软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2014,25(8):1844−1857 [doi: 10.13328/j.cnki.jos.004490]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 一种均衡可扩展计算机体系结构分布式模拟方法*

徐传福, 车永刚, 王正华, 彭宇行

(国防科学技术大学 计算机学院,湖南 长沙 410073)

通讯作者: 徐传福, E-mail: xuchuanfu@nudt.edu.cn

**摘 要**: 分布式并行模拟是提高体系结构模拟速度的有效技术手段之一.首先,建立了分布式并行模拟的通用性能分析模型,并对典型系统的并行加速比、并行效率等性质进行了理论分析,得出了一些有用的结论.在此基础上,提出了均衡可扩展分布式并行模拟方法 SEDSim(scalable and evenly distributed simulation).SEDSim 针对模拟节点负载不均衡问题,提出了开销模型指导的指令区间均衡分割和分配策略 CoMEPA(cost model guided evenly partition and allocation);针对分布式并行模拟与非连续、任意数量抽样模拟区间的高效集成,提出了基于最小等价距离(minimum equivalent cost,简称 MinEC)的指令区间分配策略 MinEC.基于 sim-outorder 实现了 SEDSim,采用 SPEC CPU2000 中的部分程序对其速度和精度进行了测试,理论分析和测试结果均表明了 SEDSim 的优势:相对于常用的方法或策略, CoMEPA 和 MinEC 分别能够获得多达约 1.6 倍和 1.4 倍的性能提升.

**关键词**: 计算机体系结构;分布模拟;负载均衡;可扩展

**中图法分类号**: TP306

中文引用格式: 徐传福,车永刚,王正华,彭宇行.一种均衡可扩展计算机体系结构分布式模拟方法.软件学报,2014,25(8):1844−1857. http://www.jos.org.cn/1000-9825/4490.htm

英文引用格式: Xu CF, Che YG, Wang ZH, Peng YX. Scalable and evenly distributed simulation method for computer architecture. Ruan Jian Xue Bao/Journal of Software, 2014,25(8):1844−1857 (in Chinese). http://www.jos.org.cn/1000-9825/4490.htm

## Scalable and Evenly Distributed Simulation Method for Computer Architecture

XU Chuan-Fu, CHE Yong-Gang, WANG Zheng-Hua, PENG Yu-Xing

(College of Computer, National University of Defense Technology, Changsha 410073, China)

Corresponding author: XU Chuan-Fu, E-mail: xuchuanfu@nudt.edu.cn

**Abstract**: Distributed simulation is an effective method to improve simulation speed for computer architecture. In this paper, a general performance model for distributed simulation is established and then some typical distributed simulation systems are analyzed based on the model. The analysis results in some important conclusions about parallel speedup and parallel efficiency for distributed simulation. Next, a scalable and evenly distributed simulation (SEDSim) approach is presented. SEDSim adopts a cost model guided even partition and allocation (CoMEPA) policy for benchmark program instructions to enhance load-balance among parallel simulation nodes. An allocation policy based on minimum equivalent cost (MinEC) is also designed to efficiently integrate arbitrary number of discrete sampling intervals in SEDSim. The study implementes SEDSim based on sim-outorder and evaluates its speed and accuracy using Benchmark programs from SPEC CPU2000. Both theoretical analysis and testing results validate some advantages of SEDSim approach. Compared with existing methods, CoMEPA and MinEC can achieve a speedup of about 1.6 and 1.4 respectively.

**Key words**: computer architecture; distributed simulation; load balance; scalability

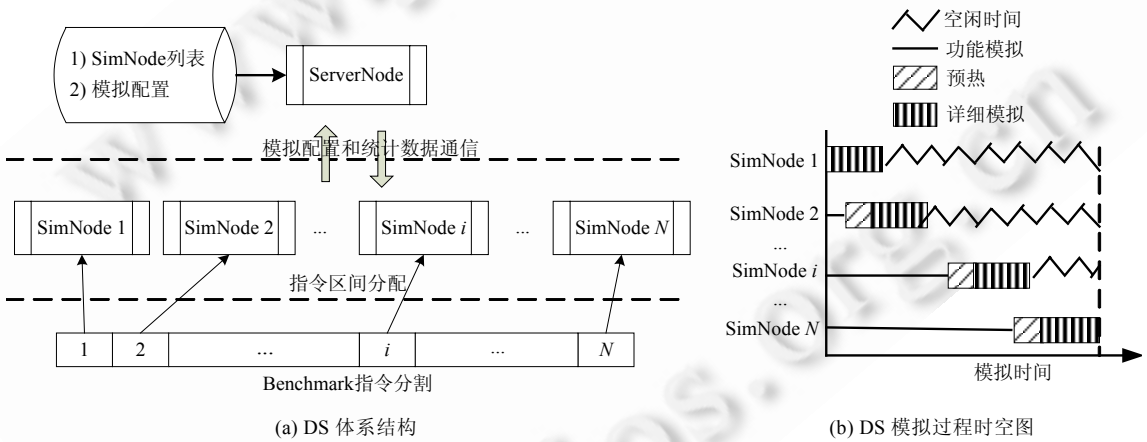
模拟技术是一种有效的计算机系统性能评测手段.研究人员常常采用模拟器对体系结构设计进行评估.相

* 基金项目: 国家自然科学基金(11272352); 国家高技术研究发展计划(863)(2007AA01Z116); 国家重点基础研究发展计划(973)(2009CB723803)

收稿时间: 2011-02-15; 修改时间: 2012-10-10; 定稿时间: 2013-07-30

对于 Benchmark 测试和分析模型,基于软件的模拟技术由于能够在性能评估的代价、时间以及灵活性之间进行很好的平衡而应用较为广泛.当前,国内外已经开发了一些支持不同模拟目标的体系结构模拟器软件,例如美国斯坦福大学以超标量乱序执行处理器为目标的 SimpleScalar[1,2]模拟器以及中国科学院计算技术研究所针对“龙芯”处理器开发的 SimGodSon[3],SimOS-Goodson[4]等.与在实际硬件上执行 Benchmark 程序相比,模拟技术的一个主要缺点是执行速度慢,我们采用 SimpleScalar(版本 3.0d)中最为详细的模拟工具 sim-outorder 在一个 2.2GHz 主频、2.0GB 内存的宿主机平台上获得的模拟速率仅为约 1MIPS(million instructions per second),Benchmark 程序实际运行 1 分钟意味着需要模拟执行几天时间,以这种速度完整模拟标准 Benchmark 程序集,如 SPEC CPU2000 等,需要多达数月.随着新型体系结构和 Benchmark 程序日益复杂和庞大,模拟器运行速度慢的问题更加突出,已严重制约了体系结构设计空间探索的效率[5].为此,研究人员提出了一些加速模拟技术如抽样模拟、Benchmark 程序输入集缩减等[6−8],这些方法主要解决串行模拟时如何选择部分模拟负载以减少模拟时间,并保证一定模拟精度的问题.

近年来,研究人员又提出了利用多个分布式节点同时运行串行模拟器实现并行模拟的方案,即所谓的分布式并行模拟(distributed simulation,简称 DS)[9−12].DS 的基本结构如图 1(a)所示:Benchmark 工作负载分为若干指令区间(interval)并分配给模拟节点(SimNode),模拟节点对指定的区间完成模拟后,将局部性能结果汇总到服务器节点(ServerNode),由后者合成全局性能指标.图 1(b)给出了模拟过程的时空图:模拟过程中,模拟节点首先采用功能模拟(functional simulation)快速推进(fast-forward)到所分配的指令区间之前进行适当的预热(warm-up),然后对该区间进行详细性能模拟(detailed simulation).




(a) DS 体系结构 (b) DS 模拟过程时空图

Fig.1

图 1

与抽样模拟等加速方法相比,DS 具有以下优点:

1) 对 Benchmark 负载进行了完整模拟因而精度较高;抽样等加速方法仅选取了负载的一部分,难以全面刻画程序特征,因而有不同程度的精度损失.
2) 基于已有串行模拟器构建因而简单且易于实现,对其核心代码改动也很少;抽样模拟往往需要专门的预处理或后处理,额外开销大.
3) 独立于具体的体系结构配置和 Benchmark 程序,较为通用;抽样模拟往往需要针对不同体系结构配置、性能指标或 Benchmark 程序重新生成抽样指令区间.

分析已有的 DS 方法,普遍存在如下问题:

1) 指令区间分割和分配不灵活,难以扩展.

通常,区间所含指令数相等(即等区间)且区间数量等于模拟节点数(即等节点).由于不同指令区间在

Benchmark 程序中的位置不同,所需的功能模拟推进开销相差较大,因此,等区间容易造成节点间模拟负载的不均衡;而等节点是由于多数 DS 方法采用的模拟器仅支持功能模拟到详细模拟的一次切换,即所谓的单向切换(unidirectional transition),而不支持再从详细模拟切换回功能模拟,即所谓的双向切换(bidirectional transition).

2) 节点间模拟负载不均衡.

如图 1(b)所示,由于指令区间 $N$"距离"最远,整个并行模拟的结束时间取决于模拟区间 $N$ 的节点(图中的 SimNode $N$),其他节点均已完成模拟处于空闲等待状态中.这种负载不均衡,部分是由于区间的均匀划分造成的:位于程序后面的指令区间在实际性能模拟之前需要花费大量的时间进行快速推进,而均匀的区间划分并未考虑这部分开销.此外,复杂的预热机制也常常导致节点间实际负载差异较大.

3) 不支持与抽样模拟区间的高效集成.

已有的 DS 方法仅考虑了完整 Benchmark 程序中连续指令区间的分配,对于由 SimPoint[13],EXPERT[14]等抽样模拟方法生成的非连续、任意数量抽样区间,需要设计新的分配策略.

上述问题在一定程度上影响了 DS 的并行加速比、并行效率以及扩展性等.本文首先建立了 DS 的通用性能分析模型,对典型的采用等区间、等节点循环分配策略的 DS 方法的并行加速比、并行效率等性质进行了理论分析.在此基础上,提出了均衡可扩展分布式并行模拟方法 SEDSim(scalable and evenly distributed simulation). SEDSim 从两个方面增强了 DS 的负载均衡和可扩展性:

1) 针对等区间导致的模拟节点负载不均衡,提出了基于开销模型的指令区间均衡分割和分配策略 CoMEPA(cost model guided evenly partition and allocation);
2) 针对 DS 与非连续、任意数量抽样模拟区间的高效集成,提出了基于最小等价距离的指令区间分配策略 MinEC(minimum equivalent cost).

基于 sim-outorder 3.0 实现了 SEDSim,采用 SPEC CPU2000 中的部分 Benchmark 程序测试了其加速比和精度.理论分析和测试结果表明:相对于已有采用等区间、等节点的循环分配,CoMEPA 总能提高并行加速比,MinEC 不仅能够获得最大加速比,而且可以预先确定获得最大加速比所需的处理器数.

## 1 相关工作

DS 思想最早起源于踪迹驱动(trace driven)模拟的抽样和并行化.在文献[15]中,Highderburger 等人将 cache 模拟器产生的内存访问踪迹分为若干踪迹区间并分布到并行模拟节点上,为减少由微体系结构状态(如 cache、分支预热器等)"冷启动(cold-start)"导致的模拟精度损失,在对踪迹区间模拟之前需要进行一定的预热.大量文献研究了如何确定预热指令的数量,例如,Nyuyen 等人[16]提出了根据 L1 cache 命中率估计预热指令数的启发式方法;文献[12]对踪迹的并行模拟进行了总结,并建立了在并行节点上分布踪迹的分析模型,支持对踪迹分布(distributing traces)和踪迹样本分布(distributing samples)进行效率分析.近年来,研究人员又针对执行驱动的指令级模拟器提出了一些分布式并行模拟方法,与踪迹驱动模拟不同的是,在对指令区间进行详细性能模拟之前,必须对体系结构状态进行功能模拟,以满足模拟状态依赖关系.Girbal 等人[9]提出了 DiST:DiST 将 Benchmark 程序指令流划分为一些相等大小的区间(chunk)分配给模拟节点,每个区间又包括若干子区间(subchunk)以控制预热粒度;DiST 设计了一个可动态调整的预热机制:指令区间 $m-1$ 与 $m$ 有若干子区间重叠,在重叠的子区间部分,不断比较指定的体系结构指标,例如 IPC(instructions per cycle),是否收敛于给定的阈值:如果满足条件,则区间 $m-1$ 模拟结束,区间 $m$ 中重叠子区间的统计数据替换为 $m-1$ 中对应的子区间.用户可以通过指定收敛阈值影响 DiST 的并行加速比和模拟精度.Masahiro 等人[10]提出了一种沿时间轴对时钟精确体系结构模拟器进行时分(time-division)并行化的方法.该方法通过失效恢复机制保证并行模拟精度与串行模拟完全相同:若第 $i$ 个区间的模拟由于近似状态而产生无效结果,则由负责第 $i-1$ 个区间的节点对其进行重新模拟,确保第 $i$ 个区间一开始就具有正确的微体系结构状态.Ramkumar 等人[11]将 Benchmark 指令分为相等的区间,按照循环分配策略映射到并行模拟节点,每个区间均采用相同的固定比率的指令数进行预热.文献[9,10]都是模拟精度优先的方法,预热机制可通过牺牲加速比换取精度的提升;缺点是实现较为复杂且预热开销难以控制,最坏情况下,并行模拟可

完全退化成单节点的串行模拟.相对而言,文献[11]的方法较为简单,并行加速比较高,但难以评估精度损失.

除了利用已有串行模拟器构建分布式并行模拟器外,研究人员也开发了一些直接支持并行执行的模拟器,这些模拟器多数面向大规模系统或者其关键部件的并行性能模拟,例如系统级并行应用性能模拟器 BigSim[17],Mpi-Sim[18],ArchSim[19]以及高性能互连网络并行模拟器 BigNetSim[17],HPPNetSim[20]等.此外,也有一些基于原有串行体系结构模拟器开发的并行版本模拟器,例如,美国佛罗里达大学的研究人员基于 SimpleScalar 开发了一个面向片上多核处理器的多线程并行模拟器 SimpleCMP[21],斯坦福大学基于全系统模拟器 SimOS[22]开发了 Parallel SimOS[23].这些并行模拟器尽管利用已有代码避免了重复开发,但需要对具体目标系统结构中的大量子单元进行高效划分,并映射到并行模拟进程以减少进程间通信、同步等开销,因此灵活性、通用性较差,且多数针对多处理器系统或片上多核处理器.

## 2 DS 性能分析模型

本节先对采用任意指令区间分割和分配策略的 DS 建立一个通用性能模型,为便于分析引入以下假设:

**假设 1(同构假设)**. 假设所有模拟节点计算、存储等能力相同,不考虑节点差异对并行性能的影响.

**假设 2(速度假设)**. 假设模拟器对同一 Benchmark 程序的各指令区间进行模拟时速度相等.

**假设 3(预热假设)**. 假设指令区间的预热指令的数量与该区间所包含的指令数量相关,采用基于详细模拟的预热(即预热过程与详细模拟过程相同,但不统计性能数据).

为便于查阅,模型的主要参数列于表 1.定义几种区间分配策略如下:

**定义 1(随机分配(random allocation))**. 任意指令区间均分配到唯一的节点,即,

1) 对任意两个不同的节点 $l$ 和 $m$, $\bigcup_{j=1}^{n(l)} I_{l_j} \cap \bigcup_{j=1}^{n(m)} I_{m_j} = \varnothing$;

2) $\bigcup_{i=1}^{N}\bigcup_{j=1}^{n(i)} I_{i_j} = \{I_k \mid 1 \leqslant k \leqslant P\}$.

**定义 2(循环分配(cyclic allocation))**. 对任一区间 $I_k$,当 $k \bmod N=i$ 时,$I_k$ 分配到节点 $i$.

**定义 3(块分配(block allocation))**. 对于块大小为 $b$(通常取 $1 \leqslant b \leqslant t$)的块分配,任一区间 $I_k$,$\exists j(j \geqslant 0)$,$bj \leqslant k \leqslant b(j+1)$,当 $j \bmod N=i$ 时,$I_k$ 分配到节点 $i$.

**Table 1** Parameters used in the analytical model

**表 1** 分析模型中的参数

| 参数 | 说明 | 参数 | 说明 |
|---|---|---|---|
| $N$ | 模拟节点数 | $w_k$ | 区间 $I_k$ 的预热比率 |
| $P$ | 指令区间数 | $I_{i_j}$ | 节点 $i$ 分配的第 $j$ 个区间 |
| $t$ | 区间倍数,$P=tN$ | $I_k$ | 第 $k$ 个区间(包括 $I_k$ 条指令) |
| $n(i)$ | 节点 $i$ 分配的区间数 | $S$ | 加速比 |
| $V_f$ | 功能模拟执行速度 | $E$ | 并行效率 |
| $V_d$ | 详细模拟和预热的速度 | $R$ | 功能模拟与详细模拟速率之比 |

下面分模拟器是否支持功能模拟与详细模拟的双向切换进行考虑:当模拟器仅支持单向切换且 $n(i)>1$ 时,对于 $I_{i_j}$,节点 $i$ 的功能模拟指令数为 $\sum_{k=0}^{i_j-1} I_k - w_{i_j} I_{i_j}$,预热指令数为 $w_{i_j} I_{i_j}$,详细模拟的指令数为 $I_{i_j}$,因此,节点 $i$ 总的功能模拟指令数为 $\sum_{j=1}^{n(i)}(n(i)-j+1)\sum_{k=0}^{i_j-1} I_k - \sum_{j=1}^{n(i)} w_{i_j} I_{i_j}$(其中,$w_1=0$ 且 $I_0=0$,表示区间 $I_1$ 无功能模拟和预热),预热指令数为 $\sum_{j=1}^{n(i)} w_{i_j} I_{i_j}$,详细模拟指令数为 $\sum_{j=1}^{n(i)} I_{i_j}$;当模拟器支持双向切换或 $n(i)=1$ 时,节点 $i$ 总的功能模拟指令数为 $\sum_{k=0}^{n(i)-1} I_k - \sum_{j=1}^{n(i)} w_{i_j} I_{i_j} - \sum_{j=1}^{n(i)-1} I_{i_j}$,预热指令数为 $\sum_{j=1}^{n(i)} w_{i_j} I_{i_j}$,详细模拟指令数为 $\sum_{j=1}^{n(i)} I_{i_j}$.

在上述两种情况下,节点 $i$ 的执行时间 $T_i$ 可以表示为

$$\left[\sum_{j=1}^{n(i)}(n(i)-j+1)\sum_{k=0}^{i_j-1} I_k - \sum_{j=1}^{n(i)} w_{i_j} I_{i_j}\right] \Big/ V_f + \sum_{j=1}^{n(i)}(w_{i_j}+1) I_{i_j} \Big/ V_d, \quad \text{单向切换且 } n(i)>1 \tag{1}$$

$$\left[\sum_{k=0}^{n(i)-1} I_k - \sum_{j=1}^{n(i)} w_{i_j} I_{i_j} - \sum_{j=1}^{n(i)} I_{i_j}\right] \Big/ V_f + \sum_{j=1}^{n(i)} (w_{i_j}+1) I_{i_j} \Big/ V_d, \quad \text{双向切换或 } n(i)=1 \tag{2}$$

整个程序的串行模拟时间为 $T_{serial}=\sum_{k=1}^{P} I_k / V_d$,而并行模拟的结束时间 $T_{parallel}$ 取决于执行时间最长的结点,并行加速比 $S=T_{serial}/T_{parallel}$,并行效率 $E=S/N$.

指令区间的分割和分配直接影响 DS 的并行性能和效率,当前,DS 实现中多采用等区间等节点的循环分配,原因在于随机分配实现复杂,而块分配中同一块中 $b$ 个连续区间的指令可以合并为一个新的区间,因而总可归为循环分配.下面主要分析这种情况下 DS 的性能.

**结论 1**. 采用等区间等节点循环分配且各区间预热比率相等时,DS 加速比上限为 $R$,其中,$R=V_f/V_d$(即功能模拟与详细模拟速率之比).

证明:等区间等节点循环分配且各区间预热比率相等时,每个节点分配一个指令数相等的区间且各区间预热比率也相等,即 $I_i=I,P=N$ 且 $w_i=w$.根据定义 2 可知,并行模拟的执行时间取决于模拟区间 $I_N$ 的节点.由公式(2)可得 $T_{parallel}=(N-1-w)I/V_f+(w+1)I/V_d$,而 $T_{serial}=NI/V_d$,因此,令 $R=V_f/V_d$,则 $S=T_{serial}/T_{parallel}=NR/[(N-1-w)+(w+1)R]$,并行效率 $E=R/[(N-1-w)+(w+1)R]$.可见,$\lim_{N\to\infty} S=R$. □

结论 1 表明:单纯通过增加模拟节点数难以突破加速比极限 $R$,且并行效率逐渐下降.以 sim-outorder 为例,其在当前主流宿主机平台上 $R$ 的范围通常为 10~40[10,11],以 $R$ 取 10、$w$ 取 0.1 为例,当 $N$=10 时,$S$=5,$E$=50%;当 $N$ 增加到 20 时,$S$ 仅增加到约 6.7,但 $E$ 却下降为 33.4%.

**结论 2**. 采用等区间循环分配且各区间预热比率相等时,等节点情况下(即区间倍数 $t$ 取 1 时)加速比和并行效率最高;当模拟器支持双向切换时,区间倍数 $t$ 对加速比和并行效率无影响;当模拟器仅支持单向切换时,加速比和并行效率随区间倍数 $t$ 增加而降低.

证明:对区间倍数 $t$ 分别讨论:

- 当 $t$=1 时,即为等区间等节点,由结论 1 可知:
  - 加速比 $S=NR/[(N-1-w)+(w+1)R]$;
  - 并行效率 $E=R/[(N-1-w)+(w+1)R]$.
- 当 $t$>1 时,每个节点分配 $t$ 个区间,设区间大小为 $I'$:
  - 当模拟器仅支持单向切换时,由公式(1)可知:此时并行模拟时间为
    $$T_{parallel}=[(t+1)t(N-1)I'/2-wtI']/V_f+t(w+1)I'/V_d,$$
    而串行模拟时间为 $T_{serial}=tNI'/V_d$,因此,加速比 $S_1=NR/[(N-1)(t+1)/2-w+(w+1)R]$;
  - 当模拟器支持双向切换时(假定模拟器切换开销忽略),根据公式(2)可知:此时并行模拟时间为 $T_{parallel}=(N-1-w)tI'/V_f+(w+1)tI'/V_d$,串行模拟时间仍然为 $T_{serial}=tNI'/V_d$,因此,加速比:
    $$S_2=NR/[(N-1-w)+(w+1)R].$$

  由于 $t$>1,因此总有 $S_2=S>S_1$,即,等节点时加速比和并行效率总是最高,且 $S_2$,$S$ 与 $t$ 无关,而 $S_1$ 随 $t$ 的增加而减少. □

结论 2 表明:当节点数固定时,单纯通过增加区间数,最多只能保持加速比和并行效率,在模拟器仅支持单向切换时,甚至会导致加速比和并行效率下降.

由结论 2 易得如下推论:

**推论 1**. 等区间循环分配且区间数固定、各区间预热比率相等时,节点数与区间数相等(即 $N=P$)时加速比和并行效率最高.

由上述分析过程和结论可知,SEDSim 需要重点解决 DS 的负载均衡和可扩展性问题:

1) 给定一个 Benchmark 程序,如何合理地进行指令区间分割和分配,以尽可能地保证模拟节点间的负载均衡?
2) 给定任意数量、大小的非连续指令区间,如何基于这些区间构建分布并行模拟,并实现最大程度的节点间负载均衡?

对于问题 1),总可以归为 $N=P$ 时的区间分割和分配的负载均衡问题.对于问题 2),则必须考虑针对 $N<P$ 设计新的区间分配策略.

## 3 均衡可扩展分布式并行模拟 SEDSim

### 3.1 $N=P$时的指令区间分割和分配CoMEPA

CoMEPA 仍然采用循环分配,但在指令区间分割时充分考虑了模拟开销以获得负载均衡:对于区间 $I_k$,随着 $k$ 的增加,因功能模拟而导致的快速推进开销会增大,因此区间大小应该随着 $k$ 的增加而递减;根据分析模型,对于包含 $N$ 个节点的 DS,设等区间时指令区间大小为 $I$,则负载均衡的 $I_k$ 应满足以下约束条件:

$$\sum_{k=1}^{N} I_k = NI \tag{3}$$

$$\left(\sum_{j=0}^{k-1} I_j - w_k I_k\right)\Big/V_f + (w_k+1)I_k/V_d = \left(\sum_{j=0}^{k} I_j - w_{k+1} I_{k+1}\right)\Big/V_f + (w_{k+1}+1)I_{k+1}/V_d \tag{4}$$

$$w_k/w_{k+1}=I_{k+1}/I_k \tag{5}$$

其中,$w_1=0$,$I_0=0$.

公式(4)称为负载均衡约束,表明任意两个节点 $k$ 与 $k+1$ 的模拟开销相等.通常认为,指令区间所需的预热指令数与其区间大小成反比[12],CoMEPA 对预热指令数的分配由公式(5)表示.由公式(4)和公式(5)可得 $I_{k+1}=aI_k$ 和 $w_k=aw_{k+1}$,即

$$I_k=a^{k-1}I_1 \tag{6}$$

$$w_k=a^{N-k}w_N \tag{7}$$

其中,$a=(R-1)/R$,称为等比因子.

公式(6)和公式(7)表明,CoMEPA 负载均衡的结果是:相邻指令区间指令数随序号 $k$ 等比递减,而预热比率相应地等比递增,两者具有相同的等比因子 $a$.公式(6)和公式(7)结合公式(3)可得 $I_1=NI/(1-a^N)R$,因此,

$$I_k=a^{k-1}NI/(1-a^N)R \tag{8}$$

而预热比率 $w_k$ 的计算由用户根据模拟精度需求首先设置参数 $w_N$.

CoMEPA 的指令区间分割和分配过程如下:

(1) 确定参数.包括采用功能模拟获得 Benchmark 程序的指令数,选择指令区间确定 $R$,设置 $w_N$.

(2) 确定详细模拟和预热的指令数.对于节点 $k$,根据公式(8)计算区间大小,根据公式(7)计算预热比率 $w_k$.

(3) 确定功能模拟指令数.对于节点 $k$,为快速推进到 $I_k$ 并在详细模拟之前预热 $w_kI_k$ 条指令,需要功能模拟的指令数为 $\sum_{j=0}^{k-1} I_j - w_kI_k$.

### 3.2 $N<P$时的区间分配MinEC

SimPoint,EXPERTT 等选择性抽样模拟方法仅选取 Benchmark 程序中若干具有代表性的指令区间进行详细模拟,以此为基础推断出整个程序的性能指标.由于 Benchmark 程序特征的差异,抽样模拟区间通常非连续、数量不确定甚至区间大小不等.例如,SimPoint3.0(其中,Max$K$ 取值为 10)对 SPEC CPU2000 中 twolf 和 equake 程序生成的 100M(million)大小的抽样区间分别为[24]$I_0$,$I_{20}$,$I_{135}$,$I_{168}$,$I_{191}$,$I_{331}$,$I_{449}$ 和 $I_0$,$I_{48}$,$I_{187}$,$I_{1011}$,$I_{1124}$.显然,利用抽样区间构建 DS 时,简单的循环分配策略难以获得最优的负载均衡.

MinEC 的基本思想是:根据第 2 节的性能模型计算所有模拟节点的开销,并转换为等价开销(将详细模拟开销折合成功能模拟开销),按指令区间序号由大到小的原则将待分配的区间分配给当等价开销最小的节点.MinEC 算法过程描述如下:

1. 分别计算 $N$ 个模拟节点的当前等价开销 $EC[\cdot]$,对于 SimNode $i$:

   a) 如果模拟器支持双向切换,则

$$EC[i] = PC_{I[i][n(i)]} + (R-1)\sum_{j=0}^{n(i)-1} I_{I[i][j]} + RI_{I[i][n(i)]} \tag{9}$$

   b) 如果模拟器仅支持单向切换,则

$$EC[i]=\sum_{j=0}^{n(i)}(PC_{I[i][j]}+RI_{I[i][j]}) \tag{10}$$

其中,$n(i)$为 SimNode $i$ 当前已分配的指令区间数,数组 $I[i][\cdot]$存储已分配的区间的索引,$PC_k$为详细模拟 $I_k$所需推进的功能指令数.这里将详细模拟指令数乘以 $R$ 转换为功能模拟指令数.

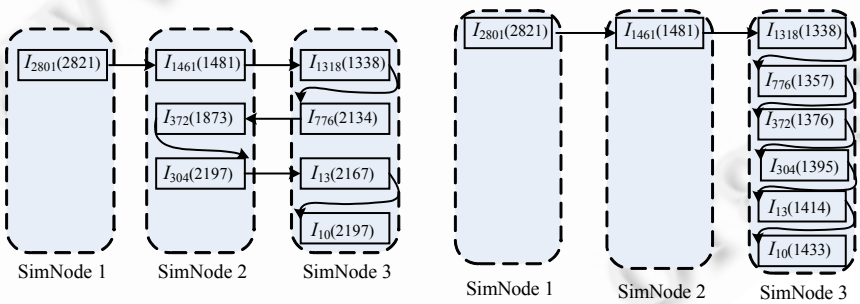
2. 获取当前 $EC[\cdot]$中等价开销最小的节点(设为 MIN),若有多个节点开销最小,则取序号最小者.
3. 按照索引由大到小的原则依次分配每个区间:

```
For (k=P; k≥1; k−−){
    调用步骤 1 和步骤 2,求 MIN 节点,
    //将区间 I_k 分配到 MIN 节点
        n(MIN)+=1;
        I[MIN][n(MIN)]=k;
        }//end for
```

图 2 给出了一个 MinEC 算法应用过程的例子.指令区间由 SimPoint3.0 针对 SPEC CPU2000 中的 ammp 生成,$R$ 取 20,模拟节点数为 3.图 2(a)给出了单向切换模拟器的分配结果,图 2(b)给出了双向切换模拟器时的分配结果.图中带有箭头的线段表明了区间的分配顺序,括号内为分配后该节点的等价开销.例如,对于图 2(a)中的 SimNode 1,首先分配了开销最大的区间 $I_{2801}$,相应的总模拟开销折合为功能模拟的等价开销为

$$2801+20\times1=2821.$$

由分配过程可以看出,MinEC 首先将区间分配给当前模拟开销最小的节点,因此总能保证各模拟节点的等价开销尽可能地接近,从而最大程度地保证了节点间的模拟负载均衡.




(a) 模拟器单向切换 (b) 模拟器双向切换

Fig.2 An example for MinEC

图 2 一个 MinEC 的例子

### 3.3 实 现

SEDSim 本身并不受限于具体模拟器,这里选择 SimpleScalar(版本为 3.0d release)中的乱序处理器性能模拟器 sim-outorder 对其进行实现.SEDSim 采用 Master/Slave 结构:Master 进程运行在 ServerNode 上,不执行实际模拟,只负责区间的分配、各 SimNode 模拟结果的接收以及汇总等;Slave 进程运行在 SimNode 上,运行模拟器对分配的指令区间进行模拟.SEDSim 首先将 sim-outorder 中的核心模拟代码进行封装:功能模拟和详细模拟分别封装为独立的函数;采用详细模拟实现预热(也作为一个独立的函数),与详细模拟不同的是,预热时不收集性能统计数据.SEDSim 增加的模块如图 3 所示,主要包括:

(1) 通信模块

SimNode 利用 socket 通信将局部性能数据上报给 ServerNode.

(2) 指令区间分配模块

根据对 SimNode 指定的序以及 $R$,$w$ 等参数实现基于 CoMEPA 和等区间的分割、循环分配以及 MinEC;根

据分配结果,可确定每个 SimNode 功能模拟、详细预热和详细模拟的指令数.

(3) 模拟模式的双向切换

sim-outorder 只支持从功能模拟切换到详细模拟,无法再从详细模拟切换回功能模拟;为实现双向切换,首先让模拟器进入等待切换状态,此时,取指部件不再为流水线取新的指令,已处于流水段中的指令随着时钟的推进继续完成详细模拟操作,流水段"排空"后重新取指,新的指令开始进行功能模拟,整个切换阶段完成.

(4) 数据统计模块

SEDSim 由 ServerNode 输出统计数据,SimNode 仅收集本地数据并上报给 ServerNode,ServerNode 需要区分不同的体系结构指标:

- 对于简单型指标(例如一级数据 cache 访问次数 *Dl*1.*accesses*),若 SimNode $i$ 模拟获得的局部统计量为 $A_i(1\leqslant i\leqslant N)$,则 ServerNode 只需计算 $\sum_{i=1}^{N}A_i$ 即可获得整个 Benchmark 程序的全局统计量;
- 对于由简单型指标 $B$ 和 $C$ 经过运算关系⊙获得的复合型指标 $D$,即 $D=B\odot C$(例如一级数据 cache 失效率 *Dl*1.*miss_rate* 等于失效次数 *Dl*1.*misses* 除以访问次数 *Dl*1.*accesses*,即 *Dl*1.*miss_rate*=*Dl*1.*misses*/*Dl*1.*acesses*),则 ServerNode 首先计算 $C=\sum_{i=1}^{N}C_i$ 和 $B=\sum_{i=1}^{N}B_i$,然后再计算复合型指标 $D=\sum_{i=1}^{N}C_i\odot\sum_{i=1}^{N}B_i$.

整个 SEDSim 实现的代码功能相对独立,与模拟器核心代码、抽样方法等无关.

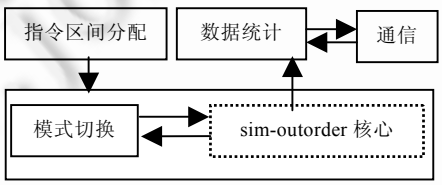



Fig.3 Main modules implemented by SEDSim

图 3 SEDSim 实现的主要模块

## 4 性能评估

### 4.1 理论分析

4.1.1 CoMEPA 理论分析

采用 CoMEPA 后,理论上各模拟节点执行时间相等,并行模拟执行时间可由任一节点执行时间计算,即

$$T'_{parallel}=[NI/(1+w_N)I_N]/V_f+(w_N+1)I_N/V_d.$$

定义理论相对加速比 TRS(theoretical relative speedup)为 $T'_{parallel}/T_{parallel}$,以分析 CoMEPA 相对于等区间策略时的性能改进.根据性能分析模型可知,$T_{parallel}=(N-1-w)I/V_f+(w+1)I/V_d$,因此,

$$TRS=[N+(R-1)(w+1)]/N[1+(1+w_N)a^N/(1-a^N)].$$

由于 $R>1$,$w$ 和 $w_N$ 总是大于等于 0,显然 TRS 值总是大于 1,即理论上 CoMEPA 总优于等区间策略.图 4 给出了 $w$ 取 30%、$R$ 分别取 10,20,30,40 时 *TRS* 随模拟节点数 $N$ 的变化情况.图 5 给出了 $R$ 取 20,$w$ 分别取 10%,30%,50% 时 TRS 值随节点数 $N$ 的变化情况.

在上述两种情况下,TRS 值至少约为 1.28,最多约为 2.1.此外,TRS 值的变化具有以下特征:

1) 随着 $N$ 的增加而降低.表明负载均衡对小规模并行宿主机的改进效果更明显,但同时说明即使改善了负载均衡,随着节点数的增加,也难以获得更高的并行加速比,且仍然不可避免并行效率的下降.
2) 随着 $w$ 的增加而增加.表明负载均衡策略能够较好地支持更高的预热比率,有利于用户提高模拟精度.
3) 随着 $R$ 的增加而增加.表明负载均衡策略能够更好地支持用户通过持续改进详细性能模拟模型而提高模拟精度.

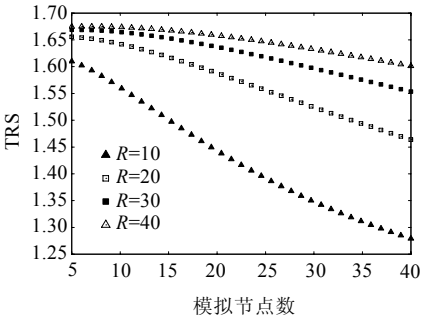



Fig.4 Variation of TRS with $R$ while $w$=30%

图 4 $w$=30%时,TRS 随 $R$ 的变化

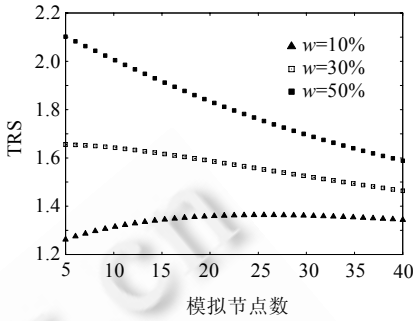



Fig.5 Variation of TRS with $w$ while $R$=20

图 5 $R$=20 时,TRS 随 $w$ 的变化

4.1.2 MinEC 理论分析

显然,对于给定的 $N$ 和 $P$,MinEC 能够获得理论上的最大加速比.此外,利用 MinEC 还可预先确定为获得最大加速比所需的最少的模拟节点数,因为最大加速比受限于等价距离最远的抽样区间,结束模拟的时间不可能小于快速推进到该区间并对其进行详细性能模拟的时间.可以不断增加模拟节点数反复运行 MinEC 算法,直到等价距离最远的抽样区间作为唯一一个分配到其中一个节点上的区间且该节点开销最大时结束,此时的节点数即为获得最大加速比所需的最少节点数.同样以图 2 的 ammp 程序为例,采用 MinEC 算法单向切换时,节点数分别为 2 和 4 时的分配结果如图 6 和图 7 所示.可见,两个节点时负载最重的为 SimNode 2,其开销为 3 615.而由图 2 可知,3 个节点时负载最重为 SimNode 1,其开销为 2 821,显然,3 个节点时的加速比更大;但若采用 4 个节点,负载最重的节点仍为 SimNode 1,加速比并未改变,因此,3 个节点便可获得最大加速比.

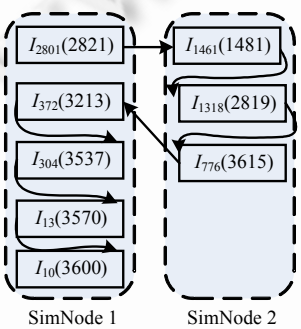



Fig.6 2 SimNodes, unidirectional transition

图 6 2 模拟节点单向切换

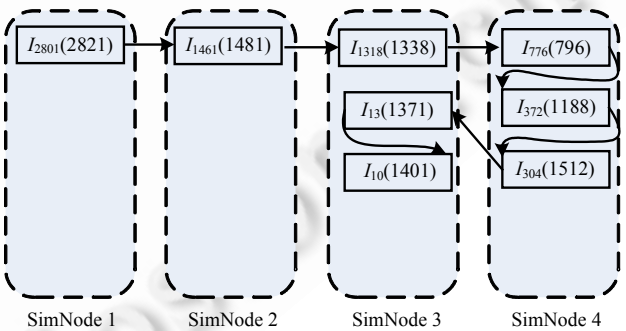



Fig.7 4 SimNodes, unidirectional transition

图 7 4 模拟节点单向切换

## 4.2 测 试

4.2.1 测试环境

并行宿主机环境为国防科技大学高性能应用中心的 1.5 万亿次集群系统,每个节点包括 1 个主频 3.3GHz 的双核 Intel Xeon 处理器,内存 4G,节点间以千兆以太网互连,操作系统为 RedHat Linux AS4.0.sim-outorder 采用默认的体系结构配置.测试程序(参见表 2)为专为 SimpleScalar 编译的 SPEC CPU2000 中的部分程序(采用 Alpha 指令集),输入集均为参考输入集.$R$ 通过 sim-outorder 快速推进 10 亿条指令后再详细模拟 10 亿条指令确定.由于 SPEC CPU2000 中的程序对于模拟而言相对庞大,因此又选取了 SimpleScalar 自带的小程序 anagram 用于测试并行模拟时预热对模拟精度的影响.此外,表 2 同时列出了测试 MinEC 算法所用到的 100M 大小的 SimPoint 区间,由 SimPoint 3.0 生成,可以参见其网站[24].

**Table 2** Benchmark programs and parameters

**表 2** 测试程序和参数

| 程序 | 输入集 | 指令数(×10$^9$) | $R$ 取值 | SimPoint 抽样区间(100M,起始索引为 0) |
|---|---|---|---|---|
| twolf | Ref | 346 | 10.2 | $I_0$, $I_{20}$, $I_{135}$, $I_{168}$, $I_{191}$, $I_{331}$, $I_{449}$ |
| gzip | Ref (graphic) | 103 | 8.4 | $I_{260}$, $I_{300}$, $I_{372}$, $I_{405}$, $I_{470}$, $I_{560}$, $I_{565}$, $I_{655}$ |
| vortex | Ref (lendian1) | 119 | 7.4 | $I_{78}$, $I_{127}$, $I_{303}$, $I_{360}$, $I_{428}$, $I_{535}$, $I_{846}$, $I_{936}$, $I_{1184}$ |
| equake | Ref | 132 | 7.1 | $I_0$, $I_{48}$, $I_{187}$, $I_{1011}$, $I_{1124}$ |
| lucas | Ref | 142 | 8.9 | $I_{207}$, $I_{209}$, $I_{380}$, $I_{597}$, $I_{887}$, $I_{1076}$, $I_{1212}$, $I_{1213}$, $I_{1268}$ |
| ammp | Ref | 326 | 12.3 | $I_{10}$, $I_{13}$, $I_{304}$, $I_{372}$, $I_{776}$, $I_{1318}$, $I_{1461}$, $I_{2801}$ |
| anagram | words, input.txt | 0.026 | 15.5 | – |

4.2.2 CoMEPA 测试结果

首先测试 CoMEPA 相对于等区间所能获得的实测相对加速比(realistic relative speedup,简称 RRS).RRS 由 $T_{EI}/T_{CoMEPA}$ 计算,其中,$T_{CoMEPA}$ 和 $T_{EI}$ 分别表示相同配置下采用 CoMEPA 和等区间时的并行模拟执行时间.图 8~图 10 分别给出了 $w$ 取 0,10%和 30%时的结果,其中,节点数分别为 5,10 以及 15.图中同时给出了相同数量节点时各测试程序 RRS 的最大值(max)、最小值(min)以及平均值(avg).此外,图中的细线给出了理论相对加速比与实测相对加速比之差的绝对值$|RRS-TRS|$.可见,RRS 的取值范围在 1.10~1.65 之间,但不同程序的 RRS 差别也较大,主要是由于各程序 $R$ 值的差异造成的.$w$ 取 30%时平均相对加速比最大,5 节点时为 1.60,10 节点时为 1.54,15 节点时为 1.47.此外,RRS 与 $R$,$w$ 以及节点数的相关性也很明显,例如:当 $w$=0 时,RRS 随节点数的增加而增加,表明采用 CoMEPA 并增加节点数加速效果会更明显;而 $w$=30%时,RRS 随着节点数的增加而迅速下降(尤其是 equake 程序),表明预热比率较大时,在大规模并行宿主机平台上采用 CoMEPA 效率较低.此外,所有情况下,RRS 与 TRS 均有一定程度的差距且基本上这种差距随着节点数增加而变大,表明采用 CoMEPA 后,尽管实现了理论上的模拟负载均衡,但运行中各节点仍然难以保证在相同时间内完成模拟,因为同一测试程序内部不同指令区间之间 $R$ 值的轻微波动、ServerNode 与 SimNode 之间的通信开销等,均可能造成模拟节点间实际执行时间的差异.

图 11 和图 12 给出了两种分配策略下预热比率对模拟精度的影响(EI 表示等区间分割,数字表示模拟节点数量).性能指标的绝对百分比误差(absolute percentage error) $M_{ape}$ 定义如下:设测试程序完整串行性能模拟时得到的性能指标为 $M_{serial}$,并行性能模拟时对应的性能指标为 $M_{parallel}$,则 $M_{ape}=|M_{serial}-M_{parallel}|/M_{serial}$.由于采用 sim-outorder 对 SPEC CPU2000 中的程序进行完整的性能模拟需要很长时间,这里仅选择了指令数相对较少的 vortex 程序,而通常情况下,处理器预热主要针对大型微体系结构部件如分支预测器、cache、TLB 等,因此这里重点给出了与上述 3 个部件有关的指标以及 IPC 的绝对百分比误差.可见,即使 $w$=0,误差也很小,IPC 的平均绝对百分比误差仅为约 0.05%,这比 SimPoint 等抽样模拟加速技术的误差要小约 3 个数量级.其原因在于,此时指令区间较大,“冷启动”的影响对整个指令区间而言很小.以 vortex 为例,其包含的指令数约为 119B(billion),而 SimPoint 通常设置的区间大小为 10M 和 100M,5 个节点时平均每个指令区间大小多达 24B,即使随着节点数的增加而导致区间减小(例如 15 节点时,vortex 的指令区间大小仍达 8B),此时模拟精度损失仍然微乎其微.此外,由于指令区间较大,区间分割和分配策略对模拟精度的影响也很小.更重要的是,DS 对于与统一二级 cache(UL2 cache)相关的性能事件(如命中 ul2.hits、失效 ul2.misses 等)也能获得较高的模拟精度:当 $w$=0 时,百分比误差最多也仅为 0.5%;当 $w$=10%时,最大误差则减小为 0.022%.由于程序中通常对 UL2 cache 的访问较少,因此对这类性能事件的精确抽样非常困难.文献[13]中报道采用单个 SimPoint 抽样区间获得的与 UL2 cache 相关的性能指标误差高达约 90%,即使采用多个抽样区间,误差也有 25%.由于 DS 方法完整模拟了整个程序,因而在这类体系结构指标方面相对于抽样模拟有较大的精度优势.

为了考察较小指令区间时预热对模拟精度的影响,我们还测试了 Simplescalar 自带的小程序 anagram.相对于 SPEC CPU2000 中的程序,该程序规模较小,约为 26M 指令(输入为随程序附带的数据文件 words 和 input.txt).图 13 给出了采用 CoMEPA 时 anagram 程序的 IPC 误差.可见,即使对于小规模程序,为了控制误差在 1%以内,也最多仅需约 30%的预热比率.

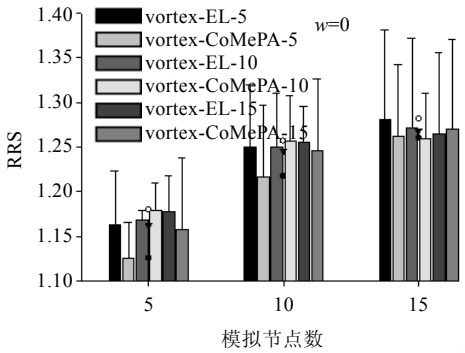



Fig.8 Realistic relative speedup while $w$=0

图 8 $w$=0 时的实测相对加速比

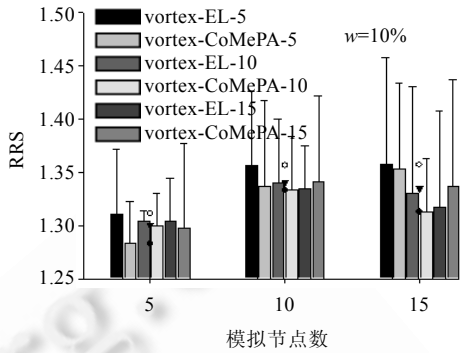



Fig.9 Realistic relative speedup while $w$=10%

图 9 $w$=10%时的实测相对加速比

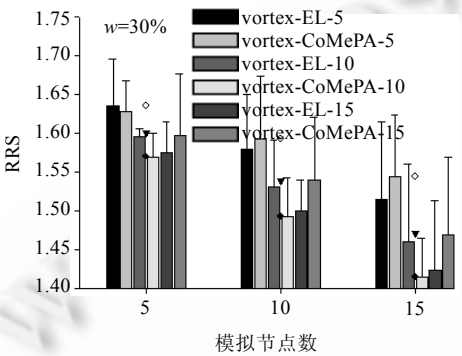



Fig.10 Realistic relative speedup while $w$=30%

图 10 $w$=30%时的实测相对加速比

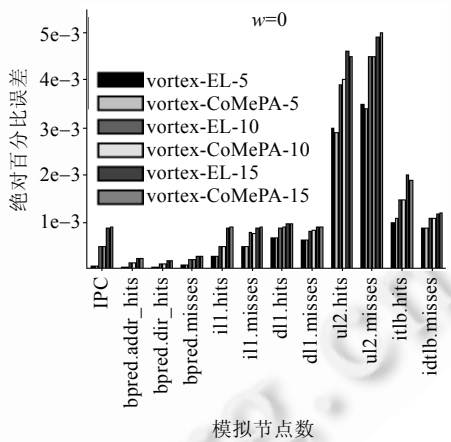



Fig.11 Absolute percentage error while $w$=0

图 11 $w$=0 时的绝对百分比误差

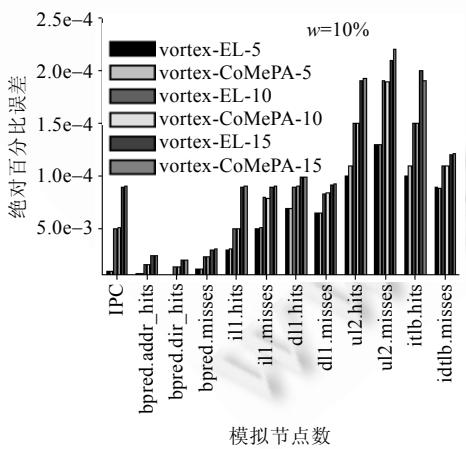



Fig.12 Absolute percentage error while $w$=10%

图 12 $w$=10%时的绝对百分比误差

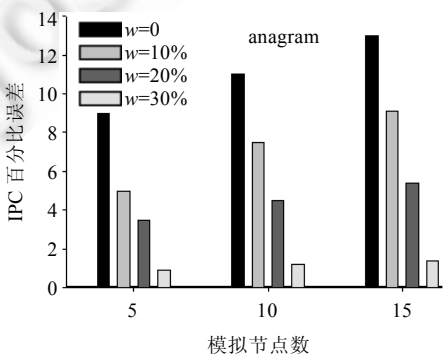



Fig.13 Absolute percentage error for anagram with different $w$s

图 13 $w$ 不同时,anagram 程序的绝对百分比误差

4.2.3 MinEC 测试结果

图 14 和图 15 分别给出了模拟器采用单向切换和双向切换时,MinEC 相对于在单个处理器上串行模拟所有 SimPoint 区间的加速比.由图 14 可见:单向切换时,MinEC 可获得最大加速比为 5.58(6 节点 lucas),平均最大加速比为 3.96;而双向切换时最多仅能获得 1.12 的加速比(2 节点 twolf).原因在于:模拟器双向切换时无需重新开始推进到后续 SimPoint 区间,因此并行效果并不明显;但单向切换时 MinEC 加速效果非常明显,尤其是节点数较少时,几乎成线性加速比.例如,对所有程序,2 节点时加速比平均达到 1.97;即便是 3 节点时,多数程序(除了 equake 和 ammp)的加速比仍然高达 2.8 以上,平均达 2.64.此外,对所有程序而言,当节点数增加到一定规模后,不仅并行效率下降,而且也无法再提高加速比,这与第 4.1.2 节的分析一致.其中,gzip 在 7 节点时达到最大加速比,而 ammp,equake,twolf 在 3 节点时即可达到最大加速比.

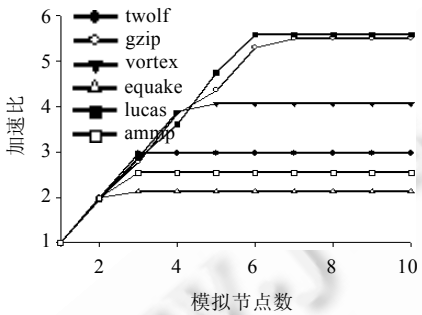



Fig.14 Speedup under unidirectional transition

图 14 单向切换时的加速比

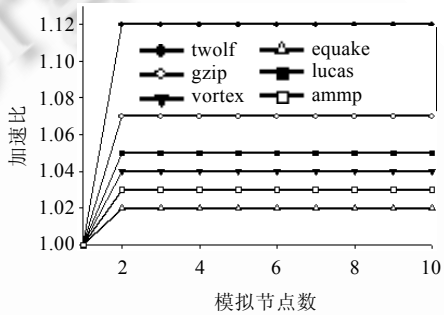



Fig.15 Speedup under bidirectional transition

图 15 双向切换时的加速比

图 16 和图 17 分别给出了模拟器单向切换和双向切换下相同数量模拟节点时,MinEC 相对于循环分配策略的加速比.这里,循环分配策略按照抽样指令区间非连续索引由小到大的顺序依次分配到模拟节点.例如,对于 twolf 程序,2 个模拟节点时,节点 1 分配 $I_0$,$I_{135}$,$I_{191}$,$I_{449}$,而节点 2 分配 $I_{20}$,$I_{168}$,$I_{331}$.可以看出,MinEC 仍然能够提高性能:单向切换情况下最多可获得 1.41 的加速比,4 节点时平均加速比最大,约为 1.19;gzip 程序加速效果最明显,平均加速比达 1.23;双向切换时加速效果相对不太明显,最多仅为 1.06.主要原因在于:简单的循环分配策略在分配指令区间时并未考虑模拟节点已有的负载,有可能加重负载不均衡.

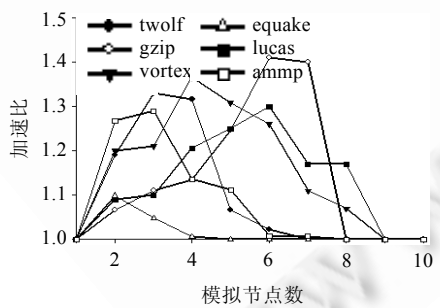



Fig.16 Speedup compared to cyclic allocation under unidirectional transition

图 16 单向切换时相对于循环分配的加速比

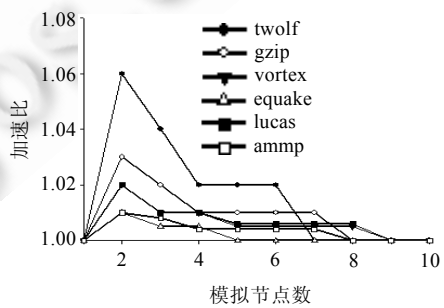



Fig.17 Speedup compared to cyclic allocation under bidirectional transition

图 17 双向切换时相对于循环分配的加速比

## 5 结束语

本文首先对分布式并行模拟建立了通用性能分析模型.利用该模型对典型系统的并行加速比、并行效率等性质进行了理论分析,得出了一些指导分布式并行模拟方法优化设计的结论.在此基础上提出并实现了均衡可

扩展分布式并行模拟方法 SEDSim.具体包括:

1) 开销模型指导的指令区间均衡分割和分配策略 CoMEPA,通过更好的负载均衡提高了分布式并行模拟的加速比;
2) 基于最小等价距离的指令区间分配策略 MinEC,使得分布式并行模拟能够与非连续、任意数量抽样模拟区间高效集成;
3) 基于 sim-outorder 实现了 SEDSim,并采用 SPEC CPU2000 进行了测试.

理论分析和测试结果均表明:相对于常用的方法或策略,CoMEPA 和 MinEC 分别能够获得多达约 1.6 倍和 1.4 倍的性能提升.

我们下一步的工作是:考虑在不改变模拟节点间耦合关系的条件下,设计动态运行时的负载均衡方法.

**References**:

[1] Austin T, Larson E, Ernst D. SimpleScalar: An infrastructure for computer system modeling. IEEE Computer, 2002,35(2):56−67. [doi: 10.1109/2.982917]

[2] http://www.simplescalar.com

[3] Zhang FX, Zhang LB, Hu WW. Sim-Godson: A godson processor simulator based on SimpleScalar. Chinese Journal of Computers, 2007,30(1):68−73 (in Chinese with English abstract).

[4] Gao X, Zhang FX, Tang Y, Zhang LB, Hu WW, Tang ZM. SimOS-Goodson: A goodson-processor based multi-core full-system simulator. Ruan Jian Xue Bao/Journal of Software, 2007,18(4):1047−1055 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1047.htm [doi: 10.1360/jos181047]

[5] Yu ZB, Jin H, Zou NH. Research on computer architecture software-based simulation. Ruan Jian Xue Bao/Journal of Software, 2008,19(4):1051−1068 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1051.htm [doi: 10.3724/SP.J.1001.2008.01051]

[6] Yi. J, Kodakara S, Resit S, David J, Douglas M. Characterizing and comparing prevailing simulation techniques. In: Proc. of the 11th Int'l Symp. on High-Performance Computer Architecture. San Francisco: IEEE Computer Society, 2005. 266−277. [doi: 10.1109/HPCA.2005.8]

[7] Yu ZB, Jin H, Chen J, John LK. TSS: Appling two-stage sampling in micro-architecture simulations. In: Proc. of the Modeling, Analysis & Simulation of Computer and Telecommunication Systems. London: IEEE Computer Society, 2009. 1−9.

[8] Yan Q, Zhang WH, Liu LL, Zang BY, Zhu CQ. A metadata-driven optimization for sampling simulation. Chinese Journal of Computers, 2008,31(11):1986−1994 (in Chinese with English abstract).

[9] Girbal S, Mouchard G, Cohen A, Temam O. DiST: A simple, reliable and scalable method to significantly reduce processor architecture simulation time. In: Cheng B, ed. Proc. of the ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems. San Diego: ACM Press, 2003. 1−12. [doi: 10.1145/781027.781029]

[10] Yano M, Takasaki T, Nakashima H. An accurate and efficient time-division parallelization of cycle accurate architectural simulators. In: Proc. of the 40th Annual Simulation Symp. Norfolk: IEEE Computer Society, 2007. 247−255. [doi: 10.1109/ANSS.2007.9]

[11] Srinivasan R, Cook J. Fast, accurate micro-architecture simulation. In: Proc. of the Applications for a Changing World, ITEA Modelling & Simulation Workshop. Las Cruces: ITEA, 2003. 23−29. [doi: 10.1109/ISPASS.2005.1430569]

[12] Eeckhout L, De Bosschere K. Efficient simulation of trace samples on parallel machines. Parallel Computing, 2004,30:317−335. [doi: 10.1016/j.parco.2004.02.003]

[13] Perelman E, Hamely G, Michael V. Using SimPoint for accurate and efficient simulation. In: Cheng B, ed. Proc. of the ACM SIGMETRICS Conf. Measurement and Modeling of Computer Systems. San Diego: ACM Press, 2003. 318−319. [doi: 10.1145/781027.781076]

[14] Liu W, Huang M. EXPERT: Expedited simulation exploiting program behavior repetition. In: Proc. of the Int'l Conf. on Supercomputing. Malo: ACM Press, 2004. 126−135.

[15] Heidelberger P, Stone HS. Parallel trace-driven cache simulation by time partitioning. In: Proc. of the Winter Simulation Conf. Piscataway: IEEE, 1990. 734−737. [doi: 10.1109/WSC.1990.129605]

[16] Nguyen A, Bose P, Wellman J. PARSIM: A parallel trace-driven simulation facility for fast and accurate performance analysis studies. In: Proc. of the IEEE Int'l Performance, Computing and Communication Conf. Phoenix: IEEE Computer Society, 1997. 291−297. [doi: 10.1109/PCCC.1997.581530]

[17] Zheng GB, Kakulapati G, Kalé LV. BigSim: A parallel simulator for performance prediction of extremely large parallel machines. In: Proc. of the 18th Int'l Parallel and Distributed Processing Symp. (IPDPS). Santa Fe: IEEE Computer Society, 2004. 89−99. [doi: 10.1109/IPDPS.2004.1303013]

[18] Prakash S, Bagrodia RL. MPI-SIM: Using parallel simulation to evaluate mpi programs. In: Proc. of the IEEE Winter Simulation Conf. Piscataway: IEEE Computer Society, 1998. 467−474. [doi: 10.1109/WSC.1998.745023]

[19] Huang YQ, Li HL, Xie XH, Qian L, Hao ZY, Guo F, Zhang K. ArchSim: A system-level parallel simulation platform for the architecture design of high performance computer. JCST, 2009,24(5):901−912. [doi: 10.1007/s11390-009-9281-9]

[20] Cao Z, Xu JW, Chen MY, Zheng G, Lü HW, Sun NH. HPPNetSim: A parallel simulation of large-scale interconnection network. In: Proc. of the SpringSim. New York: Society for Computer Simulation Int'l, 2009. 43−51. [doi: 10.1145/1639809.1639843]

[21] Chidester M, George A, Radlinski M. Multiple-Path execution for chip-multiprocessors. Technical Report, HCS Research Laborotary, Department of Electrical and Computer Engineering, University of Florida, 2001.

[22] Rosenblum M, Stephen A, Emmett W, Anoop G. Complete computer system simulation: The SimOS approach. IEEE Parallel and Distributed Technology, 1995,3(4):34−43. [doi: 10.1109/88.473612]

[23] Lantz B. Parallel SimOS: Performance and scalability for large system simulation [Ph.D. Thesis]. Stanford: Stanford University, 2007.

[24] http://cseweb.ucsd.edu/~calder/simpoint/multiple-standard-simpoints.htm

**附中文参考文献**:

[3] 张福新,章隆兵,胡伟武.基于 SimpleScalar 的龙芯 CPU 模拟器 Sim-Godson.计算机学报,2007,30(1):68−73.

[4] 高翔,张福新,汤彦,章隆兵,胡伟武,唐志敏.基于龙芯 CPU 的多核全系统模拟器 SimOS-Goodson.软件学报,2007,18(4):1047−1055. http://www.jos.org.cn/1000-9825/18/1047.htm [doi: 10.1360/jos181047]

[5] 喻之斌,金海,邹南海.计算机体系结构软件模拟技术研究.软件学报,2008,19(4):1051−1068. http://www.jos.org.cn/1000-9825/19/1051.htm [doi: 10.3724/SPJ.1001.2008.01051]

[8] 严强,张为华,刘力力,臧斌宇,朱传琪.一种基于元数据的采样模拟技术优化计.计算机学报,2008,31(11):1986−1994.

**徐传福**(1980－),男,安徽六安人,博士,助理研究员,CCF 会员,主要研究领域为并行计算.
E-mail: xuchuanfu@nudt.edu.cn

**王正华**(1962－),男,博士,教授,博士生导师,主要研究领域为并行计算.
E-mail: zhhwang@nudt.edu.cn

**车永刚**(1973－),男,博士,副研究员,CCF 高级会员,主要研究领域为并行计算.
E-mail: ygche@nudt.edu.cn

**彭宇行**(1963－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为并行与分布系统软件.
E-mail: pengyuxing1963@yahoo.com.cn