一个针对并行模拟引擎的性能评测实例

吴志敏 吕慧伟 陈明宇
(中国科学院计算技术研究所 北京 100190)

摘 要 SimK是由中科院计算所体系结构国家重点实验室开发的一个并行离散时间模拟引擎。基于已经发布的SimK1.0版本,对任务划分及同步推进阻塞控制进行了功能扩展,开发了SimK的1.1版本。同时由于缺乏一个专门对SimK模拟性能评测的Benchmark以及全面的评测结果,首先讨论了并行模拟引擎Benchmark的设计准则,之后介绍了开发的Benchmark-PassBall,并且使用它对SimK的强弱扩展性、组件负载不均衡情况下的强扩展性进行了评测,同时对比了组件负载不均衡和均衡情况下的加速比,探讨了模拟计算量的变化对模拟加速比的影响,并讨论了Benchmark的适用性。通过实验讨论得出:a)PassBall可以作为并行模拟引擎SimK性能评测的Benchmark,亦可用于其他并行模拟引擎性能的评测;b)SimK具有良好的强弱扩展性;c)负载平衡和模拟计算量都会对并行模拟加速比产生影响。

关键词 并行模拟,模拟引擎,扩展性,SimK,Benchmark
中图法分类号 TP302 **文献标识码** A

Parallel Benchmark for Evaluating Parallel Simulation Engine

WU Zhi-min LV Hui-wei CHEN Ming-yu
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract SimK is a parallel discrete event simulation engine developed by state key laboratory of computer architecture in institute of computing technology, chinese academy of sciences. Based on the released SimK-1.0, we extended the function of task partition and the blocking controlling in the process of synchronization. We released version 1.1 of SimK. On the other hand, as it lacks a benchmark to specifically SimK simulation performance and there is no comprehensive evaluation data, we first proposed the rules of development Benchmark for parallel simulation engine. Then we introduced the example "PassBall". We used it to do the evaluation of SimK on the weak and strong scalability, as well as the strong scalability in unbalanced workload condition. Then we compared the speed-up ratio between the balanced workload and unbalanced workload condition in strong scalability test. The influence of simulated computing workload on speed-up ration from was also explored. We also discussed the applicability of the Benchmark. It can be concluded from our experiments as follows: a) our example "PassBall" is available to be the benchmark for SimK, as well as other parallel simulation engine. b) SimK has favorable strong and weak scalability. c) Both the load balance and the simulated computing workload will have effect on the speed-up ratio.

Keywords Parallel simulation, Simulation engine, Scalability, SimK, Benchmark

1 引言

并行模拟通过开发模拟模型潜在的并行性,突破模拟在模拟时间和模拟内存方面的限制,适用于对模拟时间或者规模要求较高的模型。通过增大并行度,模型模拟的加速比将会显著提升。并行模拟技术是解决超大规模计算机系统研究的重要手段。

基于此,中科院计算所体系结构国家重点实验室开发了一个开源的并行离散时间模拟引擎SimK1.0[14],并使用它开发了曙光5000的胖树网络模拟器[7]以及实现了GodSon-T[9]众核模拟器的并行化[8]。我们在此工作的基础上新增了部分功能,形成了SimK1.1,并发布在文献[1]。

与此同时,对于并行模拟引擎SimK的性能评测并不完善,因为基于SimK开发的真实模拟器负载波动很大,不适合作为模拟引擎自身性能的评测工具。因此我们设计实现一个针对并行模拟引擎进行多方面性能评测的Benchmark-PassBall。并行模拟引擎的Benchmark设计原则如下:

- Benchmark应基于并行模拟引擎开发,模拟模型应具有具体的拓扑结构。
- Benchmark模拟的模型应该具有良好的并行性。

到稿日期:2012-10-19 返修日期:2012-12-12 本文受NSFC国家自然科学基金项目(61272132),NSFC国家杰出青年科学基金(60925009),NSFC创新研究群体科学基金(60921002),973项目(2011CB302502),中科院战略性先导专项(XDA06010401)资助。
吴志敏(1989-),男,硕士生,主要研究领域为高性能计算,E-mail:wuzhimin@ict.ac.cn(通信作者);吕慧伟(1983-),男,博士生,主要研究领域为高性能计算;陈明宇(1972-),男,博士,研究员,主要研究领域为计算机体系结构、高性能计算。

• Benchmark 设计的任务量及任务划分应该可控，从而满足多方面的性能测试。

• 并行程序只有随着并行计算机的规模增大效率按比例增加，才有意义。这就意味着我们需要足够的任务来使所有 cpu 或节点都处于忙碌的状态。任务量小的情况下采取多线程增加的同步和调度等开销将超过并行节省的开销，达不到效果且浪费了资源。因而 Benchmark 需要保证一定的计算粒度，即保证一定的功能模拟开销。

• Benchmark 应该提供具体的能反映性能的评测指标。

我们使用 PassBall 进行了 SimK 的强扩展性、弱扩展性、不平衡负载下扩展性以及随机通信的实验和评测，并对 Benchmark 中的功能模拟开销设定进行了简要的分析。验证了 PassBall 可以很好地完成并行模拟引擎各方面性能的测试。同时，其易于实现，设计思路具有通用性，可以用于比较不同的并行模拟引擎。

本文第 2 节对 SimK 的结构、SimK1.1 新增功能及基于 SimK1.1 开发流程进行简介；第 3 节重点描述针对并行模拟引擎 Benchmark 的设计；第 4 节列出使用 PassBall 对 SimK 进行多方面评测的结果及分析；最后是总结。

2 并行模拟引擎 SimK 简介

SimK 是一个并行离散事件模拟(PDES, Parallel Discrete Event Simulation)[10]引擎，其基于现有积累的模拟器实现及优化技术提取出一个通用框架，名称取自 Simulation Kernel，目前发展到 1.1 版本。实现中所采用的算法是 CMB 保守同步算法，并针对某些并行机体系结构和存储模型作了一些优化。SimK 的结构与操作系统内核有着众多相似之处，它提供了任务部署、缓冲管理、逻辑进程调度、通信和同步等基本服务，其优点是：易于开发、调试和调优；易于维护；为遗留代码重用提供可能性；支持多形态模块的生成。它使得并行模拟器的开发者无需考虑不同并行环境下模块的协作部署问题，只需考虑模拟的功能性问题，同时模拟器开发者可在 SimK 之上通过很少量的编码对目标系统建模，从而得到一个自动并行、效率较高的模拟器[15]。详细的 SimK 并行模拟引擎的各部分的设计及结构请参见文献[14]。SimK 采用了 MPI+Pthreads 两级并行的模式，适用于 SMP 集群，目前只支持 x86_64 平台。

在之前的工作中，并行模拟引擎 SimK 经过不断完善，形成了一个比较完整的版本 1.0 并发布在了开源网站 sourceforge.net。在此基础上，我们对一些功能进行了扩展，形成了 SimK 的 1.1 版本并发布之。本节将对并行模拟引擎SimK1.1 新增功能进行简述，主要包括以下两部分。

2.1 SimK1.1 新增功能-任务划分策略扩展

在开发并行模拟器的过程中，任务划分策略的设计是非常关键的一部分，将直接影响到并行效率。并行模拟的任务划分需要满足以下两方面的要求：a)所有处理器上的负载尽可能相等；b)不同处理器上模块之间的通信开销总和最小。因而任务划分可转化为无向赋权图的最优划分问题，这里每一个被模拟组件都可被定义为图中的一个节点，整个图代表着被模拟系统的拓扑结构，任务划分即将所有模拟组件划分到不同的节点/进程上运行。虽然这是一个 NP 完全问题[11]，但目前已经存在大量的启发式算法来对其求近似最优解[3,12]。

SimK1.0 中任务分配策略依赖于外部图划分库 Chaco[4,13]。调用其提供接口进行后台任务划分，SimK1.1 提供了用户定义/配置文件任务划分策略。形成了 3 种任务划分模式：内部手动分配(MANUAL-INSIDE)、Chaco 划分(AUTO)、配置文件 &Chaco 划分(HALF-MN-HALF-AUTO)。详情参见文献[1]中的 SimK_Programmer_Manual。

2.2 SimK1.1 新增功能-组件推进阻塞控制

在 SimK1.0 运行模拟的过程中，当组件处理完自身消息后无新消息可以继续处理时，可调用 wait_msg 接口进行消息阻塞，下个模拟周期将不再运行阻塞组件的功能函数，直到新消息到来。但还存在着另一种情况，即根据 CMB 保守算法，受同步限制，组件无法继续推进时间，因而无法继续处理消息，此时调用 wait_blk 进行时钟阻塞，记录当前阻塞时钟。下个模拟周期将不再运行阻塞组件的功能函数，直到组件可推进时间上限发生改变唤醒。

3 针对并行模拟引擎的性能评测 Benchmark-PassBall

本节将描述 Benchmark-PassBall 的设计，不涉及内部具体代码，可在文献[1]中下载代码及详细说明文档。

3.1 PassBall 设计概述

基于引言中并行模拟引擎性能评测 Benchmark 的设计原则，设计实现了一个 Benchmark-PassBall。它模拟多个 Player 之间进行 Ball 的传递行为。每一个 Player 即为模拟系统的一个组件，我们通过设计不同的拓扑和通信规则，保证 Benchmark 运行时的任务划分、通信规则及总体负载符合并行模拟引擎的多方面性能评测需求。为此，其包含 4 个评测模拟模型子程序，对应 4 种通信拓扑结构，下面介绍每一个子程序的测试功能。4 种结构分别是分组分类通信结构、组间通信结构、TORUS 互连结构、随机通信结构。

3.2 模拟模型一：分组分类通信结构

模拟模型一的设计目的是进行并行模拟引擎的弱扩展性测试。弱扩展性在于当每个线程负载不变时，线程数目线性增长，则模拟开销线性增长；当线程数目不变时，线性增大每个线程负载，模拟开销线性增长。为此而设计的分组分类通信结构中，通过参数的配置可实现线程负载的控制。

此模拟模型中 Player 只与“相邻”Player 进行通信。分组分类即先分组，后分类：

• 将所有的 Player 分为若干小组，位于同一小组的 Player 运行时被分配在同一线程中。同时若干小组组成一个大组，每一大组的 Player 都被分配在同一进程中。

• 根据运行时候的 Ball 传递关系，所有的球手又分成 3 类：所有通信的邻居都位于同一线程内部；所有通信邻居位于不同线程但同一进程内；所有通信的邻居位于不同进程内。

根据以上设计构建出的拓扑如图 1 所示。实连接表示的是 Player 之间的邻居通信关系，双向箭头表示连接的两个 Player 之间可以互发消息。虚连接的建立是为了保证拓扑图是一个连通图，与实连接的不同在于虚连接并不是一个模拟的消息通道，虚连接上不会出现消息传递。这里是动态地进行分组，可以通过调整运行输入参数(例如 Player 数目)以及进程数和每个进程内启动的线程数来保证单线程负载相同，

从而满足弱扩展性测试需求。

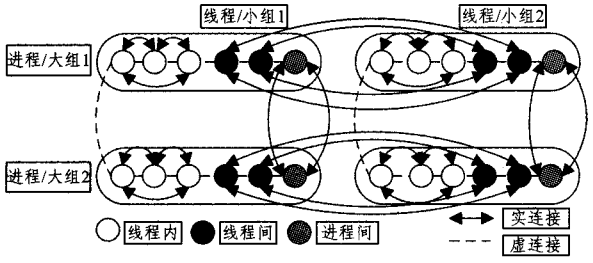

图 1　分组分类通信结构

3.3　模拟模型二：组间通信结构

模拟模型二的设计目的是进行并行模拟引擎的强扩展性测试。强扩展性测试即模拟模型结构不变，开销不变，随着并行度的线性增大，其模拟加速比线性增大。组间通信结构采用的是第一种分组分类模式的一个精简固定拓扑，组间通信结构不根据线程数目或者进程数目改变拓扑，并且每一组内并不区分不同类的通信。这种情况下通信总开销固定，同时这个结构可以保证每个 Player 的任务量相等，在每个线程分配到 Player 的个数相同的情况下，每个线程任务负载都是相等的。根据以上设计构建的拓扑如图 2 所示。

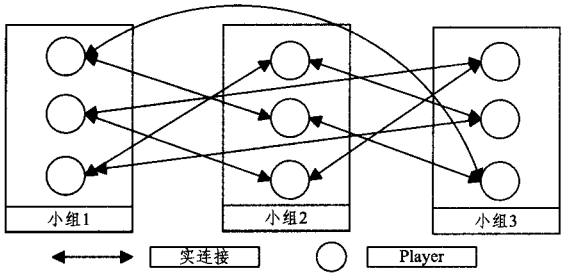

图 2　组间通信结构

3.4　模拟模型三：TORUS 互连结构

模拟模型三的设计目的是在不平衡负载情况下并行模拟引擎的强扩展性测试，这里采用的是较为通用的一个网络互联结构——TORUS[2]。如图 3 所示，每一个 Player 直接与其 4 个相连的 Player 进行通信。每一个 Player 既是消息的产生者，也是消息的转发者，类似终端和交换机功能合一。每一个 Player 都将向目的端发送自身消息，同时转发其他 Player 发向目的端的消息。

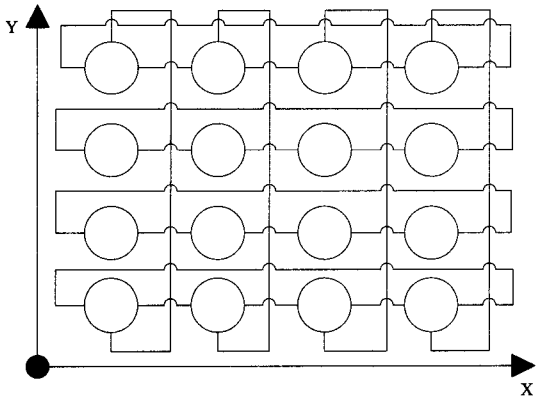

图 3　TORUS 互连结构

这里对整个 TORUS 结构采用坐标进行标记。每一个 Player 使用 $src(x,y)$ 进行标识，整个 TORUS 的规模用 $M*N$ 表示，M 代表横轴最大刻度，N 代表纵轴最大刻度。则每个 Player 发送消息的目的坐标 $dst(x,y)$ 为：

$$dst(x)=M-src(x);dst(y)=N-src(y)$$

则从 Player 发出的每一个消息都将携带 $dst(x,y)$ 信息，同时还应该携带将要发送到目的端的消息的总数量。在发送和转发的过程中，应该遵循的规则有：

- 优先判断 x，根据横坐标决策转发端口。在 x 相同的情况下再进行 y 的判断。
- 在进行转发的过程中，应该进行最短路径判断，保证发送出的节点数目最少，路程最短。
- 进行转发的时候，应该记录上一站转发的 Player 坐标，保证在选择路径的过程中不会选择重复路径。
- 对于 $M*N$ 型 TORUS，若存在中心节点，则中心节点只作为 SWITCH，自身不产生消息。

遵循以上规则进行转发，很明显这个结构中的每一个 Player 的任务量是不一样的，是一个负载不平衡的结构。

3.5　模拟模型四：随机通信结构

模拟模型四的设计目的是判断功能性测试并行模拟引擎是否支持多端口随机互连情况下的模拟，以避免出现阻塞等异常。随机通信结构，即所有 Player 之间的连接拓扑是随机无规律地产生，但需要保证任意一个端口的连接数目为 1，不能连接端口所在 Player 的其他端口，同时每个 Player 的端口数目 n 需要能够满足每个 Player 都能够被连通到全局拓扑图中。拓扑图如图 4 所示。

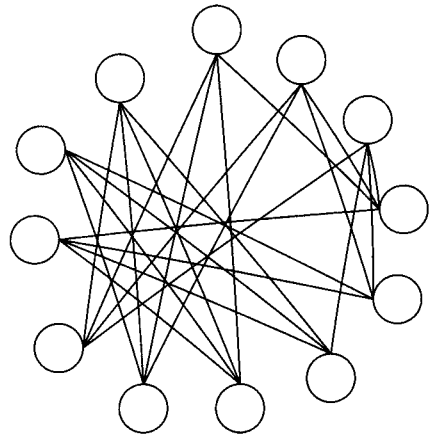

图 4　随机互连结构

3.6　功能模拟附加开销

如 3.1 节所述，对于本身开销并不大的问题，并行带来的调度同步开销等会超过其并行所节省的时间。因而对于我们的测试实例，在通信外需要增加一些伪计算来增大开销，使得每一次消息的传递同时伴随 CPU 的计算，让 CPU 繁忙起来。这里我们在每一次消息传递时附加循环 C 次加减乘除浮点运算，C 的取值在下一节讨论。

4　基于 PassBall 的 SimK 性能评测

本节我们采用 PassBallBenchmark，选取 2 个平台进行了并行模拟引擎 SimK 的性能评测，包含了强，弱扩展性测试、不平衡负载测试、两个平台上加速比的比较以及附加开销与加速比之间的关系测试。

4.1　测试平台

本实例在平台 A 和平台 B 下进行测试，两个测试平台的硬件配置如表 1 所列。

表 1　测试平台配置

节点名称	CPU 路数	CPU 厂商	CPU 类型	CPU 核数	处理器频率	内存大小	GCC 版本	MPI 库	MPI 库版本
平台 A	4	AMD	Opteron	4	800MHz	64GB	4.3.4	MPICH	1.5.4
平台 B	8	Intel	Xeon	8	1.064GHz	256GB	4.3.4	MPICH	1.5.4

实验中，加速比亦即相对于单线程运行实例性能上的提升比例。这里引入单位时间传球数(千次/s)-KPPS 作为实例性能的评价单位，根据 KPPS 得出加速比。在所有的试验中，我们保证每个核上只运行一个线程，对于平台 A 和平台 B 进行最大 16 线程的实验比较。在平台 B 上，将单独进行最大 32 线程的实验。

4.2　弱扩展性测试

3.2 节给出了弱扩展性的定义，采用的是模拟模型一，即分组分类通信结构进行测试。

多线程情况，保证线程数目不变，增大线程负载，从 2 线程到 16 线程进行测试，每个核上运行一个线程。

平台 A 的测试结果如图 5 所示。

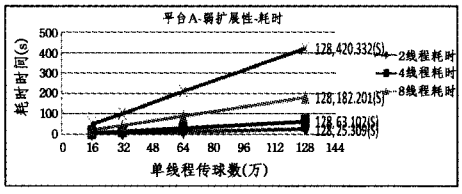

图 5　平台 A 线程负载和运行时耗关系

平台 B 的测试结果如图 6 所示。通过图 5、图 6 可以看出，在平台 A、B 上运行实例，若线程负载成倍增加，实例运行时耗也线性增加，同时我们测得实例运行性能 KPPS 是保持稳定的。对于同样的单位线程传球数，线程数目增大，时耗也成比例增大，验证了在平台 A、B 下 SimK 具有良好的弱扩展性。这里线程数目增大 2 倍，时耗会增大到大于 2 倍，原因在于线程数目增大 2 倍，总负载在通信外还会增加同步和调度的开销，导致总时耗会超过 2 倍。另外，可以看出在两个平台上因为配置的不同，实例运行性能 KPPS 是有差别的，平台 B 明显优于平台 A，但总体测试验证的弱扩展性是一致的。

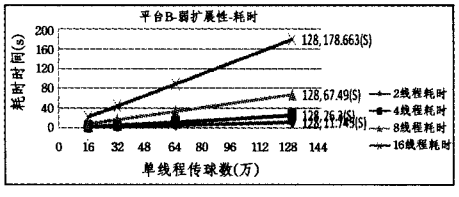

图 6　平台 B 线程负载与运行时耗关系

4.3　强扩展性测试

3.3 节给出了强扩展性的定义，采用的是模拟模型二，即组件通信结构进行测试。我们将附加开销循环次数 C 设定为 10000，实验分为两部分：a)对比 A、B 台下从单线程到 16 线程的 KPPS，选用两种规模进行测试，得出两个平台下的模拟加速比。b)在平台 B 上进行单线程到 64 线程的 KPPS 测试，得出加速比。这里，每一个线程运行时对 Player 的调度次数取决于每个 Player 所产生的 Ball 的个数，而非部署在线程内的 Player 个数，当 Ball 个数成倍增长时，Player 调度平均次数也是成倍增长的。

4.3.1　平台 A、B 模拟 KPPS 对比

首先进行 16 * 16 大小的对称的组间通信测试，即分 16 组，每组 16 个 Player，每个 Player 的每个端口持有 500 个 Ball，即每个 Player 自身传出 1000 个 Ball。在平台 A 和平台 B 的运行结果如图 7 所示。

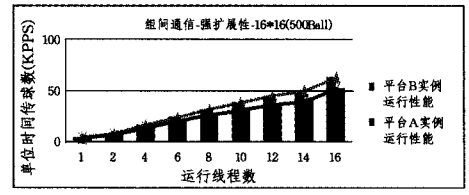

图 7　组间通信测试 1

之后采用 15 * 25 大小的非对称的组间通信测试，每个 Player 的每个端口持有的 Ball 数目依然为 500 个，在平台 A 和平台 B 上的运行结果如图 8 所示。

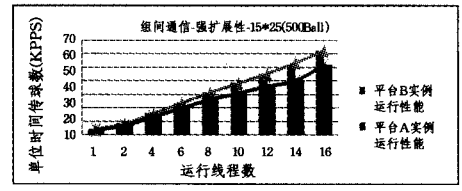

图 8　组间通信测试 2

与此同时，我们测得：a)对于 16 * 16 规模，平台 A、B 单线程情况下运行时间分别约为 200 分钟、156 分钟，16 线程情况下分别约为 12.5 分钟、10 分钟。b)对于 15 * 25 规模，平台 A、B 单线程情况下运行时间分别约为 304 分钟、235 分钟，16 线程下分别约为 19 分钟、15.5 分钟。

4.3.2　平台 B 上 32 线程模拟 KPPS

我们选取 18 * 25 规模的分组通信，即每组 18 个 Player，总共 25 组。从单线程到 32 线程，运行结果如图 9 所示。对于最大 32 线程的测试，SimK 依然保持着良好的强扩展性。图 9 中曲线的波折源于在一定数目运行线程时划分任务的不均匀导致的性能下降，但最终 32 线程的加速比达到约 27 倍。

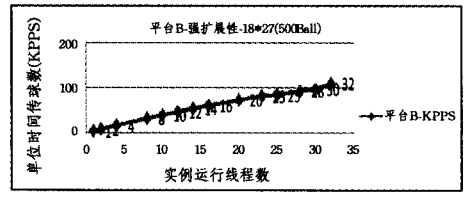

图 9　平台 B 最大 32 线程组间通信

4.4　不平衡负载下的强扩展性测试

对于此处的 TORUS 互联结构，根据设计的通信方式，TORUS 的每个节点如前所述，是负载不平衡的。这种不平衡带来的是部分节点推进快而部分推进慢，导致出现较长的空消息发送处理时间，也会带来一些负载迁移的开销。这些额外时间会降低并行性能，测试结果如图 10 所示。可以看出，TORUS 结构的运行性能随着线程数的增大而增大，最终

16 线程情况下加速比约为 10 倍。

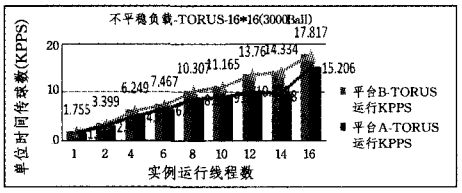

图 10　TORUS 互连结构测试

4.5　强扩展性测试加速比比较

在 4.3 节和 4.4 节中分别进行了组间通信的强扩展性测试和 TORUS 不平衡负载下的强扩展性测试，这里比较平衡和不平衡负载情况下两种测试加速比从单线程到 16 线程的差异。这里选择平台 B 测试数据，如图 11 所示。

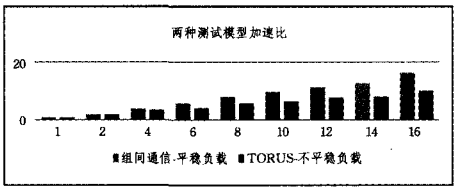

图 11　两种测试模型加速比比较

可以明显看出，TORUS 结构加速比低于组间通信。之前提到过组件推进速度不一的情况，每一个模拟周期不同，组件处理的消息数目是不一样的，推进的时钟也就不一样，这导致了推进快的将进行等待（如果阻塞的话）或者进行空循环监听（未阻塞）。这种等待带来的开销降低了模拟的性能。可以得出结论，负载的不平衡将会导致模拟系统性能的下降[5,6]。

4.6　功能模拟附加开销与加速比相关性探讨

在 3.6 节提到过，为了保证并行的效率，需要为实例添加功能模拟附加开销，使得实例的计算量达到一定规模，并行提升的性能远大于并行带来的额外负载。之前测试时，采取的是加入循环浮点运算，浮点运算次数 C 在之前的强扩展性测试中取值为 10000。本节测试了浮点运算次数 C 对加速比的影响，这里采用的是在平台 B 上运行模拟模型二（组间通信）的数据，结果如图 12 所示，CT 代表的是 C 的数值。

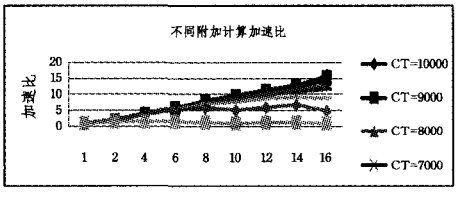

图 12　附加计算量与加速比关系

从图中可以看出，在 C 大于 5000 时，加速比递增维持着近乎线性，基本没有出现明显的降低；但到 5000 以下后，出现了较为明显的降低；到 1000 时，加速比大幅度递减；当计算量为 0 时，出现了负加速。

结束语　本文对并行模拟引擎 SimK 及 SimK1.1 新增功能进行了简介。与此同时，介绍了针对并行模拟引擎的性能测试实例 PassBall，并使用其对 SimK 性能进行了全面的评测。通过数据可以得出，SimK 拥有良好的强弱扩展性，同时负载平衡和功能模拟附加开销会对模拟性能产生较大影响。这验证了我们的 PassBall 实例是一个可以很好地对并行模拟引擎 SimK 进行全面评测的 Benchmark，设计思路具有通用性，易于在其他并行模拟引擎上实现。

参 考 文 献

[1] SimK Project [OL]. https://sourceforge.net/projects/simk/files/

[2] Torus[OL]. http://en.wikipedia.org/wiki/Torus

[3] Andersen R, Lang K J. An algorithm for improving graph partitions[C]// Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, California, 2008: 651-660

[4] Hendricksom B, Leland R. The chaco user's guide: Version 2.0 [M]. Sandia National Laboratories, 1994

[5] Boukerche A, Das S K. Dynamic load balancing strategies for conservative parallel simulations[C]// Proceedings of the Eleventh Workshop on Parallel and Distributed Simulation. Lockenhaus, Austria, 1997: 20-28

[6] Boukerche A, Das S K. Null messages cancellation through load balancing in distributed simulations[C]// Euro-Par'99: Parallel Processing. vol. 1685, 1999: 562-569

[7] Cao Z, Xu J, Chen M, et al. HPPNetSim: a parallel simulation of large-scale interconnection networks[C]// Proceedings of the 2009 Spring Simulation Multiconference. San Diego, California, 2009: 1-8

[8] Cheng Y, Bai L, Chen M, et al. P-GAS: Parallelizing a Cycle-Accurate Event-Driven Many-Core Processor Simulator Using Parallel Discrete Event Simulation[C]// Principles of Advanced and Distributed Simulation. 2010: 1-8

[9] Fan D R, Yuan N, Zhang J C, et al. Godson-T: An Efficient Many-Core Architecture for Parallel Program Executions[J]. Journal of computer science and technology, 2009, 24(6): 1061-1073

[10] Fujimoto R M. Parallel Discrete Event Simulation[J]. Communications of the ACM, 1990, 33(10): 30-53

[11] Garey M R, Johnson D S, Stockmeyer L. Some simplified NP-complete problems[C]// Proceedings of the Sixth Annual ACM Symposium on Theory of Computing. Seattle, Washington, United States, 1974: 47-63

[12] Pothen A. Graph Partitioning Algorithms with Applications to Scientific Computing[R]. Old Dominion University, 1997

[13] Chaco P A. a sacred center-excerpt[C]// ACM SIGGRAPH 98 Electronic art and animation catalog. Orlando, Florida, United States, 1998: 112

[14] Xu J-W, Chen M, Zheng G, et al. SimK: A Large-Scale Parallel Simulation Engine[J]. Journal of computer science and technology, 2009, 24(6)

[15] 许建卫. 高性能计算机的并行模拟技术研究[Z]. 中国科学院计算技术研究所, 2008