协作式协同调度：一种面向非专用异质工作站网络的协同方法

有研究证明，工作站网络（NOW, Network of WorkStation）中存在着大量空闲的计算资源（如 CPU 和内存）。本文主要研究使用这些空闲计算资源执行分布式应用的可能性，并且要求在不影响每 个工作站本地应用性能的情况下，使其性能与大规模并行处理机（MPP，massively parallel processors）相当。

上述两个目标可以通过协同调度技术达到。传统的协同调度算法是通过同时调度分布式应用的组成任务（forming tasks） 或 组成任务的子集来减少分布式应用的通信等待时间。这样的同时执行是由群调度（gang scheduling），又被称为显示的协同调度，所支持。 在群调度中，一个作业的所有进程被同时调度或不调度（de-schduled），因此，需要事先知道一个作业由哪些进程组成。但是，在类似工作站网络的分 布式系统中，这样的信息是难以获得的。另外一种方案则是根据协同调度的需要来确定进程，又称协作任务，也就是在执行的过程中收集和分析隐式的运行时信息， 这些信息通常是通信事件。这样，只有进程的一个子集被同时调度，由此产生的是隐式的协同调度，而不是群调度。

以前的研究表明 在不损害并行作业性能的情况下分配多个并行作业是合理而且可能的。隐式的协同调度方案不仅降低作业的响应时间，同时也增加整个系统的吞吐量，这是因为多道 程序释放了其它的结点。然而，从本地用户的角度来看，当并行应用（MPL）的数量高于 1 时，这个方案并不能保证本地用户的性能。就是说，当 MPL 增长时， 简单地分析通信事件不足以保证本地作业和并发作业的性能。基本上，这是由于：（a）通信驱动的调度没有限制分配给并发用户的资源；（b）由于本地用户存在 而产生的负载不均衡严重降低并发作业的性能；（c）必须根据集群中结点间的异质度来分配 CPU 和内存；（d）当 MPL>1 时，过度的页面调度会降低 整个系统的性能。

为解决上述问题，一个有效的协同调度技术必须处理以下方面：

1. 自适应且平衡的资源分配。必须根据每个作业的CPU和内存需求和集群的异质性来动态调整MPL的值。类似地，对于同一个作业，分配给不同任务的计算资源必须是均衡的。

2. 通信同步进程的协同调度。任何进程都不应该等待一个还未被调度的协作进程与之进行同步/通信操作，并且在同步点上的等待时间必须被最小化。

3. 本地用户和并行用户间的社会化契约。必须保证本地应用的性能在某个特定的阀值之上。

本 文提出一种称为协作式协同调度（CCS，Cooperating CoScheduling）的新技术，它拥有上述的协同调度的特性。不同于传统的协同调度技术，在CCS中，每个结点结合本地事件（基本上是网络通信、内 存、I/O和PU）和来自远程结点的外部事件来决定自己的调度决策。这就允许CCS通过预留一定比例的CPU和内存资源来提供社会化契约，以保证并行作业 不会影响到本地用户，同时保证通信任务的协同调度。此外，在需要时，CCS算法还使用协作结点的状态信息来均衡集群的资源。

为详细说明 CCS算法，本文提出一个针对非专用工作站网络系统的完整的协同调度模型。基于这个模型，详细描述CCS 协作调度算法，同时展示一个Linux-PVM环境下的CCS实现。通过这个实现，可以在两种不同的异质环境，即控制式工作站网络和产生式 （production）工作站网络，中比较CCS和传统的协同调度技术的性能。在这样的场景中，三个有不同通信需求的并行工作负载（workload） 被执行。并且，每个并行工作负载都是与由多个local user profiles组成的本地工作负载一起执行。一般地，不论从本地用户还是从并行用户的角度来说，实验显示CCS的性能都优于其它被评价的协同调度算法。 实现结果还表明对于非专用工作站网络，倾向于短期策略，将使得空闲资源的利用更有效率，因此能获得令人满意的性能提高（在许多情况下给定应用的半数以上的 任务都得到提速），并且本地用户几乎感觉不到其所产生的额外开销。