基于 OpenCL 的网络演算数值计算库的改进算法

李宝亮

***Abstract*—网络演算已经取得了巨大的成功，但是以往针对网络演算的研究主要集中于对实际系统的分析和基于网络演算的建模，但是很少有研究现解决网络演算模型的快速计算问题。网络演算的数据计算库是网络演算应用的基本前提。尽管目前已经有一些面向分段线性伪周期函数的的网络演算数值计算库，但是这些库在处理阶梯函数时效率较低，特别是在进行卷积和闭包计算时。本文，我们提出一种基于区别覆盖的阶梯函数快速卷积计算方法，通过与 Java 计算库 DISCO 的比较，我们发现我们的方法在分析处理阶梯函数时效率大幅度提高。通过对该算法的分析，我们发现该算法的具有较高的可并行性，为了进一步提高计算性能，我们还基于 OpenCL 对这一算法进行了并行加速。实验结果表明，我们的计算方法较相应的串行方法性能提高 10 倍左右，较之前的基于 java 的算法相比，性能提高了近 xxx 倍。网络演算中卷积运行的快速计算，使得网络演算的应用更容易。**

***Keywords*—数值计算、并行算法、网络演算、延迟和积压界**

I. Introduction

网络演算的基本概念和应用情况。

网络演算的数值计算是网络演算面向实用的关键，没有一个高效的计算库，那么网络演算的性能分析则只能停留在理论层面，无法用于指导实际的网络分析和设计。网络演算是基于极小加代数得出的一种性能分析工具，由于其基于特殊的代数结构，而且卷积的计算和分析非常困难，因而使得其实际的分析和计算非常困难。目前很多的基于网络演算的研究都只考虑了采用网络演算来建模，但是在计算结果时则只能假设非常简单的到达和服务曲线模型，根本原因是网络演算的数据计算是非常复杂和困难的，即使是对非常简单的网络拓扑和模型，计算其端到端性能是也会涉及非常复杂的公式和推导，这使得计算结果变得非常困难。为了简化计算，在计算过程中通过会做一些简单，假设用一个较宽松但是非常简单的到达曲线（例如 RB）和服务曲线（LR）来代替实际的网络服务曲线。又比较利用连续的到达曲线模型和服务曲线模型来简化实际的离散系统。这样做的直接后果是，所得到的结果较实际性能有更大的差异，原因是当采用近似的方法来简单系统时，会牺牲精度。另一方面，实际的通信系统（如计算机网络、ATM 网络等）都是离散时间离散值的系统，因而采用离散模型来建模也更加直接，而且所得的结论也更好。另一方面，很多系统的分析过程中，必须采用离散的网络演算模型，例如基于实时网络演算的 Flexray 性能分析模型 [1], [2]，对于这类模型，如果采用连续服务模型会导致错误的结论；另外，还有一些情况，当采用离散时间模型时，也会非常方便和直观，一个典型的例子是基于实时网络演算的非抢占系统的性能分析 [3]。

但是，我们发现，当采用实时网络演算的工具箱来分析离散时间的到达和服务曲线模型时，是非常耗时的。以我们的论文中的例子，计算一组数据需要几十分钟的时间。通过对这类网络演算数据计算工具箱数值算法的分析我们发现，这些工具箱都基于相同的算法 [4]，这类算法面向分段线性伪周期函数类型的到达和服务曲线。所谓分段线性伪周期函数是指:。这类函数基本涵盖了网络演算理论中涉及的所有的模型。该算法假设所有的模型都是由无穷多的直线段组成的，直线段之间首尾相连，但是模型可以是不连续的，即可以存在瑕疵点。而分段线性伪周期函数函数都是由一些起始的非周期部分和后续的周期部分组成的，每一部分都是由若干个线段和瑕疵点组成。当整个函数中包含的线段数量（非周期部分 + 周期部分中一个周期内的线段数量）较少时，已有的算法便可以快速的给出结果，但是，当线段较多时，整个计算过程便会非常耗时。由于实际的系统都是离散的，因此，我们考虑一类非常普遍存在的情况：周期部分和非周期部分都很长，但是，每一部分中的线段数量也可以很多，但是，这些线段的斜率都为 0，即整个的函数是阶梯函数。对于这类函数，本文，我们提出一类高效的计算方法，该方法的计算效率是原算法实现的数十倍。事实上，自然界的大部分模型都可以规约到这一类模型，因为通信系统都是离散时间系统。因而，我们的算法对网络演算的应用是非常重要的。

本文余下部分的安排如下：第二节介绍片上实时通信的设计方案以及性能分析的相关研究现状；第三章给出我们的研究假设、虫孔交换的片上网络模型以及实时网络演算的基本理论进行简要的介绍；第四章给出建模过程和分析实例；第五章开展实验验证本文提出的实时演算模型的正确性并与其它理论方法进行比较；第六章对全文进行总结。

II. Related Work

1. 网络演算的应用 2. 网络演算应用的重要步骤是数据计算工具箱，因为网络演算的模型是符号形式，如何推导计算结果是非常关键的一步。如果将网络演算应用于设计空间搜索的过程中，需要有一个数据计算库将网络演算模型在不同参数下的性能结果做为设计参数。3. 现在已经有的嵌入 SCILab 的 min-max 计算工具箱，该工具箱不支持某些操作，而且不是专门面向网络演算的工具箱，而是面向 min-max 的。NC-madue[?] 也是一个面向网络演算的工具箱。DISCO[?], [?] 可能是最好用的数值计算工具箱，该工具箱甚至还推出了面向随机网络演算的工具箱。另外还有 COINIC 等。另外，还有面向实时网络演算的 RTC Toolbox[?]。但是这些工具箱都基于同样的数值计算算法，该算法于 [4][?] 中提出。关于对这些工具箱的性能比较可以参考 [?]。但是，这些工具箱在计算过程中，速度比较慢。因此，我们考虑对其进行优化。4. 尽管采用简化的服务曲线和到达曲线模型可以加速和简化数值计算（如 RB 到达曲线和 RL 服务曲线），但是这样的简化会进一步 loosen 性能界。为了使得计算结果紧致，应尽可能的让到达曲线和服务曲线都尽可能的紧。对于到达曲线和服务曲线的因果性的研究可以参考。本文中我们不研究到达曲线和服务曲线的生成方法，只研究几个重要的网络演算算符的 GPU 加速实现方法。5. 我们采用的方法是 OpenCL，Opencl 是。。。6. 考虑能否将 multi-mode rtc 的研究成果集成到加速库中来。因为它是状态机，状态机在 GPU 上实现是非常高效的。

It has been reported that 70% of the time is spent in the validation and verification phase of the design cycle [1] where verification by simulation is used.

K.-D. Schubert, “Improvements in functional simulation addressing challenges in large, distributed industry projects,” in Proceedings of the 40th annual Design Automation Conference, ser. DAC ’ 03. New York, NY, USA: ACM, 2003, pp. 11–14.

网络演算的最成功应用是对空客 A380 的延迟分析，原因是它可以处理大规模系统的性能分析。而传统调度论是无法胜任的。但是，对于大规模系统的网络演算分析，完全采用手工计算是非常困难的，原因是网络演算的卷积、反卷积和闭包运算是非常困难的。而且为了实现网络演算结论的自动化应用，也迫切需要强大的数学计算库的支持。

关于 NC 计算工具箱的更多研究请参考 NC-Maude: A Rewriting Tool to Play with Network Calculus.

列出目前已经有的 NC 计算工具箱，并比较其各自的优缺点。最后指出，没有一个是面向高性能计算应用的。都无法满足大规模系统的快速计算。而没有一个快速的计算工具箱，那么网络演算的优势也就无从谈起。

III. 改进算法

A. 基础数学概念

B. 原有算法简介

C. 我的改进方法

IV. 基于 OpenCL 的并行算法

A. 开发算法中的并行性

B. 实现基于 OpenCl 的并行算法

1.OpenCL 的简介 2. 基于 OpenCL 实现的核函数

V. 实验验证

1. 实验平台 2. 实验数据 3. 实验结果

VI. 总结

网络演算面向实用的重要一步就是数学模型的快速计算。

References

[1] Devesh B. Chokshi and Purandar Bhaduri. Performance analysis of flexray-based systems using real-time calculus, revisited. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC ’10, pages 351–356, New York, NY, USA, 2010. ACM.

[2] Andrei Hagiescu, Unmesh D. Bordoloi, Samarjit Chakraborty, Prahladavaradan Sampath, P. Vignesh V. Ganesan, and S. Ramesh. Performance analysis of flexray-based ecu networks. In *Proceedings of the 44th Annual Design Automation Conference*, DAC ’07, pages 284–289, New York, NY, USA, 2007. ACM.

[3] D.B. Chokshi and P. Bhaduri. Modeling fixed priority non-preemptive scheduling with real-time calculus. In *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA ’08. 14th IEEE International Conference on*, pages 387–392, 2008.

[4] Anne Bouillard and Éric Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18:3–49, March 2008.