BDSim：面向大数据应用的组件化高可配并行模拟框架

李文明[1),2)]，叶笑春[1)]，张洋[1),2)]，宋风龙[3)]，王达[1)]，唐士斌[4)]，范东睿[1)]

[1)](中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 中国 100190)

[2)](中国科学院大学 北京 中国 100049)

[3)](华为技术有限公司中央研究院 北京 中国 100085)

[4)](高效能服务器和存储技术国家重点实验室 北京 中国 100085)

摘 要 大规模并行模拟是研究大数据体系结构的重要方法，对大数据应用及众核体系结构的发展有着不可替代的推动作用。然而，目前的模拟技术不能满足大数据体系结构研究的需求，主要体现在模拟速度慢、配置过程复杂、可扩展性差等方面。为了解决此问题，评估面向大数据应用的高通量众核体系结构的性能与功耗，本文提出了面向大数据应用的并行模拟框架——BDSim。该框架基于组件化思想，将功能组件与框架服务单元组成并行功能单元，并可根据负载情况，自由配置组件与框架服务单元之间的映射关系。为了提高组件之间的通信和同步效率，本文提出了一种非阻塞无锁通信优化方法，和一种 CMB 保守同步算法的优化算法——NMTRT-CMB 同步算法。通过模拟不同并发规模的基于 2D-Mesh 网络的众核系统的实验结果表明，与基于锁的并行通信方法相比，框架采用的非阻塞无锁通信优化方法可以提高并行模拟速度约 10%，与 CMB 同步算法相比，NMTRT-CMB 同步算法可以减少空消息数量达 90%以上。

关键词 组件化并行模拟框架；非阻塞无锁通信；CMB 算法；高可配

中图法分类号 DOI 号：

BDSim : A component-based high configurable parallel simulation framework for big-data application evaluation

Li Wen-ming[1),2)], Ye Xiao-chun[1)], Zhang-Yang[1),2)], Song Feng-long[3)], Wang Da[1)], Tang Shi-bin[4)], Fan Dong-rui[1)]

[1)](State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China)

[2)](University of Chinese Academy of Sciences, Beijing, 100049, China)

[3)](Central Research Institute, Huawei Technology Co. Ltd, Beijing, 100085, China)

[4)](State Key Laboratory of High-end Server & Storage Technology, Beijing, 100085, China)

Abstract Large-scale parallel simulation is becoming an important researching method in big-data architecture，which acts as an irreplaceable active driving role for big data application and many-core architecture development. However, the simulation techniques currently cannot meet big data architecture, mainly reflected in respects as flowing, the low simulation speed, the complicated configuration, the scalability and so on. In order to solve this problem, this paper proposes BDSim, a high configurable parallel simulation framework for big data simulation. It can evaluate the performance and power of High Throughput Computing, which aiming at big data application. The framework is based on the thought of component. Parallel function cell consists of components and parallel

收稿日期：年-月-日*投稿时不填写此项*；最终修改稿收到日期：年-月-日 *投稿时不填写此项*。本课题得到国家“973”重点基础研究发展规划项目基金(2011CB302501)；国家“863”高技术研究发展计划项目基金(2012AA010901)；国家自然科学基金项目(61100013,61100015, 61173007, 61202059, 61204047, 61221062, 61332009)资助。**李文明**，男，1988 年生，博士研究生，学生，主要研究领域为高通量处理器设计及软件模拟技术，E-mail: liwenming@ict.ac.cn. **叶笑春**，男，1981 年生，博士，副研究员，主要研究领域为高性能计算机体系结构及软件模拟技术，E-mail：yexiaochun@ict.ac.cn. **张洋，**男，1981 年生，博士研究生，学生，主要研究领域为高通量处理器设计及实时系统，E-mail: liwenming@ict.ac.cn. **宋风龙**，男，1980 年生，博士，高级工程师，主要研究领域为片上存储系统及高性能计算机体系结构，E-mail: songfenglong@gmail.com. **王达**，女，1981 年生，博士，副研究员，主要研究领域为处理器微体系结构，E-mail：wangda@ict.ac.cn. **唐士斌**，男，1986 年生，博士，主要研究领域为高性能计算机体系结构及片上存储系统，E-mail: tangshb@inspur.com. **范东睿**，男，1979 年生，博士，研究员，博士生导师，主要研究方向为处理器体系结构，E-mail: fandr@ict.ac.cn. 第 1 作者手机号：15201294897，E-mail：liwenming@ict.ac.cn

service cell and the mappings between components and parallel service cell are determined by loadings. To improve communication efficiency, this paper proposed optimal non-block lock-free communication method. And also presented NMTRT-CMB synchronization algorithm based on CMB conservative synchronization algorithm. Proved by many-core architecture based on 2D-Mesh NOC experiments under different parallel scalability, non-block lock-free communication method can help improving simulation speedup with 10%, compared to communication based on locking method. NMTRT-CMB can reduces almost 90% null message when running with 16 threads, compared to CMB.

Key words: Component modular parallel simulation framework; Non-block lock-free communication; CMB algorithm; High configurable

1 引言

随着网络购物、物联网、舆情监测等应用的迅速发展，数据中心需要处理的数据量呈爆炸式增长。面对快速增长的数据规模，面向传统应用的处理器体系结构在处理速度、功耗以及数据带宽等方面都表现出了不足与缺陷。大数据处理不同于用于科学计算的超级计算机，不是追求尽量缩短单个任务的计算时间，而是在允许的时间范围内处理尽可能多的任务（数据或线程）[1][2]。因此，现有的众核结构已不能满足实际需要，体系结构需要重大变革[3]。近年来，为了满足大数据应用的实际需求，众核体系结构的研究在学术界和工业界都得到了一定的发展。针对众核处理器性能、功耗、温度等方面的改善也成为当前的研究热点[4][5][6][7][8][9][10]。然而，目前的模拟技术，显然已不能满足针对大数据处理的大规模众核处理器的模拟要求，限制了众核体系结构的进一步发展，主要体现在模拟速度慢、配置过程复杂、可扩展性差等方面。

通常，一款模拟器的开发需要权衡以下四个方面：精度、速度、可扩展性以及可配置性。随着众核体系结构各功能部件数量与实现复杂度的增加，传统串行模拟器因为速度的原因已经严重不适于大规模众核处理器的模拟，成为限制模拟器发展的主要因素。例如，目前常用的串行模拟器 GEM5[11]、MARSS[12]，模拟速度大约在 200KIPS 左右。基于此速度，模拟一个真实物理核的一秒钟大约需要几个小时，模拟一千个核的一秒钟，几乎需要一年的时间。而与串行模拟器相对的并行模拟器能并发模拟功能部件执行状态，显著提高模拟速度，被广泛用于众核体系结构的模拟。

然而，大规模并行模拟器的开发面临诸多挑战。首先，被模拟结构复杂。随着体系结构不断发展，不仅模拟系统内模块的数量逐渐增加，模块的功能也更加复杂，导致模拟器开发任务量大、周期长。其次，模拟器并行困难。由于模拟器系统是紧耦合设计，导致并行过程中任务难以划分、并行单元难以管理、同步控制难以实现，此外并行模拟过程中的消息通信也成为制约性能提升的重要因素。最后，系统构造过程复杂。不同的配置形成的不同的体系结构（包括：网络拓扑、存储层次、一致性协议等）导致目标系统性能迥异，而大规模模拟带来了配置上的沉重负担。所以，灵活的配置功能对体系结构研究人员是至关重要的。

在对大规模并行模拟系统的强烈需求下，学术界与工业界涌现出大量的并行模拟器。比较有代表性的研究工作有 Simics[13]、Graphite[14]、SimFlex[15]、SST[16]、Hornet[17]、SlackSim[18]、ZSim[19]等等。但是这些模拟器研发初期便以不同的侧重点为目标，导致其并不能完全满足高通量众核处理器的模拟需求，例如 Simics 拥有及其丰富的模拟组件用于搭建系统模型，在学术和工业界都得到了广泛的应用，但对基于共享访存的多核模拟系统并没有得到很好的支持；SST 采用模块化搭建方法，具有较好的可配置和扩展性，但是 Barrie 同步方法导致其速度不尽人意；ZSim 则采用不同的加速方法来提高模拟速度，但专用性太强，灵活性和可扩展性较差。由此可见，现存的并行模拟器在模拟速度、灵活性、可扩展性及对大规模并发模拟等方面的支持上均不能满足其要求。

针对目前的现状，为了找到高效的解决方案，本文提出了支持大规模并行系统模拟的并行模拟框架——BDSim。该模拟框架基于组件化思想，将功能模块抽象成功能组件，以动态链接库的形式存在和加载，增强了模拟的灵活性和模块的复用性。模拟框架采用高效的通信方式和同步算法，提高了执行效率。同时支持高可配设计模式，提供文件和

图形化两种配置方法，降低了配置的复杂性，提高了框架的易用性。通过实验分析和应用举例表明，BDSim 并行模拟框架能够很好地支持大规模并发系统的模拟，适用于大数据处理的众核处理器体系结构的研究。

本文的主要贡献如下：

- 基于组件化思想。所有功能模块以组件库的方式加载至模拟框架中,已开发的组件以资源库的形式存在。提出了 FS(Framework Service)概念，作为组件运行的代理，是并行的最小单位。
- 实现高效的非阻塞无锁通信方法。显著加快消息的处理速度，提高整体框架的执行效率。
- 实现高效的 NMTRT-CMB（基于空消息时间戳请求标志位）同步算法，同时支持粒度可调的松散同步算法。BDSim 提供两种同步算法，一种是基于 CMB（Chandy-Misra-Bryant）同步算法[20]的改进算法,NMTRT-CMB 同步算法。另一种是粒度可调的松散同步。
- 支持高可配设计模式。BDSim 提供两种配置方式，一种是通过配置文件，另一种是通过图形化操作界面。

文章组织结构如下：第 2 节描述相关工作及研究动机；BDSim 模拟框架的实现细节及各种算法应用将在第 3 节讨论，包括基础框架、端口、组件和 FS 等模块的实现及非阻塞无锁通信方法和 NMTRT-CMB 同步算法；实验和结果分析将会在第 4 节给出；第 5 节通过应用举例，描述如何使用 BDSim 搭建高效的大数据模拟评估模型；最后对论文进行总结及阐述未来的研究方向。

2 相关工作和研究动机

随着众核体系结构规模日益庞大，大规模并行体系结构的模拟变成巨大的挑战。目前，大多数模拟器都是串行执行。串行模拟器用一个主机线程来模拟整个目标系统，当目标系统的核数增加时，分配给单个核的模拟性能就会下降。目前已经有许多方法被用来加速模拟，包括并行模拟、直接执行[21]、FPGA 加速[22]等等。下面简要介绍几款支持大规模体系结构模拟的模拟器及其关键技术。

ZSim[19]，是 MIT 和斯坦福大学推出的一款千核级并行模拟器，目标是解决众核并行模拟的速度问题。ZSim 通过使用基于指令驱动的时序模拟配合二进制翻译机制实现单核模拟组件加速。在并行加速方面，提出了 Bound-Weave 执行算法，将模拟分成 Bound 和 Weave 两个阶段。Bound 阶段首先进行一定时间的功能模拟，同时记录必要的访存序。Weave 阶段根据之前记录的访存序驱动并行模拟，输出时序统计信息，极大的加快了模拟速度。虽然对于众核模拟能够达到很高的速度，但是对于其它功能模块的模拟，例如网络、存储等模块，并不是其研究重点。同时不支持组件化，不能够灵活配置并行度，限制了其使用范围。

MARSS[23]，是一款结合了 QEMU 和 PTLsim 的支持多核模拟的全系统模拟平台。MARSS 具有 QMEU 的全系统支持功能及二进制翻译加速功能，也具有 PTLsim 精确的乱序核模拟功能。同时也提供详细的性能统计分析功能和配置功能。不足之处是 MARSS 模拟平台是基于 X86 指令集的全系统模拟，且可扩展性比较差，严重影响了二次开发，且对于大数据众核模拟并不能提供很好的支持。

Asim[24]，是 Intel 研发的一款针对复杂计算机系统的模块化模拟框架。Asim 通过模块化和重用性解决模拟的复杂性。通过模块化的实现方法，可以将复杂的性能模拟分解成离散的单模块，简化了实现的复杂性。模块的可重用性可以将已经验证的模块重新用于新环境中，既保证了模块的可靠性，也加速了模拟模型的实现过程，提高了开发效率。然而，Asim 的最大缺陷在于其模拟的非并行化，这对于大数据的模拟需求来说是难以回避的问题。

SST[16]，是一款开源，模块化，支持并行执行的模拟框架。模块主要包括处理器、储存器及一些网络模型。SST 已经广泛应用于 HPC 系统的模拟。它采用了模块化的搭建方法，允许目标系统使用已经存在的组件来组装实现。在并行模拟方面，使用 Barrie 的方法进行同步操作。然而，此种同步方法会降低模拟的速度。

Manifold[25]，是一项开源软件工程，旨在为多核体系架构提供建模和模拟的可扩展性基础框架。方法是通过使用成熟的并行离散事件模拟(PDES)算法及并行时间步进技术实现粗粒度的并行模拟以获取模拟的可扩展性。Manifold 对于并行模拟具有很好的支持，但是并不支持松散同步机制，影响模拟的速度。

SIMFLEX[15]，是一款基于组件设计思想的模拟框架，具备精确的采样统计功能，支持复杂模型开发，能够确保在快速模拟过程中获取到具有代表性的统计数据。SIMFLEX 的创新之处在于它将独特的编译时方法应用于组件链接过程，能够运行不

经修改的商业负载，且能获取到精确的模拟数据。SIMFLEX 使用基于采样的方法来提高模拟速度，缺点是不支持并行模拟机制。

Simics[13]，是一款商用模拟框架，拥有丰富的模拟组件，在学术和工业界都得到了广泛的使用。处理器、存储器、系统控制器、各种总线和网络结构等目标系统都可以模拟实现。同时提供方便的软件测试和调试环境，也支持回滚调试和执行。为了提高模拟速度，Simics 推出了 Simics Accelerator 加速器[27]。它能充分地利用多核宿主机来提高目标系统的模拟速度。然而，大规模并发系统及共享访存式多核系统并没有得到很好的支持。

除此之外，还有众多针对大规模体系结构的模拟模型，例如 BGLsim[28]、BigSim[29]、Hornet[17]、PartitionSim[25]、 SlackSim[18]、SimHPC[30]、WWT[31]等并行模拟器。在众多模拟加速技术中，并行模拟因为能够利用目标结构天然的并行特征，能在低成本 CMP 计算机上实现，成为一种常用的有效加速技术。

但是，针对大数据处理的众核模拟系统需达到千核万线程量级。以上各模拟器在模拟规模一旦达到千核万线程量级时，在速度、灵活性、易用性、可扩展性等方面会暴露出各种缺点，并不能很好的满足需求。另一方面，传统的体系结构已经不能满足大数据高通量的需求。Michael Ferdman 等人在当前流行的处理器硬件上针对规模化大数据应用做了大量的实验[32]。结果表明，在传统的处理器架构中表现出色的核内乱序、指令级并行、层级 Cache、访存预取等结构设计在大数据应用下表现平平，甚至成为影响性能的瓶颈问题。因此，亟需一款灵活、易用、高可配、高效率的并行模拟器，以解决大数据处理结构研究和制造的模拟需求。

考虑到以上各模拟器的优势和缺点以及针对大数据应用的模拟特点，本文提出了一种基于组件化的面向大数据应用模拟的并行模拟框架 BDSim。BDSim 采用离散并行事件模拟机制，首次提出了 FS(Framework Service)概念，支持灵活的并行粒度调节。支持 NMTRT-CMB 同步算法和松散粒度同步算法。使用优化的非阻塞无锁通信方式，显著提高了模拟速度。在配置方面，BDSim 提供了方便灵活的配置方法，相对于其它模拟器，在高通量众核系统建模过程中，极大地减少了使用的复杂度，减轻了研究人员的负担。

3 BDSim 结构与特点

本节从组件化、通信、同步及配置等方面对 BDSim 实现结构及特点进行详细描述。

3.1 组件化框架

3.1.1 相关概念

PDES(Parallel Discrete Event Simulation)：并行离散事件模拟。PDES 根据事件和时间戳驱动模拟执行，相对于传统的系统时间驱动方法，PDES 具有较高的执行效率[33]。图 1 展示了 SDES（串行离散事件模拟）和 PDES 之间的关系。可以看出，SDES 可以由一个或多个逻辑同能单元构成，而多个 SDES 按照时间轴同时运行构成 PDES。

组件：逻辑功能单元。使用者在 BDSim 提供的接口协议之下定义的为实现特定功能的模块单元，以动态连接库的形式存在，可根据配置，动态加载至 BDSim 模拟框架上。

端口：组件属性之一，消息通信处理单元。端口的数量是由与其相联的其它组件的个数决定。当目标模型拓扑结构确定后，端口会得到一个全局编址。端口地址会插入到消息中，用于路由和寻址。

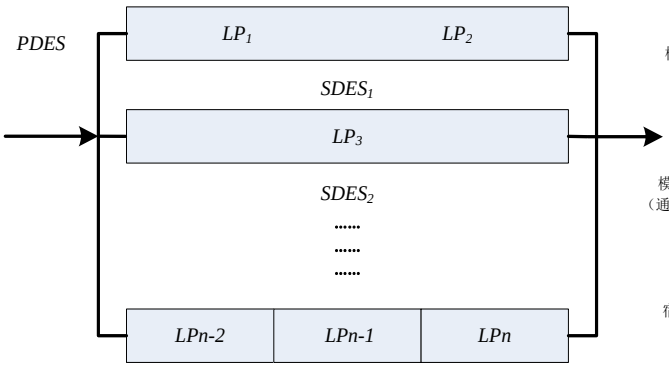


图 1 PDES 和 SDES 关系示意图

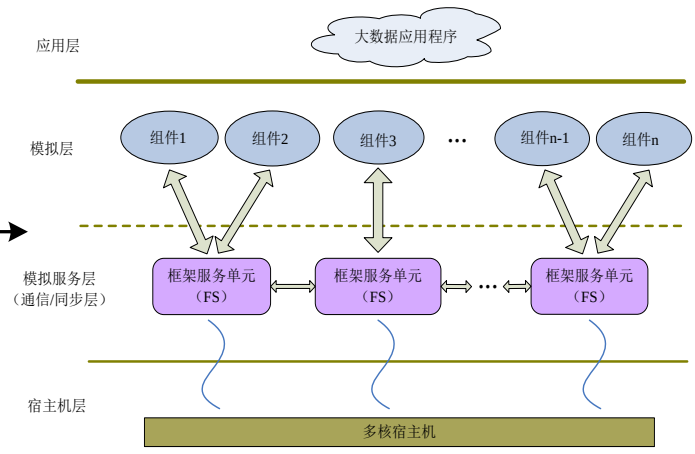


图 2 BDSim 框架结构示意图

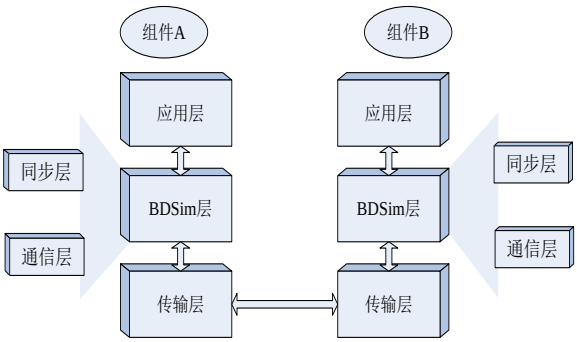


图 3 BDSim 通信协议栈示意图

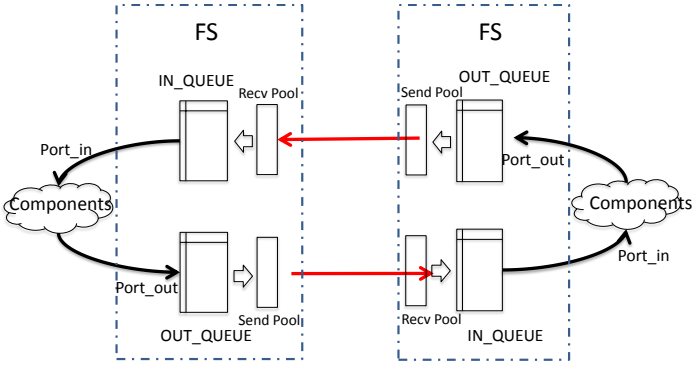


图 4 通信结构示意图

FS（Framework Service）：框架服务单元，并行执行的基本单元。所有功能组件必须挂载到 FS 之上。FS 相当于组件代理，处理拓扑、通信、同步等关键问题。通过使用 FS，一方面可以将用户从繁杂的并行同步和通信处理中解脱出来，交由 FS 处理，用户只关注于组件的功能实现；另一方面，FS 的使用使得模拟框架高度模块化，这使得大规模数据处理并行模拟成为现实。

3.1.2 框架架构

BDSim 框架结构如图 2 所示。最底层是多核宿主机。之上的模拟服务层是框架的主体部分，由 FS 组成，实现整个模拟框架的同步和通信等功能，为上层的各种组件运行提供支持。模拟层即按照模拟需求实现的各功能组件，所有组件的功能之和构成了整个目标系统的模拟功能。最上层是应用层，具体实现目标二进制程序的加载和用户控制的响应。BDSim 将框架拆分成离散的 FS 单元，与功能组件绑定在一起。每个 FS 和与其绑定在一起的组件运行在一个线程之上。通过这种方式，一方面可以方便地实现并行粒度调节；另一方面为组件开发提供了良好的模块化基础。

3.2 非阻塞无锁消息通信

在 BDSim 模拟框架中，拓扑关系由组件端口的互连确定，消息的收发由 FS 处理。类似 TCP/IP 协议，组件间的通信可以看成三层结构，层与层之间消息的传递是透明的，每一层负责完成自己所在层的任务，消息按照层次之间的协议分别进行处理。通信协议栈如图 3 所示，同步和通信功能在 BDSim 层中实现，主要维护各并行单元之间的同步关系及消息传递和处理。

组件之间进行通信时，发送方将消息打包，利用模拟框架提供的通信接口将消息送往发送方组件所在的 FS。FS 收到消息之后，根据源端口通过查找拓扑表找出目的端口，由目的端口号获得目的端口所在组件的全局编号，根据组件的全局编号，进而可以得到组件所在 FS 的全局编号，将消息和目的端口号打包之后，发往通信层；通信层根据 FS 的编号将消息发往目的 FS。最后，目的 FS 根据目的端口号判断出消息的目的组件，将消息发往目的组件端口。

在 BDSim 中，每个 FS 拥有一个输入队列，一个输出队列。FS 提供 *Port_out* 接口给组件，组件可以使用此接口将发出消息传至 FS 的输出队列。组件需要提供 *Port_in* 接口给 FS，FS 可以通过此接口将消息发给相应的目的组件，如图 4。通过这种消息处理方式，可以实现不同组件之间的消息传递。在并行环境中，消息是线程/进程间数据和控制信号传递的基本方式。消息的传递和处理会占用很多资源，执行过程中大量的消息传递会严重影响模拟速度。在 BDSim 中，组件间的通信是靠消息的传递来完成，通信开销有时会占到系统运行时的一半左右，所以通信效率的高低对于模拟框架的性能有着非常大的影响。例如，一个 Core 组件和 Memory 组件通过三级总线互连，两者之间的通信消息包需要经过 4 跳才能到达目标组件。理想情况下，4 跳至少需要 4 个 Cycle 的延迟。通常情况下，由于路由阻塞等原因，一个消息包所消耗的时间往往大于 4 跳。而消息传递的长延迟会严重影响模拟器的性能，所以，优化通信和降低消息数量对模拟框架的性能也具有重要的意义。

3.2.1 非阻塞无锁消息通信

消息传递方式分为两类：阻塞和非阻塞。阻塞算法允许缓慢或延迟的处理单元阻止更快的处理单元完成操作，从而无限期地共享数据结构。对于非阻塞算法，如果有一个或者多个处理单元将要操作共享数据结构，可以保证操作在有限的时间或者步骤内完成。在同步的多处理器系统中，当处理单元被阻塞在不合适的地方，将会严重影响系统性能。因此，BDSim 采用非阻塞消息传递方式。另一方面，

BDSim 采用共享存储通信方式，消息以先进先出队列的形式被存储和访问。对共享队列进行并行访问时，需加锁以保证访问的正确性。

对于共享先进先出队列的锁避免算法已经有了非常多的研究。Hwang 和 Briggs 提出了一种基于 *Compare_and_swap* 指令的锁避免算法[34]。在这些算法中通常忽略了空队列的处理、队列中只有一个元素时的情况，以及当插入队列与删除队列同时发生时的情况。Lamport 指出利用单读单写队列可以实现无锁算法[35]。

3.2.2 BDSim 通信

BDSim 模拟框架中组件间的通信主要是由 FS 代理完成，FS 内部串行执行，FS 之间并行执行。BDSim 模拟框架中 FS 的个数相对来说比较少，所以可以在每一个 FS 中，为每对通信的 FS 建立一个缓冲队列，FS 之间的通信对于共享缓冲队列的操作则转变为单读单写队列。根据 Lamport 理论[35]，可以实现非阻塞无锁的操作。由于与每个 FS 相连的其它 FS 的个数不确定，为了减少通信中发送端查找通信队列的开销，根据 FS 的个数为每一个 FS 建立对应数目的通信队列，发送端在发送消息时，只需根据自身 FS 的编号，将消息插入到与此编号对应的通信队列的末尾即可。但是，模拟框架针对的是大规模众核系统的模拟，当模拟目标规模达到一定的程度，FS 及 FS 之间的通信连接数量急剧增加，会造成空间的浪费。为了减少冗余队列的开销，提高通信队列的空间利用率，FS 中的通信队列采用动态生成机制，即 FS_i 与 FS_j 首次通信时动态建立两者之间的通信队列 Q_{ij}。因为不同负载和拓扑结构下，通信数量会有变化，所以，当前 BDSim 对通信队列长度不做限制。且消息的传递只是事件消息内容指针和路由信息的传递，并不传输正真的数据内容，这在一定程度上减轻了存储空间的开销。通信队列内存开销实验及分析结果见第 5 部分性能评估举例。

当插入一个节点时，按节点携带时间戳大小搜索队列，选择插入点并执行插入操作，*tail* 指针更新指向队尾节点。当删除一个节点时，将 *head* 指针所指的下一个节点返回，然后 *head* 指针后移即可。将 *head* 和 *tail* 指针指向空节点，且 *tail* 指针并不设置为最后一个节点，这种方法可以将对 *head* 和 *tail* 指针的修改分别限制在 *Dequeue* 和 *Enqueue* 操作中。具体的队列数据结构及算法操作步骤如算法 1。

算法 1 单读单写消息队列插入/删除算法

算法：单读单写消息队列插入/删除操作

队列插入操作：

输入：待插入消息节点

输出：插入成功队列

步骤：

（1）搜索 FS 中与此连接端口相对应的消息队列；

（2）创建节点，将待插入消息内容赋值给新节点；

（3）按时间戳大小，搜索队列中待插入点，将节点插入；

队列删除操作：

输入：待插入消息节点

输出：插入成功队列

步骤：

（1）若队头指针不等于队尾指针，返回队头指针指向节点并更新队列；

（2）否则，返回 NULL；

3.2.3 BDSim 通信扩展

1）直接通信

BDSim 在提供消息通信的同时，也提供了直接通信方式——*Callback* 函数(回调函数)。组件可以直接调用其它组件在框架中注册的回调函数来实现信息传递。直接通信方式可以对并不关心的模拟功能模块实现加速，提高模拟效率，也为紧耦合的组件实现提供了灵活的外部调用接口，方便使用。同时也可以作为辅助调试接口，加速模拟器的调试。

2）进程通信

并行模拟是利用多核平台提高模拟器速度的有效手段，随着模拟的处理器核数的增加，利用多进程、多机模拟是提高模拟速度的有效方法。BDSim 在向多进程、多机方向扩展时，主要解决的问题是保证组件间的消息能够正确传递至目的地址。为了实现多进程、多机之间的消息传递，在 BDSim 模拟框架中增加单独的消息处理单元—CP (Communication Process)，专门处理进程间消息收发。扩展通信结构如图 5 所示。

处于同一进程中的 FS 利用共享内存的方式进行通信，不同进程间的 FS 利用消息传递的方式进行通信。CP 作为逻辑处理单元与其它模拟模块一样参与调度，执行到 CP 模块时，将本进程需要发送的消息统一发送出去，同时将发往本进程的消息统一接收回来。对于本进程内部线程间的通信，只是完成消息指针的传递，不同进程间的通信才需要消息的拷贝。

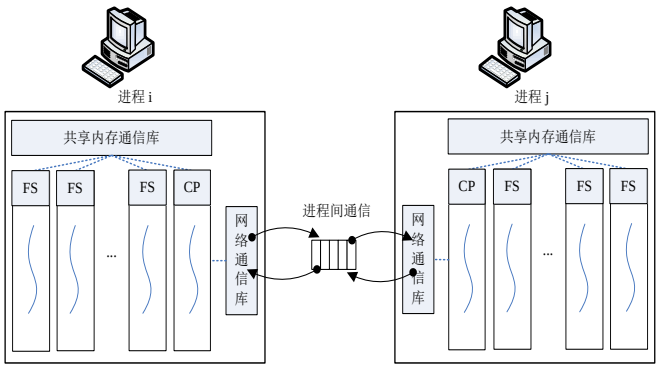


图 5 扩展通信结构示意图

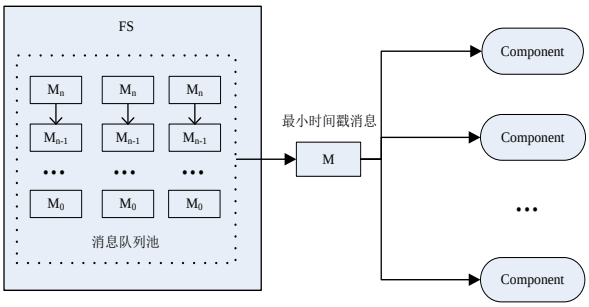


图 6 FS 维护消息队列示意图

3.3 NMTRT-CMB同步算法

在并行模拟中，同步策略是模拟速度的主要制约因素。不同的同步策略会带来模拟性能上的巨大差异。下面从同步控制角度来优化 BDSim 模拟框架。

3.3.1 同步算法

并行模拟中同步问题的本质是在不同逻辑单元之间高效地维护实体间的因果关系，根据是否严格按时间戳顺序来处理事件可以把 PDES 同步分为两种：保守同步和乐观同步。

在保守同步机制中，局部因果关系需要严格保证，任何可能违背因果关系的操作都被禁止，在确保它的事件列表中最小时间戳事件可以被安全处理之前，该逻辑单元被阻塞执行。相反，乐观同步机制中则允许事件被乱序执行。如果收到一个会影响后续事件的小时间戳事件，系统就需要进行回滚以恢复到该时间戳之前的状态。虽然乐观同步减少了系统等待时间，但是为了支持回滚，乐观同步机制中需要提供保存和恢复机制，这会带来整个系统的计算资源和网络资源的开销。因此，BDSim 采用保守同步算法。

CMB 算法[15]是保守同步经典算法，其基本思想为逻辑处理（LP）单元之间的消息按非递减的顺序发送，消息接收者为每一个连接维护一个队列来保存收到的消息。每个队列都有一个时间戳，为队列中时间戳最小的消息，通过最小时间戳的不断增长，整个系统得以向前推进。如果队列为空，那么队列时间则是最后一个收到的消息的时间戳。

CMB 算法步骤：

（1）找出所有队列中时钟值最小的队列；

（2）如果该队列中有消息，则取出消息并处理，转（1）；

（3）如果该队列中没有消息，则阻塞等待直到这个连接上有一个新的消息来临；

（4）新的消息将该队列的时间戳更新，转（1）。

在 CMB 算法中，当三个 LP 之间有消息同步时，可能会因互相等待对方消息而发生死锁现象。在死锁处理方面，CMB 算法是通过发送空消息的方法解决。当 LPi 向 LPj 发送消息时，需要同时向其它相连 LP 发送同样时间戳的空消息。空消息不携带有用信息，只包含时间戳。时间戳给出了本 LP 以后发出的消息的时间戳的最小值，在收到一个空消息之后，接收端可以将与连接上的时间戳更新至这个时刻，从而使得该 LP 可以重新选择时间戳最小的队列，则系统可以避免死锁，继续执行。

3.3.2 NMTRT-CMB 同步算法

在 CMB 算法中， FS 为每一个组件中的所有端口通道都维护一个接收队列，那么接收队列将会非常多，相应的空消息数量会变的非常巨大，这会消耗大量的系统资源，严重影响模拟的速度。

为此，本文提出基于 CMB 算法的 NMTRT-CMB (基于 CMB 的空消息时间戳请求标志位)同步算法。对于处在同一个 FS 中的组件，可以近似认为它们的局部时钟一致，组件时钟以 FS 时钟为准。将 FS 及 FS 上的组件作为一个整体看作一个逻辑处理单元，只需为与其相连的 FS 维护队列保存接收到的消息即可。局部 FS 的时钟等于此 FS 维护的接收队列的最小时钟。

CMB 算法要求逻辑处理单元之间的消息按非递减的顺序发送，而在同一个 FS 中的组件由于处理消息的能力不同，所产生的延迟也会不同。为了实现消息按非递减的顺序发送，FS 中需维护一个发送队列，消息按发送时间在发送队列中有序排列。FS 在发送消息时，如果消息的时间戳小于等于 FS 的局部时钟则将此消息发送出去，如图 6 所示。

FS 从维护的通信队列中选取时间戳最小的队列，如果队列为空，则向与此相连的所有 FS 发送

```
/* 组件，初始化/结束函数及向FS提供的端口函数声明 */
DECL_COMPONENT ( router )
DECL_INITFUNC ( Init )
DECL_FINFUNC ( Fini )
DECL_CALLBACK ( Port_In )

/* 端口声明 */
DECL_IN_PORT     ( north_in, 64 )
DECL_OUT_PORT  ( north_out, 64 )
DECL_IN_PORT     ( south_in, 64 )
DECL_OUT_PORT  ( south_out, 64 )
DECL_IN_PORT     ( east_in, 64 )
DECL_OUT_PORT  ( east_out, 64 )
DECL_IN_PORT     ( west_in, 64 )
DECL_OUT_PORT  ( west_out, 64 )
DECL_IN_PORT     ( self_in, 64 )
DECL_OUT_PORT  ( self_out, 64 )


/* 用户自定义命令声明 */
DECL_COMMAND  ( showreg, “show register information\n,” Com_Showreg)
DECL_COMMAND  ( showport, “show port information\n,” Com_Showreg)
DECL_COMMAND  ( routerstart, “start running a router\n,” Router_Start)

/* 组件参数声明 */
DECL_PARAMETER  ( coordinate_x, “enter x value:”, 1, “normal”, “0” )
DECL_PARAMETER  ( coordinate_y, “enter y value:”, 1, “normal”, “0” )
```

图 7 Router 组件配置文件

```
/* 组件类型及数量说明 */
[ COM ]
1
16 , router
/* FS数量及每个FS挂载组件数量和编号 */
[ FS ]
4
4 , 0 , 4 , 8 , 12
4 , 1 , 5 , 9 , 13
4 , 2 , 6 , 10 , 14
4 , 3 , 7 , 11 , 15
/* 端口连接决定拓扑结构 */
[ TOPO ]
 0 , -1
 1 , -1
 2 , 41
 3 , 40
 4 , 17
 5 , 16
 6 , -1
 7 , -1
 8 , -1
 9 , -1
 10 , -1
 11 , -1
 12 , 51
 13 , 50
 14 , 27
 15 , 26
 16 , 5
 17 , 4
 …
```

（a）*Topology* 文件

```
/* 组件类型及数量说明 */
router
16
default_value

/* Router组件编号、x及y坐标值 */
0  x  0  y  0
1  x  1  y  0
2  x  2  y  0
3  x  3  y  0
4  x  0  y  0
5  x  1  y  0
6  x  2  y  0
7  x  3  y  0
8  x  0  y  0
9  x  1  y  0
10  x  2  y  0
11  x  3  y  0
12  x  0  y  0
13  x  1  y  0
14  x  2  y  0
15  x  3  y  0
```

（b）*Parameter* 文件

图 8 4x4 Mesh 拓扑配置文件

空消息且将时钟请求标志位置位。对于收到的空消息，如果时钟请求标志位被置位，则向与此相连的所有其它 FS 发送空消息；如果时钟请求标志位没有被置位，则 FS 只是更新对应的接收队列时钟，释放空消息。

NMTRT-CMB 算法步骤为：

（1）找出所有队列中时钟值最小的队列；

（2）如果该队列中有消息，如果是普通消息则取出处理转（1），如果是空消息且时钟请求标志位置位，则向与此相连的 FS 发送空消息，如果时钟请求标志位没有置位，那么只更新队列时间戳，转（1）；

（3）如果该队列中没有消息，则阻塞等待直到这个连接上有一个新的消息来临，同时向与此队列连接的 FS 发送时钟请求消息；

（4）新的消息将该队列的时间戳更新，转（1）。

相对于 CMB 算法， NMTRT-CMB 算法中空消息数量明显下降，性能得到了明显的提高。通过实验表明，优化后带来的模拟精度损耗非常小，可忽略。

3.4 高可配

BDSim 模拟框架是基于组件思想开发的，这使得高可配成为可能。然而，虽然高可配能够带来模拟模型构建的灵活性，但是也带来了沉重的配置负担。使用者需要按照拓扑结构实现所有模块的通信连接。当模拟组件达到上千规模时，配置操作变成了几乎不可完成的任务。为了减轻使用者的配置负担，

BDSim 提供了两种不同的配置方法：文件配置和图形化配置。

3.4.1 文件配置

文件配置意味着用文件的方式描述拓扑结构，配置内容包括：目标模型有几种组件，每种组件的个数，每个组件端口和其它端口互连信息等。以 Router 组件为例，首先声明 Router 组件对象，定义初始化函数，定义 Callback 函数等。然后描述组件的端口属性，支持 2D-Mesh 结构的路由器需要五个端口：四个方向端口和一个与 Core 组件连接的端口。之后是调试命令注册，将组件支持的用户命令注册至框架中，运行时通过框架来识别和执行。最后是初始化一些参数，如路由的坐标信息等，配置格式如图 7。在实现单个组件配置后是对整个拓扑结构的配置。图 8(a) (b)分别是相关两个配置文件：Topology 和 Parameter。Topology 文件描述了整个模拟目标系统的拓扑结构，以端口之间的连接方式描述。Parameter 文件记录着组件的初始化参数，示例中主要描述每个 Router 的坐标。

3.4.2 图形化配置

文件配置需要使用文件描述语言编写配置文件，繁琐且容易出错。为此，BDSim 提供给使用者更为便捷的配置方式——图形化配置。组件库中已有的组件和新添加的组件会以图标的方式展示给用户，同时提供组件的各个属性信息供使用者查看。使用者只需要将需要的组件拖拽至工作区，然后将需要互连的端口通过引线连通即可完成模拟系统的配置，方便快捷。

3.4.3 配置流程

BDSim 配置流程如图 9 所示。首次配置时，在

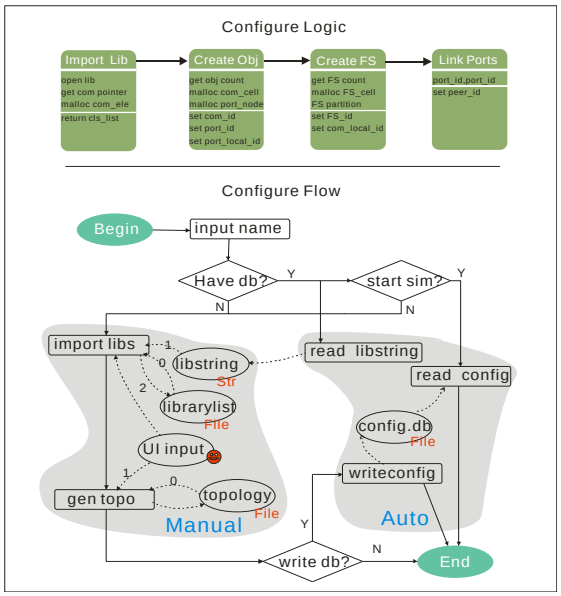


图 9 BDSim 配置流程

拓扑结构和组件种类、数量确定之后，通过图形化操作界面或编写文件描述符的方法生成目标系统的配置文件。框架对配置文件进行解析，生成二进制配置数据库。在之后的运行中，可以直接使用此配置数据库，无需再次解析生成。如若修改已生成二进制配置文件，则选择启动后重新配置。整个配置数据库以树形结构存储，如图 10 所示，先根据配置形成相应数量的 FS， FS 是并行的最小单元。然后将每个功能组件挂载到相应 FS 之上，一个 FS 可以挂载多个功能组件，同一个 FS 之上的功能组件之间串行执行。最后是端口配置。配置完成后，相当于完成了拓扑结构的初始化。使用树形结构，可以快速定位模块的位置，实现不同组件之间的通信。

通过这种配置方式，使用者可以按照自己的需求，任意配置并行粒度，方便快捷地构建目标模型。

4 实验及分析

实验部分主要对框架中的同步和通信优化算法进行测试和分析。

4.1 实验环境

实验基于 Intel 多核处理器平台，实验环境及参数配置如表 1 所示。

表 1 实验平台配置

软件平台	
操作系统	CentOS 5.10
编译器	GCC 4.6.1

硬件平台	
CPU	4 块 Intel Xeon(R) E7420 CPUs
	每块 4 核，共 16 个物理线程
L1 Cache	L1 Cache 私有， 每个核有 32KB I-Cache 和 32KB D-Cache
L2 Cache	L2 Cache 共享，每个 CPU 3MB，共 12MB
LLC	LLC 共享，每个 CPU 8MB，共 32MB

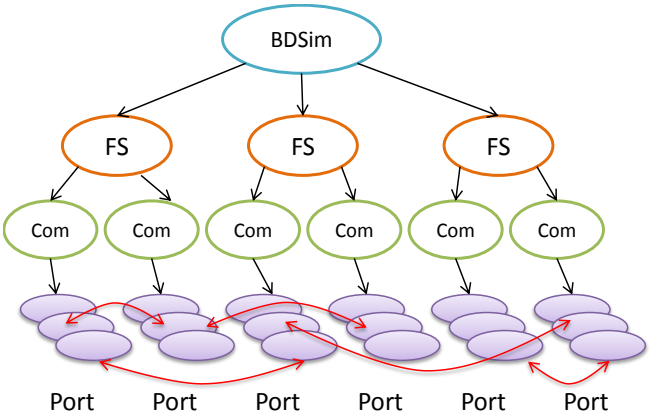


图 10 BDSim 树形配置结构

4.2 实验模型

本实验选择基于 2D-Mesh 片上网络结构的众核结构作为模拟模型，配置 VCore、Memory、Router 三种组件，Mesh 大小为 4x4。通过并发模拟 VCore、Memory 之间的数据传输，控制网络消息流量大小，可以达到充分测试的目的。

VCore 为虚拟核组件，使用 Trace 生成访存消息。Router 为片上网络路由组件，根据 X-Y 静态路由算法向相邻的四个方向传递消息。考虑到 Router 可能会有消息发往自己（连接 VCore 的情况），所以为 Router 保留了发往自己的端口。每个 Router 组件有 10 个端口，端口的编号顺序为上、下、右、左，最后是连接 VCore 的端口。 图 11 为由 16 个 Router 组件组成的 4x4 2D-Mesh 网络拓扑结构图。

Router 的拓扑位置以坐标的形式标记，坐标轴以左上角为原点，横坐标方向水平向右递增，纵坐标方向垂直向下递增。消息由 VCore 根据 Trace 生成并发送至 Mesh 网络中，通过控制 Cycle 内注入消息数量来控制网络流量。测试程序如表 2 所示。12 Packets/Cycle 和 110 Packets/Cycle 是根据测试程序发送包的总数以及测试程序运行后的总 Cycle 数计算得到，分别代表不同程度的流量大小。

表 2 测试程序分析

测试程序	程序特点	流量大小
router_small	低网络流量	12 packets/cycle
router_big	高网络流量	110packets/cycle

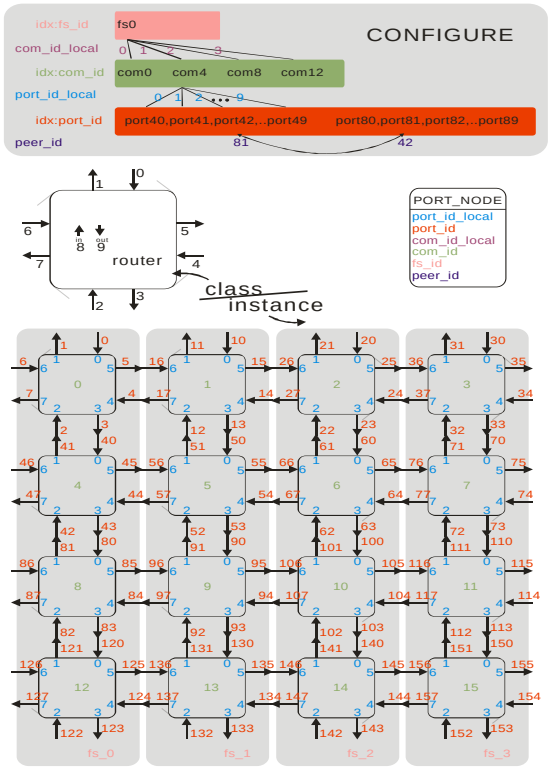


图 11 4x4 2D-Mesh 网络拓扑配置结构图

4.3 实验结果及分析

4.3.1 非阻塞无锁通信性能测试

非阻塞无锁队列与阻塞有锁队列相比，对于共享队列的访问不需要加锁。因而，共享队列的访问竞争越激烈，非阻塞无锁队列的优势会更加明显。利用通信量比较小的测试用例 Router_Small 分别测试阻塞有锁和非阻塞无锁两种情况下所用时间，测试结果如表 3 所示。

当线程数比较少且通信量比较小时，对共享队列的访问竞争程度相对较小，非阻塞无锁与阻塞有锁两种实现方式所用的时间是相差不大。当线程数比较多时，对于同一个 FS 中的共享队列的访问竞争程度变大，在阻塞有锁的情况下，线程阻塞等待的时间将会变大。非阻塞无锁通信相对于阻塞有锁通信效率提高的比例如图 12 所示。当线程数为 16 时，效率的提高约为 7.5%。利用通信量比较大的测试用例 Router_big 分别测试阻塞有锁和非阻塞无锁两种情况下测试用例所用时间，测试结果如表 4 所示。线程数越多共享队列的竞争越大，所以，随着线程数的增加，非阻塞无锁相对于阻塞有锁的效率的提高会越来越明显。图 13 展示了在 Router_Big 测试用例下，非阻塞无锁通信相对于阻塞有锁通信的效率的提高比。当线程数为 16 时，效率提高 10% 左右。

表 3 Router_Small 测试结果（ms）

线程数目	阻塞有锁	非阻塞无锁	时间差	提高比
1	974.209	968.845	5.364	0.50%
2	618.996	615.825	3.171	0.51%
4	492.711	489.768	2.943	0.60%
8	475.023	463.481	11.542	2.40%
16	444.538	411.354	33.184	7.50%

表 4 Router_Big 测试结果（ms）

线程数目	阻塞有锁	非阻塞无锁	时间差	提高比
1	10800.414	10798.474	1.94	0.02%
2	6504.203	6450.324	53.879	0.80%
4	5234.491	5184.320	893.118	1.00%
8	4677.888	4569.724	108.164	2.00%
16	4673.814	4199.697	474.117	10.1%

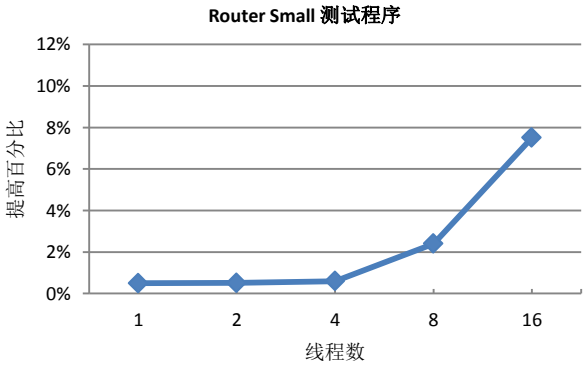


图 12 Router_Small 测试程序下
非阻塞无锁队列相对于阻塞有锁队列效率提高比

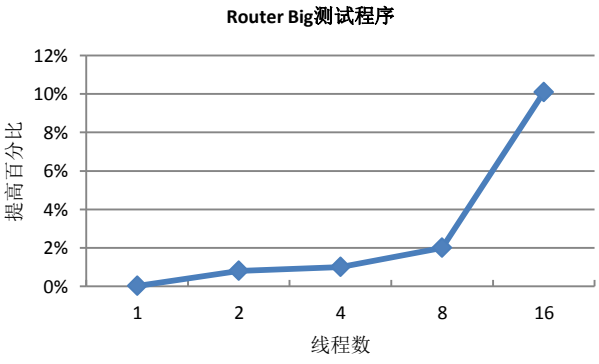


图 13 Router_Big 测试程序下
非阻塞无锁队列相对于阻塞有锁队列效率提高比

4.3.2 NMTRT-CMB 算法性能测试

实验以 4 线程为例，测试在 NMTRT-CMB 算法下，空消息的发送量的减少比以及性能损耗。如表 5 所示，随着每个 FS 绑定组件数的增加，在优化前需要同步的逻辑处理单元的数目随之增加，因而用于同步的空消息的数量也不断增加。优化后，随着组件数的增加，空消息的发送量得到了明显的减少，变化趋势如图 14 所示。

衡量并行效果的另一个重要标准是精度损耗。NMTRT-CMB 算法将处在同一个 FS 中的组件近似认为它们的局部时钟一致，势必会对精度产生一定的影响。本实验对精度损耗也做了充分的评估，分别利用通信量比较小和通信量比较大时的测试用例测试优化后相对于传统 CMB 算法的精度损耗。

实验表明，在两种测试用例情况下精度的损耗都是非常小的。结果如表 6 和图 15 所示。即便是损耗比较大的 Router_Small 测试用例，精度的损耗最大也只有 0.7%左右。线程数较少时，逻辑处理单元的时钟推进可以依靠收到的普通消息，此时最后程序运行的 cycle 数跟逻辑处理单元的延迟有一定的关系。而线程数较多时逻辑处理单元时钟的推进会依靠收到的空消息，测试程序运行的 cycle 数跟同步用到的 Lookhead 和逻辑处理单元的延迟都有一定的关系，在 Router 测试用例中 Lookhead 比逻辑处理单元的延迟要小，所以线程数多时精度损耗反而较小。

表 5 CMB 与 NMTRT-CMB 算法空消息数量对比

组件数	CMB	NMTRT-CMB	减少比
1	7621	7454	2.2%
2	83540	15863	81.0%
4	439685	36917	91.6%

表 6 不同线程下 NMTRT-CMB 算法精度损耗

线程数	Router_small	Router_big
16	0.01%	0.01%
8	0.19%	0.02%
4	0.50%	0.10%
2	0.70%	0.20%

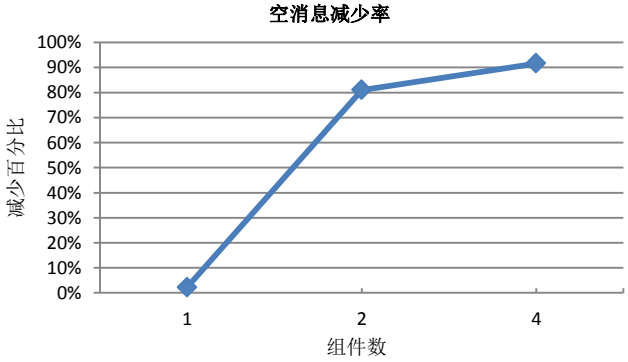


图 14 不同组件数时空消息减少率变化

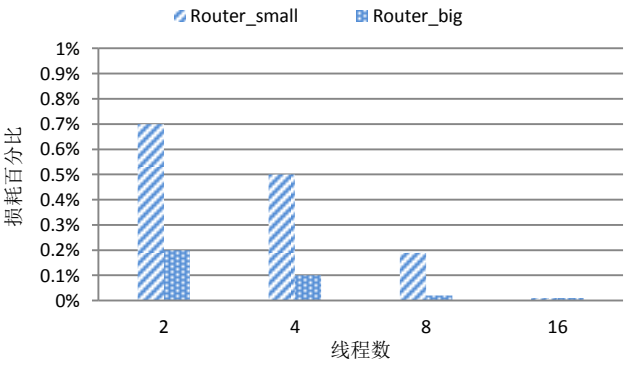


图 15 不同线程数时 cycle 数的损耗

5 大数据性能评估举例

随着大数据的兴起，“存储墙”的问题越来越严重，对于如何提高访存带宽，实时处理高通量的数据成为学术界和工业界研究的热点。本实验目的是评估 MACT（访存控制表）在大数据应用体系结构中对访存的性能影响，验证 BDSim 模拟框架如何在大数据处理模拟方面发挥作用。

MACT 结构实现于核中，在核发出访存请求时对访存事件进行收集、分类和批量处理，能够减少网络拥塞、提高带宽利用率。如图 16 所示，MACT 结构内置两个表 Rmact 和 Wmact，分别用于读/写记录。表是一个队列，由一个数组和头尾指针构成。队列每一项包含标志读/写记录的 Bitmap、需要结算的时间 Deadline 以及该项的基址 Base addr。其中 Bitmap 中共有 1024 位，每一位代表了一个字节地址的读/写情况。当接收到一条访存请求时，MACT 行为如下：

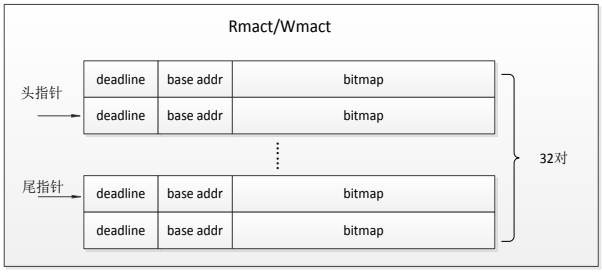


图 16 Rmact/Wmact 结构示意图

（1）读数据。清除 Rmact 中相关表项，并将表中 Bitmap 为 1 的地址的数据发送给总线;

（2）写数据。直接返回 ACK;

（3）读/写地址。将该地址插入到 Rmact/Wmact 中，如果现有表项中存在该地址的基址记录，则在此表项的 Bitmap 中将对应 Bit 位置 1 并记录消息内容，如有重复置 1 的情况则需要同样需要记录多个消息内容，最后返回 ACK。如表项中不存在该地址的基址记录，且表已满，则向总线发送 NACK 消息，若不满则增加一项基址为该地址基址一致的一项，用当前周期加上固定延时（实验中设置为 16cycle）作为此新项的 Deadline，并发送 ACK 消息。

5.1 组件

模拟模型包括：XBAR、VCore、MCU 三类组件，如图 17。XBAR 即为 Crossbar 总线，本例中使用 XBAR 组件搭建三级总线。VCore 代表虚拟核，因本实验主要研究访存性能，所以无需模拟完整的核内结构，可以降低模拟的复杂度。VCore 组件需要实例化两类：带 MACT 结构的和不带 MACT 结构的。MCU 为存储控制单元。实验 Benchmarks 为大数据测试程序的访存 Trace，虚拟核解析 Trace 并在一定的时序控制下执行相应的访存事件。

5.2 拓扑结构

本例中，以 128 核的众核结构作为实验对象，目标模型如图 17 所示。每 8 个核连接到一个三级总线，形成 Core Group。每个二级总线挂载 4 个三级总线，4 个二级总线连接到一级总线，形成 128 核的多级总线互连结构。4 个 MCU 分别挂载到 4 个二级总线之上。

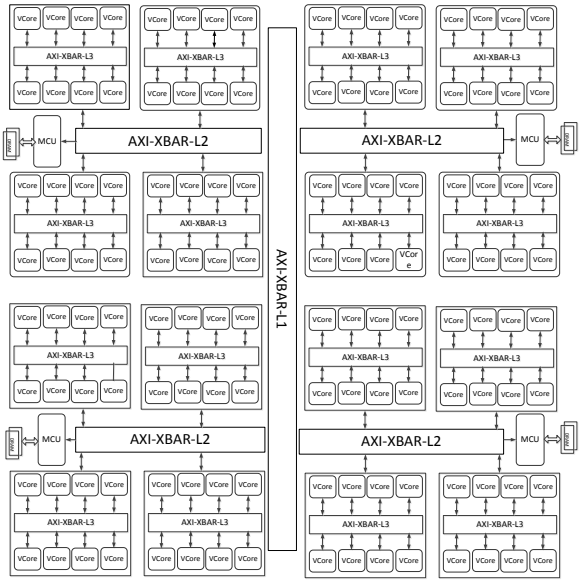


图 17 多级总线拓扑结构图

表 7 传统结构测试结果

	pin_search560	wholeWC	wholeTS
整体执行时间(cycles)	19162908	70877323	502965521
单个消息平均延迟(cycles)	16.58999266	18.64520255	17.25216839
访存带宽比(次/cycle)	0.345696854	0.411295486	0.077403161
访存请求(次)	6624557	29151523	38931121

表 8 包含 MACT 结构测试结果

	pin_search560	wholeWC	wholeTS
整体执行时间(cycles)	8343595	30134820	494104785
单个消息平均延迟(cycles)	29.04942188	51.27353451	68.15464703
访存带宽比(次/cycle)	0.491548427	0.620428959	0.060601146
访存请求(次)	4101281	18696515	29943316

表 9 加速比统计

	pin_search560	wholeWC	wholeTS	average
整体执行时间	2.30	2.35	1.02	1.89
单个消息平均延迟	1.75	2.75	3.95	2.82
访存带宽	1.42	1.51	0.78	1.24
访存请求	1.62	1.56	1.30	1.49

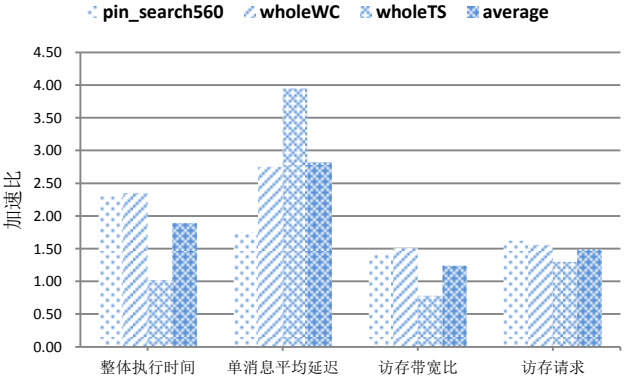


图 18 加速比统计图

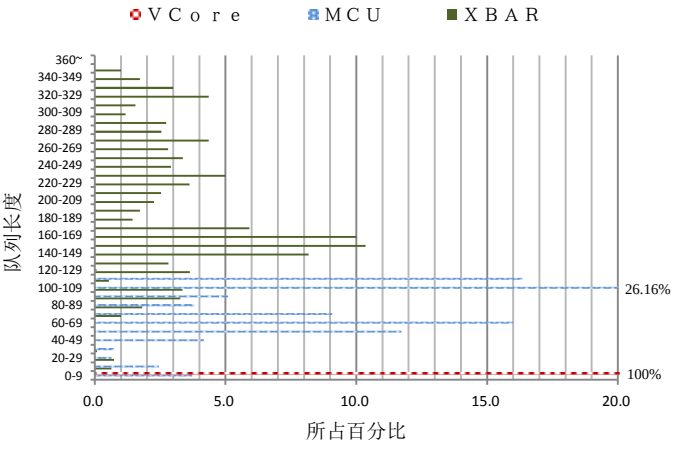


图 19 消息队列长度统计图

5.3 测试环境及结果分析

测试环境如表 1。

实验比较了包含 MACT 结构和传统结构的程序的执行效率。以搜索引擎程序(pin_search560)、WordCount 程序(wholeWC)及 Terasort 程序（wholeTS）三个大数据应用生成的访存 Trace 作为本次实验的 Benchmarks。Trace 大小分别为 280MB、1.3GB、1.9GB。VCore 组件按照访存 Trace 的时序标签发出访存请求。因为本实验是对并行框架的测试和验证，所以使用 Trace 和运行真实的 Benchmarks 不会对框架的测试结果产生影响。实验统计数据如表 7、表 8、表 9 三表及图 18 所示，使用 MACT 结构的运行 Cycle 值明显下降，单消息延迟增加，访存带宽比提高以及请求次数降低。

实验结果表明，通过 MACT 对访存请求的收集和批量处理，大大减少了访存消息的数量，可以有效提高带宽的利用率。由于访存消息统一在 MACT 中收集并等待一定 Cycles 才继续发送，所以延迟会相应增大，但总体执行时间得到了有效的缩减。可见，MACT 结构可以通过对访存事件的收集和批量处理，有效减少网络拥塞，增加带宽利用率，缩短执行时间，相对于传统结构具有更强的性能。

关于运行过程中存储空间的消耗，图 19 展示了运行时消息队列长度统计结果。可知，最大队列长度不会超过 360，单消息大小为固定 32B。在图 17 的配置下，128 核与三级 XBAR 互连，共 128 对队列。二三级 XBAR 互连共 16 对队列，一二级 XBAR 互连共 4 对队列。MCU 与 XBAR 互连共 4 对队列。整个配置共需 2*(128+16+4+4)=204 个队列。最大所需内存 204*32B*360≈2.24MB 大小。可知，即使千核量级配置下，当前通用处理器也完全可以承受。

综上所述，实验以评估 MACT 结构在大数据处理体系结构中的作用为例，验证了 BDSim 框架的有效性。通过实验过程和结论可以看出，BDSim 适用于构建大规模众核体系结构模拟平台，能够在大数据处理研究过程中发挥重要的评估作用。

6 结 论

大规模并行模拟已经成为大数据体系结构研究的主要方法。但目前的模拟技术却不能满足大数据体系结构研究的需求，主要体现在模拟速度慢、配置过程复杂、可扩展性差等方面。针对此，本文提出了 BDSim 并行模拟框架，支持大规模并发模拟。该并行模拟框架具有以下优势：首先，模拟框架基于离散组件化开发模式，功能模块以组件形式加载到模拟框架中运行，提高了模型开发的自由性及组件的复用性。其次，使用高效的非阻塞无锁消息通信机制，加快了消息处理速度，16 线程配置下，执行效率提高约 10%左右。再次，提出基于 CMB 的 NMTRT-CMB 同步算法，4 线程 16 组件配置下，减少控制消息数量 90%以上。同时，为了满足不同的同步需求，模拟框架也支持粗粒度同步方法，使用者可以随意配置同步粒度以满足对模拟速度的需求。最后，模拟框架提供了灵活方便的配置方法，文件或图形化配置方法，提高了框架的易用性。

通过实验分析和评估举例表明，BDSim 并行模拟框架以其灵活方便的搭建方式和快速精确的执行过程，符合大规模并行系统的模拟需求，适用于大数据处理体系结构的评估和研发。

7 未来工作

为了提高模拟框架的速度、精确度和易用性，未来研究工作主要集中在三个方面。

首先，提高模拟框架的模拟速度。模拟速度一直是困扰软件模拟发展的难题。针对大规模数据处理体系结构的模拟，更需要速度的保证。围绕此问题，加速工作主要从两个方面展开：从框架角度，通过进一步优化通信和同步算法，提高框架的执行效率；从组件角度，通过使用动态二进制翻译机制、指令 Trace Cache 以及采样等技术加速单个组件的执行速度。

其次，提高模拟框架的精确度。模拟精确度越高就会使模拟行为和结果更接近于真实硬件，能够更加准确地指导体系结构的研究。目前，以本研究的模拟框架为基础，千核级处理器的 RTL(Register Transfer Level, 寄存器传输级)仿真模型正在开展。之后，框架中的功耗、时延、面积等工艺参数信息会与 RTL 模型进行对比调试，逐步调优 BDSim 模拟框架的精确度。

最后，开发完善的组件库。组件库使已存在的组件被无限次重复使用成为可能。因此，需要继续开发不同功能的组件或是整合现有的功能准确、使用度广的组件添加至组件库，降低并行模拟器开发的复杂性，形成一个功能丰富的模拟开发平台，方便研发人员快速搭建目标模型。

参考文献

[1] 李国杰. 大数据对计算机系统的挑战. CNCC 大数据论坛, 长沙, 2013

[2] Wang Yuanzhuo, Jin Xiaolong, Cheng Xueqi. Network Big Data: Present and Future. Chinese Journal of Computers, 2013, 36(6):1125-1138 (in Chinese)
(王元卓, 靳小龙, 程学旗. 网络大数据: 现状与展望. 计算机学报, 2013, 36(6):1125-1138)

[3] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, Babak Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS`12). London, UK, 2012:37-48

[4] Tianshi Chen, Qi Guo, Ke Tang, Olivier Temam, Zhiwei Xu, Zhi-Hua Zhou, Yunji Chen. ArchRanker: a ranking approach to design space exploration. Proceeding of the 41st annual international symposium on Computer architecuture (ISCA`14). Minneapolis, USA, 2014:85-96

[5] Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao, Zhen Jia, Yingjie Shi, Shujie Zhang, Chen Zheng, Gang Lu, Kent Zhan, Xiaona Li, Bizhu Qiu. BigDataBench: A big data benchmark suite from internet services. IEEE 19th International Symposium on High Performance Computer Architecture (HPCA`14). Orlando, USA, 2014:488-499

[6] Hamid Reza Ghasemi, Nam Sung Kim. RCS: runtime resource and core scaling for power-constrained multi-core processors. Proceedings of the 23rd international conference on Parallel architectures and compilation (PACT`14). Edmonton, Canada, 2014:251-262

[7] Abeyratne Nilmini, Das Reetuparna, Li Qingkun, Sewell Korey, Giridhar Bharan, Dreslinski Ronald G., Blaauw David, Mudge Trevor. Scaling towards kilo-core processors with asymmetric high-radix topologies. IEEE 19th International Symposium on High Performance Computer Architecture (HPCA`13). Shenzhen, China, 2013:496-507

[8] Guthmuller E., Leti CEA Grenoble France, Miro-Panades I., Greiner A.. Architectural exploration of a fine-grained 3D cache for high performance in a manycore context. 2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI`13). Istanbul, Turkey, 2013:302-307

[9] Kapil Dev, Abdullah Nazma Nowroz, Sherief Reda. Power mapping and modeling of multi-core processors. 2013 IEEE International Symposium on Low Power Electronics and Design (ISLPED`13). Beijing, China, 2013:39-44

[10] Filippo Sironi, Martina Maggio, Riccardo Cattaneo, Giovanni F. Del Nero. ThermOS: System support for dynamic thermal management of chip multi-processors. 22nd International Conference on Parallel Architectures and Compilation Techniques (PACT`13). Edinburgh, UK, 2013:41-50

[11] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. SIGARCH Computer Architecture News, 2011, 39 (2):1-7

[12] Avadh Patel, Furat Afram, Shunfei Chen, Kanad Ghose. MARSS: a full system simulator for multicore x86 CPUs. Proceedings of the 48th Design Automation Conference (DAC`11). 2011:1050-1055

[13] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. IEEE Computer, 2002, 35(2):50–58

[14] Miller, J.E, Kasture, H, Kurian, G, Gruenwald, C, Beckmann, N, Celio, C, Eastep, J, Agarwal, A. Graphite: A distributed parallel simulator for multicores. 2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA`10). Bangalore, India, 2010: 1-12

[15] Nikolaos Hardavellas, Stephen Somogyi, Thomas F. Wenisch, Roland E. Wunderlich, Shelley Chen, Jangwoo Kim, Babak Falsafi, James C. Hoe, Andreas Nowatzyk: SimFlex: a fast, accurate, flexible full-system simulation framework for performance evaluation of server architecture. SIGMETRICS Performance Evaluation Review, 2004, 31(4): 31-34

[16] A.F.Rodrigues. The structural simulation toolkit, http://www.cs.sandia.gov/sst. 2007

[17] M. Lis, P. Ren, M. Cho, K. Shim, C. Fletcher, O.Khan, and S. Devadas. Scalable, accurate multicore simulation in the 1000-core era. IEEE International Symposium on Performance Analysis of Systems and Software

(ISPASS`11). Austin, USA, 2011:175-185

[18] Jianwei Chen, Murali Annavaram, Michel Dubois. SlackSim: a platform for parallel simulations of CMPs on CMPs. SIGMETRICS Performance Evaluation Review , 2009, 37(2):77-78

[19] Daniel Sanchez and Christos Kozyrakis. ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems. Proceedings of the 40th annual International Symposium in Computer Architecture(ISCA`13). Tel-Aviv, Israel, 2013: 475-486

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

[21] Miller J.E., Kasture H., Kurian G., Gruenwald C., Beckmann N., Celio C., Eastep J., Agarwal A.. Graphite: A distributed parallel simulator for multicores. IEEE 16th International Symposium on High Performance Computer Architecture (HPCA`10). Bangalore, India, 2010:1-12

[22] Derek Chiou, Dam Sunwoo, Joonsoo Kim, Nikhil A. Patil, William Reinhart, Darrel Eric Johnson, Jebediah Keefe, Hari Angepat. FPGA-Accelerated Simulation Technologies (FAST): Fast, Full-System, Cycle-Accurate Simulators. Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO`07). Chicago, USA, 2007: 249-261

[23] Avadh Patel, Furat Afram, Shunfei Chen, Kanad Ghose. MARSS: a full system simulator for multicore x86 CPUs. Proceedings of the 48th Design Automation Conference (DAC '11). New York, USA, 2011:1050-1055

[24] J. Emer, P. Ahuja, E. Borch, A. Klauser, Chi-Keung Luk, S. Manne, S. S. Mukherjee, H. Patil, S. Wallace, N. Binkert, R. Espasa, and T. Juan. Asim: A performance model framework. IEEE Computer, 2002, 35(2):68–76

[25] Manifold Project. http://manifold.gatech.edu. 2011

[26] Jiao Shuai, Xu Weizhi, Tang Shibin, Fan Dongrui, Sun Ninghui. PartitionSim: A Parallel Simulator for Many-Cores. Chinese Journal of Computers, 2011, 34(11):2084-2091(in Chinese)
(焦帅, 徐卫志, 唐士斌, 范东睿, 孙凝晖. PartitionSim: 一个面向众核结构的并行模拟器. 计算机学报，2011, 34(11):2084-2091)

[27] J. Engblom. Simics Accelerator. Virtutech White Paper. 2009

[28] X. Martorell, N. Smeds, R. Walkup, J. R. Brunheroto, G. Almási, J. A. Gunnels, L. DeRose, J. Labarta, F. Escalé, J. Giménez, H. Servat, J. E. Moreira. Blue Gene/L performance tools. IBM Journal of Research and Development , 2005, 49(2):407-424

[29] Gengbin Zheng, Gunavardhan Kakulapati, Kale, L.V. BigSim: a parallel simulator for performance prediction of extremely large parallel machines. Proceedings of the 18th International Parallel and Distributed Processing Symposium(IPDPS`04). Santa Fe, USA, 2004: 78-87

[30] Liu Yi, Zhi Yuzhe, Zhang Xin, Li He, Jiao Lin, Zhang Peng, Su Yangming, Ni Zehui, Qian Depei. SimHPC: An Execution-Driven Simulator for High-Performance Computers. Chinese Journal of Computers, 2013, 36(4):738-746 (in Chinese)
(刘轶, 支予哲, 张昕, 李鹤, 焦林, 张鹏, 苏阳明, 倪泽辉, 钱德沛. SimHPC：一种基于执行驱动的大规模并行系统模拟器. 计算机学报，2013, 36(4):738-746)

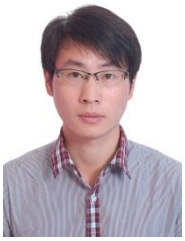
[31] Steven K. Reinhardt, Mark D. Hill, James R. Larus, Alvin R. Lebeck, James C. Lewis, David A. Wood. The Wisconsin Wind Tunnel: virtual prototyping of parallel computers. Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and modeling of computer systems(SIGMETRICS'93). New York, USA, 1993, 8(4):12-20

[32] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, Babak Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS`12). New York, USA, 2012, 40(1):37-48

[33] Richard M. Fujimoto. Parallel Discrete Event Simulation. Communications of the ACM, 33(10):30-53

[34] K. Hwang and F. A. Briggs. Computer Architecture and Parallel Processing. New York: McGraw-Hill, 1984

[35] L. Lamport. Specifying Concurrent Program Modules. ACM Transactions on Programming Languages and Systems, 1983, 5(2): 190-222



Li Wen-ming, born in 1988, Ph.D. Candidate. His research interests include high throughput computing architecture and software simulation.

Ye Xiao-chun, born in 1981, Ph.D., associate professor. His research interests include high-performance computer architecture and software simulation.

Zhang Yang, born in 1981, Ph.D. Candidate. His research interests include high throughput computing architecture, real-time system.

Song Feng-long, born in 1980, Ph.D., senior engineer. His research interests include high-performance computer architecture, and on chip memory systems.

Wang Da, born in 1981, Ph.D., associate professor. Her research interests include IC testing， analysis， and microarchitecture design.

Tang Shi-bin, born in 1986, Ph.D.. His research interests include high performance computer architecture, debugging parallel program and on chip memory system.

Fan Dong-rui, born in 1979, Ph. D., professor. His research interests include low-power design and processor micro-architecture.

Background

Large-scale parallel simulation is becoming an irreplaceable researching method in big data application and many-core architecture, since it can achieve high speedup compared to sequential simulation. However, the parallel simulation techniques currently cannot meet requirements of researching, mainly reflected in low simulation speed, complicated configuration, scalability and so on. Especially to simulate large scale architecture, such as 1000-core scale system.

In this paper, BDSim, a high configurable parallel simulation framework for big data simulation, is proposed. The framework is based on the thought of component. Function modules exist as components with uniform interface. Users can adjust the parallel granularity according to the loadings and simulation requirements arbitrarily.

This paper proposes optimal non-block lock-free communication method to improve communication efficiency and also presents NMTRT-CMB (null message timestamp require token CMB) synchronization algorithm based on CMB synchronization algorithm. Proved by many-core architecture based on 2D-Mesh NOC under different parallel scalability, NMTRT-CMB can reduces almost 90% null message running with 16 threads compared to CMB. And non-block lock-free communication method can help improving simulation speedup with about 10% compared to traditional communication on lock-based method.

This work is under the support of the National Grand Fundamental Research 973 Program of China under No. 2011CB302501, National High-Tech Research & Development Program of China under No. 2012AA010901 and the National Natural Science Foundation of China under No. 61100013, 61100015, 61173007, 61202059, 61204047, 61221062, 61332009.