在硬件加速中，主要考虑并行加速。目前主流的并行编程环境主要有基于共享存储平台的OpenMP和基于消息传递平台的MPI。OpenMP是应用在共享储存多处理器上的并行编程标准，是一种易于使用的并行编程标准，是线程级的并行。它不是一种语言，而是用指令集扩展语言，采用的是Fork-Join并行编程模型。MPI (message passing interface) 是在分布式内存（distributed-memory）之间实现信息通讯的一种规范/标准/协议（standard），是进程级的并行。

本文采用的并行策略如下：

（1）MPICH2

MPICh2是MPICH的一个版本，它是MPI标准最重要的一种实现。在本文中，MPI并行硬件加速的实现就是通过MPICH2实现的。MPI 允许静态任务调度，显示并行提供了良好的性能和移植性，用 MPI 编写的程序可直接在多核集群上运行。在集群系统中，集群的各节点之间可以采用 MPI 编程模型进行程序设计，每个节点都有自己的内存，可以对本地的指令和数据直接进行访问，各节点之间通过互联网络进行消息传递，这样设计具有很好的可移植性，完备的异步通信功能，较强的可扩展性等优点。MPI 模型存在一些不足，包括：程序的分解、开发和调试相对困难，而且通常要求对代码做大量的改动；通信会造成很大的开销，为了最小化延迟，并行可能伴随大量的通信；动态负载平衡困难；并行化改进需要大量地修改原有的串行代码，调试难度比较大

（2）OpenMP

OpenMP在节点内执行的基于共享内存的编程模型。OpenMP是针对单主机上多核/多CPU并行计算而设计的工具，换句话说，OpenMP更适合单台计算机共享内存结构上的并行计算。由于使用线程间共享内存的方式协调并行计算，它在多核/多CPU结构上的效率很高、内存开销小、编程语句简洁直观，因此编程容易、编译器实现也容易（现在最新版的C、C++、Fortran编译器基本上都内置OpenMP支持）。不过OpenMP主要缺陷是只能在单台主机上工作，不能用于多台主机间的并行计算。

在本案例中我们对例题2，例题4，例题5，进行了计算，得到的加速比，如图：

（3） MPICH2/OpenMP的混合并行

在大规模节点间的并行时，由于节点间通讯的量是成平方项增长的，所以带宽很快就会显得不够。所以一种思路增加程序效率线性的方法是用mpi/openmp混合编写并行部分。大致思路是每个节点分配1-2个mpi进程后，每个mpi进程执行多个openmp线程。openmp部分由于不需要进程间通信，直接通过内存共享方式交换信息，不走网络带宽，所以可以显著减少程序所需通讯的信息。在本文中对MPICH2/OpenMP也进行了实现。