Subject: Re: Core 2019
Posted by mirek on Sun, 09 Jun 2019 21:25:20 GMT
View Forum Message <> Reply to Message

Here is the code for logging all huge allocations (replace in Core/hheap.cpp):

```
void *Heap::HugeAlloc(size_t count) // count in 4kb pages
{
 ASSERT(count);

#ifdef LSTAT
 if(count < 65536)
  hstat[count]++;
#endif

 huge_4KB_count += count;

 if(huge_4KB_count > huge_4KB_count_max) {
  huge_4KB_count_max = huge_4KB_count;
  if(MemoryUsedKb() > sKBLimit)
   Panic("MemoryLimitKb breached!");
  if(sPeak)
   Make(*sPeak);
 }

 if(!D::freelist[0]->next) { // initialization
  for(int i = 0; i < 2; i++)
   Dbl_Self(D::freelist[i]);
 }

 if(count > HPAGE) { // we are wasting 4KB to store just 4 bytes here, but this is >32MB after all..
  LTIMING("SysAlloc");
  byte *sysblk = (byte *)SysAllocRaw((count + 1) * 4096, 0);
  BlkHeader *h = (BlkHeader *)(sysblk + 4096);
  h->size = 0;
  *((size_t *)sysblk) = count;
  sys_count++;
  sys_size += 4096 * count;
  return h;
 }

 LTIMING("Huge Alloc");

 word wcount = (word)count;

 if(16 * free_4KB > huge_4KB_count) // keep number of free 4KB blocks in check
```

```
  FreeSmallEmpty(INT_MAX, int(free_4KB - huge_4KB_count / 32));

 for(int pass = 0; pass < 2; pass++) {
  for(int i = count >= 16; i < 2; i++) {
   BlkHeader *l = D::freelist[i];
   BlkHeader *h = l->next;
   while(h != l) {
    word sz = h->GetSize();
    if(sz >= count) {
     void *ptr = MakeAlloc(h, wcount);
     if(count > 16)
      RLOG("HugeAlloc " << asString(ptr) << ", size: " << asString(count));
     return ptr;
    }
    h = h->next;
   }
  }

  if(!FreeSmallEmpty(wcount, INT_MAX)) { // try to coalesce 4KB small free blocks back to huge
storage
   void *ptr = SysAllocRaw(HPAGE * 4096, 0);
   HugePage *pg = (HugePage *)MemoryAllocPermanent(sizeof(HugePage));
   pg->page = ptr;
   pg->next = huge_pages;
   huge_pages = pg;
   AddChunk((BlkHeader *)ptr, HPAGE); // failed, add 32MB from the system
   huge_chunks++;
  }
 }
 Panic("Out of memory");
 return NULL;
}

int Heap::HugeFree(void *ptr)
{
 BlkHeader *h = (BlkHeader *)ptr;
 if(h->size == 0) {
  LTIMING("Sys Free");
  byte *sysblk = (byte *)h - 4096;
  size_t count = *((size_t *)sysblk);
  SysFreeRaw(sysblk, (count + 1) * 4096);
  huge_4KB_count -= count;
  sys_count--;
  sys_size -= 4096 * count;
  return 0;
 }
 LTIMING("Huge Free");
 if(h->GetSize() > 16)
```

```
  RLOG("HugeFree " << asString(ptr) << ", size: " << asString(h->GetSize()));
 huge_4KB_count -= h->GetSize();
 return BlkHeap::Free(h)->GetSize();
}

bool Heap::HugeTryRealloc(void *ptr, size_t count)
{
 bool b = count <= HPAGE && BlkHeap::TryRealloc(ptr, count, huge_4KB_count);
 if(b)
  RLOG("HugeRealloc " << asString(ptr) << ", size: " << asString(count));
 return b;
}
```

(please test with active MemoryTryRealloc)