理解内存屏障（一）

2015年5月25日 14:28 | 阅读 7465

作者：新浪微博（@NP等不等于P）
计算机学习微信公众号（jsj_xx）

1 前言

内存屏障是搞软件的需要面对的一个涉及硬件cpu的问题，很多人困惑不解。本文是我们对linux内核内存屏障的理解，参考linux内核（4.0版本）的Documentation/memory-barriers.txt。

2 内容

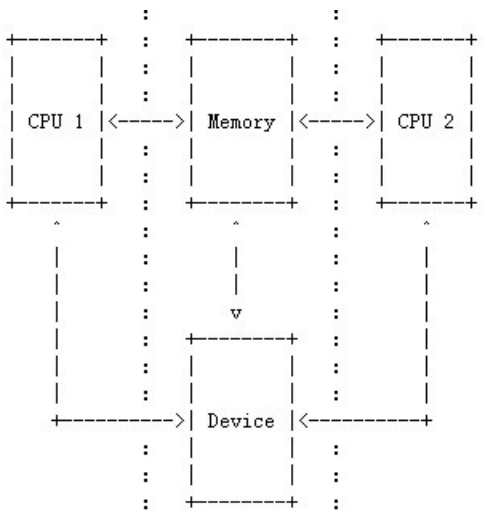
主要内容如下：

- 2.1 内存访问的的抽象模型
- 2.2 什么是内存屏障
- 2.3 内核中的显式和隐式的内存屏障
- 2.4 cpu间的锁和屏障的关系
- 2.5 哪里需要使用屏障
- 2.6 内核中io屏障的作用
- 2.7 执行有序的最小假想模型
- 2.8 cpu cache对屏障的影响
- 2.9 alpha cpu
- 2.10 一个环形缓冲区的使用样例

好，让我们开始遐想（本文需要借助想象力，否则。。。）吧！

2.1 内存访问的抽象模型

如下图，多个cpu共同访问一个内存的场景（模型）：



我们的理解是：只要能保证程序逻辑正确执行，cpu和complier（为了提升性能）怎么个乱序（优化）都行！

能？怎么“乱序”（优化）都行！

举例说明，如下：

```
CPU 1CPU 2
==========================
{ A == 1; B == 2 }
A = 3;x = B;
B = 4;y = A;
```

cpu1有2条指令，cpu2也有2条指令，共4条指令，总共有24种执行顺序（其实就是4的阶乘），够多了吧！（更可怕的是，这还是建立在一个重要假设之下的：假设cpu上执行顺序和其它cpu感知的是一样的！否则。。。）

就这个例子而言，我们只关注x和y组成的可能结果，不外乎4种（其实就是2*2）：

```
x == 2, y == 1
x == 2, y == 3
x == 4, y == 1
x == 4, y == 3
```

再举个例子：

```
CPU 1CPU 2
==========================
{ A == 1, B == 2, C = 3, P == &A, Q == &C }
B = 4;Q = P;
P = &BD = *Q;
```

只关注Q和D组成的结果的话，会出现“Q == 3”的结果么？cpu2这里有数据依赖性（或者说程序逻辑性）：必须先取Q，再取*Q！这样看，cpu2肯定是先执行“Q = P”，所以一定不会出现“Q == 3”的结果了。

再看一个例子：

```
*A = 5;
x = *D;
```

A是地址端口寄存器，D是数据端口寄存器。很明显，此时必须先放地址，再读数据，我们肯定认为只能这一种顺序，但是谁也保证不了！

至此，我们停顿下来，做个分析。貌似很乱了，软件根本搞不定，感觉是个硬件问题啊，那就让硬件做些限制（保证）吧！（cpu必须得做一些前提保证，否则软件世界大乱。。。）

前面说了，cpu会按照自己的（优化）顺序去执行指令，但一定不能违反一个大准则：程序本身的逻辑顺序。它会做如下保证：

1）有依赖的，保证保持现有顺序。

比如下面指针使用的例子：

```
ACCESS_ONCE(Q) = P; smp_read_barrier_depends(); D = ACCESS_ONCE(*Q);
```

smp_read_barrier_depends()一般为空，也就是说大部分的cpu是（不需要特殊处理）保证这种顺序的：因为得保持程序自身的依赖关系！

2）保证对重叠操作的处理保序

所谓重叠操作就是对同一内存地址的连续处理。

比如：

```
a = ACCESS_ONCE(*X); ACCESS_ONCE(*X) = b;
```

对地址X的处理顺序的两条指令就是重叠操作，cpu会保证现有顺序。

3）对毫无关系的指令，cpu保证你猜不出顺序！比如：

```
X = *A; Y = *B; *D = Z;
```

这样的指令序列，会有几种可能的顺序？6种（3的阶乘）顺序，哪种都可能！

4）cpu保证对重叠部分可能合并，可能覆盖！

比如：

```
X = *A; Y = *(A + 4);
```

此时可能有：

```
X = LOAD *A; Y = LOAD *(A + 4);
Y = LOAD *(A + 4); X = LOAD *A;
{X, Y} = LOAD {*A, *(A + 4) };
```

再比如：

```
X = *A; Y = *(A + 4);
```

此时可能有：

```
STORE *A = X; STORE *(A + 4) = Y;
STORE *(A + 4) = Y; STORE *A = X;
STORE {*A, *(A + 4) } = {X, Y};
```

可见，重叠时，cpu可能会合并（又可能导致覆盖）。

cpu能做出以上保证，算给软件稍微（一点点）减负了。

特别需要注意的是位域结构：一般地，位域操作不是线程安全的！就是说，对同一个结构体内的不同位域成员（即使连续定义的成员）的多线程访问是无法保证线程安全的：

> Do not attempt to use bitfields to synchronize parallel algorithms.

我们来仔细分析这个问题，先看跟位域相关的一个memory location定义：

> memory location
>
> either an object of scalar type, or a maximal sequence
>
> of adjacent bit-fields all having nonzero width

所谓memory location，指的是一个标量类型对象或一个最大的连续非0长度位域组。特别地，0长度位域会单独霸占一个memory location，从而隔离出memory location!

那memory location到底对线程安全有何影响？对同一memory location的访问（包括更新）不是线程安全的；对不同memory location的访问（包括更新）则是线程安全的。

更具体地讲，对于一个结构体（各个字段的类型，可能是位域，也可能不是）而言，我们总结如下几个要点：

- 位域类型和非位域类型之间的并发访问（包括更新），是线程安全（两个线程分别访问其中一个类型字段）的。
- 此结构体内部和该结构体的嵌套子结构体之间的位域字段，是线程安全的。
- 位域之间如果有0长度位域分割，则是线程安全的。
- 位域之间如果被一个非位域分割，则是线程安全的。
- 位域之间所有的位域都是非0位域，则是线程不安全的。

综上，要使访问位域线程安全化，可以采用锁，也可以在两个位域之间插入0长度位域（虽然有点浪费空间）。

好了，我们这次就讲到这里。总之，每个控制主体（compiler、各个cpu、程序逻辑本身）都会有自己所期望的顺序，那如何协调呢？下次开始讲什么是内存屏障。。。（未完待续）

关于我们

赏

打赏的人

还可以输入116字

我分享了@NP等不等于P 的文章 http://t.cn/RyhYpuk

分享

分享　赞过的人（共12人）

永科田

理解内存屏障 | 理解内存屏障（一） | 理解内存屏障（二） | 理解内存屏障（三） | 理解内存屏障（四） | 理解内存屏障（五）

2015-7-4 15:05 来自 卖代码买的Android

收藏　转发　评论

屈春河

@NP等不等于P 写的五篇文章介绍理解内存屏障 | 理解内存屏障（一） | 理解内存屏障（二） | 理解内存屏障（三） | 理解内存屏障（四） | 理解内存屏障（五）

2015-6-26 18:30 来自 微博 weibo.com

收藏　转发 4　**评论 4**

storm_0326：很不错的文章！有没有博客可以提供?

2015-7-5 06:44　回复

roylieu：

2015-6-30 00:45　回复

littlenorth：不错

2015-6-27 20:39　回复

chasex_

@femrat memory barrier | 理解内存屏障（一）

2015-6-24 14:33 来自 微博 weibo.com

收藏　转发　评论

Kinvali

理解内存屏障（一） | 理解内存屏障（一）

2015-6-7 22:07 来自 iPad客户端

收藏　转发　评论

灬毛宏斌灬

mark | 理解内存屏障（一）

2015-6-2 23:52 来自 iPhone 6

收藏　转发　**评论 1**　1

葬寒澈：屌屌的
2015-6-3 17:08 回复

此v间
@我的印象笔记 理解内存屏障（一）
2015-6-2 23:23 来自 荣耀6 Plus
收藏 转发 评论

TsingTao_皂户_小明
理解内存屏障（一） 理解内存屏障（一）
2015-5-30 09:12 来自 iPad客户端
收藏 转发 评论

竹澈2002
@我的印象笔记 理解内存屏障（一）
2015-5-30 01:00 来自 nubia Z5S mini
收藏 转发 评论

野骆驼祥子
@我的印象笔记 理解内存屏障（一）
2015-5-29 17:39 来自 一加手机 不将就
收藏 转发 评论

胖鱼的微笑
@印象笔记 理解内存屏障（一）
2015-5-29 17:39 来自 iPhone客户端
收藏 转发 评论

NP等不等于P
内存屏障是cpu设计者甩给软件设计者的包袱？我们一起看看。。。 我分享了@NP等不等于P 的文章 理解内存屏障（一）
2015-5-25 14:30 来自 微博 weibo.com
收藏 转发 100 **评论 7** 3

袖手旁观ML：mark
2015-8-3 18:23 回复

路易斯偶偶：@mywiz
2015-6-3 09:25 回复

csddl：@mywiz
2015-6-3 00:26 回复