مدیریت حافظه cottontail: روشی برای تخصیص, آزادسازی و حسابرسی حافظه

سیستم مدیریت حافظه cottontail فضای آدرس دهی حافظه را برای یافتن یک فضای آدرس یکنواخت ردیابی می کند. تقریبا به پلت فرم (در اینجا منظور پردازنده میباشد) بستگی دارد, و برای سیستم های اینتل طراحی شده و برای حداکثر 4GB فضای ۳۲ بیتی, و از ویژگی صفحه بندی(paging) پردازنده های x86 بهره می برد. و این به این معنی است که ۴,۳ میلیارد بایت جداگانه قابل آدرس دهی در این فضای ۴ گیگابایتی وجود دارد که به 1.05 میلیون صفحه ۴ کیلو بایتی تقسیم می شود. تنها چیزی که یک نفر برای تخصیص فضای آدرس دهی باید بداند این است که آیا یک صفحه از فضای آدرس دهی قبلا تخصیص داده شده است یا نه.

این موضوع توسط صفر و یک ها نمایش داده می شود, ۱ برای در حال استفاده و ۰ برای فضای آزاد. اندازه bitmap (در اینجا منظور از bitmap فایل گرافیکی نمیباشد بلکه فضایی میباشد که در حال حاضر از حافظه در اختیار ما میباشد و در حال کار در آن میباشیم) که برای این اطلاعات احتیاج است از طریق زیر محاسبه می شود:

1 bit per page x 2^20 pages x 1 byte per 8 bits = 131,072 bytes or 128KB

از دیدگاه برنامه نویسی یک جستجوی خطی از این bitmap بسیار آهسته خواهد بود و این عمل هر بار که حاظه نیاز به تخصیص دوباره داشته باشد انجام خواهد شد. این برنامه کوچک C یک جستجو برای فضای آزاد را نمایش می دهد:

```
unsigned char page_usage_bitmap[131072];
int i=0;
while(page_usage_bitmap[i / 8] & (1 << (i % 8)))
{
   i++;
}
```

فقط برای درک بیشتر فرض کنیم اولین ۴ مگا بایت حافظه اشغال شده است. و این یعنی اولین ۱۰۲۴ صفحه اشغال شده است و برنامه برای یافتن یک فضای خالی باید از سطح ۱۰۲۵ به بعد که خالی است جستجو کند. ۴ مگ برای یک سیستم عامل خیلی زیاد نیست. حال بیایید فرض کنیم بازی half-life اجرا شده و ۲۰۰ مگ از فضای آدرس دهی اشغال شده است. من از لفظ فضای آدرس دهی استفاده می کنم زیرا ممکن است تمام اطلاعات روی حافظه فیزیکی یا RAM نباشد و ممکن است به خارج از دیسک انتقال یافته باشد. ولی به هر حال مقداری از فضای آدرس دهی را اشغال کرده است. به یاد داشته باشید که این چیزی است که ما اینجا تخصیص می دهیم نه در RAM حقیقی. ولی بیایید به بازی half-life باز گردیم.

با اشغال شدن ۲۰۰ مگ از فضا برنامه باید ۵۰۰۰۰ صفحه را برای یافتن یک فضای خالی بگردد. این ممکن است کمی بیشتر یا کمتر باشد بسته به این که حافظه کجای فضای آدرس دهی را اشغال کرده باشد ولی به طور کلی باید تقریبا پیوسته باشد. این به معنی ۵۰۰۰۰ بار تکرار حلقه بالا است اما این کار در زمان واقعی چه قدر طول می کشد؟ بعد از کامپایل کردن اسمبلی با GCC انالیز نشان داد که ۲۰ دستور زبان برنامه نویسی برای هر تکرار به کار برده می شود. یعنی ۱۰۰۰۰۰۰=۵۰۰۰۰*۲۰ دستور برنامه نویسی برای هر تخصیص حافظه. با متوسط ۲ سیکل ساعت در هر دستور ۲ میلیون سیکل ساعت در هر تخصیص حافظه وجود دارد. در یک کامپیوتر ۵۰۰ مگا هرتز با ۵۰۰ میلیون سیکل ساعت در ثانیه تنها ۲۵۰ تخصیص حافظه در هر ثانیه امکان پذیر است آن هم در صورتی که کامپیوتر کاری به غیر از تخصیص حافظه انجام ندهد! تخصیص حافظه یک کار سطح پایین است که به طور وسیعی توسط پروسه های مهمتری مانند توابعی که اطلاعات را از روی هارد می خوانند به کار برده می شود. همین امر دلیل کافی برای آنست که بدانیم تخصیص حافظه باید به سرعت برق انجام پذیرد. برنامه بالا برای یک سیستم عامل مناسب نیست.

چند تغییر و بهبود ساده می تواند الگوریتم را بسیار سریعتر کند. اول این را به خاطر داشته باشید که x86 یک پردازنده ۳۲ بیتی است و سرعت خواندن یک بایت(۸ بیت) از حافظه در آن با سرعت خواندن DWORD (۳۲ بیت) برابر است. سپس بیایید Bitmap خود را از ۱۲۸کیلو بایت به چهار قسمت یعنی ۱۲۸ تقسیم بر ۴ یا ۳۲ کیلوبایت DWORD تغییر دهیم. سپس مشاهده می کنیم که یک DWORD که ۳۲ صفحه را توصیف می کند که تمام آن استفاده شده است در هگزادسیمال برابر 0xFFFFFFFF است یا ۱- . به جای چک کردن هر یک بیت حال هر DWORD باید چک شود. که تعداد تکرار های حلقه را ۳۲ برابر کمتر می کند. وقتی که یک صفحه خالی پیدا شد تعیین کردن افست یک صفحه خالی در یک DWORD یک امر ناچیز است. و این کد تغییر یافته است که تنها ۱۵۶۳ تکرار را برای یافتن یک فضای خالی به کار می برد:

```
unsigned long page_usage_bitmap[32768];
int i=0;
while(page_usage_bitmap[i] == 0xFFFFFFFF)
{
  i++;
}
```

برای داشتن یک الگوریتم ساده تر تعداد دستور های ماشین که برای توصیف آن نیاز است خیلی کمتر شده, یعنی ۸ عدد.

8x1563=12504 instruction x 2 clock cycles per instruction = about 25000 clock

۲۵۰۰۰ سیکل ساعت در هر تخصیص حافظه. خیلی کمتر از ۲ میلیون در الگوریتم اول. اما این هم خیلی آهسته است . اگر فضای بیشتری از حافظه اشغال شده باشد چه؟ ملاحظات واقعگرایانه زیادی برای طراحی یک سیستم عامل با قابلیت اجرای بالا و موفق وجود دارد. و بنابراین بهینه سازی های زیادی نیاز است. فضای آدرس دهی را به ۱۰۲۴ "superpage" ۴ مگی تقسیم کنید و Bitmap دومی بسازید که تنها یک عامل ساده را ردیابی کند: که آیا دراین superpage حداقل یک صفحه خالی وجود دارد یا خیر. سایز این Bitmap به این اندازه است:

۱۲۸بایت=۱ بایت به ازای هر ۸ بیت*۱بیت در هر superpage ۱۰۲۴*superpage

که مقدار ناچیزی است. اگر بهینه سازی ۳۲ بیتی هم اعمال شود این مقدار به اندازه یک آرایه ۳۲ DWORD کاهش می یابد. با جستجوی این آرایه در ابتدا می توان از مقدار متنابهی صفحه عبور کرد. بگذارید به مثال خود برگردیم ۲۰۰مگابایت در half-life اشغال شده است. چون هر ۳۲ superpage که بصورت DWORD باشد برابر است با ۱۲۸ مگابایت تمام ۲۰۰ مگابایت در اولین ۲ دو کلمه ای قرار می گیرد. پس در یک جستجوی خطی پنجاه superpage اول کنار گذاشته میشود و پنجاه و یکمین superpage دارای مقداری فضای خالی خواهد بود. تعداد تکرار برای یافتن فضای خالی ۳۲ بار است چون تعداد صفحه در هر superpage ۱۰۲۴ تا است. و این بدترین حالت ممکن است. و این نتیجه می دهد:

۵۱۲ سیکل ساعت در هر تخصیص حافظه=۲سیکل ساعت*۸ دستور*۳۲ تکرار

ولی اگر احتیاج به تخصیص بیش از یک صفحه در هر بار داشته باشیم مثلا ۶۴؟ در آن صورت تنها چیزی که شما نیاز دارید انجام دهید این است که دو تا DWORD ۰ پشت سر هم یا یک DWORD ۰ با تعداد مشخصی فضای خالی قبل و بعد از آن بیابید. و این ممکن است اندکی پیچیده تر باشد و کمی بیشتر طول بکشد مثلا ۲۰۰۰ سیکل ساعت اما با در نظر گرفتن این که حتی یک پردازنده ۸۰۴۸۶ قدیمی ۵۰۰۰۰ سیکل ساعت در یک میلی ثانیه دارد این عمل واقعا سریع انجام می پذیرد. هنگامی که یک تخصیص حافظه به طور کامل انجام شد باید :

الف) Bitmap صفحه را با صفحاتی که اشغال کرده به روز کنید.

ب) تمام superpage هایی را که فضای آدرس دهی را به آن تخصیص داده چک کنید تا مطمئن شود حداقل یک صفحه خالی مانده است.

این یک عمل خیلی سریع است و صحبت درباره آن به این بحث مربوط نمی شود. آزادسازی تقریبا ساده تر است. به سادگی فقط باید Bitmap صفحه را با صفحه هایی که آزاد شده اند به روز کنید و بیت های تمام superpage هایی را که ازاد شده اند در bitmap سوپر صفحه به ۰ تغییر دهید. زیرا که ما از آزاد سازی در می یابیم که حداقل یک صفحه خالی در superpage وجود دارد.

در نتیجه سیستم مدیریت حافظه cottontail بهترین برنامه‌ای است که من تا این لحظه به ذهنم رسیده.

Translated by ALIPACINO
Persian OS group(_LOVE_CODER_,MASTER,NETSPC)
Please contact us at

os@persiasecure.com
http://groups.google.com/group/Persian_OS
http://www.persiasecure.com/OS

Released in 2006 April