第46卷 第2期 西 安 交 通 大 学 学 报 Vol. 46 No. 2
2012年2月 JOURNAL OF XI'AN JIAOTONG UNIVERSITY Feb. 2012

并行片上网络仿真器 ParaNSim 的设计及性能分析

唐轶轩[1]，吴俊敏[1,2]，陈国良[1]，朱小东[1]，胡蝶[1]
(1. 中国科学技术大学计算机科学与技术学院，230027，合肥；
2. 中国科学技术大学苏州研究院，215123，江苏苏州)

摘要：为了减少使用仿真器对片上网络的性能、结构等进行仿真的时间，提高仿真效率，利用当代计算机的并行计算能力，设计并实现了一个并行片上网络仿真器 ParaNSim. 该仿真器可配置拓扑、路由算法以及虚通道等参数，既可以作为独立的仿真器使用，也可以作为一个子模块嵌入其他仿真器(如 Multi2Sim)中；经过实验验证，其并行仿真能达到的加速比平均约为 210%，最大加速比可达 250%，因此它能有效地减少仿真时间，为大规模片上网络的仿真提供支持.

关键词：仿真器；并行仿真；片上网络
中图分类号：TP302 **文献标志码**：A **文章编号**：0253-987X(2012)02-0024-07

ParaNSim: a Parallel Network-on-Chip Simulator

TANG Yixuan[1], WU Junmin[1,2], CHEN Guoliang[1], ZHU Xiaodong[1], HU Die[1]
(1. School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China;
2. Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou, Jiangsu 215123, China)

Abstract: It is a very popular approach to use simulators to evaluate the performance and cost of different network-on-chips (NOCs) for determining the best network designs and configurations. Most of the traditional simulators are of single thread, which is a computational bottleneck of these simulators because single thread renders them cannot take advantages of new chip multiprocessors. A parallel NOC simulator, ParaNSim, is designed and implemented in this paper. The simulator supports large-scale NOC simulation, and can effectively reduce the simulation time for large-scale NOC simulation. Experimental result shows that the speedup of parallel simulation can reach 210% on average, and 250% the maximum.

Keywords: simulator; parallel simulation; network-on-chip

半导体技术和集成电路技术的高速发展使得大规模片上系统(system-on-chip, SoC)得以实现，微处理器、存储器、I/O 系统等都可以大量地集成到单个芯片中，因此多核处理器系统的出现成为必然. 多核处理器对片上通信提出了更高的要求，然而传统总线结构的扩展性、通讯效率以及功耗和面积均不能满足这些要求，使片上通信成为片上多处理器系统的瓶颈，因此片上网络(network on chip, NoC)便应运而生. 片上网络的基本特征是模块化设计片上互连结构，各个模块之间通过片上的微型网络进行并行通信，提高带宽以降低延迟，同时提高可扩放性. 类似于传统的计算机网络，片上网络有拓扑结构、路由器、路由算法等设计因素，但是片上网络又有其特有的设计因素，例如功耗、面积限制及复杂连线等问题. 因此，片上网络的设计是一种关于开销、复杂度、性能等设计标准的权衡，设计目标是在尽可能不增加开销和复杂度的前提下，提高通信性能.

在设计片上网络时，构建一个具备分析处理器

收稿日期：2011-05-16. 作者简介：唐轶轩(1984—)，男，博士生；吴俊敏(通信作者)，男，副教授. 基金项目：国家高技术研究发展计划资助项目(2008AA01Z111)；IBM 大学合作联合研究项目(JSA200906010)；中央高校基本科研业务费专项资金资助项目(WK0110000020).
网络出版时间：2011-12-01 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20111201.1640.002.html

核、片上互连、外设等性能参数的仿真平台，对研究片上网络的设计有重要意义. 然而，现有的片上网络仿真器，如 Noxim[1]、Booksim[2]、SICOSYS[3] 等，都是基于单线程的，这使得在仿真大规模片上网络时会消耗大量的时间. 为了提高仿真速度，利用当代计算机的多线程能力开发一个并行片上网络仿真器是十分有必要的.

本文实现了一个可配置拓扑结构、路由算法、虚通道、链路等参数的并行片上网络仿真器 ParaNSim，可以进行多处理器系统中的片上网络仿真和性能分析，能够在硬件实现前评测各种方案的性能，找出设计瓶颈并加以改进，为系统选择最优的片上网络结构、路由算法等提供灵活有效的设计研究平台，并可以进行应用程序的仿真和调试.

1　片上网络仿真器的整体架构

片上网络借鉴了计算机网络中的概念，使用路由器互联单个芯片内的各个 IP 核. 常见的片上网络结构如图 1 所示，图中的每一个资源块都可以放置一个固定的 IP，例如处理器核、存储器或带总线结构的小系统. 资源块通过网络接口（NI）连接路由器，路由器之间通过双向链路连接，资源块及其网络接口构成一个节点.

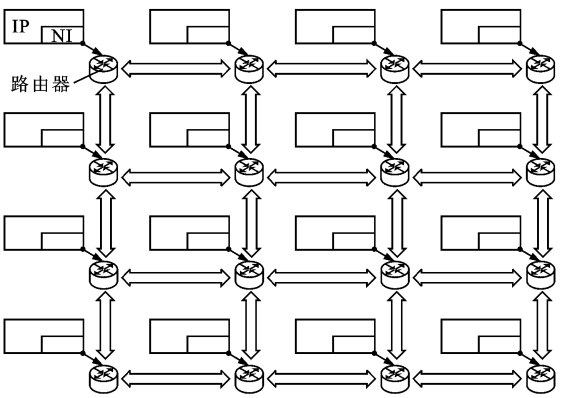


图 1　典型的片上网络结构

对于 NoC 仿真器而言，拓扑结构、路由策略、路由器结构和路由算法决定了 NoC 的基本特性. 路由器是 NoC 中最为核心的组成部分，其基本结构和传统的路由器区别不大，但是由于受到芯片面积和功耗的限制，NoC 路由器在一些设计细节上呈现出独有的特性，如内部的缓冲数目相对较小，端口数目较少，路由器之间的链路较宽，路由器的仲裁逻辑通常较为简单，等等.

ParaNSim 仿真器主要分为 3 个模块：网络接口模块，链路模块，路由功能模块.

网络接口模块主要用来产生符合网络协议的流量（即数据包格式的流量），并提供与系统仿真器的接口. 网络仿真数据的统计功能也在该模块中.

链路模块主要负责数据片的传输，因为 ParaNSim 采用了基于 Credit 的流控机制，所以链路模块还负责 Credit 的传输.

路由功能模块是 ParaNSim 中最复杂、最核心的模块，它负责数据片的路由、虚通道的选择以及负载平衡等功能. ParaNSim 采用了经典的路由模块结构，如图 2 所示.

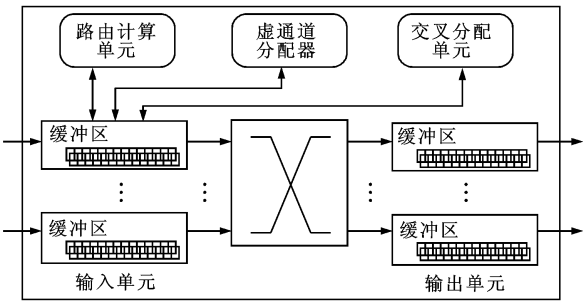


图 2　ParaNSim 路由模块结构

整个路由模块由 6 个功能单元组成：路由计算单元，虚通道分配单元，交叉分配单元，交叉开关，输入单元，输出单元. ParaNSim 默认的结构是二维 Mesh，因此每个路由模块有 5 个输入和输出单元. 仿真器初始化时，接收配置文件信息，对拓扑结构、路由策略、虚通道数量、端口数目及缓冲区深度进行配置. 输入单元由多个虚通道组成，每个虚通道有一组状态参数，其原始状态为空闲 I，当一个数据片传入通道后，状态变为 R，经过路由计算，虚通道状态变为 V，再通过虚通道分配，状态变为 A，每个周期路由单元会在所有状态为 A 的虚通道中选择一个进行发送，这样就完成了一个数据片从输入单元到输出单元的传输. ParaNSim 在单线程状态下每个周期的运行流程图如图 3 所示，每个周期结束后，仿真器会检查网络或者网络接口控制器（NIC）中是否还有数据包，若有则循环此流程，若无则输出统计数据后退出运行.

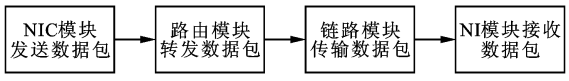


图 3　ParaNSim 单线程运行流程

ParaNSim 实现的片上网络模型具有良好的灵活性和扩展性，片上网络的拓扑结构、路由算法、仲

裁策略和流控策略等均可方便地进行配置,而且由于仿真器的模块化较好,用户也可以较容易地添加其他结构和算法策略.该仿真器的默认配置:拓扑结构为二维 Mesh,路由算法为 XY 路由算法,仲裁策略为 Round Robin 仲裁策略[2],流控策略为基于 Credit 的流控策略[2].

2 仿真器的并行化方案

2.1 并行仿真背景

随着片上资源的增多,片上网络的规模也越来越大,使得大规模片上网络的仿真往往需要运行很长时间.把大的网络模型划分为子模型并行执行,可大大提高网络仿真的运行效率.并行仿真的核心是精确地同步在互联的各个处理器上运行的仿真过程,这是由于仿真在空间上被划分开后,各个模块以自己的本地逻辑时钟前进,为了保证并行仿真与串行仿真的结果能够高度匹配,并行执行的各模块必须服从逻辑时钟上的顺序性及系统状态的依赖性,这样每个模块必须按照一致的时间顺序处理事件[4].同步的方法分为保守同步和乐观同步 2 类.

保守同步基于阻塞,基本思想是每个模块必须先确认某事件是安全的,然后才能处理该事件."安全"是指一个模块不会从其他模块收到具有比本地逻辑时钟更小时戳的事件,也就是说,当一个远程事件来得太晚、本应该在过去发生时,安全性就被违反了.保证安全性的关键是系统内在的时间依赖关系.如果一个远程发送来的事件应该在 x 时刻发生,那么模块的本地逻辑时钟就不能前进到 $x+1$ 时刻,必须在 x 时刻阻塞并等待事件的到来.保守同步的机制有 Null Message[5]、BSP(bulk synchronous parallel)[6]等.保守同步机制中每个模块反复执行一个包含以下步骤的过程:

(1)全局同步,计算时间下限;

(2)处理所有时戳不超过时间下限的事件;

(3)本地时间前进.

乐观同步的提出是为了避免保守同步中的阻塞,基本思想是每个子模块可以"大胆"地向前执行,如果发生错误则用回滚机制纠正.常见的乐观同步机制是 TW(time warp)算法[7].

当发生了预测失败,也就是确实发生了因果错误的情况时,乐观同步提供了一套状态保存与回滚机制来解决这个问题.每个事件发生时,逻辑处理器先保存一个状态的临时副本,然后再处理事件,并在临时副本上更新状态,直到全局时钟超过事件的时间戳、不再会有更早的事件从其他逻辑处理器发送过来时,副本才会被提交合并到持久状态中.如果在提交之前侦测到因果逻辑错误发生,逻辑处理器将撤销所有因果逻辑错误发生时刻之后的状态副本,回滚到一个状态一致的时间点重新开始执行.可以看出,影响乐观同步性能的主要因素有保存的状态量及因果逻辑错误发生的频率.在保存的状态量较少的情况下,预测执行期间不会有过多的额外开销,而当逻辑处理器间的通信事件比较少时,预测能够保持较高的成功率,降低宿主节点间的通信开销,保持更好的负载平衡性,从而提供比保守同步更好的性能.但是,就片上网络的并行仿真来说,由于系统复杂度高,状态量大,而且各个模块之间耦合度高,数据在模块之间传输的逻辑时间差很小,导致状态保存的开销远高于并行度上升带来的收益,使得乐观同步技术并不适宜于该仿真领域.因此,我们采用了保守同步技术.

2.2 ParaNSim 采用的并行技术

并行化的首要问题是要确定可并行运行的资源,然后是为每个子线程分配资源.在片上网络仿真器中,NI 模块、路由模块和链路模块都是相对独立的,如果它们之间存在通信,也只会发生在每个周期的开始或者结束的阶段,即数据片只会在每个周期的开始或者结束的时候在 3 个模块之间移动,因此从理论上来说,可以为每个模块分配一个线程.但是,实际运行时我们发现,路由模块消耗的时间远远大于另外 2 个模块消耗的时间,因为路由模块在每个周期内要经过路由计算、虚通道分配、交叉开关分配等多级流水,而 NI 模块和链路模块仅仅是读写缓冲区,所以假若每个模块分配一个线程,就会造成速度不匹配,即 NI 和链路模块的线程大部分时间处于空闲或者阻塞状态,这样大大降低了系统性能,因此,ParaNSim 采用了主线程运行 NI 和链路模块、子线程运行路由模块的模式.如图 4 所示,主线程负载所有的 NI 和链路模块的仿真,其余的路由模块由子线程仿真,子线程的数量可通过参数设置,系统自动为每个子线程划分任务.

在同步技术方面,ParaNSim 采用的是保守同步策略,子线程和主线程每个周期同步一次,具体的实现方法是采用 pthread 库中的栅栏(barrier)机制,即等待 N 个线程都到达了某一阶段后同时唤醒.N 可以由 pthread_barrier_init 函数设置.被唤醒的所有线程中,有一个 pthread_barrier_wait 函数的返回值为 PTHREAD_BARRIER_SERIAL_

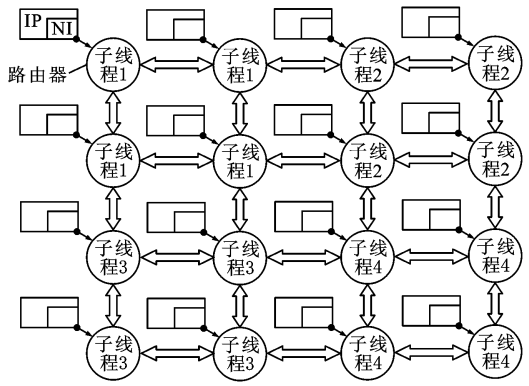


图 4　ParaNSim 线程任务划分方案(以 4 个子线程为例)

THREAD，其他为 0，唤醒后 barrier 对象会被重置到最近一次初始化的状态. ParaNSim 中采用了 2 个 barrier —— t_start 和 t_end，分别用来同步所有线程每个周期的开始和结束. 主线程和子线程的运行伪代码如图 5 所示.

主线程：	子线程：
While(网络中有数据包){	While(网络中有数据包){
pthread_barrier_wait(t_end)；	pthread _ barrier _ wait (t _ start)；
//等待所有子线程结束	//等待开始
周期计数增加；	驱动当前子线程所属的所有路由模块；
驱动 NIC 模块；	统计当前子线程所属的所有仿真数据；
驱动链路模块；	pthread_barrier_wait(t_end)；
pthread _ barrier _ wait (t _ start)；	//当前周期运行完毕
//启动所有子线程	}
}	

图 5　ParaNSim 的主要运行伪代码

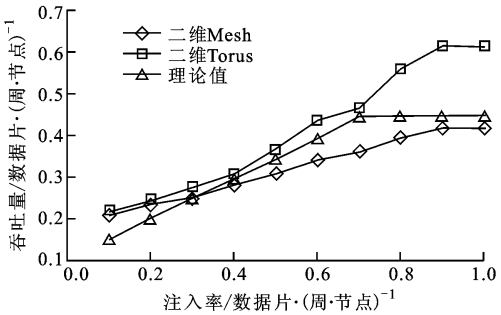
图 5 展示的是 ParaNSim 的核心代码，主线程除了图中所示的功能之外，还具有仿真器的初始化、启动子线程以及统计全局数据的功能. 关于子线程任务的划分，由于过多的子线程会造成很大的同步开销，所以本仿真器未采用每个子线程对应一个路由模块的结构，而是将路由模块均匀分配给各个子线程，以确保子线程的负载均衡.

2.3　正确性验证

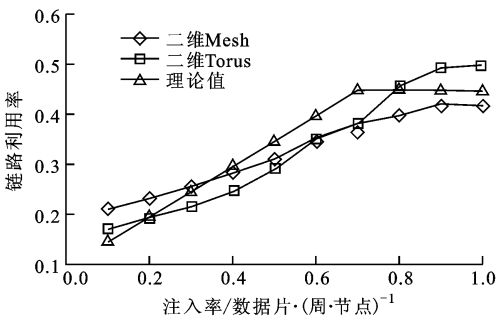
为了验证本仿真器实验结果的正确性，我们采用了如下 2 种实验方案.

(1)随机生成几组测试数据，这些测试数据有几万个甚至几百万个，将这些数据分别放在 2 个拓扑结构(二维 Mesh 和二维 Torus)的片上网络仿真器上运行，然后画出延时、吞吐量、链路利用率随注入率变化的曲线，如果曲线和正确的相符，就证明该仿真器是正确的. 本设计中采用的正确模型来自于参考文献[2].

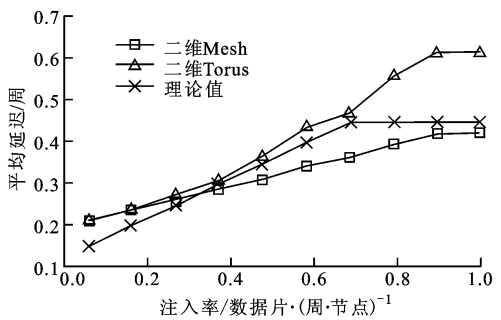
如图 6 所示，网络吞吐量随着注入率的增加而线性增加，当达到一定程度(饱和度)后便停止继续线性增加，并相对保持不变；网络平均延迟随着注入率的增加而逐渐增加，并且增加的速度越来越快；网络链路利用率随着注入率的增加而增加，达到一定程度后便相对保持不变. 由此，可以推断本设计实现的仿真器是正确可用的.



(a)吞吐量



(b)链路利用率



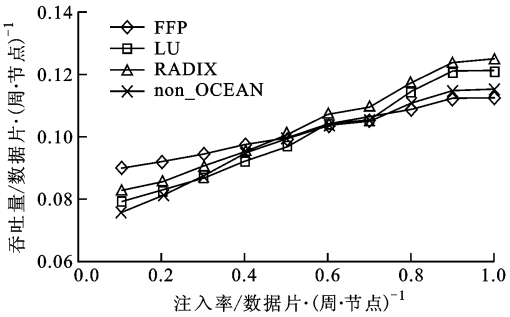
(c)平均延迟

图 6　方案 1 测试数据比较图

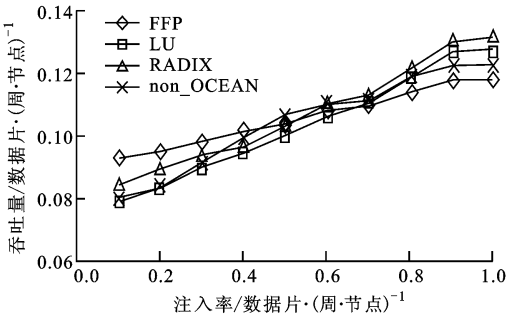
(2)利用实际 benchmark 的 trace 进行测试. 本设计采用的 trace 是由多核版本的 Sim-Godson 模拟器[8] 分别运行程序 FFP、LU、RADIX、non_OCEAN生成的. 如果在 ParaNSim 仿真器上运行这些程序生成的 trace 能够顺利通过，且测得的

性能指标与源仿真器测得的指标类似,则证明该仿真器的结果是正确的.

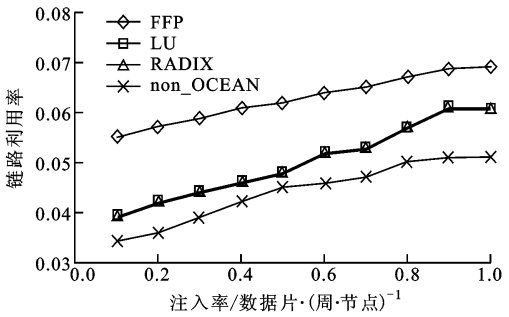
实验中用 ParaNSim 运行生成的 trace 文件获得了通过,而且片上网络的各项性能指标也符合标准,因此我们认为 ParaNSim 的仿真结果是正确可信的. 部分实验数据见图 7.



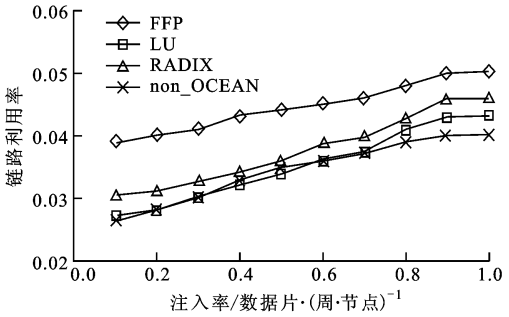
(a)二维 Mesh 吞吐量



(b)二维 Torus 吞吐量



(c)二维 Mesh 链路利用率



(d)二维 Torus 链路利用率

图 7 方案 2 测试数据比较图

2.4 并行化的优化策略

并行程序设计中的一些传统问题在并行仿真中仍然存在,如共享变量的竞争访问、锁的相对较大的性能开销、线程间存储访问的假共享等,这些问题容易形成并行程序中的性能瓶颈,阻碍同步技术有效性的评估. 根据仿真器的软件架构,我们制定了解决上述问题的多种优化策略,这些策略包括:线程局部存储,用于解决共享变量竞争访问的问题;无锁化读写队列,用于解决互斥锁开销大的问题;线程私有空间,用于解决假共享的问题;等等. 具体方案如下.

(1)利用 GNU C 中提供的线程局部存储机制,把全局变量申明为线程私有变量,从而避免并发写行为. 然而,有些全局变量可以线程私有化,而有些私有化之后必须保证私有值之间的一致性. 对于并不频繁访问的全局变量,用锁来保证互斥一致性,这时锁的开销对性能影响有限. 对于私有化后的全局变量,如果需要保证一致性,可通过一种线程间的通信机制——无锁化队列来实现.

(2)虽然线程共享所属进程的存储空间,线程间的通信机制似乎显得多余,但当考虑到各线程之间必须依靠一些变量来实现同步,即私有化之后的"全局变量"的读写顺序决定了程序行为时,这种通信机制就显示出必要性与高效性了. 无锁化队列的实现方法是:在每一对需要通信的线程间建立 2 个循环队列,每条线程分别充当这 2 个队列的生产者与消费者,在每个队列上,生产者写队列尾,消费者读队列头,通过头、尾的标志判断队列的空、满状态,读、写并行执行,从而所有的通信都可以并行执行. 这种方法避开了锁的竞争及造成的陷入开销,缓解了程序热点对性能的影响.

(3)假共享问题主要是由数据排列位置引起的,我们通过线程私有空间彻底解决了这个问题. 线程私有空间是 DLMALLOC 库的一种机制,允许程序 malloc 时指定一个私有空间为参数,DLMALLOC 只在此私有空间分配内存,私有空间以较大的块(例如 64 kB)为单位增长,块边界保持对齐,因此可以有效避免假共享问题,并且无空间浪费的缺陷. 上述优化策略保证了并行仿真器不受程序热点数据的影响,有效提高了仿真性能.

(4)保护对共享资源的并发访问是保障并行程序正确性和性能的重要因素,而锁是将共享资源的访问串行化的常用技术之一. 在 ParaNSim 中存在着大量共享资源,当核线程对这些资源进行访问的

时候，我们使用操作系统级的互斥锁来对其进行保护，同时合理安排锁的粒度和数量，以最大化并行多核功能仿真器的加速比. 另外，在多线程负载的多个上下文之间往往存在着大量的同步原语（如锁、路障等），串行仿真器执行某个上下文的同步原语时，只需要在相应的系统调用中将该上下文插入到适当的上下文状态链表中，如果某个上下文不能获得所请求的锁，则将其插入到挂起链表中. 对于并行仿真器，在将上下文插入到适当的状态链表的同时，还要将其所在的线程切换到合理的状态，为此在 ParaNSim 实现同步原语的系统调用函数中增加了相应的操作系统级同步操作，以达到此目的.

3　实验结果与分析

3.1　测试平台及参数

实验平台是曙光天演 EP850-GF 小型机，EP850-GF 服务器采用了高性能服务器芯片组、高速 HyperTransport 直连架构，配有 32 GB 内存、8 颗 AMD Opteron 8000 8346 1.8 GB 中央处理器. 实验使用的参数除了上文提到的拓扑结构、路由算法、仲裁机制和流控机制外，虚通道数量为 4，链路延迟为 2 周期，路由延迟为 1 周期，流量为 50 万个均匀模式的数据包，每个数据包包含 2 个数据片（flit）.

3.2　单独测试加速比结果

为了确定 ParaNSim 中子线程数量与加速比的关系，测试了多种规模的网络在注入 50 万个数据包的情况下，使用多线程的加速比，结果如图 8 所示.

从图 8 可见：在大多数情况下，小于 4 个子线程时的加速比变化很大，但未达到峰值；8 个子线程时的加速比基本达到峰值；子线程数量大于 8 之后，加速比变化不是很明显. 因为线程过多造成的共享资源的竞争会导致同步开销增大，造成仿真器整体性能下降，因此我们认为，在本实验平台下，4 个和 8 个子线程对测试并行片上网络的加速比具有代表性.

对 4 个和 8 个子线程在大规模网络情况下的测试结果如图 9 所示，从中可以看到：ParaNSim 在多线程情况下相对于单线程具有良好的加速比，尤其对于 10 000 个网络节点的规模，加速比达到峰值——4 线程的 1.44 和 8 线程的 2.42. 从平均值来看，4 线程和 8 线程的加速比分别达到了 1.35 和 2.1，由此可见，ParaNSim 能够有效地提高片上网络的仿真速度.

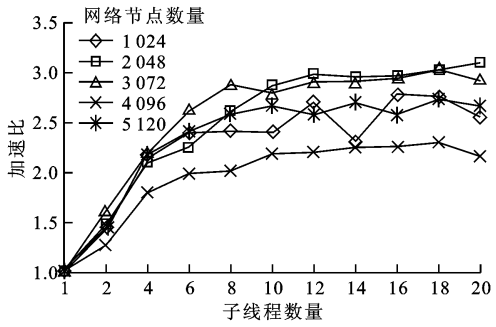


图 8　不同网络规模下的加速比

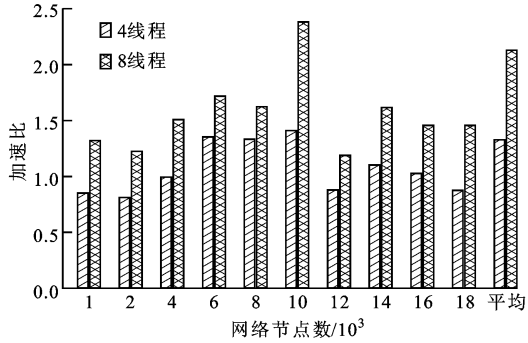


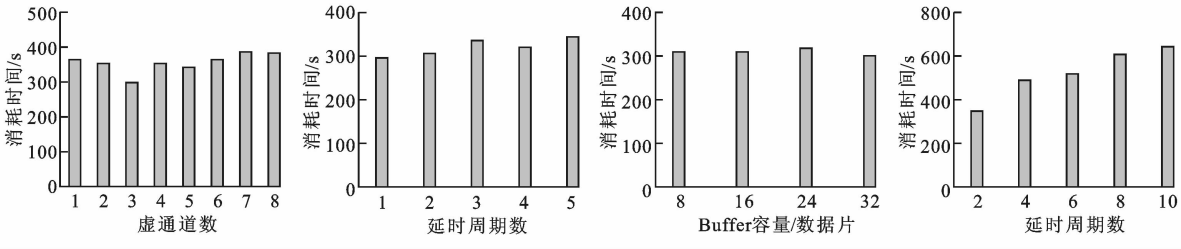
图 9　大规模网络的加速比测试结果

3.3　内部参数对仿真性能的影响分析

为了分析各种配置参数对 ParaNSim 性能的影响，我们在 3.1 节中描述的基本配置的基础上，单独改变某个参数，其他参数不变，通过对比使用 4 个子线程仿真所消耗的时间来分析相关参数对仿真性能的影响. 仿真结果如图 10 所示，可以看出：虚通道数量、路由延时、Buffer 容量这 3 个参数的变化对仿真性能的影响较小（消耗时间的差别不明显），这主要是因为这 3 个指标对应的模块都是由子线程完成的，而子线程的数量较多，能最大程度地并行化参数改变带来的仿真任务；链路延时对仿真性能的影响相对较大，这是因为链路的仿真是由主线程完成的，主线程的仿真速度对整体速度影响较大.

3.4　ParaNSim 与 Multi2Sim 集成的测试结果

将 ParaNSim 作为 Multi2Sim 模拟器[9]的一个子模块，通过运行 SPLASH2 负载测试集成仿真器的模拟速度加速比. 由于运行了实际的负载，所以片上网络的规模应与 Multi2Sim 的配置文件相对应，本实验设置了 32 节点和 64 节点 2 种规模. 此外，作为单独模块运行时，要考虑宿主仿真器的运行，若子线程数量过多，将会造成线程之间的资源竞争，从而导致性能下降，所以，我们在实验中只测试了 2、4、8 个子线程的加速比数据. 如图11所示，在2核、4线



(a)虚通道数量的影响 (b)路由延时周期的影响 (c)Buffer 容量的影响 (d)链路延时周期的影响

图 10 各种参数对仿真性能的影响

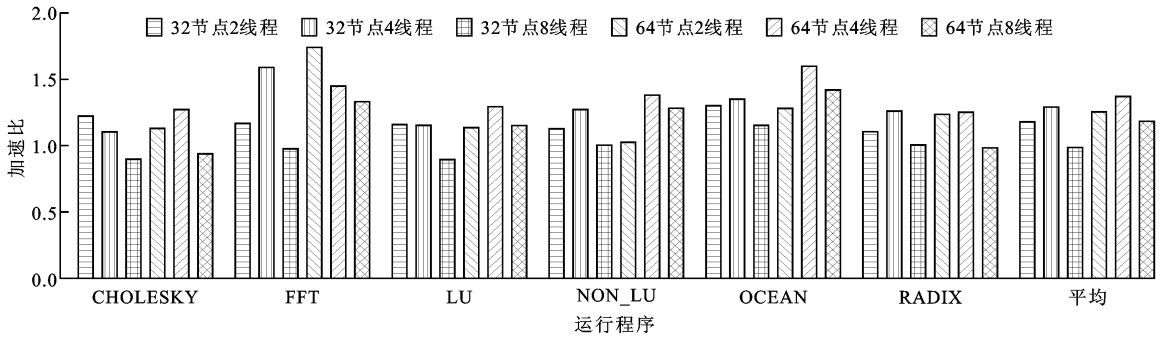


图 11 ParaNSim 与 Multi2Sim 集成的加速比测试结果

程的条件下,仿真速度均获得了不同的加速比:对于 32 节点网络,最高加速比为 159%,对于 64 节点网络,最高加速比可达 174%;2 种规模网络的最大平均加速比分别为 129%和 140%. 然而,在 8 线程的情况下,由于线程间的资源竞争,导致加速比增加不明显,尤其对于 32 节点网络,负载的平均加速比还略有下降,但在 64 节点的情况下,平均加速比则达到了 118%.

4 相关工作概述

国内外对片上网络已经进行了多年的研究工作,并且已开发出多种仿真器,例如在一些全系统仿真器(如 RSIM[10],SESC[11])中的片上网络模块,但这些全系统仿真器只是对片上网络实现了功能上的模拟,而模拟精度不够精确. 此外,还有一些独立的片上网络仿真器,如 NOXIM[1],SICOSYS[3] 等,这类仿真器只能对纯粹的片上网络进行性能测试,不能融入全系统模拟器. 而且,无论是 RSIM、SESC 中的片上网络模块,还是纯粹的片上网络仿真器,它们的仿真时间均随着负载规模的增大而增大,极大地影响到了研究效率. 本文研制的 ParaNSim 克服了这些缺陷,既能作为独立的片上网络仿真器,又能作为系统仿真器(如 Multi2Sim)的一个模块,此外还可以进行并行仿真,有效地降低了仿真时间,提高了研究效率.

5 结 语

使用仿真器已经成为对片上网络进行测试和性能评价的重要手段. 为了仿真大规模片上网络,传统的片上网络仿真器会消耗大量的时间. 本文设计并实现了一种并行片上网络仿真器 ParaNSim,通过实验证明它能够有效地提高片上网络的仿真效率. 此外,由于 ParaNSim 采用模块化设计,所以具有良好的扩放性,能够集成进其他模拟器(如 Multi2Sim)中. 在作为其他仿真器的片上网络模块时,ParaNSim 的性能受宿主仿真器性能的限制,相信随着性能更高的仿真器的出现,ParaNSim 的可扩性将会得到进一步的体现. 总体的测试结果表明,ParaNSim 能够大幅减少仿真时间,同时具有可行度高、配置方便及性能分析效率高等特点.

参考文献:

[1] PALESI M, PATTI D, FAZZINO F. NOXIM [EB/OL]. [2011-02-25]. http://noxim.sourceforge.net.

[2] DALLY W J, TOWLES B P. Principles and practices of interconnection networks [M]. San Francisco, USA: Morgan Kaufmann Publishing Inc., 2004: 521.

[3] PUENTE V, GREGORIO J A, BEIVIDE R. SICOSYS: an integrated framework for studying intercon-

nection network performance in multiprocessor systems [C] // Proceedings of Euromicro Workshop on Parallel, Distributed and Network-Based Processing. Washington DC, USA: IEEE Computer Society, 2002: 15-22.

[4] FUJIMOTO R M. Parallel and distributed simulations systems [C] // Proceedings of the 2001 Winter Simulation Conference. Washington, DC, USA: IEEE Computer Society, 2001: 147-157.

[5] CHANDY K M, MISRA J. Distributed simulation: a case study in design and verification of distributed programs [J]. IEEE Transactions on Software Engineering, 1979, SE-5(5): 440-452.

[6] VALIENT L G. A bridging model for parallel computation [J]. Communications of ACM, 1990, 33(8): 103-111.

[7] JEFFERSON D R. Virtual time [J]. ACM Transactions on Programming Languages and Systems, 1985, 7(3): 404-425.

[8] 黄琨，马可，曾洪博，等. 一种分片式多核处理器的用户级模拟器[J]. 软件学报，2008，19(4)：1069-1080.
HUANG Kun, MA Ke, ZENG Hongbo, et al. A use-level simulator for tiled chip multiprocessor [J]. Journal of Software, 2008, 19(4): 1069-1080.

[9] UBAL R, SAHUQUILLO J, PETIT S, et al. Multi2Sim: a simulation framework to evaluate multicore multithreaded processors [C]. Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing. Washington DC, USA: IEEE Computer Society, 2007: 62-68.

[10] PAI V S, RANGANATHAN P, ADVE S V. RSIM: an execution driven simulator for ILP-based shared-memory multiprocessors and uniprocessors [C] // Proceedings of Third Workshop on Computer Architecture Education. Washington DC, USA: IEEE Computer Society, 1997: 32-38.

[11] RENAU J, FRAGUELA B, TUCK W L J, et al. SESC simulator [EB/OL]. [2011-03-01]. http://sesc.sourceforge.net.

（编辑　葛赵青）