第一次实验方法简介

一些基本前提

- Cache 大小是 2 的整数次幂
  - 4,8,16… …(KB)
- Cache block 大小是 2 的整数次幂
  - 8,16,32,64,128… …(B)
- Cache 相连度大小是 2 的整数次幂
  - 2,4,8,16,32,64,128(way)

测量 Cache 大小

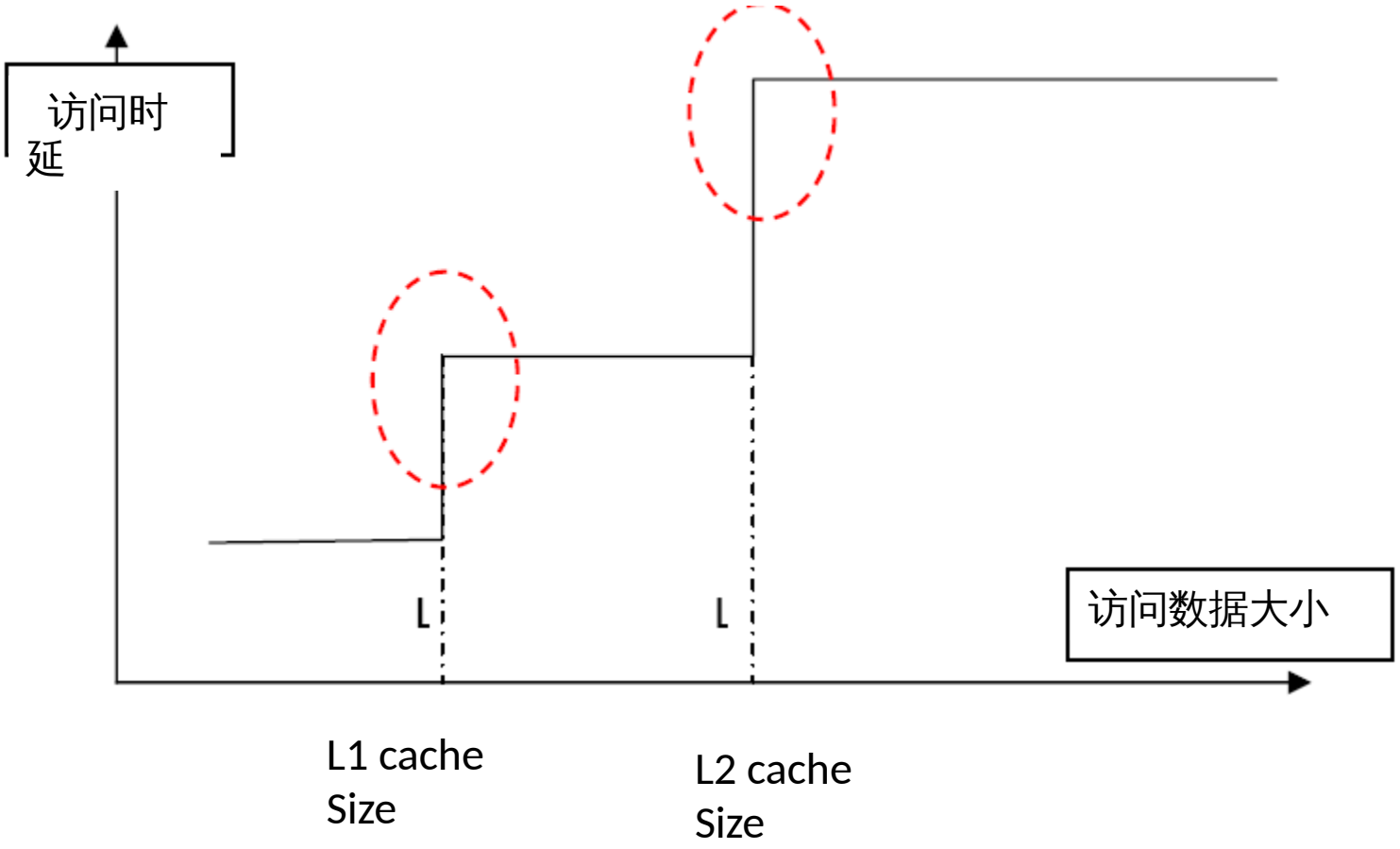
- 最简单的办法是不断的在内存中读取连续数据，数据块的大小可以从 1kB 一直到 16MB ，然后观察平均读取速度。
- 当数据块大小超过 L1 Cache 后，会出现 Cache 读缺失，平均读取速度会有一个突然增加的过程
- 同理，在数据块大小超过 L2 cache 后，平均读取速度也会有一个突然增加的过程

测量 Cache 大小

- 测量 Cache 块大小，主要是利用 L1 Data Cache 和 L2 Data Cache 和主存之间的速度差异。如果一个数组能够完全存放在 L1 Data Cache 中，那么在一个循环里不停的访问数组，第一遍循环数组就会从主存被搬到了 L1 Data Cache 中，以后每次循环就会直接从 L1 Data Cache 中找数组数据，这样速度就会很快。

- 如果一个数组超过了 L1 Data Cache 的大小，比如 L1 Data Cache 的两倍，那么整个数组就不能被存放在 L1 Data Cache 中，此时采用某种访问方式，可以使得数组的循环访问每次都是缺失的 ，都必须从 L2 Data Cache 中找， L2 Data Cache 的速度低于 L1 Data Cache 的速度，从而使得这个大小数组的数组访问延迟明显比其它小数组访问延迟要大。

一个简单的示意图



测量 Cache block 大小

- CPU 与 Cache 之间的数据交换是以字为单位，而 Cache 与主存之间的数据交换是以块为单位，一个块是由若干字组成，是定长的。
- 也就是说，当某个字节访问缺失时，读进来的是一个块的大小。

- 对于数组访问，如果是连续访问，第一个字节缺失的话，一个块就被存放在了 cache 中，以后 cache 块顺序访问的字节就命中了，因此命中率就会很高。
- 如果访问是间断的，对数组间隔顺序访问，命中率就会降低，平均访问延迟增大。当某次间隔达到一定大小，即超过 cache 块大小，有可能造成每次都缺失的最坏情况。
- 结合数组的大小和间隔的大小，依次增加间隔的大小和数组访问的大小，会使得访问同样次数的时间变大。然后处于跳变的值即为 cache 块大小。

测量 Cache 的相连度

- 为了测量相连度，用一个 2 倍 cache 大小的数组。
- 比如如果是 2 路组相连的结构，
- 如果将数组分为 4 块，访问数组的第一块和第三块，由于是 2 路组相连，第一块对应的 cache 块和第三块对应的 cache 块映射到同一个组里，可以同时被放到这个组里。
- 如果将数组分为 8 块，访问数组的第一块、第三块、第五块、第七块，这时第一、三、五、七块对应的 cache 块映射到同一个组里，但一个组只能容纳两个 cache 块，因此不停的循环访问会造成 cache 块不断被替换出去，使得访存变得很慢。
- 因此可以逐渐增大数组的分组，当某一次访问时间变慢时，前一次就是相连度的值

测量 Cache write back/write through

- 对于写直达，数据在写到 Cache 块的同时还要写入更低一层的存储器块中。
- 对于写回法，数据仅被写入 cache 中，只有 cache 中数据块被替换出去时，才会写回到主存中。
- 因此写直达会一直访问下一层存储，而写回法，如果不是被替换的话，一直是在 cache 中进行访问。
- 因此设计两种情况：
- 1）数组被赋值修改，但是数组大小小于 cache 的大小，数组没有被替换，如果是写回法，没有对主存储器访问。如果是写直达法则不停访问主存储器。
- 2）数组的访问一直是缺失的，这样一直访问存储器。
- 对于两种情况的时间进行对比：如果是写回法，差距很大，如果是写直达法则差距不大。

测量 Cache 替换策略

- 根据前面测得的相连度，自行设计不同的访问序列，产生不同次数的缺失。
- 如果实验能够做到这里，就不需要更加详细的解释了吧☺