内存池

什么是内存池

在进程的堆中分配的一块连续内存，构成一个内存池，提供内存池的创建/销毁，内存的分配/释放等接口。后续的内存申请和释放在该内存池中进行，替代调用系统的分配和释放函数。

内存池的好处

1、减少内存分配和释放时调用系统函数的次数，减少了用户态和核心态切换次数，提高了CPU cache的命中率，从而提升程序的性能。

2、减少内存碎片，提升程序性能和健壮性。
3、减少内存泄漏的概率，提升程序的健壮性。

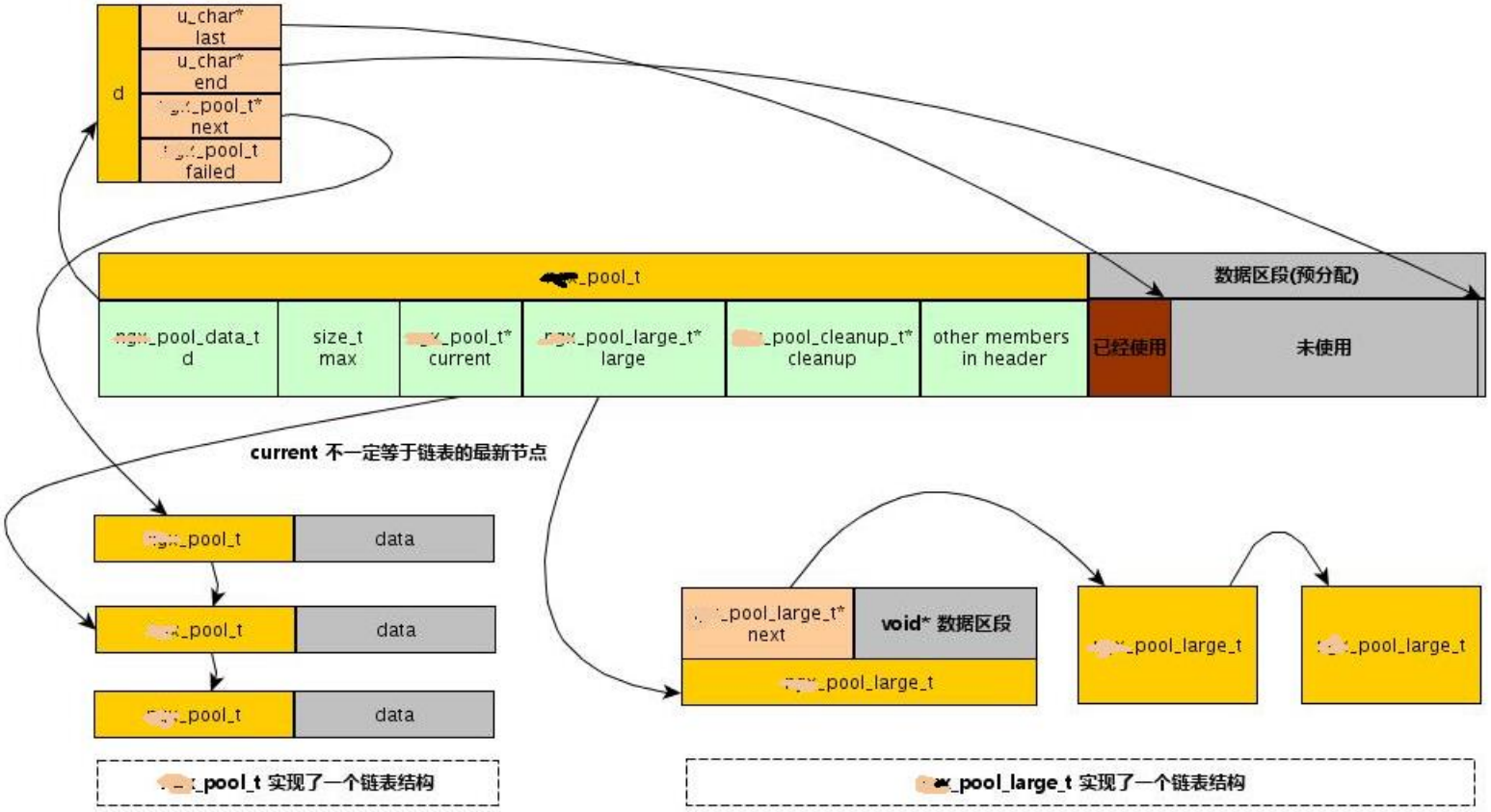
内存池的类型

1、多次内存分配和释放的内存池，如slab内存池。
2、多次分配一次释放的内存池。

内存池的应用场景

多应用于后端服务器程序，如数据库、中间件等关键系统，如Redis/Memcache/kafka/RabbitMQ/Nginx等。

作业：使用单向链表创建一个内存池



创建一个内存池

- **定义三个数据结构，内存池，小数据块，大数据块**
  三个数据结构组成内存池链表、大数据块链表，见右图。

- **实现内存池分配和释放函数**

```
scp_t *scp_create(size_t  size);
void scp_destroy(scp_t *s);
```

- **实现内存分配的接口函数，按照入参由下述三个子函数组合而成**

```
void *scp_alloc(scp_t *pool, size_t size);
```

- **实现上述接口函数的子函数**

```
//size<= SCP_MAX_SIZE调用
static void *scp_small_alloc(scp_t *pool, size_t size);

//size>= SCP_MAX_SIZE调用
static void *scp_big_alloc(scp_t *pool, size_t size);

//内存池链表内所有的空间不够时，新分配一个内存池并加入链表
static void *scp_block_alloc(scp_t *pool, size_t size);
```

说明：

- **使用malloc和free分配和释放内存**
- **申请内存的空间以SCP_MAX_SIZE为界限：**

  > SCP_MAX_SIZE则调用malloc分配实际大小的整块内存；
  <= SCP_MAX_SIZE则在一个数据块内分配。
- **考虑返回的指针地址有可能是奇数的问题，在池内分配的空间的首地址要转换成偶数地址。**

```
#define SCP_MAX_SIZE 4095

#define scp_align_ptr(p, a)                                   \
    (u_char *) (((uint64_t) (p) + ((uint64_t) a - 1)) & ~((uint64_t) a
- 1))

//大数据块
struct scp_big_t
{
    scp_big_t    *next; //下一个scp_big_t
    void         *buf;  //缓冲区指针
};
//小数据块
typedef struct
{
    char        *last; //当前内存池内未使用的区域首地址
    char        *end;  //当前内存池内的尾指针
    scp_t       *next; //下一个内存池指针
} scp_data_t;

//内存池
typedef struct
{
    scp_data_t     dt;  //内存池的数据块
    scp_t          *current; //内存池链表的最近使用的内存池指针
    scp_big_t      *large;   //内存池的大数据块链表的头指针
} scp_t ;

scp_t *scp_create(size_t  size);
void scp_destroy(scp_t *s);
void *scp_alloc(scp_t *pool, size_t size);
static void *scp_small_alloc(scp_t *pool, size_t size);
static void *scp_big_alloc(scp_t *pool, size_t size);
static void *scp_block_alloc(scp_t *pool, size_t size);
```

使用内存池分配内存

- 编写一个试验程序
- 创建一个2m大小的内存池
- 在内存池分别分配一个2k和1m大小空间给变长结构体scp_info_t。
- 销毁内存池

```
struct scp_info_t
{
    char name[32];
    int    age;
    char  origin[16]
    char  data[1];
};
```