  * Allocate pages, preferring the node given as nid. The node must be valid and
478  * online. For more general interface, see alloc_pages_node().
479  */
480 static inline struct page *
481 __alloc_pages_node(int nid, gfp_t gfp_mask, unsigned int order)
482 {
483     VM_BUG_ON(nid < 0 || nid >= MAX_NUMNODES);
484     VM_WARN_ON((gfp_mask & __GFP_THISNODE) && !node_online(nid));
485     return __alloc_pages(gfp_mask, order, nid);
487 }
488
```

```

include > linux > C gfp.h > ...
470 static inline struct page *
471 __alloc_pages(gfp_t gfp_mask, unsigned int order, int preferred_nid)
472 {
473     return __alloc_pages_nodemask(gfp_mask, order, preferred_nid, NULL);
474 }

```

```

4504 struct page *
4505 __alloc_pages_nodemask(gfp_t gfp_mask, unsigned int order, int preferred_nid,
4506                      nodemask_t *nodemask)
4507 {
4508     struct page *page;
4509     unsigned int alloc_flags = ALLOC_WMARK_LOW;
4510     gfp_t alloc_mask; /* The gfp_t that was actually used for allocation */
4511     struct alloc_context ac = { };
4512
4513     /*
4514      * There are several places where we assume that the order value is sane
4515      * so bail out early if the request is out of bound.
4516      */
4517     if (unlikely(order >= MAX_ORDER)) {
4518         WARN_ON_ONCE(!(gfp_mask & __GFP_NOWARN));
4519         return NULL;
4520     }

```

## 2、伙伴系统释放页框步骤（free\_pages()）：

```

mm > C page_alloc.c > EXPORT_SYMBOL(__free_pages)
4603
4604 void __free_pages(struct page *page, unsigned int order)
4605 {
4606     if (put_page_testzero(page))
4607         free_the_page(page, order);
4608 }
4609 EXPORT_SYMBOL(__free_pages);
4610
4611

```

```

mm > C page_alloc.c
4595 static inline void free_the_page(struct page *page, unsigned int order)
4596 {
4597     if (order == 0) /* Via pcp? */
4598         free_unref_page(page);
4599     else
4600         __free_pages_ok(page, order);
4601 }
4602
4603

```

```

mm > C page_alloc.c
1292
1293 static void __free_pages_ok(struct page *page, unsigned int order)
1294 {
1295     unsigned long flags;
1296     int migratetype;
1297     unsigned long pfn = page_to_pfn(page);
1298
1299     if (!free_pages_prepare(page, order, true))
1300         return;
1301
1302     migratetype = get_pfnblock_migratetype(page, pfn);
1303     local_irq_save(flags);
1304     __count_vm_events(PGFREE, 1 << order);
1305     free_one_page(page_zone(page), page, pfn, order, migratetype);
1306     local_irq_restore(flags);
1307 }
1308

```

### 三、伙伴系统实现

```
vico@ubuntu: ~/Desktop/ap
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
vico@ubuntu:~/Desktop/ap$ ls
ap.c  Makefile
vico@ubuntu:~/Desktop/ap$ make
make -C /usr/src/linux-headers-5.4.0-109-generic M=/home/vico/Desktop/ap modules
make[1]: 进入目录“/usr/src/linux-headers-5.4.0-109-generic”
CC [M] /home/vico/Desktop/ap/ap.o
Building modules, stage 2.
MODPOST 1 modules
CC [M] /home/vico/Desktop/ap/ap.mod.o
LD [M] /home/vico/Desktop/ap/ap.ko
make[1]: 离开目录“/usr/src/linux-headers-5.4.0-109-generic”
vico@ubuntu:~/Desktop/ap$ ls
ap.c  ap.mod  ap.mod.o  Makefile      Module.symvers
ap.ko  ap.mod.c  ap.o      modules.order
vico@ubuntu:~/Desktop/ap$
```



```
root@ubuntu: /home/vico/Desktop/ap
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

vico@ubuntu:~/Desktop/ap$ clear

vico@ubuntu:~/Desktop/ap$ ls
ap.c Makefile
vico@ubuntu:~/Desktop/ap$ make
make -C /usr/src/linux-headers-5.4.0-109-generic M=/home/vico/Desktop/ap modules
make[1]: 进入目录"/usr/src/linux-headers-5.4.0-109-generic"
  CC [M] /home/vico/Desktop/ap/ap.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M] /home/vico/Desktop/ap/ap.mod.o
  LD [M] /home/vico/Desktop/ap/ap.ko
make[1]: 离开目录"/usr/src/linux-headers-5.4.0-109-generic"
vico@ubuntu:~/Desktop/ap$ ls
ap.c ap.mod ap.mod.o Makefile Module.symvers
ap.ko ap.mod.c ap.o modules.order
vico@ubuntu:~/Desktop/ap$ insmod ap.ko
insmod: ERROR: could not insert module ap.ko: Operation not permitted
vico@ubuntu:~/Desktop/ap$ su root
密码:
root@ubuntu:/home/vico/Desktop/ap# insmod ap.ko
root@ubuntu:/home/vico/Desktop/ap#
```

```
root@ubuntu: /home/vico/Desktop/ap
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

[ 9.300501] AVX2 version of gcm_enc/dec engaged.
[ 9.300503] AES CTR mode by8 optimization enabled
[ 9.909127] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[ 9.909128] Bluetooth: BNEP filters: protocol multicast
[ 9.909130] Bluetooth: BNEP socket layer initialized
[ 50.798023] perf: interrupt took too long (2518 > 2500), lowering kernel.perf_event_max_sample_rate to 79250
[ 54.781170] Bluetooth: RFCOMM TTY layer initialized
[ 54.781175] Bluetooth: RFCOMM socket layer initialized
[ 54.781178] Bluetooth: RFCOMM ver 1.11
[ 56.702257] rfkill: input handler disabled
[ 57.369051] perf: interrupt took too long (3233 > 3147), lowering kernel.perf_event_max_sample_rate to 61750
[ 62.626949] perf: interrupt took too long (4385 > 4041), lowering kernel.perf_event_max_sample_rate to 45500
[ 126.305213] perf: interrupt took too long (5487 > 5481), lowering kernel.perf_event_max_sample_rate to 36250
[ 1280.954685] perf: interrupt took too long (6983 > 6858), lowering kernel.perf_event_max_sample_rate to 28500
[ 1522.255804] ap: loading out-of-tree module taints kernel.
[ 1522.255837] ap: module verification failed: signature and/or required key missing - tainting kernel
[ 1522.256249] alloc_pages Allocation succeeded.
[ 1522.256250] page_address(allocpages) = 0xffff985c07fd0000
root@ubuntu:/home/vico/Desktop/ap#
```

```
root@ubuntu: /home/vico/Desktop/ap
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@ubuntu:/home/vico/Desktop/ap# rmmod ap.ko
root@ubuntu:/home/vico/Desktop/ap# dmesg -c
[ 1805.970571] __free_pages is Release successful!
[ 1805.970573] Prompt : The program exits normally
root@ubuntu:/home/vico/Desktop/ap#
```