第 34 卷 第 6 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.6 Computer Engineering March 2008
·软件技术与数据库· 文章编号：1000—3428(2008)06—0085—03 文献标识码：A 中图分类号：TP311

一种适合 P2P MMOG 的动态调度策略

石祥滨[1,2]，王 越[2]，李 强[2]，王晓光[2]，王 晶[2]
(1. 沈阳航空工业学院计算机学院，沈阳 110034；辽宁大学信息科学与技术学院，沈阳 110036)

摘 要：移动代理在 P2P MMOG 中的应用带来了新的实时任务调度问题。该文提出了一种适合 P2P MMOG 的优先级动态变更调度策略，根据代理服务信任度值、代理服务所在玩家机器的 CPU 使用率、玩家机器的网络流量和任务在调度队列中等待的时间动态调度玩家任务。实验证明，该方案满足游戏实时性要求，降低了系统延迟，解决了 P2P MMOG 中玩家节点的负载均衡问题。

关键词：P2P MMOG 结构；移动代理；任务调度

Dynamic Schedule Strategy Suitable for P2P MMOG

SHI Xiang-bin[1, 2], WANG Yue[2], LI Qiang[2], WANG Xiao-guang[2], WANG Jing[2]
(1. Department of Computer , Shenyang Institute of Aeronautical Engineering, Shenyang 110034;
2. School of Information Science and Technology, Liaoning University, Shenyang 110036)

【Abstract】 Mobile agent brings new problems about real-time scheduler in the applications of P2P MMOG. This paper proposes a dynamic priority-variation strategy which can schedule the players' tasks dynamically according to the trusted degree of agent services, the CPU occupating coefficient of agent service in the players' computers, the network traffic of the players' computers and the tasks' latency time in the scheduler queue. The simulation results show that this strategy fulfills the real-time demand of the games, shortens the system latency time, and solves the problem about load balance in P2P MMOG.

【Key words】 P2P MMOG; mobile agent; task schedule

1 概述

基于P2P的MMOG(Massively Multiplayer Online Games)通常采用混合式P2P结构，该结构能够高效地利用区域内所有玩家节点的计算资源，但是一些玩家节点经常因任务过重导致超载，从而影响P2P MMOG的可伸缩性。移动代理凭借其根据运行环境的变化携带任务动态迁移的能力成功地解决了上述问题[1]。该方案将MMOG中的典型功能抽象成各个代理服务，这些代理服务部署在区域内的若干玩家节点上。区域内所有玩家的任务请求被调度给这些代理服务完成。因此，将玩家的任务请求迅速地调度给各个代理服务处理是解决P2P MMOG可伸缩性的关键，此外MMOG的实时性需求给任务调度策略提出了更高的要求。

目前，基于优先级的调度算法在实时系统中有着广泛的应用，这类算法主要包括单调速率算法(RM)、截止期最早最优先算法(EDF)、空闲时间最短最优先算法(LSF)、价值最高最优先算法(HVF)和价值密度最大最优先算法(HVDF)等[2]。在这些算法中，任务的优先级都是基于任务的某个特征参数，如截止期、空闲时间或关键性(即任务的重要程度或者价值)等计算而得。然而，如果优先级仅由某个特征参数来确定是不够的，如EDF策略是将最高优先级指派给具有最早截止期的任务，LSF策略是将最高优先级指派给具有最短空闲时间的任务，尽管在正常的系统负载下这些算法表明了其最优性，但是在过载的情况下，系统不可能保证所有的任务都能够满足截止期，这时EDF或者LSF算法会出现急剧的性能降级，甚至导致多米诺现象[2]。

此外，蚁群算法具有较强的稳定性、分散性和鲁棒性，算法中的信息素不断加强，采用正反馈原理，利用随机选择策略，加快进化过程。分布式环境中的任务调度问题是NP问题。大量实验表明蚁群算法是一种有效的求解NP 类问题的新型算法，能够综合考虑多个特征参数并且易于和其他方法结合。虽然蚁群算法[3]能成功解决复杂的任务调度问题，但不能满足实时任务调度系统的需求。

为了解决 P2P MMOG 中的任务调度问题，本文提出了一种优先级动态变更调度策略。其中，任务优先级采用 EDF 算法计算，代理服务优先级采用蚁群算法计算。该策略综合考虑 P2P MMOG 中的网络流量、代理服务信任值、代理服务所在玩家机器的 CPU 使用率和任务在调度队列中等待时间，动态决定任务调度。此外，针对不同的实时调度系统，该策略还可以综合考虑更多的特征参数。本文给出了优先级动态变更调度的策略实现，并进行了仿真实验与分析。

2 P2P-Agent 框架的网络游戏中间件体系结构

本文设计的优先级动态变更调度策略是基于目前正在实现的一种P2P-Agent框架的网络游戏中间件系统，该中间件体系结构如图 1 所示。系统中，玩家代理服务被部署在每个玩家机器上。它是静态代理，负责玩家与游戏世界的沟通。此外，系统游戏任务被抽象成同步代理服务[4]、兴趣管理代理服务[5]、逻辑判定代理服务和世界管理代理服务，这些代理

基金项目：辽宁省自然科学基金资助项目“基于 P2P 的 MMOG 关键技术的研究”(20052007) 辽宁省教育厅攻关计划基金资助项目“网络游戏引擎及其相关技术的研究”(2004D116)

作者简介：石祥滨(1963－)，男，教授、博士，主研方向：分布式系统，网络游戏，数据库技术；王 越、李 强、王晓光、王 晶，硕士研究生

收稿日期：2007-03-30 **E-mail：**sxb@lnu.edu.cn

是动态代理。动态代理负责实时处理玩家的任务请求。区域内所有代理服务存储在各个玩家节点上，玩家的任务请求由相关的代理服务完成。于是，根据P2P MMOG的特点量身定做任务调度策略是本文拟解决的主要问题。

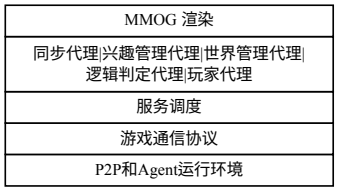

图 1　P2P-Agent 框架的网络游戏中间件体系结构

3　优先级动态变更调度策略

优先级动态变更调度策略中包含任务优先级计算算法、代理服务优先级计算算法和优先级动态变更调度算法，该策略是非抢占的调度策略。其中，玩家任务请求和代理服务的优先级是根据代理服务信任度值、代理服务所在玩家机器的CPU 使用率、玩家机器的网络流量和任务在调度队列中等待的时间 4 个因素动态确定的。优先级动态变更调度策略的思想是将优先级最高的玩家任务请求分配给具有最高优先级的代理服务，同时该代理服务的优先级立刻变为最低。当任务完成之后，综合代理服务原来的优先级以及玩家对代理服务的服务评价 2 个因素动态确定代理服务的新优先级。游戏过程中，玩家将任务请求信息发送给协调者，协调者将这些任务组成任务队列。协调者采用优先级动态变更调度策略进行动态调度，从而保证各个玩家任务被实时地处理，进而满足了 MMOG 动态变化的需求。

(1)任务优先级计算算法

本文采用 EDF 算法确定任务的优先级。EDF 算法是动态优先级驱动的算法，该算法根据任务的开始截止时间确定任务的优先级。截止时间越早，其优先级越高。算法具体介绍如下：

调度队列表示为 $Q=\{REQ_1,REQ_2,\cdots,REQ_n\}$，其中 n 为队列中任务的数量且任务按照优先级降序排列。每个任务请求由 $REQ=<Pid,PJ>$ 表示，其中，Pid 表示发送该请求消息的玩家；PJ 表示玩家任务。$PJ=(ST,DT,ET,T\text{-}PRI)$，其中，ST 表示任务的提交时间； DT 表示任务的截止时间； ET 表示代理服务对象处理该任务的时间； T-PRI 表示任务的优先级。调度队列每隔 t 时间段就行进一次更新。当协调者接收一个新的任务请求信息后，立刻将其按优先级降序插入到调度队列中。

任务优先级计算算法：

$$REQ_i.PJ_i.T\text{-}PRI_i=REQ_i.PJ_i.DT_i-REQ_i.PJ_i.ST_i-REQ_i.PJ_i.ET_i$$

调度队列按照任务优先级由高到低的顺序排列任务。调度队列任务排序算法伪代码如下：

```
TASK-SORT(Q)
{
    for(i=1；i<=n;i++)
    REQi.PJi.T-PRIi=REQi.PJi.DTi－REQi.PJi.STi－REQi.PJi.ETi;
    按照T-PRI的值进行降序排序;
}
```

(2)代理服务优先级计算算法

由于 MMOG 动态地变化，因此代理服务的优先级需要动态地确定。本文采用蚁群算法根据代理服务信任度值、代理服务所在玩家机器的 CPU 使用率和玩家机器的网络流量 3 个因素动态确定代理服务优先级。优先级表明代理服务提供的服务质量，优先级越高，服务质量越好。

代理服务队列用 $AQ=\{A_i\},(i=1,2,\cdots,s)$ 表示，其中 s 表示代理服务的数量。每一个代理服务对象用 $A_i=(\lambda(A_i,t),\mu(A_i,t),tra(A_i,t),\zeta(A_i,t))$ 表示，其中 $\lambda(A_i,t)$ 表示代理服务信任度值；$\mu(A_i,t)$ 表示该代理服务所在玩家机器的 CPU 使用率，$\overline{\mu}$ 是 $\mu(A_i,t)$ 的阈值；$tra(A_i,t)$ 表示 t 时刻代理服务所在玩家机器的网络流量；$\zeta(A_i,t)$ 表示代理服务的优先级。

该算法将代理服务信任度值 $\lambda(A_i,t)$ 作为信息素，初始时刻 $\lambda(A_i,t)=C$(C表示常数)。玩家将根据代理服务的服务质量给予一个信任评价 ω，代理服务的信任值根据 ω 进行更新。当代理服务迁移到其他玩家机器后，该代理服务的信任值将变为 C。算法中信息素浓度用代理服务信任值 $\lambda(A_i,t)$ 的密度 $\tau(A_i,t)$ 表示，该密度可用公式

$$\tau(A_i,t)=\begin{cases}\dfrac{\lambda(A_i,t)}{\sum_{m=1}^{s}\lambda(A_m,t)} & t\neq 0\\ \dfrac{1}{s} & t=0\end{cases}$$

求出。为了防止 $\tau(A_i,t)$ 无限制累加。本算法采用了信息素挥发系数 χ，随着时间的推移，信息素调整规则为 $\tau(A_i,t+1)=\chi\tau(A_i,t)+\omega(\chi\in[0,1])$。此外，与 $\tau(A_i,t)$ 相关联的基于问题的启发式信息值有 $\mu(A_i,t)$ 和 $tra(A_i,t)$。

$tra(A_i,t)$ 的计算方法采用文献[6]中的游戏流量模型 $f(x)=\frac{1}{b}e^{\frac{x-a}{b}}e^{-e^{\frac{x-a}{b}}}(b>0)$，其中，$a$，$b$ 为参数，x 为数据包发送的时间间隔，$f(x)$ 表示 x 时间间隔内发包的概率。由该模型可以预测目的玩家节点的未来某时刻的网络流量 $tra(A_i,t)$，这里 $tra(A_i,t)=\lim_{x\to x_0}(f(x)package\_size)$，其中 $package\_size$ 为游戏中数据包大小。若目的节点能够满足网络流量的需求，应有 $tra(A_i,t)<\overline{Tra}$。$tra(A_i,t)$ 最终可以化简为

$$tra(A_i,t)=\frac{e^{\frac{a}{b}}}{b}package\_size\lim_{x\to x_0}\left[\left(e^{-e^{\frac{x}{b}}}\right)^{e^{\frac{x}{b}}}e^{-\frac{x}{b}}\right]$$

最后，代理服务优先级可根据公式

$$\zeta(A_i,t)=\frac{\tau^{\alpha}(A_i,t)\mu^{\beta}(A_i,t)tra^{\gamma}(A_i,t)}{\sum_{j=1}^{s}\tau^{\alpha}(A_j,t)\mu^{\beta}(A_j,t)tra^{\gamma}(A_j,t)}$$

求出。其中 α,β,γ 分别是 $\tau(A_i,t)$、$\mu(A_i,t)$ 和 $tra(A_i,t)$ 在概率 $\zeta(A_i,t)$ 中权重的参数。

代理服务队列排序算法伪代码如下：

```
AGENT-SORT(AQ)
{
    for(i=1；i<=s;i++)
        Ai.ζ(Ai,t)=Ai.τ^α(Ai,t)*Ai.μ^β(Ai,t)*Ai.tra^γ(Ai,t) / Σ_{j=1}^{s} Aj.τ^α(Aj,t)*Aj.μ^β(Aj,t)*Aj.tra^γ(Aj,t);
    按照代理服务优先级进行降序排列;
}
```

(3)优先级动态变更调度算法

优先级动态变更调度算法将优先级最高的任务分配给优先级最高的代理服务，同时将代理服务的优先级设置为最低，从而使得这个代理服务专心地完成任务。当任务完成之后，玩家将根据代理服务服务的质量对该代理服务进行评价。最后代理服务的优先级根据评价值以及原来的优先级进行更

新。优先级动态变更调度算法伪代码如下：

```
Schedule(AQ, Q)
{   i = 1;
    while(调度队列Q不为空)
    {从调度队列Q中获取玩家任务请求 REQ_i;
        j = 1;
        while(代理服务队列AQ不为空 & &j ≤ s)
     { 从代理服务队列AQ中获取代理服务A_j;
          if(μ(A_j, t) < μ̄)
             {λ^0 = λ(A_j, t);
              将代理服务A_j的优先级ζ(A_j, t)设置;
并且将A_j放到队列的最末端;
          将代理服务A_j分配给REQ_i;
              代理服务A_j处理REQ_i;
          从调度队列Q中将玩家任务请求 REQ_i删除;
          代理服务A_j获得此次服务的信任评价ω;
              λ(A_j, t) = λ^0;
              τ(A_j, t) = λ(A_j, t) / Σ_{m=1}^{s} λ(A_m, t);
              τ(A_j, t+1) = χτ(A_j, t) + ω;
          重新计算代理服务A_j的优先级;
          将代理服务队列AQ重新排序;
              break;}
        else j = j + 1;}}}
```

该算法根据 MMOG 中代理服务信任度值、代理服务所在玩家机器的 CPU 使用率、玩家机器的网络流量和任务在调度队列中等待时间动态决定任务的调度，符合 MMOG 运行的实际情况。

4 实验与分析

本文通过应用移动代理的 P2P MMOG 的中间件系统开发了一个简单的 P2P 游戏，玩家用一个简单的球表示，魔法物体用简单的圆锥或者八面体表示。玩家通过吃掉魔法物体来改变自己的形态和颜色。P2P 环境采用 JXTA 来实现，代理平台采用 IBM Aglets V1.0。实验采用校园网中的 200 个计算机资源，分别在这些计算机上部署代理服务。实验对延迟时间、任务在截止时间之前完成百分比和某个区域内 11 个玩家的 CPU 使用率方差在优先级动态变更调度算法和 EDF 两种算法作用下进行了比较。

图 2 的实验结果表明，在优先级变更调度算法作用下，代理服务能够将玩家的任务请求快速分配给区域中的代理服务，使得玩家的请求被快速解决，从而降低了系统延迟。因此，本实验从系统延迟方面证明优先级变更调度算法更适合 P2P MMOG。

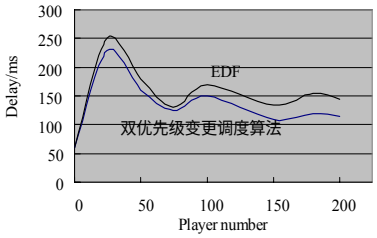

图 2 延迟比较

图 3 的实验结果表明，系统资源使用率在 0.5 以下时，2 个算法在满足任务截止时间百分比的比较结果相差不大。这个结果是由于系统中玩家的数量很少，玩家的任务请求全部可以被区域内的代理服务快速地处理。但是当系统资源使用率达到 0.55 以上时，实验结果相差很大。这主要是因为优先级变更调度算法在系统负载比较重的情况下能更加快速地调度玩家的任务请求。因此，本实验从截止时间满足率方面证明了先级变更调度算法更适合 P2P MMOG。

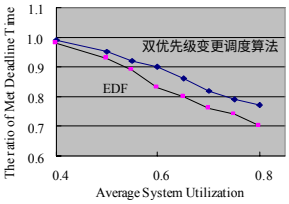

图 3 任务在截止时间之前完成百分比比较

表 1 记录了某区域内 11 个玩家的 CPU 使用率在 2 种调度算法作用下第 12 min 时刻的方差比较。方差越小，越能够符合负载均衡的需求。该实验结果表明优先级变更调度算法对应的方差值最小。因此，本实验从负载均衡方面证明优先级变更调度算法更能满足 MMOG 的需求。综上 3 个实验结果得出优先级动态变更调度策略更适合 P2P MMOG。

表 1 CPU 使用率方差比较

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	方差
优先级动态变更调度算法	0.60	0.65	0.58	0.72	0.81	0.68	0.73	0.76	0.70	0.74	0.73	0.31
EDF	0.55	0.67	0.85	0.75	0.76	0.70	0.80	0.65	0.60	0.58	0.81	0.39

5 结束语

本文提出了一种适合 P2P MMOG 的任务调度策略，该策略综合考虑了 P2P MMOG 中的代理服务信任度值、代理服务所在玩家机器的 CPU 使用率、玩家机器的网络流量和任务在调度队列中等待时间 4 个因素。实验证明，该任务调度策略满足 MMOG 的实时性需求，同时保证了系统的负载均衡。今后将对代理的内部处理机制进行更深入的研究。

参考文献

[1] Ranjan S, Gupta A, Basu A. Adaptive Mobile Agents: Modeling and a Case Study[C]//Proc. of Workshop on Distributed Computing. Calcutta, India: [s. n.], 2000-12.

[2] 王永炎, 王　强, 王宏安, 等. 基于优先级表的实时调度算法及其实现[J]. 软件学报, 2004, 15(3): 360-370.

[3] Kopuri S, Mansouri N. Enhancing Scheduling Solutions Through Ant Colony Optimization[C]//Proc. of International Symposium on Circuits and Systems. [S. l.]: IEEE Press, 2004.

[4] 石祥滨, 周兴海, 高　鹏, 等. 一种基于事件关联的 Timewarp 算法[J]. 小型微型计算机系统, 2006, 27(11): 2067-2071.

[5] 石祥滨, 孟令强, 佟艳阳, 等. 一种基于 N-Tree 的可预测兴趣管理机制[J]. 小型微型计算机系统, 2006, 27(12): 2206-2211.

[6] Faerber J. Network Game Traffic Modelling[C]//Proceedings of the 1st ACM Workshop on Network and System Support for Games. [S. l.]: ACM Press, 2002-04.