并行计算机博弈系统设计与改进

李天明 中共天水市委党校 甘肃天水 741000

【文章摘要】
计算机博弈的核心技术是搜索。本文针对单核条件下的搜索算法、并行条件下的搜索算法两个方面进行了研究。单核搜索算法方面，简单地介绍了博弈单核条件下的选择性搜索思想，本文研究了几种已有的多核条件下博弈并行搜索方法，分析了其优劣，并针对性地提出了改进方案，给出了系统实践的结果，比较了改进方法与原有方法的效果差异。最后，本文探讨了 Alpha-Beta 并行搜索系统实现中一些特殊的程序设计技巧。

【关键字】
计算机博弈；选择性搜索；并行 Alpha-Beta 搜索

1 前言

随着并行计算技术和框架的日益成熟，搭建并行计算环境并非难事。对于博弈并行计算来说，如何设计更好的并行算法模型，以尽量减少三种额外开销称为了关键问题。在大多数系统实践中，由于无法很好的避免额外开销，在多核计算中系统的加速比较低，甚至在计算核数达到一定数量后，加速比会出现负增长，这使得博弈计算的进一步提速遇到了瓶颈。

2 单核搜索算法简介

早期的博弈程序大多会展开一棵完整的博弈树。尽管 Alpha-Beta 等思想利用启发式信息对博弈树的实际展开进行了剪枝，但逻辑上仍然是展开了一棵完全树。随着计算机博弈研究的进展，学者认为原有的搜索方法不能体现人类智能在下棋中的思维习惯。上世纪 80 年代出现了选择性搜索的思想，这一思想更有助于提高搜索效率，因此逐渐成为主流方法。本章主要介绍计算机博弈单核条件下选择性搜索的基本思想，以及选择性搜索算法的代表——1997 年著名弈棋机"深蓝"使用的"DC 延展搜索"。在此基础上，针对"DC 延展搜索"的缺陷提出改进方案，并给出了这一改进的实验结果。

3 博弈并行搜索模型研究

博弈中的并行计算是为了加强计算机的运算能力，从而获得更强的智能表现。在以搜索为主结构的博弈系统中，并行计算模型主要是指对 Alpha-Beta 搜索进行分解，也即并行 Alpha-Beta 搜索。目前主要的做法有顶层分支模型、PVS(Principle Variation Splitting)、YBWC(Young Brother Wait Concept) 等。这些算法本身并不涉及实现方式，但为讨论方便，下文在提及这些算法及其改进的时候，都是以多线程实现为例。

3.1 顶层分支并行模型

3.1.1 顶层分支模型简介

顶层分支模型，即将博弈树根结点的所有子结点交付不同线程进行计算。这种做法其实几乎完全放弃了通信，因此通信开销几乎是零。然而相对应的，造成了大量的搜索开销，每个并行分支的平均剪枝效率远小于串行搜索。由于分解粒度较粗，各分支内部的搜索速度也可能很不均衡，会造成一些线程早早结束工作，而另一些线程则慢很多的现象，这样造成的同步开销也比较大。

3.1.2 顶层分支模型的缺陷

顶层分支模型存在三个主要缺陷。

1. 负载不均衡

若所有可用的计算核的搜索速度都相同，其每秒搜索的结点数 NPS(Node Per Second)均为 S，则每棵子树的搜索时间为 $T_i = N_i / S, 1 \le i \le n$。由于每棵子树的内部搜索结点树与其剪枝情况有关，因此结点数并不相同。需搜索结点数最大的子树结点数为 N_{MAX}，搜索时间为 T_{MAX}，则平均运算速度 $\bar{S} = \sum_{j=1}^{n} N_i / (n * T_{MAX}) < (n * N_{MAX}) / (n * N_{MAX}) = S$，也即没有发挥出处理器的实际运算能力。

2. 搜索开销较高

在串行 Alpha-Beta 搜索中，窗口的搜索小是造成剪枝的原因，从而减少搜索结点数。在基础顶层分支并行模型下，由于各子树分别维护一个搜索窗口而无法共享，因此各子树剪枝率均小于等于串行搜索下的剪枝率。

3. 并行能力受到分支因子限制

对于一个给定的游戏，其每个局面下的着法是有一定范围的，如中国象棋一般不超过 60 个着法。如果此时的可用核数超过 60，则有一部分核无法得到利用。

3.1Principle Variation Splitting(PV splitting)

对于顶层分支模型的最早的改进是 PV 分解(Principle Variation Splitting)。PV 分解模型将搜索树中的每个结点按 Knuth and Moore[6] 提出的分类方法进行分类：

PV 结点——根结点以及每个 PV 结点的第一个子结点

CUT 结点——PV 结点除第一个孩子外的所有结点，以及所有 ALL 结点的所有孩子

ALL 结点——CUT 结点的第一个孩子

其他结点——所有不符合上述定义的结点

3.2 改进的顶层分支模型

在目前主流的计算机博弈系统中，很少有采用 PVS 与 YBWC 模型的，大多数还是采用顶层分支模型。原因是 PVS 与 YBWC 虽然在理论上是成功的，但在实际操作中会造成更多的控制与分配开销，且带来的改进效果受不同棋种、不同系统自身特点影响，在很多系统中实际效果与基础顶层分支模型相差不大。而顶层分支模型简单、易于根据不同系统的特点进行改进，为大多数系统所采用。

3.3 动态延展

顶层分支模型只在顶层对所有子结点进行搜索。这里对这一做法加以扩展，得到动态延展的方法。

对于给定的空闲核数 n，游戏的分支因子为 b，则从根节点展开 k 层，使得 $b^k \ge n$，然后再将所有的以叶子结点为根节点的子树按上述动态分配方法，分配给空闲核进行计算。可以看出，原有的顶层分支模型是这里的一个特例，即 n 较小，使得 k 为 1 时即满足条件并展开。

动态延展不仅是用以应对大规模并行的需求。这种方法的实质是减小了工作的粒度，配合后面将要提出的"动态分配着法"以及 "共享高层窗口"的改进，可以有效的减少同步开销，并在通信开销和搜索开销之间取得较好的平衡。

4 结论

博弈并行计算目前的研究主要是对于博弈树进行并行分解，这一做法确实有较大的瓶颈，每一点改进都较艰难，且效果并不显著。然而，如果将博弈并行的重心放在对博弈中复杂运算函数的并行，将能取得很明显的效果。本文工作结束后，作者开始尝试使用多显卡技术和多 CPU 技术融合，实现双并行的结构，将博弈树分解交给 CPU 处理，而运算函数的并行则在 GPU 中。

【参考文献】
[1]Manohararajah, Valavan. Parallel Alpha-Beta Search on Shared Memory Multiprocessors[M] . Canada: National Library of Canada,2001. 19-27.
[2]Russell, Stuart J. and Norvig, Peter. Artificial Intelligence: A Modern Approach, Second Edition[M]. NJ: Pearson Education Inc,2003. 161 - 171.
[3]Hsu, Feng-hsiung. Behind Deep Blue: Building the Computer that Defeated the World Chess Champion[M], US:Princeton University Press, 2002. 105-172.