特立独行MVP

【第三章：BAT等名企面试真题解析8讲】第11节：阿里面试真题解析之内存对齐 已购

来自【《收割BAT：C++校招学习路线总结》】 | 111 浏览 | 0 回复 | 2020-02-16


特立独行MVP 专栏作者 +关注

前言

阿里的面试非常的喜欢问体系结构相关的问题。比如我在秋招面阿里云polardb团队终面当中被问到的这个问题：

> 你知道什么是内存对齐以及为什么要内存对齐么？

相信大家都思考或者看到过这个问题，看似离我们平时写代码很远的细节却能考察出我们对计算机体系结构的了解，这也是为什么在阿里的面试当中会出现这个问题的原因。接下来我将会通过几个例子讲解我对内存对齐的理解，然后以阿里面试真题为例对内存对齐相关问题进行解析。

内存对齐

内存对齐：**编译器**将程序中的每个“数据单元”安排在适当的位置上。

简单理解：按照**某种规则**将我们定义的结构体成员放在合适的地址偏移位置上存储。

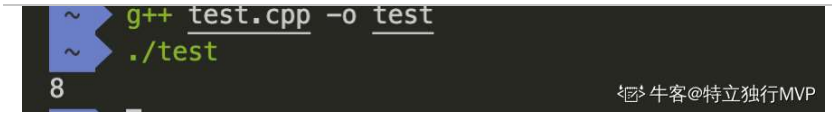
举一个例子：

```
//32位系统
#include<iostream>
using namespace std;

struct{
    int x;
    char y;
}s;

int main()
{
    cout << sizeof(s) << endl;
    return 0;
}
```

问题：上述代码的输出是多少？

答案：8

特立独行MVP



为什么要额外的3字节去填充这个结构体？一个原本5字节的结构现在变成8 字节，几乎扩大了 2 倍的存储空间，这样的空间开销是否值得？又是什么样的原因导致这样的设计？

内存对齐的原因

1.内存以字节为单位：

内存是以字节为单位存储，但是处理器并不会按照一个字节为单位去存取内存。处理器存取内存是块为单位，块的大小可以是2，4，8，16字节大小，这样的存取单位称为**内存存取粒度**。如果在64位的机器上，不论CPU是要读取第0个字节还是要读取第1个字节，在硬件上传输的信号都是一样的。因为它都会把地址0到地址7，这8个字节全部读到CPU，只是当我们是需要读取第0个字节时，丢掉后面7个字节，当我们是需要读取第1个字节，丢掉第1个和后面6个字节。所以对于计算机硬件来说，内存只能通过特定的对齐地址进行访问。

2.内存存取效率：

从内存存取效率方面考虑，内存对齐的情况下可以提升CPU存取内存的效率。比如有一个整型变量(4 字节)，现在有一块内存单元: 地址从 0~7。这个整型变量从 地址为 1 的位置开始占据了 1,2,3,4 这 4 个字节。 现在处理器需要读取这个整型变量。假设处理器是 4 字节 4 字节的读取，所以从 0 开始读读取 0,1,2,3发现并没有读完整这个变量,那么需要再读一次,读取 4,5,6,7。然后对两次读取的结果进行处理,提取出 1,2,3,4 地址的内容。需要**两次**访问内存，同时通过一些**逻辑计算**才能得到最终的结果。如果进行内存对齐，将这个整型变量放在从0开始的地址存放，那么CPU只需要一次内存读取，并且没有额外的逻辑计算。可见内存对齐之后存取的效率提升了1倍。

面试真题

1.C++当中一个空的结构体或者类的对象的大小是多少？

答案： 空的类或者结构体的大小是1个字节，因为C++当中每个实例在内存中都有一个独一无二的地址，为了达到这个目的，编译器往往会给一个空类隐含的加一个字节，这样空类在实例化后在内存得到了独一无二的地址。

2.结构体成员的声明顺序会影响结构体的大小么？比如下面两个结构体A，B他们大小是多少？

```
1 struct A  // sizeof (A) == 12
2 {
```

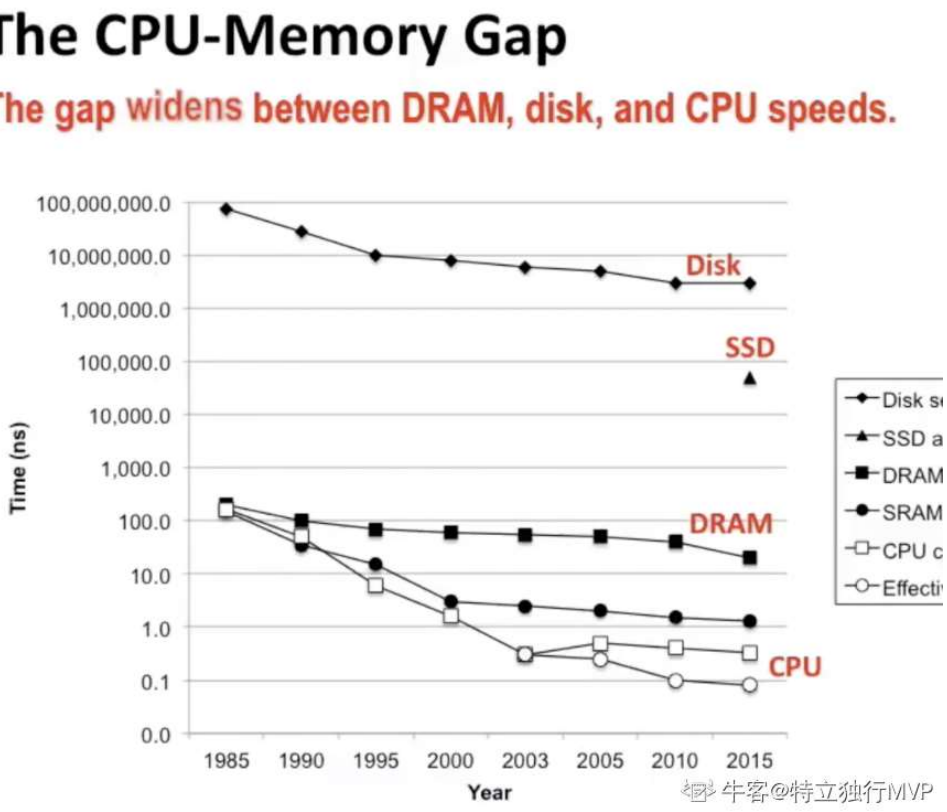
特立独行MVP

```
};
struct B  // sizeof (B) == 8
{
 char b;
 char c;
 int a;
};
```

答案：成员声明顺序会影响结构体大小。

3.内存对齐的作用是什么？

答案：提升性能：减少CPU读取内存的次数，提升程序执行的效率



上图是CPU和几种存储之间的存取速度在这30多年的发展对比（图片来自CMU的深入理解计算机系统课程）。内存就是上述的DRAM存储，CPU的速度和内存 的速度之间差距接近1000倍，3个数量级的差距。可见如果能够减少对内存的读取次数可以极大的提升程序的执 行效率。**移植原因**：有的硬件体系不支持非对齐内存地址的电路系统.当遇到非对齐内存地址的存取时,它将抛出一个异常，可能导致程序崩溃。

4.内存对齐的原则是什么？

答：三原则:

面补充空白字节；

3. 结构体总体大小能够**被最宽的成员的大小**整除，如不能则在**后面**补充空白字节；

分析：编译器在编译的时候是可以指定对齐大小的，实际使用的有效对齐其实是取指定大小和自身大小的最小值，一般默认的对齐大小是4。可以通过预编译命令#pragma pack(n)。除了上述3原则之外还有其他的对齐规则：计算机体系结构当中缓存是很重要的一环，CPU不是直接读取内存而是读取缓存：高速缓冲存储器。其作用是为了更好的利用局部性原理，减少CPU访问主存的次数。因为存取内存相对存取缓存是慢很多的，cache也可以看做是一种空间换时间的做法。实际读取内存的是缓存。所以内存对齐有的时候还需要考虑缓存更新的读取策略，一些规则如下：

1.对较大结构体进行cache line对齐：Cache与内存交换的最小单位为cache line。一个cache line大小以64字节为例。当我们的结构体大小没有与64字节对齐时，一个结构体可能就要占用比原本需要更多的cache line,同时还会带来**错误共享**问题，大家可以自行google。

2.对只读字段和读写字段分离对齐： 只读字段和读写字段分离对齐的目的是为了让只读字段和读写字段分别存储在缓存的不同cache line中，使得读写字段的淘汰尽量少的影响只读字段，因为只读字段不会被改变所以应该尽量少的被缓存换出。

5.什么是指令乱序？

答：从编译器的角度其实是对我们写的代码的一种优化，按照机器的角度讲一些指令代码执行顺序进行改变，优化程序实际执行的效率。

分析：之所以出现编译器乱序优化是因为**编译器能在很大范围内进行代码分析**,从而做出更优的执行策略,可以充分利用处理器的乱序执行功能。

指令乱序的问题：编译器优化产生的指令乱序可能会导致多线程程序产生意外的结果。

6.如何解决指令乱序问题？

答：**内存屏障**。

分析：内存屏障，是一类同步指令，是对内存随机访问的操作中的一个同步点。此点之前的所有读写操作都执行后才可以开始执行此点之后的操作。因为指令乱序执行的存在，就需要内存屏障保证程序执行的可靠。

总结

通过填充字段padding使得结构体大小与机器字倍数对齐是一种常见的做法。显然内存对齐是会浪费一些空间的。但是这种空间上得浪费却可以减少存取的时间。这是典型的一种以空间换时间的做法。在内存越来越便宜的今天，这一点点的空间上的浪费就不算什么了。**因为访问内存的速度对于处理来说是非常非常的慢， 内存访问速率对于现在 CPU 来说越来越跟不上， 额外的内**

特立独行MVP

结语

感谢阅读，如果文中有任何错误欢迎你给我留言指出，也欢迎分享给更多的朋友一起阅读。

举报

收藏 赞

相关专栏

《收割BAT：C++校招学习路线总结》

19篇文章 | 95订阅

已订阅

0条评论 默认排序

没有回复

请留下你的观点吧~

发布

专栏推荐

《收割BAT：C++校招学习路线总结》

《收割BAT：C++校招学习路线总结》，专栏共计17节。专栏分为五大主要内容，包括后台开发学习...

19篇文章 | 95阅读