第44卷 第2期 西 安 交 通 大 学 学 报 Vol. 44 №2
2010年2月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Feb. 2010

一种适于主-从模式网络计算的事件驱动架构

韩彪[1]，吴众欣[2]，栾钟治[1]，王永剑[1]
（1.北京航空航天大学中德联合软件研究所，100191，北京；2.西安交通大学电子与信息工程学院，710049，西安）

摘要：为了满足大规模网络计算系统在高并发、动态资源管理、稳定性等方面的需求，提出了一种适于主-从模式网络计算的事件驱动架构.它综合了多线程任务处理和事件驱动任务处理的优点，基于排队论推导出了针对主-从模式网络计算的线程资源管理方法，通过引入网络队列将分阶段事件驱动架构的应用范围由单机环境扩展到广域网环境，利用延时队列改善了系统的响应性和可靠性，优先级队列的使用有效地支持了各种作业调度机制.此外，系统还具备了模块化构建和快速开发的特征.实验和药物发现网格应用的实践表明，应用该架构可使系统1 000个作业的平均提交时间由1 850 s缩短为1 350 s，作业的平均处理时间由1 910 s缩短为1 420 s，系统资源得到了更合理的利用.

关键词：主-从模式；分阶段事件驱动架构；动态资源管理
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2010)02-0039-05

An Event Driven Architecture for Master-Worker Network Computing

HAN Biao[1]，WU Zhongxin[2]，LUAN Zhongzhi[1]，WANG Yongjian[1]
（1. Sino-German Joint Software Institute，Beihang University，Beijing 100191，China；2. School of Electronics and Information Engineering，Xi′an Jiaotong University，Xi′an 710049，China）

Abstract：An event-driven architecture for master-worker network computing is proposed to meet the needs in large-scale network computing systems such as high-concurrency，dynamic resource control and stability. The architecture combines the advantages of multi-threaded mode and event-driven mode，and a new strategy of thread management that is suited for master-worker network computing is derived based on queuing theory. The introduction of network queue extends the application domain of staged event-driven architecture from single host to wide area network environment，and the use of delay queue improves system response and reliability. The priority queue is designed for effectively supporting a variety of job scheduling mechanisms. Moreover，systems have the characteristics of modular construction and rapid development. Experiment results and the application of drug discovery grid with systems based on the proposed architecture show that the average job submission time is reduced from 1 850 s to 1 350 s and the average job processing time is reduced from 1 910 s to 1 420 s for 1 000 jobs，and that system resources have more rational use.

Keywords：master-worker paradigm；staged event driven architecture；dynamic resource control

网络计算作为一种能够有效整合跨广域网分散的计算资源，可提供庞大的聚合计算能力，受到了学术界和工业界越来越广泛的重视.网络计算系统是网络计算的基础设施，它利用广域网络上的各种资源，为用户提供有价值的服务.主-从模式是构建网络计算系统的一种常用手段，它基于分而治之的原

收稿日期：2009-05-06. 作者简介：韩彪(1984—)，男，硕士生；栾钟治(联系人)，男，副教授，硕士生导师. 基金项目：国家高技术研究发展计划重大专项资助项目(2006AA01A118，2006AA01A124)；国家高技术研究发展计划资助项目(2009AA01Z144)；科技部国际合作重点资助项目(2006DFA11080)；欧盟FP6项目BRIDGE资助项目(045609).

则来满足大规模计算需求，较好地解决了网络计算环境下，由资源的分布性、动态性和异构性带来的整合困难，充分发挥了网络计算的潜力[1].

目前，主-从计算模式网络计算系统的相关研究，主要集中于如何通过改进调度策略来提高系统性能和资源的利用率[2]，在如何利用动态、自适应的调节方法[3-4]来提供高性能计算能力方面缺乏系统架构层面的支撑，同时较少关注如何改善系统的高并发性、可控性和可靠性支持. 本文以分阶段事件驱动架构为基础，提出了一种适于主-从模式网络计算系统构建的事件驱动架构(MEDA). 该架构将事件驱动用于主-从模式网络计算系统的构建，较好地满足了系统在并发性、可控性和可靠性等方面的需求.

1 分阶段事件驱动架构

在传统的基于多线程的请求处理模式基础上，分阶段事件驱动架构(SEDA)[5]吸收了事件驱动型并发处理模式的优势，具备了支持高并发、自适应调节和模块化构建的新特征. 它将用户请求以事件的方式进行封装，处理流程被切分成若干个阶段，阶段之间通过事件队列互相连接.

每个阶段的处理部件包括事件队列、线程池、控制器以及用来封装应用逻辑的事件业务处理部件(IEventHandler). 控制器负责监测事件队列中的事件，利用从线程池中获取的工作线程来执行 IEventHandler 封装的应用逻辑. 完成本阶段处理后，事件将被放入下一阶段的事件队列中等待处理. IEventHandler 的引入使得应用开发人员能够专注于开发应用逻辑. 控制器通过监测事件队列的变化来动态调整线程池中工作线程的数量，预设阈值可避免一个阶段获取过多的资源而导致系统过载，同时这种模块化的构建方式也使得系统故障的定位和处理变得更加容易. SEDA 阶段结构示意图如图 1 所示.

目前，分阶段事件驱动架构已广泛用于高负载、高并发系统. 然而，面对分布、动态、异构的网络计算环境，分阶段事件驱动架构无法完全满足系统在功能和性能上的需求.

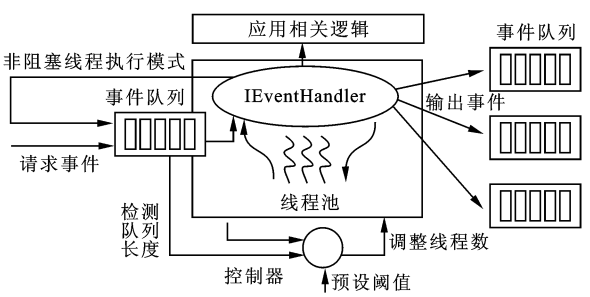

图 1 SEDA 阶段结构图

2 主-从模式下的事件驱动架构

以分阶段事件驱动架构为基础，本文提出了适于构建主-从模式网络计算系统的事件驱动架构 MEDA. 主节点包含初始化、切分、分发、合并、完成 5 个阶段，与之相对应的每个计算节点由提交、查询和完成 3 个阶段构成. 阶段之间通过队列连接，整个系统构成一个阶段网络，从而实现了全系统层面的资源自适应控制和模块化构建，其功能模型如图 2 所示.

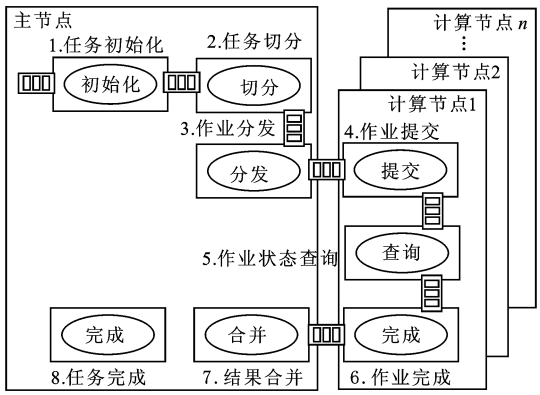

图 2 基于 MEDA 的主-从模式网络计算系统功能模型

为了适应网络计算环境分布、异构、动态的特点，MEDA 关注的问题主要集中于以下 3 个方面：①如何将分阶段事件驱动机制的应用环境由单机范围扩展到广域网范围；②如何根据主-从模式网络计算的应用特点配置合理的资源管理策略，以进一步提高性能；③如何有效支持主-从模式网络计算系统中的各种调度机制.

2.1 网络队列提供通信保障

传统的分阶段事件驱动应用只局限于单机范围，不同阶段之间通过简单的队列连接，这种连接不存在网络通信代价. 但是，在网络计算环境下，通信却是网络计算系统能否安全可靠地进行资源动态整合和任务分布协同的关键. MEDA 以网络队列的形式解决了位于不同主机上的阶段之间的通信问题.

网络队列提供同普通队列一样的操作接口，支持入队操作 put 和出队操作 take. 对于网络通信来讲，可靠传输是最为关键的技术之一. 网络队列的可靠传输模型如图 3 所示，在简单的入队、出队操作背

后，MEDA 对底层通讯协议进行了进一步封装，利用内部队列支持通信消息的缓存和持久化，由发送模块和接收模块协作控制通信过程中消息的确认、重发、重排等操作，提供可靠的传输保障.

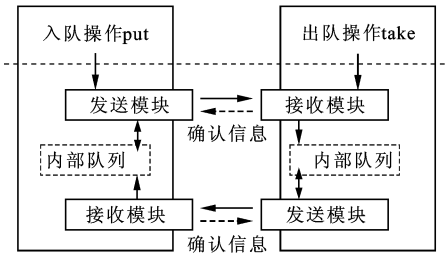

图 3　网络队列的可靠传输模型

网络队列支持同步非阻塞和异步消息两种通信方式，它们之间优势互补，保证了网络队列的可靠性和灵活性，从而实现将分阶段事件驱动机制由单机环境扩展到广域网环境下的目标.

2.2　网络计算环境下的资源管理

对于主-从模式网络计算系统的资源管理，MEDA 主要关注作业提交阶段的线程池阈值配置和作业状态查询阶段的资源消耗与响应性能的均衡.

2.2.1　*作业提交阶段的线程池阈值配置*　节点的计算能力取决于节点的 CPU 数和单个 CPU 的处理能力. 利用排队论理论，作业提交阶段可以建模为一个包含 N 个终端、M 个 CPU 的分时系统（其中 $N \geqslant M$，终端即为线程池中的工作线程）. 作业提交过程可以抽象为：某个用户从一个终端登陆到系统，在经过一个平均长度为 R 的初始化阶段后，提交一个平均耗时为 P 的任务. 根据某个不确定的优先级和分时系统的规则，该任务进入计算机的任务队列中并等待 M 个 CPU 中的一个来处理. 接下来，可以应用 Little 定理计算系统吞吐量的范围. 假设系统始终处于满负荷状态，以保证系统中始终有 N 个任务. 分时系统模型如图 4 所示.

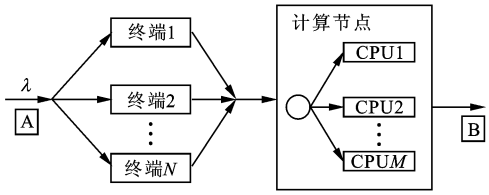

图 4　分时系统模型

对系统入口 A 点和系统出口 B 点之间的部分应用 Little 定理，可以得到系统吞吐率

$$\lambda = \frac{N}{T} \tag{1}$$

式中：T 是任务从进入系统到最终处理完毕所耗费的平均时间，简称任务的平均延迟时间

$$T = R + D \tag{2}$$

其中 D 指从任务提交到任务完成的平均延迟时间. 由于 D 可以在 P 到 NP/M 之间取值，这样由等式(2)可以得到

$$R + P \leqslant T \leqslant R + (N/M)P \tag{3}$$

由关系式(1)和式(3)，可以得到

$$\frac{N}{R + (N/M)P} \leqslant \lambda \leqslant \frac{N}{R + P} \tag{4}$$

同样，吞吐量 λ 还受到计算节点处理能力的限制. 每个 CPU 执行一个任务平均需要 P 个单位时间，单位时间内节点不能处理多于 M/P 个任务，即

$$\lambda \leqslant \frac{M}{P} \tag{5}$$

综合关系式(4)和式(5)，可以得到吞吐量的取值范围

$$\frac{N}{R + (N/M)P} \leqslant \lambda \leqslant \min\left\{\frac{M}{P}, \frac{N}{R + P}\right\}$$

通过应用 $T = N/\lambda$，得到当系统始终处于繁忙状态情况下，任务的平均延迟时间 T 的范围

$$\max\left\{\frac{N}{M}P, R + P\right\} \leqslant T \leqslant R + \frac{N}{M}P$$

通过以上分析可以看出，随着系统中终端数量 N 的增加，系统吞吐量将趋向于最大值 M/P，而任务的平均延迟时间 T 与 N 成正比，所以 N 会随着延迟的增加而增加. 当 $N \leqslant M + RM/P$ 时，系统终端的数量成为限制吞吐量的瓶颈，此时节点 CPU 资源会出现闲置现象. 相反，当 $N > M + RM/P$ 时，节点有限的处理能力则成为瓶颈. 由此可见，作业提交阶段最大线程数的合理值应该是 $\lfloor M + RM/P \rfloor$，设置超过此值的线程数只会白白浪费系统线程资源，因为此时节点的处理能力已经成为系统的瓶颈，特别是当 $R \ll P$ 时，设置最大线程数为节点的 CPU 数即可.

2.2.2　*作业状态查询中的资源消耗与响应性能*　由于作业处理时间受诸多不确定因素的影响，因此要想尽可能早地获取作业完成信息只有不断查询作业状态. 然而，频繁地查询状态会导致过多的并发请求，影响系统的稳定性. 针对这一问题，MEDA 中提出了利用延时队列实现待处理事件休眠的方法来确定状态查询的频率. 其中，待处理事件即为被封装成队列元素的作业状态查询请求. 每个事件进入延时队列前需要预先指定一个延迟时间，延迟时间满的事件将从队列中取出并执行对应的查询请求.

在延迟时间的设置上，MEDA 采用自适应折半查询算法，前提是：越接近作业的平均执行时间，作业完成的概率越大，那么相应的查询频率也应该加大. 算法要求：每个作业第一次进入延时队列时，其延迟时间要设为平均执行时间的一半，以后每次查询结束后，如需再次查询，在事件重新入队前，新的延时时间都要在原值的基础上减半，除非原值已达到最小延迟时间. 算法采用加权平均的方式确定平均执行时间. 初始时，平均执行时间采用经验值，在处理过程中，不断利用最新的作业执行时间数据和原平均执行时间加权平均来修正平均执行时间，以使此估计值尽量接近真实值. 采用自适应折半查询有效地减少了作业初始执行阶段的低命中率查询，提高了查询的整体命中率. 其中，设置最小延迟时间的目的是为了限制系统并发量，以降低负载，保证系统的稳定性.

2.3 基于优先级队列的作业调度

任务调度系统是网络计算系统的重要组成部分. 大部分任务调度机制都有设置优先级的需求，比如文献[6]提出的事件驱动的完全优先级调度算法和非完全优先级调度算法. 为了支持系统调度，MEDA 实现了优先级队列机制. 通过动态修改优先级，可以支持更加灵活的调度策略.

此外，本质上优先级队列提供了一种作业分类机制，而同种类型的作业并发执行能够极大限度地发挥程序执行过程中时间和空间的局部性优势，提高整体的执行效率，所以优先级队列不仅仅为调度机制提供使能性支持，而且具备通过增加同类作业并发执行的概率来提高系统效率的能力.

3 实验与应用

实验测试环境由若干台服务器节点构成，根据具体实验项目的不同可能分布于同一局域网或广域网. 它们的硬件配置相同：4 个 2.4 GHz 的 CPU，2 GB 内存，80 GB SCSI 硬盘，局域网带宽为 1 GB/s. 各节点的软件环境为：基于 2.6 内核的 Linux 操作系统，基于 Sun JDK1.5 开发的 Java 程序.

3.1 网络队列测试

实验对比了网络队列分别在同步非阻塞、异步消息通信方式下，支持的连接数和吞吐率情况. 每次通讯中传输的有效数据量为 5 kB，实验结果如图 5 所示. 需要说明的是，当连接数达到 1 200 时，异步通信模式已无法进行工作，所以后续数据缺失. 总体上同步通信模式由于采用了非阻塞机制，所以其对并发量的支持要优于异步通信模式，不过异步通信模式也基本满足了通常情况下网络计算的通信需求.

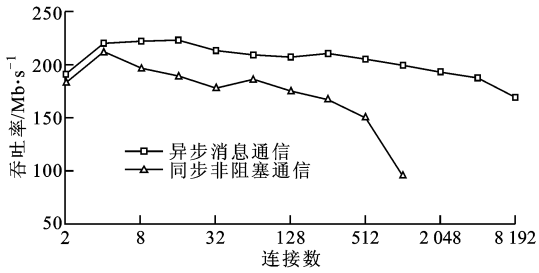

图 5 同步非阻塞通信与异步消息通信的对比

3.2 延迟队列测试

实验对比了理论上具有最低延迟效果的无休眠多线程循环查询、普遍使用的查询线程定时休眠和采用延时队列的待处理事件休眠 3 种作业状态查询方法. 作业的处理时间在[85 s，90 s]区间内随机分布，以 1 s^{-1} 的速率发送，总共向处理单元派发 500 个作业. 实验结果如图 6 所示.

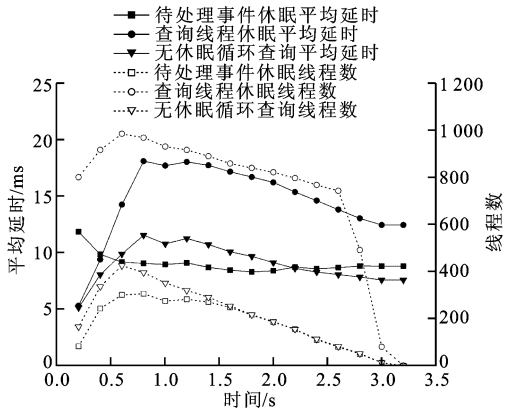

图 6 待处理事件休眠查询效果评测

由图 6 可以看出，普通的查询线程定时休眠方法所需要的线程数要远远高于另外 2 种方法，而且平均延迟时间始终最长. 尽管待处理事件休眠和无休眠查询的线程数和平均延时比较接近，但无休眠查询的平均查询次数要远远高于待处理事件休眠方式，所以待处理事件休眠方法是最佳的状态查询方法，它在资源占用同系统响应性之间达到的平衡性是最优的.

3.3 优先级队列测试

实验中，作业依类别被分为 A、B、C 3 个等级，每类作业以 1∶1∶1 的比例顺序派发，发送速率分别为 1 ms^{-1}，总共向处理单元派发 500 个作业. 在

处理过程中 A、B、C 3 种作业分别要从不同的磁盘文件中读取数据，文件大小均为 512 kB. 处理单元的最大线程数为 10. 实验对比了普通先进先出队列和优先级队列的作业处理总耗时和作业的平均耗时两项指标，结果如图 7 所示.

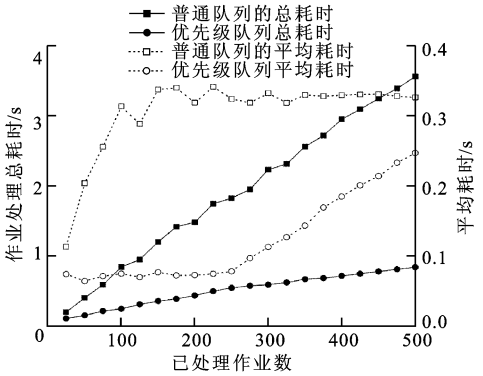

图 7　普通队列与优先级队列的效能对比

可以看出，采用优先级队列的作业处理总耗时为 0.85 s 左右，远远低于普通队列的 3.5 s；就作业的平均耗费时间而言，优先级队列的 0.25 s 也要低于普通队列的 0.32 s. 实验证明了优先级队列具有提高系统处理效率的能力.

3.4　MEDA 在药物发现网格中的应用

目前，MEDA 已经用于药物发现网格[7]（Drug Discovery Grid，DDGrid）的系统改进. 与原有的基于多线程的处理模式相比，DDGrid 的作业处理性能有了明显提高. 如图 8 所示，应用此架构使得 1 000个作业的平均提交时间由 1 850 s 缩短为 1 350 s，作业的平均处理时间由 1 910 s 缩短为 1 420 s. 改进后的 DDGrid 网格无论是作业提交效率还是作业处理效率都有了显著提高，资源得到更加合理的利用，系统的稳定性和可控性也得到了很大改善.

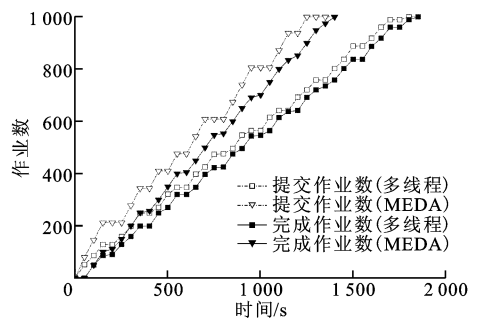

图 8　DDGrid 的作业提交和作业完成情况统计

4　结　论

事件驱动的体系结构满足目前动态多变的大规模网络计算环境的需求，有着广阔的应用前景. 本文提出的 MEDA 作为一种用于主-从模式网络计算系统构建的事件驱动架构，适应了网络计算环境分布性、异构性、动态性的特点，为动态、自适应地调节资源，改善系统性能提供了系统架构层面的支撑，增强了系统的并发性、可控性和可靠性.

参考文献：

[1] DUBREUIL M, GAGNE C, PARIZEAU M. Analysis of a master-slave architecture for distributed evolutionary computations [J]. IEEE Transactions on Systems, Man and Cybernetics, part B: Cybernetics, 2006, 36(1): 229-235.

[2] HEYMANN E, SENAR M A, LUQUE E, et al. Efficient resource management applied to master-worker applications [J]. Journal of Parallel and Distributed Computing, 2004, 64(6): 767-773.

[3] CESAR E, MORENO A, SORRIBES J, et al. Modeling master/worker applications for automatic performance tuning [J]. Parallel Computing, 2006, 32(7/8): 568-589.

[4] MORAJKO A, MARGALEF T, LUQUE E. Design and implementation of a dynamic tuning environment [J]. Journal of Parallel and Distributed Computing, 2007, 67(4): 474-490.

[5] WELSH M, CULLER D, BREWER E. SEDA: architecture for well-connected scalable internet services [J]. Eighteenth Symposium on Operating Systems Principles, 2001, 35(5): 230-243.

[6] 刘洋，桂小林，徐玉文. 网格工作流中基于优先级的调度方法研究 [J]. 西安交通大学学报，2006，40(4)：411-414.
LIU Yang, GUI Xiaolin, XU Yuwen. Study on scheduling methods based on priorities in grid workflow [J]. Journal of Xi′an Jiaotong University, 2006, 40(4): 411-414.

[7] WANG Yongjian, LUAN Zhongzhi, QIAN Depei, et al. DDGrid: a grid computing environment with massive concurrency and fault-tolerance support [C]// Proceedings of 7th International Conference on Grid and Cooperative Computing. Piscataway, NJ, USA: IEEE Computer Society, 2008: 5-14.

（编辑　刘杨　苗凌）