manycore系统的发展，预计到十年底将具有数百个核心，需要有效的设计空间探索和调试解决方案。 在相关的现有解决方案中，众所周知的gem5模拟器提供了丰富的架构描述框架。 然而，这些特征是以极高的模拟时间为代价的，这将限制了可能探索到数十个核心的配置的范围。 为了解决这一局限，本文提出了一种新颖的跟踪驱动模拟方法，用于有效地探索manycore架构。

一，引言

为了有效地探索新兴的人鱼系统，灵活且可扩展的模拟器是强制性的。 这种模拟器应该结合有效的建模，调试和仿真功能来验证软件开发，同时满足平台的预期功能和性能目标。 可用的工业和学术模拟器在模拟速度/精度权衡方面有所不同，它们的采用通常由所需的勘探水平确定。 准循环精确模拟器[1]对于建筑勘探是受欢迎和有吸引力的。 gem5环境是一种流行的开源准周期精确模拟器[2] [3]，支持丰富的指令集架构（ISA）。 它有一个积极的发展和支持社区。

即使启用灵活且详细的微架构评估，gem5需要较慢的模拟速度，从而将其适用范围限制在具有数百个内核的系统中。 这需要提供高仿真速度的替代方法，同时保持对建筑探索至关重要的准确性。

所提出的工作旨在为事件驱动的计算机体系结构模拟器（如gem5）提出一种新颖的可扩展跟踪仿真方法。 该方法的基本原理在于通过将核心执行抽象为跟踪来降低仿真复杂度，如下所示：

1. 在全系统仿真中捕获核心执行跟踪，即进/出存储器事务;
2. 这些轨迹用同步语义进行增强，然后复制到所谓的增强矢量轨迹中，以模拟由较高核心数组成的系统;
3. 增强的矢量轨迹通过交通注射器重播到最终的gem5模拟中; 实际上仅模拟互连和存储器子系统，从而导致显着的性能提升。

这种方法在gem5中实现，并在ARM ISA上进行验证（但是在其他ISA上运行）。 已知的局限性在于该方法的跟踪驱动性质：线程被固定到核心，并且运行时功能（例如负载平衡）不会进一步建模。 我们的解决方案通过其精确模拟基于M核心的计算机体系结构的能力，可以准确模拟基于M包N核心的参考模拟中捕获的踪迹的能力，这得益于跟踪复制 。

在本文的其余部分，第二部分讨论相关工作。 第三部分提供了跟踪驱动方法的基本概念，其开发和集成到gem5中。 第四节对我们的模拟方法的加速，精度和成本进行了广泛的评估。 最后，第五节指出结论和未来工作。

二， 相关工作

已经对有效的系统仿真研究了几种技术，并将它们分为三大类。 第一个家庭关系到依赖于即时（JIT）动态二进制翻译的模拟器，例如。 OVP [4]和QEMU [5]。 这样的模拟器可以以有限的精度为代价实现接近千个MIPS [6]的速度。 他们往往侧重于功能验证，而不是建筑探索。 然而，在[7]中已经进行了一些应用于建筑探索的努力。 这些方法错过了与缓存层次结构和一致性协议相关的表达式建模支持。

第二个家庭强调在多台主机上分布仿真[8]。 Graphite [9]是处理功能行为的分布式模拟器。 它通过抽象化沿模拟的事件排序来最小化同步开销。 因此，在降低模拟成本的同时，这种轻松的同步视觉限制了架构探索，如通信瓶颈。 ZSIM [6]通过在x86多核主机上并行仿真来提高模拟速度。 作者声称比Graphite和gem5高出约2/3和4个数量级的加速。 更一般地说，使用分布式模拟器是微妙的，因为用户必须仔细地处理可用CPU之间的模拟分区和同步，这限制了模拟加速。

第三个系列集中于在事务级建模（TLM）[10]之后或通过使用跟踪驱动模拟[11]来优化组件描述（例如CPU，互连基础架构）在减少模拟事件的数量的同时，由于缺乏准确的微体系结构建模功能，TLM对架构探索的使用受到严重的惩罚。 另一方面，跟踪驱动建模是降低模拟成本的高性能和嵌入式计算的相关方法。 在[12]中，64位系统的tracedriven仿真平均达到了150x模拟速度w.r.t参考的基于Simics的模拟器[13]。 [14]中的作者将网络仿真器与基于跟踪的MPI模拟器相结合，探讨了互连网络对并行应用性能的影响。 在[15]中，跟踪驱动的注射器可以根据应用程序的流程图驱动片上网络（NoC）的优化。 这种方法并不关注微架构问题。 另一个跟踪驱动的模拟器是TaskSim [11]，致力于设计共享内存多核系统的空间探索。 它集中在多线程应用程序行为而不是微体系结构方面。

我们的贡献不同于以前的所有作品，提出了一种新颖的跟踪驱动模拟方法，适用于对manycore架构的有效探索。 其优势之一是线程同步的源级依赖关系管理和跟踪复制技术。 它能够正确地考虑到线程导向的API（如POSIX线程和OpenMP）的控制和数据依赖性。

三， 追踪方法

提出的跟踪驱动（TD）仿真方法由三个阶段组成，如图1所示：（a）跟踪收集，（b）跟踪处理和（c）跟踪模拟。

第一阶段专门用于追踪收集。我们将目标系统定义为硬件架构和软件系统。在图1（a）中，硬件架构包括N个内核，每个内核都有其私有指令（I $）和数据（D $）高速缓存。核心和外部存储器之间的通信通过可包括高速缓存（L2，L3）的任意通信基础设施来实现。软件系统包括应用程序代码和操作系统。由于建模系统是共享内存类型，所有通信都通过内核位置进行，这些内存位置正在内核上执行的代码访问。图1描述了通过通信和存储器基础设施作为请求和响应事件流的通信。跟踪的结构包含内存访问类型（读取或写入），目标地址，数据值，请求发送和满足的时间。图1（a）所示的跟踪收集接口（TCI）跟踪并存储所有请求和响应跟踪。它位于每个专用高速缓冲存储器的接口处。这允许捕获关于核心侧存储器请求和对应的存储器侧响应的信息。作为收集阶段的全局输出，获得由每个核心的一个跟踪文件组成的跟踪集。只有与缓存未命中事件相关的跟踪信息才会保留，只能导致外部存储器流量。

为了确定TD方法的增益，我们从两个角度考虑模拟过程：全事件驱动模拟和相应的TD模拟。 这些观点如图2所示，其中目标架构由对应于核/ TI，专用L1高速缓冲存储器，互连部件和外部存储器的四个通信层组成。 时间间隔T1和T2分别表示在缓存未命中和高速缓存命中的情况下的诱导持续时间。 根据事件驱动的范式，我们假设全事件驱动仿真和跟踪驱动仿真的仿真时间与模拟事件的数量成比例。 因此，使用提出的TD方法在模拟时间方面的增益取决于在跟踪收集阶段收集和过滤的事件数。

让我们考虑根据图1收集的参考轨迹。 假设目的在于确定存储器性能的影响，与用于跟踪收集的参考模拟中记录的时间线相比，在TD仿真的不同时刻可能会发生存储器响应。 这在图2（c）中示出，其中TI在时间t1发出请求，其在时间t'2到达存储器，而在参考模拟中，该请求预期在t2（由具有虚线边界的子弹表示） 。 这意味着推迟1）在时间t'3而不是t3的对喷射器的响应的时移，以及2）下一个TI注入时间。 这是通过处理来自TI的响应和下一个请求之间的时间间隔而不是绝对时间来处理的。

集成在gem5中

在gem5中实现我们的TD模拟方法遵循前三个阶段。

跟踪收集和处理。 第一阶段是同时处理的。 跟踪收集包括两个重要方面：硬件架构和软件系统。 在硬件结构方面，为了实现TCI，gem5已经具有必要的功能，它是其基于跟踪的调试的一部分，其中包含DPRINTF语句。 它提供了一种灵活的文本格式文件，在某些情况下可能在空间和处理时间方面是次优的。 幸运的是，诸如编码和压缩等众所周知的优化技术解决了这个问题。

gem5内存子系统基于端口，数据包和请求/响应的概念。 端口将内存组件相互连接。 请求和响应用于封装CPU或I / O设备消息[2]。 我们的跟踪收集界面的实现依赖于这些概念。 图3（a）显示了收集的迹线的提取。

在软件系统方面，我们考虑多线程编程模型，其中多个线程存在于同一进程中并共享资源，例如内存。 在一个manycore系统中，每个核心同时执行一个单独的线程。 为了处理资源共享，线程配备了同步机制。 同步点（障碍和连接）引入了通过跟踪同步机制解决的控制流依赖关系。

我们的同步机制的第一步是收集附加的同步跟踪。 为此，我们添加宏以输入应用程序源代码，以捕获同步点。 gem5指定操作rpns（）以内部时间计数的格式返回已调用的时间，例如 蜱虫 我们基于POSIX线程API和gem5的rpns（）操作创建了三个宏CREATE，BARRIER和JOIN。 用这种宏注释的应用程序的执行输出如图3（b）所示。 这使得可以自动地附加在稍后使用的执行跟踪同步信息中。

跟踪驱动模拟。 对于增强矢量轨迹的模拟，在gem5模拟器中实现了两个主要组件：（i）跟踪注射器和（ii）跟踪仲裁器。

TI由两个基本模块组成：第一个专用于解析跟踪文件并提供请求/响应结构，而第二个触发器请求注入和时间间隔管理，即根据响应时间对即将到来的请求进行时移。

第二个重要组成部分是追踪仲裁器。 它连接到每个注射器，并具有整个系统的全局视图。 仲裁器处理通过宏收集的依赖关系。 仲裁器和注射器之间的通信通过简单的协议实现。 TI有两个基本状态：LOCK和UNLOCK。 初始化后，它被解锁并读取跟踪文件。 一旦解析器到达跟踪中的同步条目，则注入器将相应的信号发送到仲裁器。 仲裁器将进样器设置为LOCK状态，并等待直到所有其他喷油器达到相同的同步点。 一旦发生，锁定的仲裁器解锁所有注射器，并继续进行仿真。 仲裁器和注射器之间的消息交换由原子包传送，不会影响运行时间。 仲裁器和注射器连接如图4所示。

一个额外的修改涉及缓存。 由于TD模拟以冷（空）存储开始，因此会导致一种偏差。 这个问题通常被称为冷启动偏差[16]。 对于私有缓存，我们利用所有内存请求都是高速缓存未命中的事实，并将其声明为TD模拟。 该解决方案消除了冷启动错误，但带来了新的限制：通过一个缓存大小收集的每个单独的跟踪文件不能与其他缓存大小一起模拟。 对于其他级别的缓存，例如 L2，有几种方法，在[16]中讨论，减少了问题。

一个重要的跟踪模拟场景是跟踪复制。 目的是从N核心系统中收集踪迹，然后复制并模拟M-N系统，其中M> N。当给定的N核心系统执行计算阶段有时间的N线程应用时，这种复制是适用的。 在这种情况下，可以复制应用程序行为模式来模拟M-core系统，从而增加应用程序问题的虚拟大小。 我们的技术的一个主要特点是直接访问内存参考地址字段，允许TI应用绕过编译器和应用程序源代码的智能存储器映射。

**限制。** 我们的做法有三个主要的限制。 第一个与核心和L1缓存抽象有关。 它需要收集痕迹并改变其配置，例如 核心频率，L1缓存大小等。第二个约束涉及考虑的应用程序的性质。 我们的跟踪复制方法不支持计算阶段不能被静态界定的应用程序，例如 数据相关的应用程序，这个阶段取决于输入数据。 第三个约束涉及模拟主机功能。 实际上，一些迹线可能需要几十GB的磁盘空间和数十GB的操作内存，用于512个核心仿真。 因此，所考虑的机器应提供足够的存储空间。

四， TRACE-DRIVEN 方法评估

我们通过对模拟过程的详细分析来验证我们的TD方法。 我们的参考平台（RP）的特点是一组参数：（i）1，2，4，8核处理器（500 MHz），（ii）4 kB私有L1 D / I高速缓存，（iii ）64位通道宽度，（iv）30 ns DDR内存延迟和（v）Linux内核2.6.38。 使用了gem5的ARM按顺序ISA模型。 处理器通过总线连接到DDR存储器。

我们考虑在POSIX线程中实现的一系列科学和多媒体计算领域的应用。 其中一些来自SPLASH-2基准测试套件[17]：由于在相应算法中存在多个依赖关系，所以与Radix，Barnes，LU和Ocean相关。 此外，我们还采用了运动JPEG（MJPEG），有限脉冲滤波器（FIR），史密斯沃特曼（SW），直方图计算直方图，合并分类，用于模拟粒子动力学系统的N体，向量减少 ，矢量运算（VO）和快速傅里叶变换（FFT）。

考虑的RP被配置为在gem5的全系统（FS）模式的时候运行上述基准。 我们通过研究在执行资源（主要是内存）方面的模拟加速，准确性和成本方面的增益来深入了解我们的TD方法。

1. 加速和精度的评估

跟踪文件从上述RP采集到8个核心，然后在我们的TD模拟器中执行，没有任何架构变化。 表I总结了仿真结果。 这些包括gem5 FS和TD模拟的仿真和应用执行时间（ET），生成的跟踪文件的大小，加速度的增益和误差百分比。

由TD模拟提供的结果表明，加速度的增益在6x800x的相当宽的范围内，这取决于应用性质，例如， 计算通信比率。 获得的轨迹尺寸和TD模拟误差分别在40Mb〜11.6Gb和0.02〜5.79％之间。

B.建筑参数探索

我们通过在跟踪收集阶段之后改变内部架构参数来评估TD模拟一致性。 使用5个内存延迟值：5ns，15ns，30ns，45ns和55ns。 对两种应用（MJPEG和FFT）执行时间的影响进行了评估，如图5所示。 结果显示，执行时间误差在6％左右。 因此，即使架构参数发生 变化，我们的TD模拟也能正确地再现应用行为。

在另一个实验中，我们使用捕获的跟踪来评估系统配置，包括在原始跟踪收集阶段不存在的组件。 为了说明这一评估，从我们的RP收集跟踪，八个运行MJPEG的内核，然后转移到一个TD系统，其中包括所有TI之间共享的L2缓存。

在这里，为了尽量减少执行时间的错误，我们必须解决第三章B节中讨论的冷启动偏差问题。 我们建议通过考虑应用程序执行阶段之前捕获的跟踪来预热二级缓存。 我们收集并比较三条迹线：（i）执行时间（ET）迹线，（ii）具有初始化相位跟踪的ET，以及（iii）ET +初始化和OS引导阶段。 结果如图7所示。 （i）中的痕迹不能提供预期的性能改进，并导致平均绝对误差的14.01％。 通过使用（ii）中的轨迹，我们观察到加速，平均获得7.88％的绝对误差，是（i）提供的2倍。 （iii）中的痕迹平均给出6.60％的绝对误差，这是最好的。

C.一致性模拟存在的依赖性

我们评估TD方法的错误百分比w.r.t. 在这种情况下，gem5 FS模拟应用程序由它们之间的依赖关系的多个线程组成，例如由同步引起的。 我们为解决这一问题而定义的机制已在第三节B节中提出。 这在整数基数排序内核中进行了说明。 在跟踪收集阶段，RP包含4个内核，Radix内核由4个线程执行。 对于每个核心，在给定时隙期间观察到的高速缓存未命中的数目和诱导误差如图6所示。 发生同步障碍的时刻用虚线垂直线突出显示。 我们观察到障碍发生的错误的波动。 然而，在整个仿真期间，观测到的误差峰值得到补偿，因此累积误差仅为1.39％。

D.大规模仿真的跟踪复制

我们评估模拟在比用于收集此跟踪集的处理元素更多的处理元素上的给定跟踪集的复制的机会。 通常，从2核的gem5 FS模拟获得的曲线可以在数十或数百个核心上复制，并且通过所提出的TD扩展更快且更准确地模拟。 关于核心缓存未命中行为的初步观察是它们非常相似（见图6）。 所有内核的高速缓存未命中都遵循相同的内存访问时间模式。 然后，可以将这样的模式用作在更多核心之间复制的可能的跟踪模板。 我们通过考虑MJPEG应用来探索识别这种模式的不同场景，其中从以下方面收集踪迹：（i）1核RP，（ii）2核RP，（iii）4核RP和（iv） 核心RP。

从这些情况获得的踪迹在具有8个TI的平台上进行复制。 我们使用归一化相关函数来产生准确的估计。 通过8核gem5 FS模拟获得的高速缓存未命中数与通过上述四种场景获得的高速缓存未命中数相关。 获得的每个场景的相关系数如表II所示。 这些结果表明，三种情况提供非常相似的行为，无论使用的核心数量如何。 与FS模拟相比，产生的执行时间误差低于4％。 因此，双核心情景与足够数量的注射器有针对性的复制目标相关。 我们使用上述复制模式来模拟由多达512个内核组成的manycore架构。 核心数量受主机功能的限制。 建筑勘察结果超出了本文的范围。

为了改善相关系数，我们研究了应用问题大小对这些系数的影响。 我们专注于四个应用：Radix，LU，Ocean和Barnes。 对于每个这些应用程序，我们选择了三个问题大小，其中Size 1 <Size 2 <Size 3。这里，size指处理的数据量。 然后，我们在RP上收集了四个核心和四个线程的相应跟踪。 表III显示了根据三个问题大小为每个应用程序跟踪计算的平均相关系数。 注意与较大的问题大小相关的显着增加，源于对存储器子系统的压力增加：随着时间的推移，高速缓存未命中率的较高动态导致更突出的相关性。

跟踪驱动模拟成本

我们在这里分析主机上的模拟工作的故障，包括跟踪注入器，缓存，总线，内存和gem5模拟器。 单核心全系统仿真也供参考。 从实现的实验中，我们观察到可以如图8所示的类似分布的MJPEG解码器。 分析使用标准gprof专业工具1执行。 TD模拟的大部分内容都是缓存一致侦听协议的实现。 由于目标系统是基于总线的，所以存储器业务拥塞和性能下降是由增加的处理器和注入器引起的。

五，结论与未来工作

在本文中，我们提出了一个着名的gem5模拟器的扩展，并提供了一个跟踪驱动的方法，以便对manycore架构进行有效的探索。 与当前的gem5版本相比，我们的解决方案提供了一个快速准确的模拟，同时保留了gem5的所有设计功能。 已经报道了一些实验，显示了我们的跟踪驱动扩展的所有好处：通过跟踪复制，加速和准确性的可扩展性。 得到的结果表明，仿真速度比定时gem5全系统模式高出800倍，而实现的精度从0.02％变化到6％。 我们的实施旨在免费提供在线2，用于manycore架构探索，同时减少模拟精度和性能之间的差距。

未来的工作包括通过解决有关跟踪文件优化，乱序处理器支持和不同内存映射算法的问题来改进当前的跟踪模拟器。 我们也计划支持其他

1http：//sourceware.org/binutils/docs/gprof 2http：//www.lirmm.fr/ADAC

支持对GPU等复杂计算机系统进行架构探索的编程模型。

六， 致谢

导致这些成果的研究已经获得了蒙特利尔项目下的欧洲共同体第七框架计划（FP7 / 2007-2013）的资助：http://www.montblanc-project.eu，授权协议号288777。