1、

UCA结构是集中式共享结构，多个处理器核通过总线或者交叉开关连接最后一级缓存(LLC)，所有处理器核对LLC的访问延迟相同。UCA结构的优点是实现简单，存储开销较小。但是，UCA结构使用总线或者交叉开关互联，缺点是拓展性差，不能用于核心众多的场景。

NUCA结构是一种分布式共享结构，拥有较多计算核心，且片上网络不再是简单的总线结构，故常常依赖于基于目录的缓存一致性协议，拥有良好的可扩展性。NUCA结构的优点是能够支持更多的核心，缓存访问延迟相对较小。但是，NUCA结构的缺点是实现复杂，存储开销较大。

2、

P1 P2

1:x=1; 3:y=1;

2:print Y; 4:print X

顺序一致性要求所有节点对于同一个对象的访问都必须按照相同的顺序进行，即所有节点对于同一个对象的操作顺序必须相同。

弱一致性则允许不同节点对于同一个对象的访问顺序不同，但是最终所有节点都会收敛到相同的值。

因此对于顺序一致性：有6种正确的执行顺序

执行顺序 输出

1234 0 1

1324 1 1

1342 1 1

3412 1 0

3142 1 1

3124 1 1

X Y的正确结果应该是1，1

弱一致性：每个处理器的各自的赋值和打印没有相关，故1234的全排列共24种执行顺序全部正确。输出值1 0，0 1，1 1，0 0均有可能

3、

在 MESI 协议中，当高速缓存行被修改时，它会被标记为 "M"（已修改），并只存在于当前高速缓存中。当其他处理器需要访问该缓存行时，它们必须先将数据写回内存，然后才能访问该缓存行。可以减少Cache到内存的写回次数，提高了处理器性能。

相比之下，ESI 协议没有 "M "状态。当高速缓存行被修改时，必须在其他处理器访问前将其写回内存。这增加了高速缓存一致性所需的通信量，降低了处理器性能。

总的来说，MESI协议相对于ESI协议来说，具有更好的性能和更少的通信量。

4、

// Fetch\_and\_Increment原子指令实现自旋锁

void lock(atomic\_int\* lock\_var) {

while (atomic\_fetch\_increment (lock\_var) != 0) {

// 自旋等待

}

}

void unlock(atomic\_int\* lock\_var) {

atomic\_store(lock\_var, 0);

}

// Cmpare\_and\_Swap原子指令实现自旋锁

void lock(atomic\_int\* lock\_var) {

int expected = 0;

while (!(Compare\_and\_Swap (lock\_var, &expected)==1)) {

expected = 0;

// 自旋等待

}

}

void unlock(atomic\_int\* lock\_var) {

atomic\_store(lock\_var, 0);

}

自旋锁虽然避免了线程切换的开销，但是会占用处理器时间。可以根据前一次在同一个锁上的自旋时间及锁的拥有者状态来决定是否自旋，从而避免了自旋锁占用处理器时间过长的问题。

锁粒度过大会导致锁争用，锁粒度过小会导致锁开销过大。可以根据具体情况对锁粒度进行优化，以提高程序的性能。

实现读写分离锁，可以将读操作和写操作分别加锁，从而提高程序的并发性能。

实现分段锁，将数据分成多个段，每个段都加锁，从而提高程序的并发性能。

5、

即应该保证多线程之间不需要共享的变量的地址应保持至少1个cache行的间隔。

1、将不同的变量放在不同的缓存行中。

2、使用填充变量来避免不同变量之间的相互影响。

3、使用缓存行对齐来避免不同变量之间的相互影响。

6、

· 虚通道可以将一个物理通道分成多个逻辑通道，从而实现多路复用。一个物理通道可包含多个虚通道，每个虚通道包含独立的缓冲区和状态标识，物理通道选择一个虚通道的数据进行传输。当某个虚通道阻塞时，可切换到其它虚通道进行传输，提高整体吞吐率。虚通道的设置，可以避免死锁。提高网络的性能和可靠性，提高网络的带宽利用率，降低网络的延迟和丢包率。

7、

Fermi GPU架构的存储层次结构包括L1 Cache、L2 Cache、Shared Memory、Global Memory等。这些存储器的带宽和延迟如下表所示：

| **存储器** | **带宽** | **延迟** | **是否共享** |
| --- | --- | --- | --- |
| L1 Cache | 128 bytes/clock | 1 cycle | 不共享 |
| L2 Cache | 64 bytes/clock | 5 cycles | 不共享 |
| Shared Memory | 128 bytes/clock | 1 cycle | 共享 |
| Global Memory | 192 bytes/clock | 400-600 cycles | 不共享 |

其中，L1 Cache和L2 Cache是私有存储器，Shared Memory是线程块内共享存储器，Global Memory是全局存储器。