清华大学博士学位论文报告

面向多核架构的操作系统可扩展性研究

Research on Scalability of Operating Systems on Multi-Core Platforms

姓　名：崔　岩

导　师：史元春　教　授

　　陈　渝　副教授

学　号：2007310406

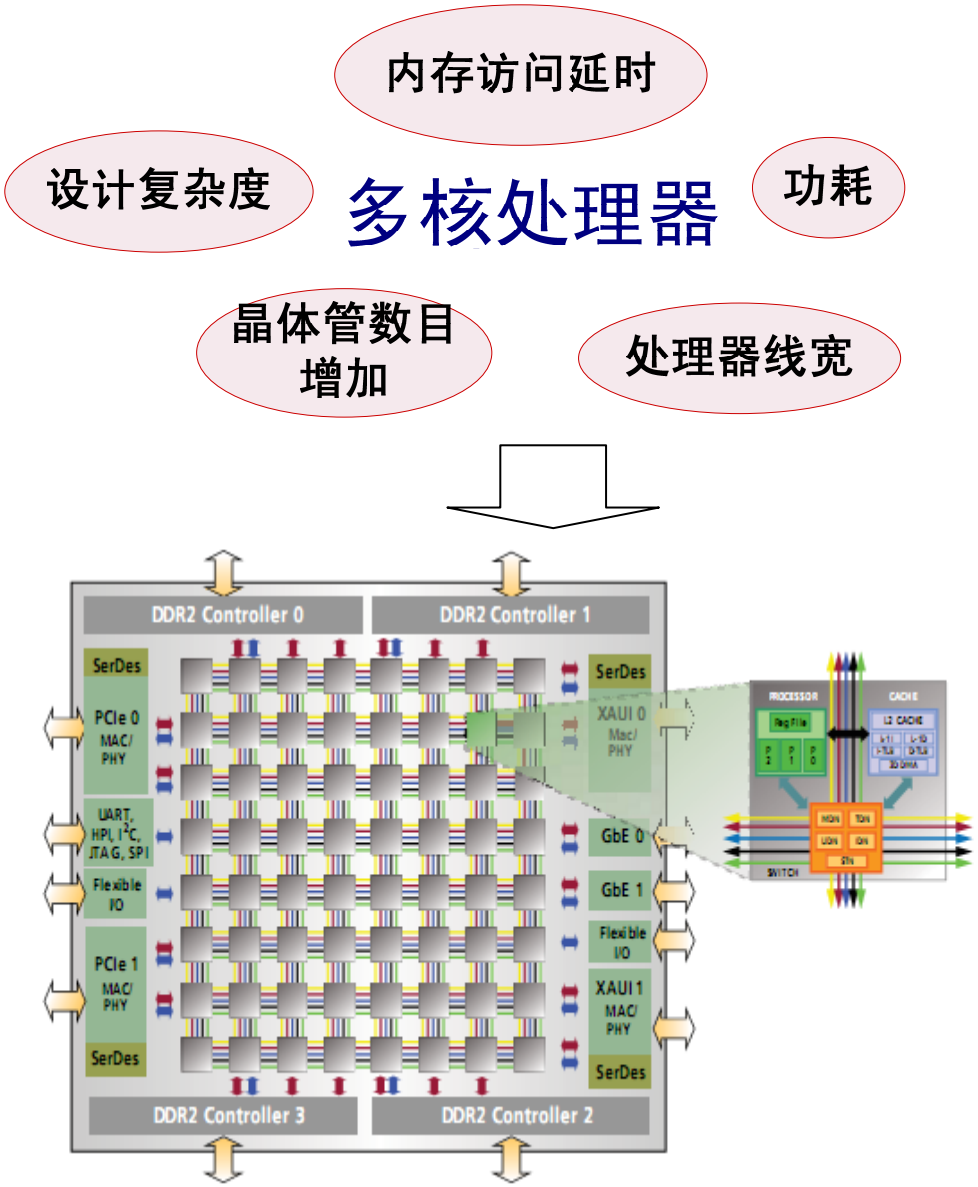
提纲

1. 选题背景
2. 论文工作
   - 分析系统服务接口、发现制约因素
   - 锁颠簸现象的模拟和避免
   - 共享硬件资源竞争降低
   - 可扩展性瓶颈定位方法
3. 总结
4. 研究成果

提纲

1. 选题背景
2. 论文工作
   - 分析系统服务接口、发现制约因素
   - 锁颠簸现象的模拟和避免
   - 共享硬件资源竞争降低
   - 可扩展性瓶颈定位方法
3. 总结
4. 研究成果

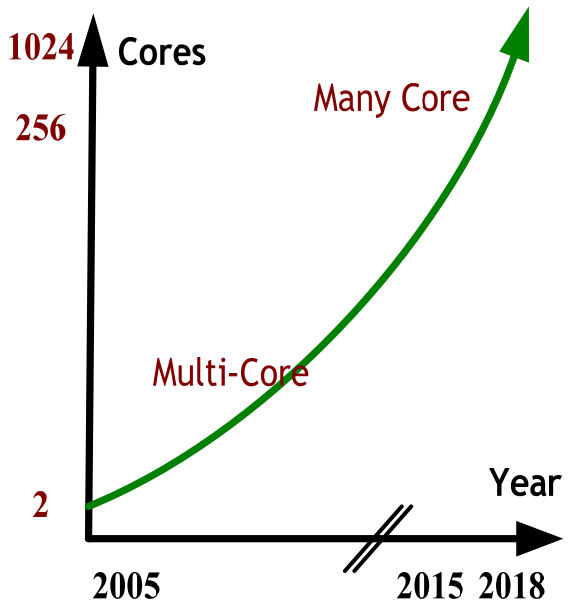
多核挑战

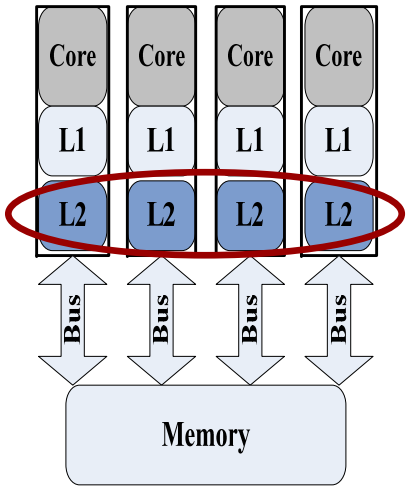


- 工业界
  - 普遍采用
  - Intel/AMD/IBM/Sun/…
- 涉及领域
  - 无处不在
  - 服务器/PC/笔记本/嵌入式系统/…

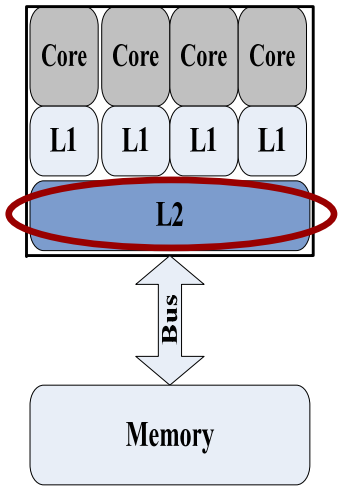
多核挑战

- 多核(CMP) V.S. 对称多处理器(SMP)
  - 核的数目更多
    - SMP: 低端(2CPU), 中级(4~8CPU),高端(>16CPU)
    - CMP: 4~8 cores 多核系统， **1000+ cores (<10年)**
      E.g. Intel's 80 cores chip & Tilera's 100 core chip
  - 硬件资源共享(e.g., 最后级缓存)





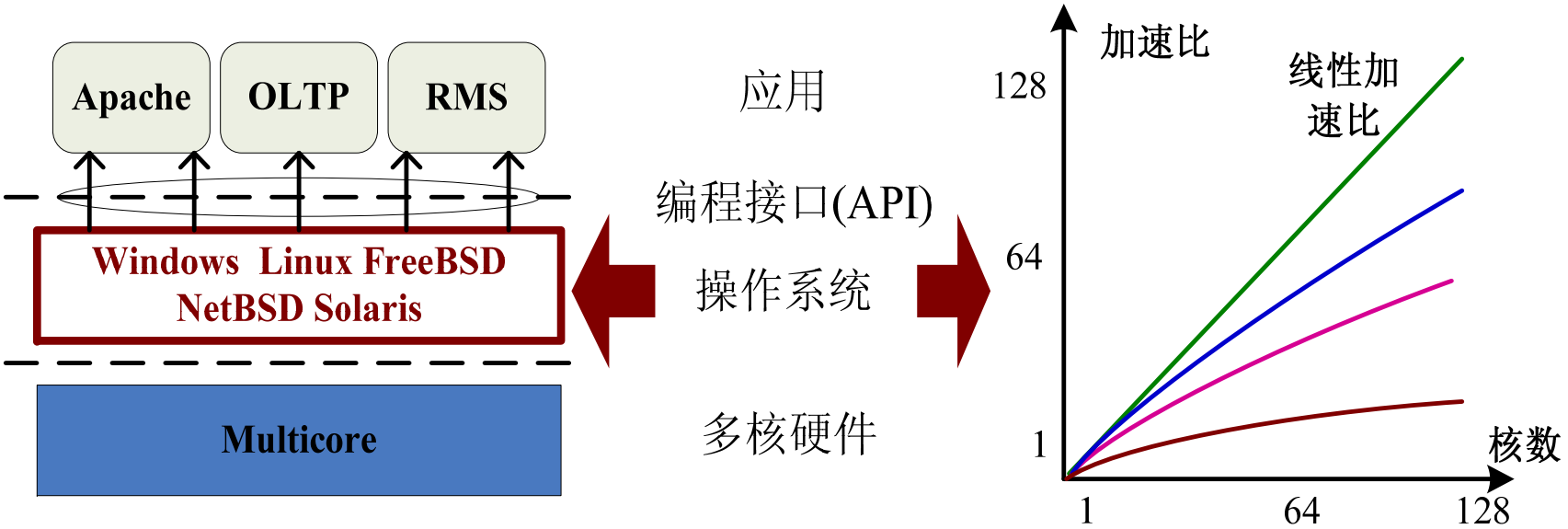
Separate



Shared

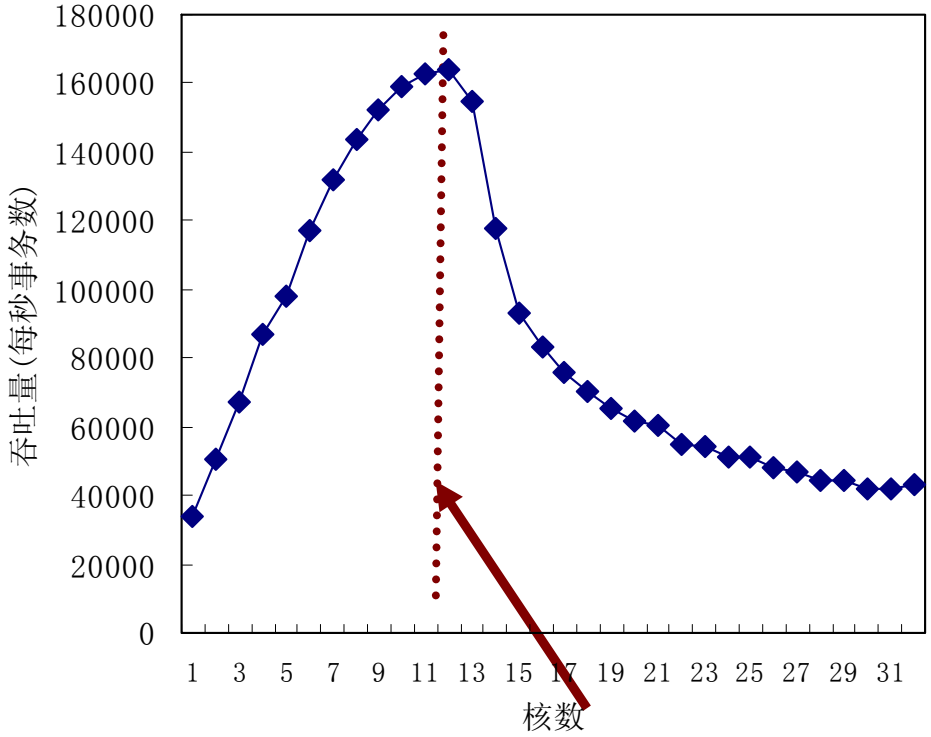
多核挑战

- 操作系统层的可扩展性
  - 性能是否随着核的数目线性增加
    - 固定时间的加速比模型 (Gustafson S=N.(1-P)+P)
  - 侧重操作系统层
    - 针对小规模多处理器设计 未考虑多核特点(核多，资源共享)



可扩展性瓶颈

- 内核锁竞争



操作系统: Linux

实验平台: AMD 32 cores

测试程序: 文件服务器模拟

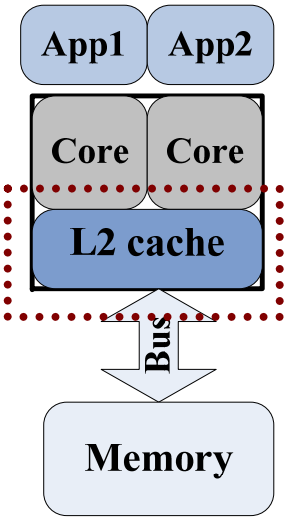
文件系统: tmpfs

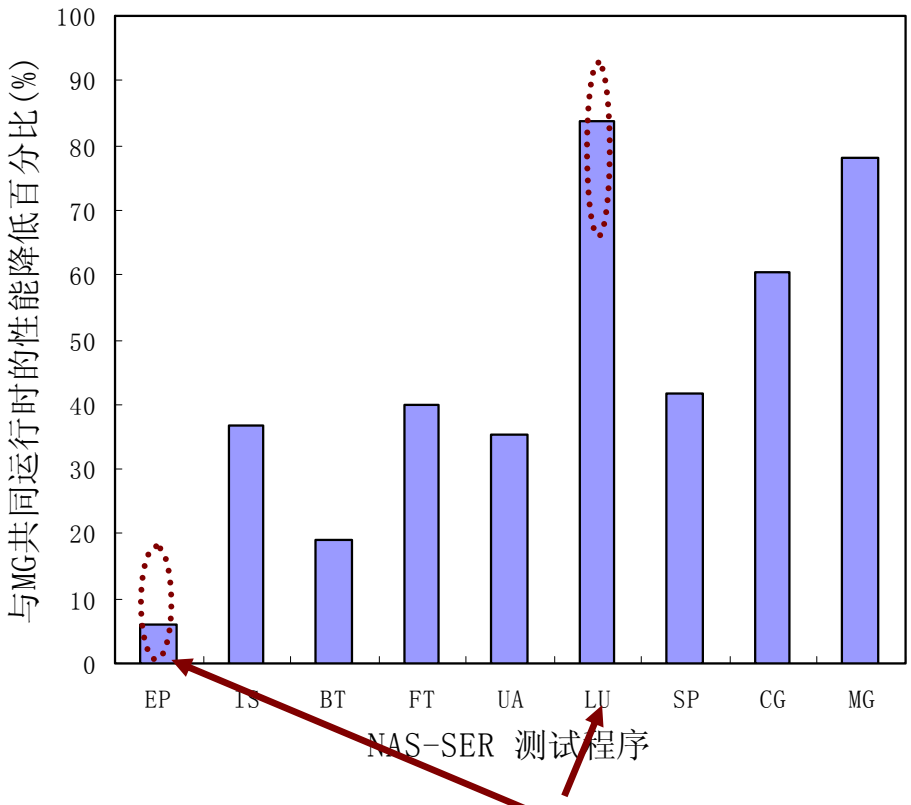
可扩展性瓶颈: 保护文件描述符表和内存文件系统统计信息的自旋锁竞争

内核锁竞争导致吞吐量下降

可扩展性瓶颈

- 共享硬件资源竞争





操作系统: Linux

实验平台: Intel 8core

测试程序: 科学计算(NAS)

可扩展性瓶颈: 共享硬件资源竞争

缓存等资源竞争导致执行时间延长6%~84%

可扩展性瓶颈

- 共享是影响操作系统可扩展性的最根本原因[OSDI'08]
  - 共享有多种体现

    内核数据的共享➔锁竞争

    硬件缓存的共享➔缓存竞争

    地址总线的共享➔总线竞争
  - 共享的两面性

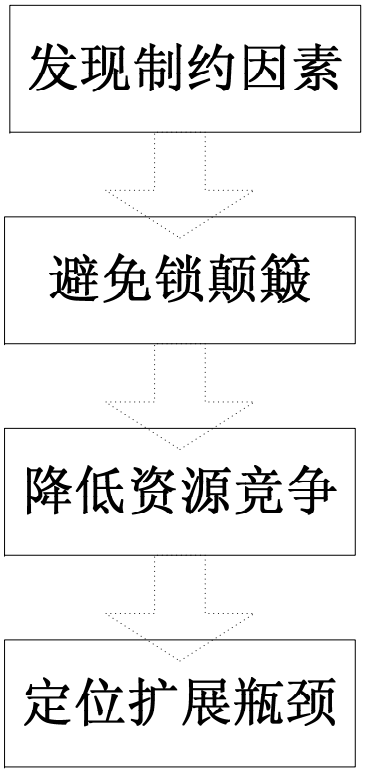
    好处: 方便、快速的通信

    坏处: 引入额外数据交互, 降低可扩展性

[1].Silas Boyd-Wickizer, et al. Corey: An Operating System for Many Cores. In Proceedings of OSDI 2008.

研究内容

- 如何发现共享导致的可扩展性制约因素
  - 分析获得了系统服务接口的加速比制约因素
- 如何避免内核锁竞争导致的锁颠簸现象[OSDI'10]
  - 提出了一种基于离散事件仿真技术的模拟器
  - 提出了一种基于等待者数目的可扩展锁机制
  - 提出了一种锁竞争感知的调度策略
- 如何降低硬件资源的竞争问题[ASPLOS'10, USENIX'05]
  - 提出了一种资源竞争感知的调度策略
- 如何定位共享导致的可扩展性瓶颈[EuroSys'10]
  - 提出了一种基于函数可扩展性值的瓶颈定位方法



[1].Slias Boyd-Wicizer, et al, "An Analysis of Linux Scalability to Many Cores ". In OSDI 2010.

[2].Sergey Zhuravlev, et al, "Addressing Shared Resource Contention in Multicore Processors via Scheduling", in ASPLOS 2010.

[3].Alexandra Fedorova, et al, "Performance of Multithreaded Chip Multiprocessors for Operating System Design", in USENIX ATC 2005.

[4].Aleksey Pesterev, "Locating Cache Performance Bottlenecks Using Data Profiling", in EuroSys 2010.

提纲

❶ 选题背景

❷ 论文工作

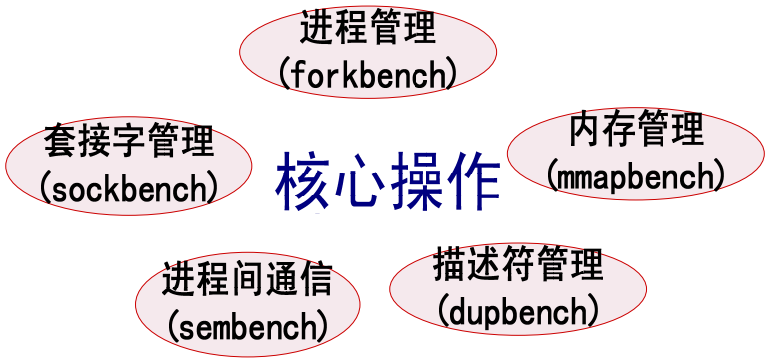
- 分析系统服务接口、发现制约因素
- 锁颠簸现象的模拟和避免
- 共享硬件资源竞争降低
- 可扩展性瓶颈定位方法

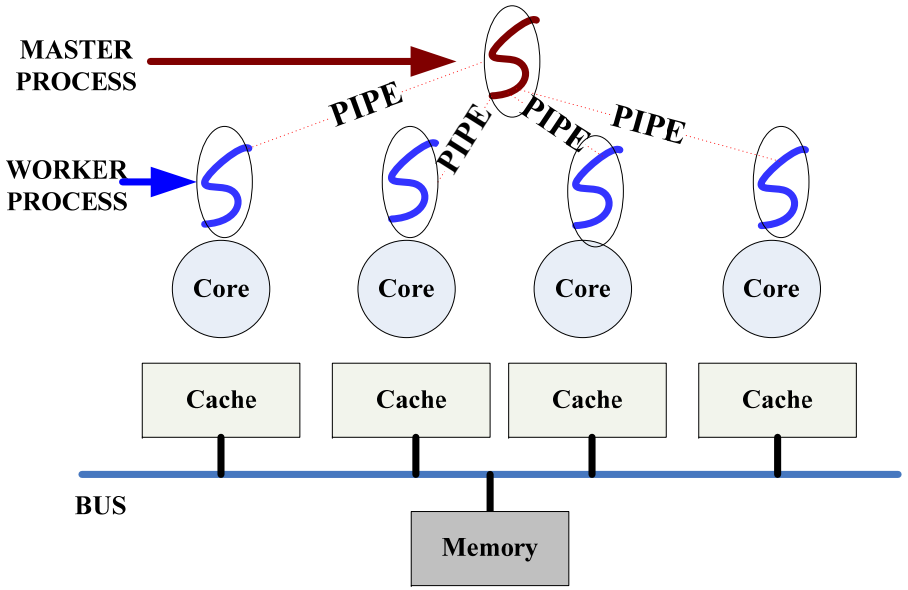
❸ 总结

❹ 研究成果

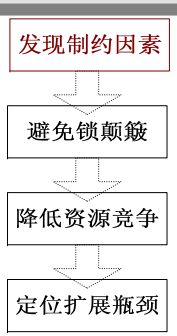
分析系统服务接口

- 设计微基准测试程序集
  - 5个 核心、重要接口 由统一框架管理





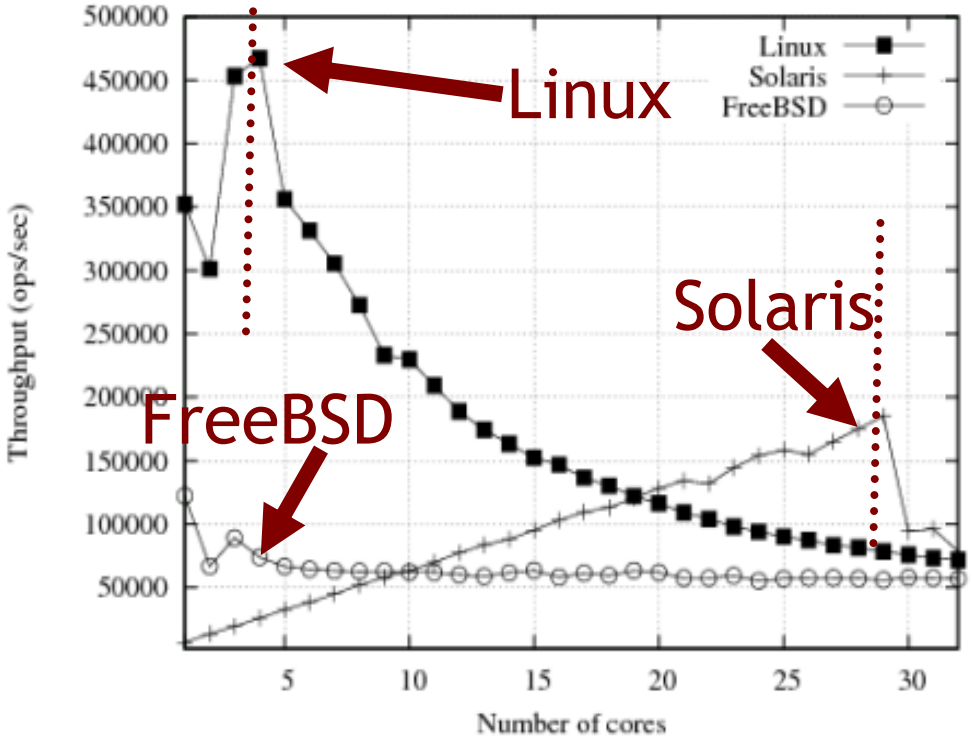
分析系统服务接口：实验方法



- 操作系统(POSIX兼容)
  - Linux 2.6.26.8
  - OpenSolaris 2008.11
  - FreeBSD 8.0-CURRENT
- 硬件平台
  - AMD NUMA 8*4 = 32
- 分析工具(函数执行时间，锁使用等)
  - Linux: Oprofile /proc/lock_stat
  - Solaris: Dtrace lockstat
  - FreeBSD: lock profiling
- 绑定接口(避免调度影响)
  - Linux sched_setaffinity()
  - Solaris pset_bind()
  - FreeBSD cpuset_setaffinity()

设计微基准测试：mmapbench

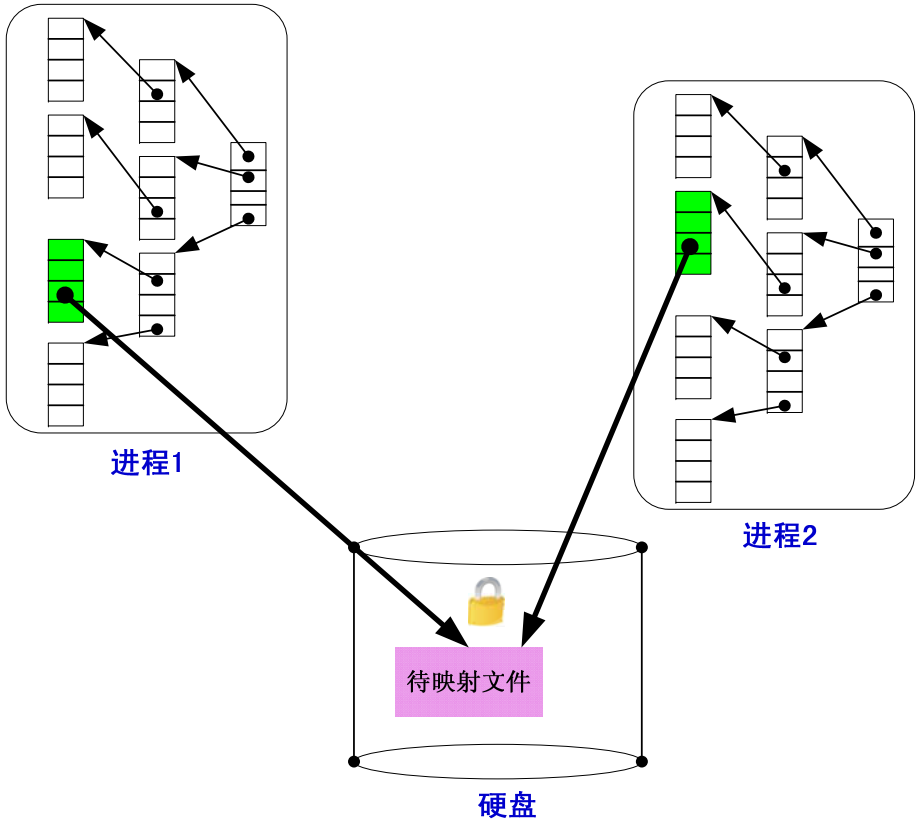
- mmapbench
  - 每个进程映不断射同一文件的500Mbytes 读取每一页的第一字节 解除映射



所有系统均有可扩展性问题

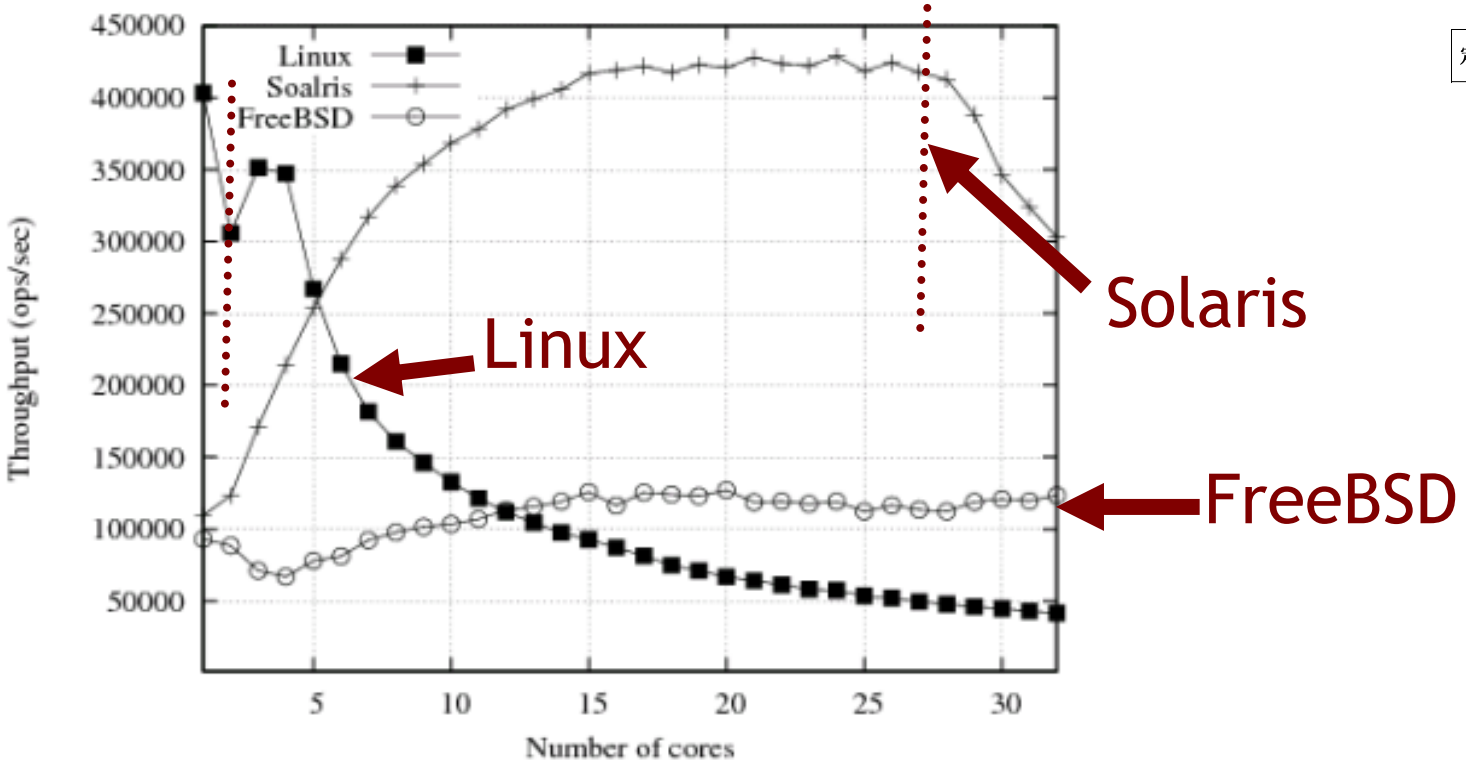
设计微基准测试：mmapbench

- 瓶颈
  - 保护相同文件的锁竞争
  - 锁获取时机不同
    - Linux 整个映射
    - Solaris 统计数据更新
    - FreeBSD 映射策略



设计微基准测试：sockbench

- sockbench
  - 每个进程不断调用socket()和close()

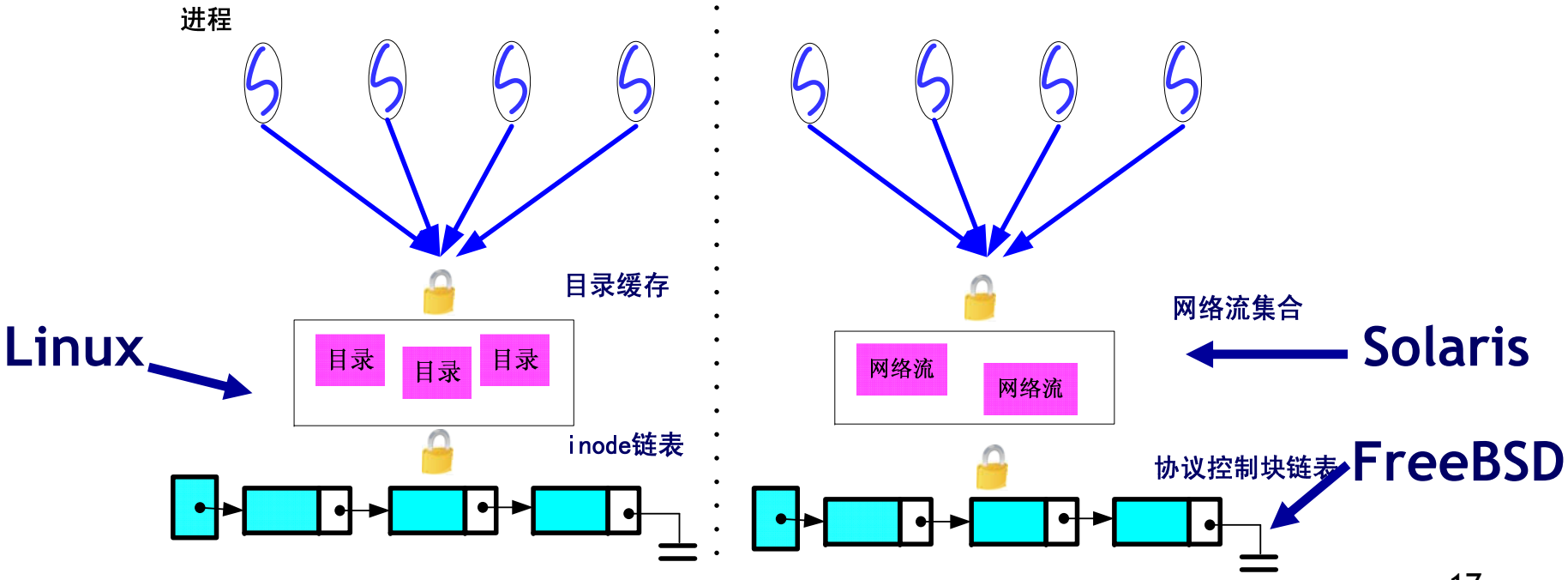


所有系统都有可扩展性问题

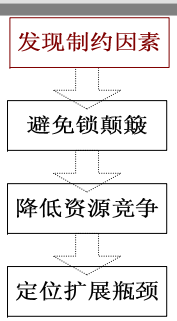
设计微基准测试：sockbench

发现制约因素
避免锁颠簸
降低资源竞争
定位扩展瓶颈

- 瓶颈
  - 目录缓存锁 inode链表锁竞争(linux)
  - 网络协议栈竞争
    - Solaris 引用计数更新
    - FreeBSD 协议控制块链表维护



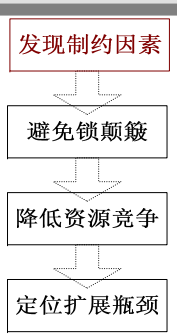
制约因素概要



	Linux	Solaris	FreeBSD
forkbench	建立、删除VMA导致保护内存映射文件的锁竞争	缺页异常在内存映射文件的读写锁竞争	缺页异常在内存映射文件的互斥锁竞争
mmapbench	建立、删除VMA导致保护内存映射文件的锁竞争	设置内存放置策略导致内存映射文件读写锁竞争	更新和查找vnode信息在内存映射文件互斥锁竞争
dupbench	完全可扩展	关闭文件描述符在哈希表的自适应互斥锁上竞争	witness开销随着核数线性增加(去掉则完全可扩展)
sembench	保护全局信号量的读锁有竞争	完全可扩展	完全可扩展
sockbench	全局目录缓存和全局inode链表的自旋锁竞争	建立和删除流导致网络协议栈的引用计数竞争	保护全局的协议控制块链表的读写锁竞争

1. 操作系统中保护共享数据的锁竞争是影响可扩展性的重要因素
2. 锁竞争的激烈程度足以导致加速比随着核数的增加下降(锁颠簸现象)

小结



- 贡献
  - 分析获得操作系统服务接口加速比的制约因素
- 成果
  - IEEE CLUSTER, IEICE Transactions 发表论文一篇
    - **Yan Cui** , Yu Chen, Yuanchun Shi, “Experience on Comparison of Operating System Scalability on the Multicore Architecture”, in **CLUSTER** 2011 (EI, **CCF rank B**).
    - **Yan Cui**, Yu Chen, Yuanchun Shi, “Comparing Operating System Scalability by Microbenchmarking”, in IEICE Transactions on Information and Systems (**IEICE Transactions**) (SCI & EI)

提纲

1. 选题背景
2. 论文工作
   - 分析系统服务接口、发现制约因素
   - 锁颠簸现象的模拟和避免
   - 共享硬件资源竞争降低
   - 可扩展性瓶颈定位方法
3. 总结
4. 研究成果

锁颠簸现象的模拟

- 如何模拟、建模锁竞争导致的吞吐量下降
- 相关工作
  - 平均值分析建模锁以及缓存缺失[OSR'78，TON'98]
  - 利用自旋锁达到的循环特征模拟[SIGCOMM'93]
  - 近似平均值分析建模缓存缺失[TR'98]
  - 利用概率模型[ISCA'10]

  没有建模排号自旋锁的特征，不能重现锁颠簸现象

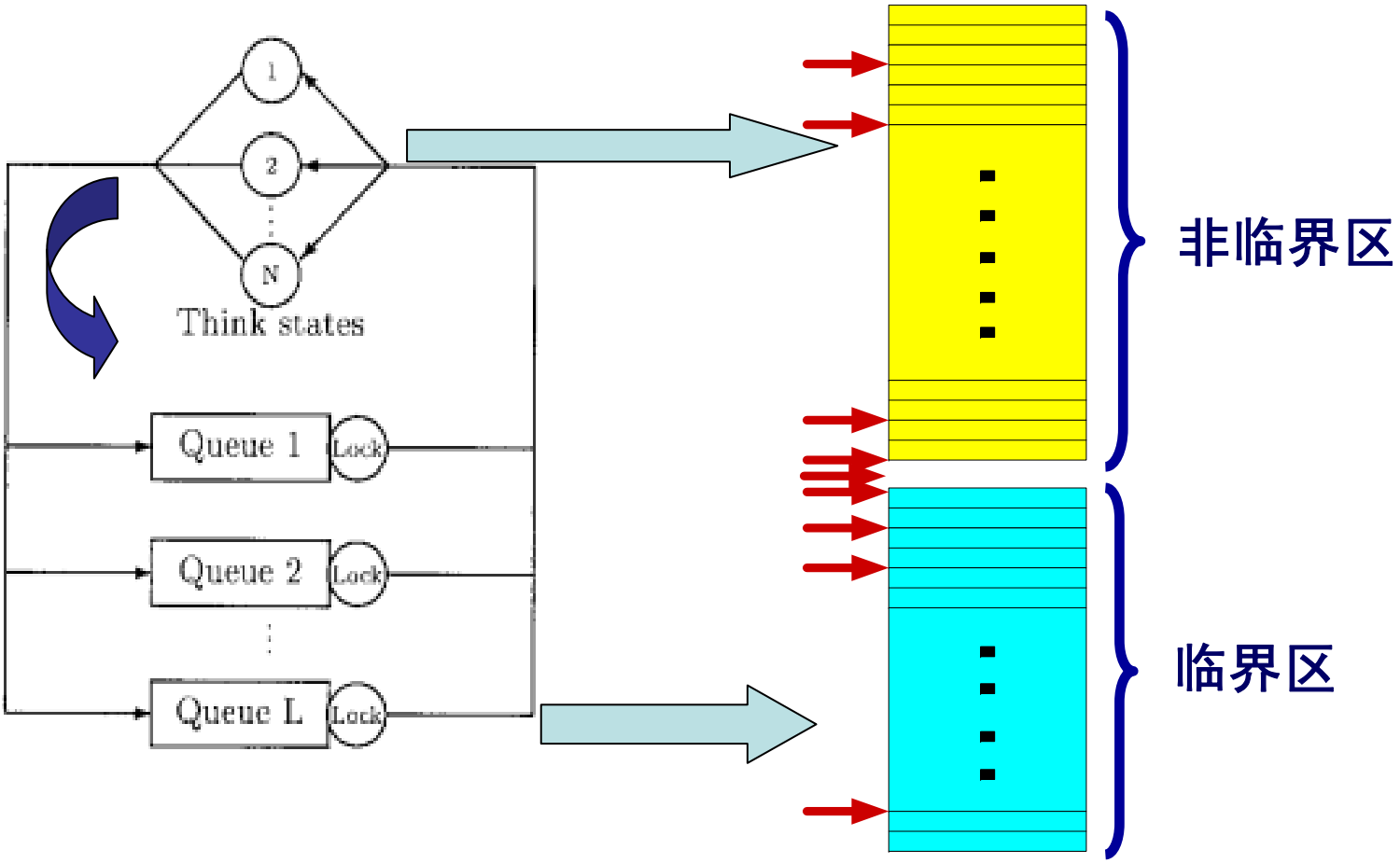
  - 马尔科夫链建模自旋锁[OLS'12]

  没有建模临界区和非临界区的缓存缺失，求解复杂

[1].D.Gillbert. "Modeling Spin Locks with Queuing Networks". In OSR 1978.
[2].M.Bjorkman and P.Gunningberg, "Performance Modeling of Multiprocessor Implementation of Protocols", in TON 1998.
[3].M.Bjorkman and P.Gunningberg, "Locking Effects in Multiprocessor Implementation of Protocols", in SIGCOMM 1993
[4].D.L.Eager, D.J.Sorin and M.K.Vernon, "Analysis Modeling of Burstiness and Synchronization Using Approximate MVA", in TR 1998
[5].Stijn Eyerman and Lieven Eeckhout, "Modeling Critical Sections in Amdal's Law and its implications for Multicore Design", In ISCA 2010
[6].Slias Boyd-Wickizer, et.al, "Non-Scalable Locks are Dangeous", in OLS 2012

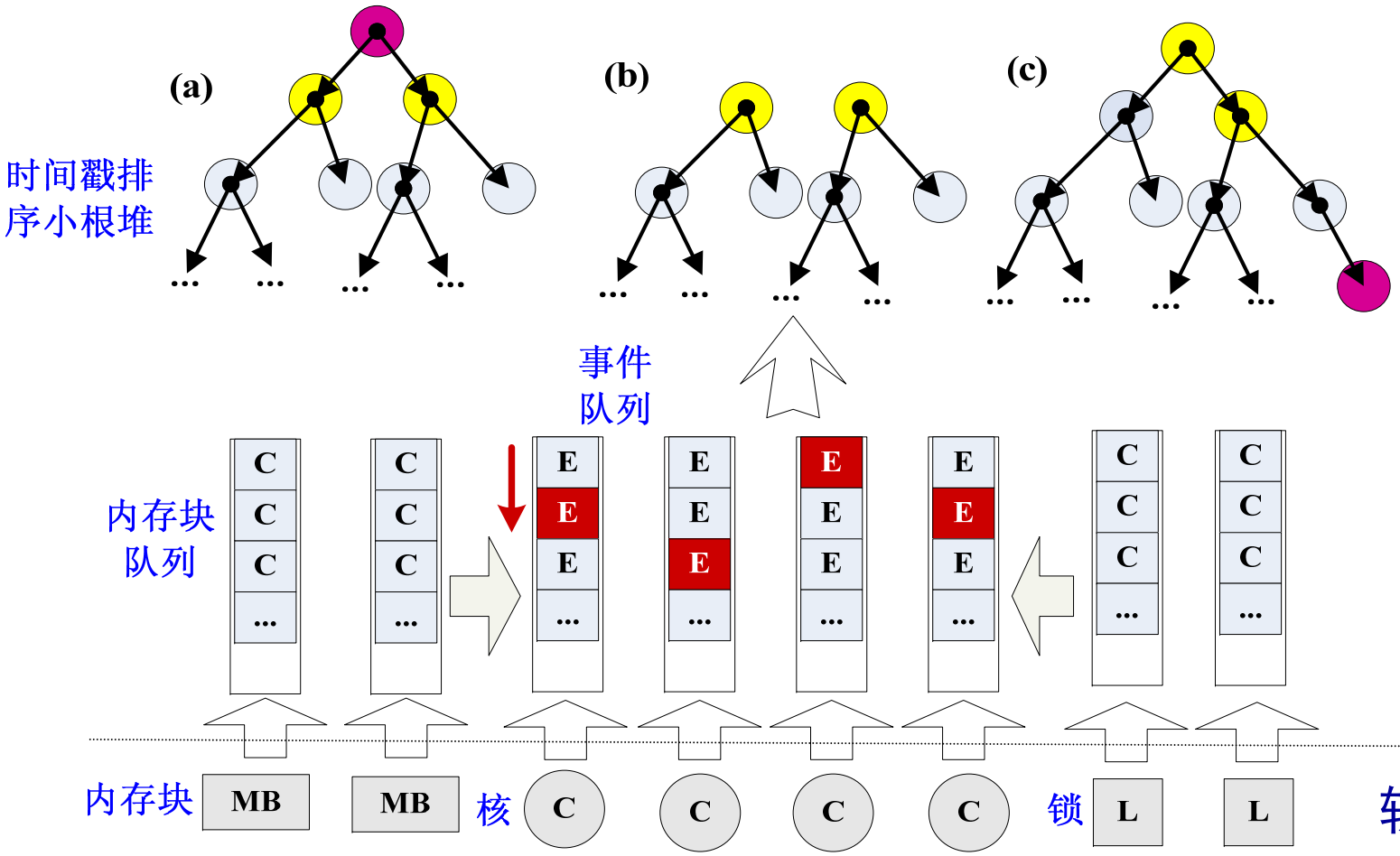
模型

- 基于排队论模型 离散事件仿真求解
  - 锁实现的开销同时兼顾临界区、非临界区内缓存缺失



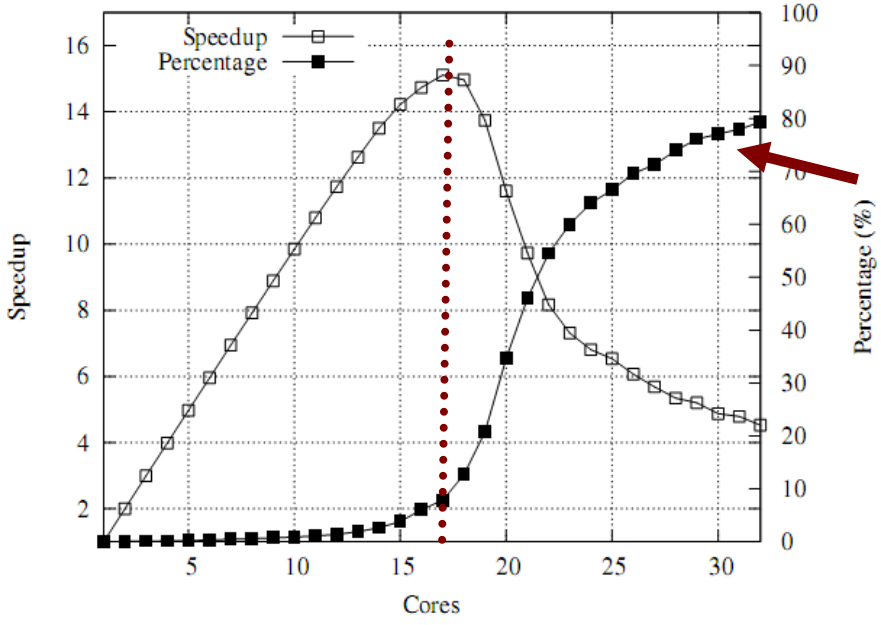
数据结构和操作

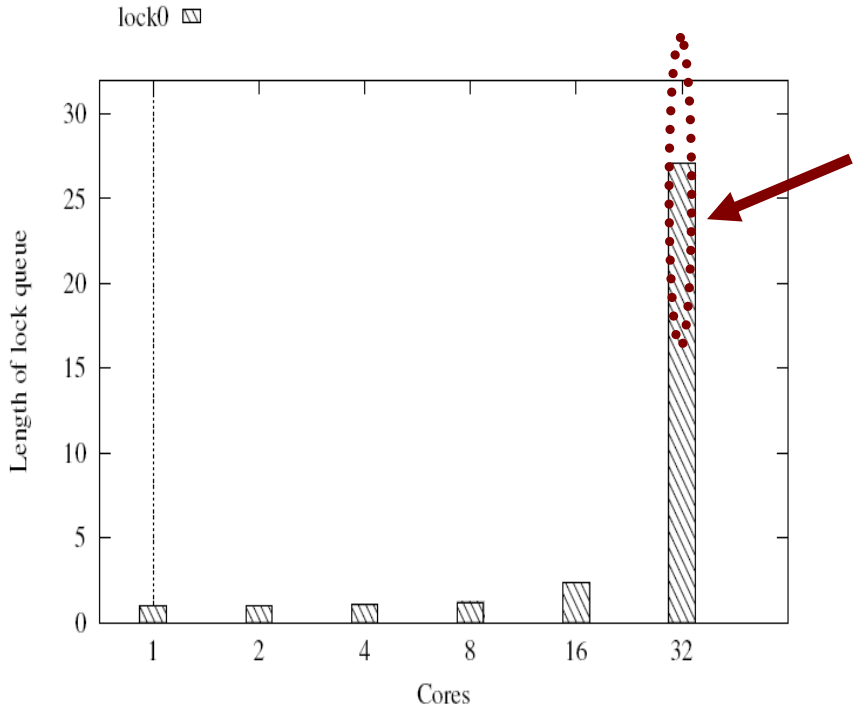
- 数据结构及操作



实验和评价

- 锁颠簸现象重现 曲线与实际系统相近
- 颠簸现象与百分比、等锁核数的强相关性





锁颠簸现象避免

- 相关工作
  - 细粒度锁: 临界区分解

耗时、困难、易错[OSR'09]

  - 互斥锁: 通过睡眠等待

上下文开销大(10000 cycles+)

  - 自适应锁: 自适应地睡眠和忙等

经验决定睡眠 自旋 难以达到全部潜力[LKML.org'09]

  - MCS锁[TOCS'91]: 等待者等在不同的变量上

计算资源有效性低、能耗高

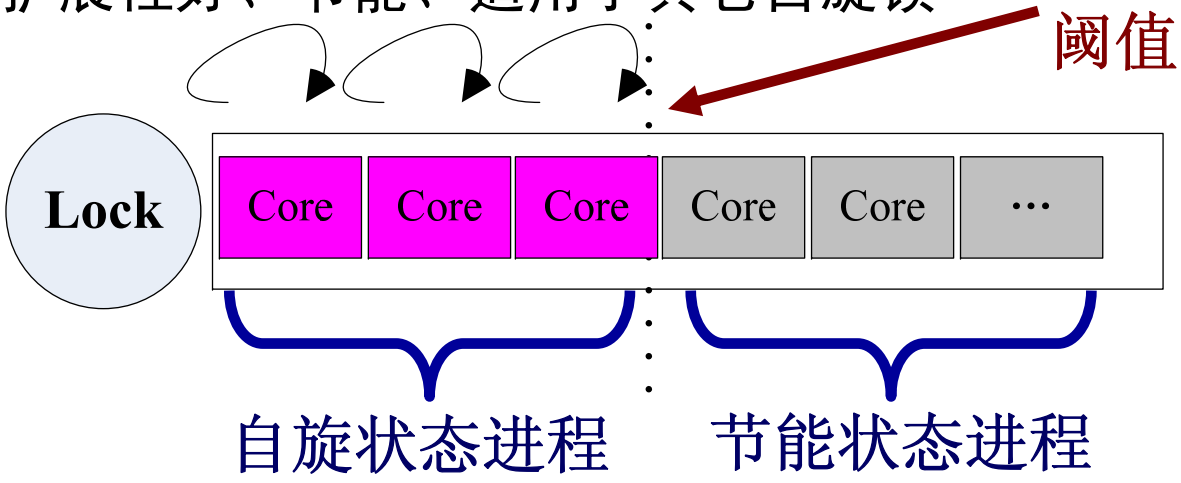
[1].D.Wentzlaff and A.Agarwal, "Factored Operating Systems (fos): the case for a scalable operating system for multicores", in OSR 2009.

[2]."Adaptive Spinning Mutexes" , http://lkml.org/lkml/2009/1/14/393.

[3].J.Mellor-Crummey and M.L.Scott, "Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors", in TOCS 1991

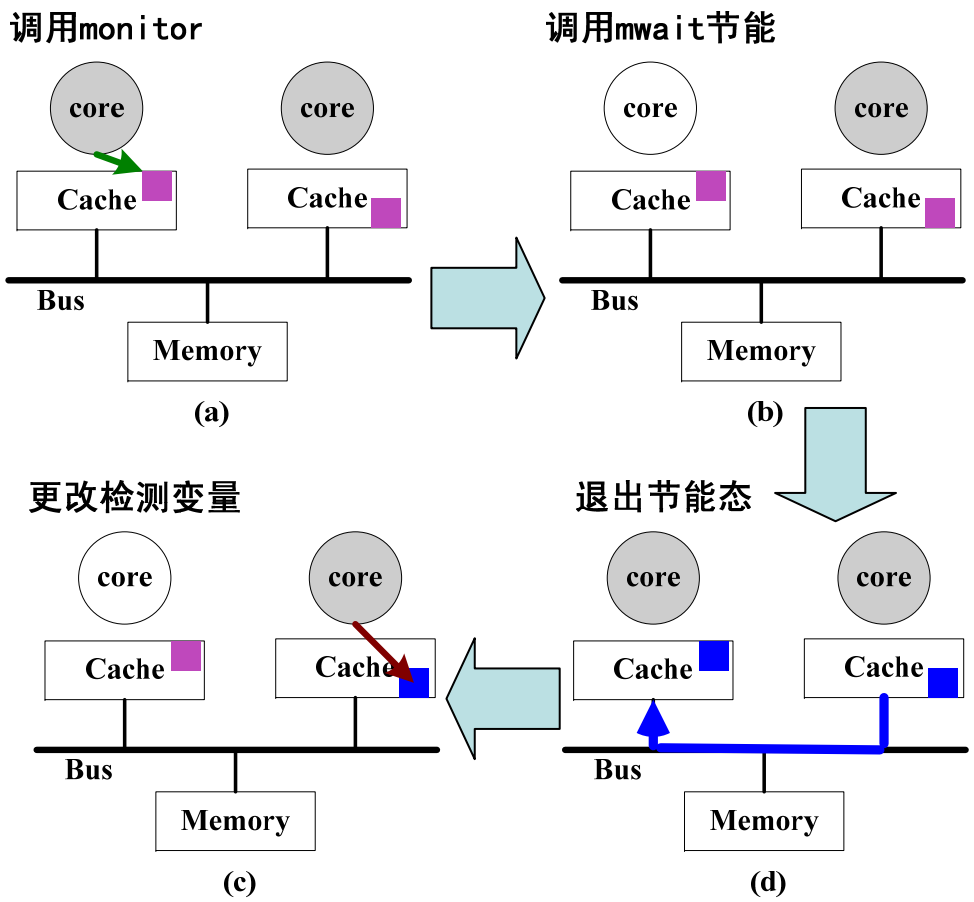
基于等待者数目可扩展锁

- 锁颠簸产生时等待某个锁的核数会大幅度增加
  - 模拟器中观察 实际系统验证
- 基本方法
  - 检测等待排号自旋锁的核数，若大于阈值则进入节能态(monitor/mwait)，否则自旋等待
  - 进入节能状态时保证每个等待者自旋在局部变量上
  - 可扩展性好、节能、适用于其它自旋锁



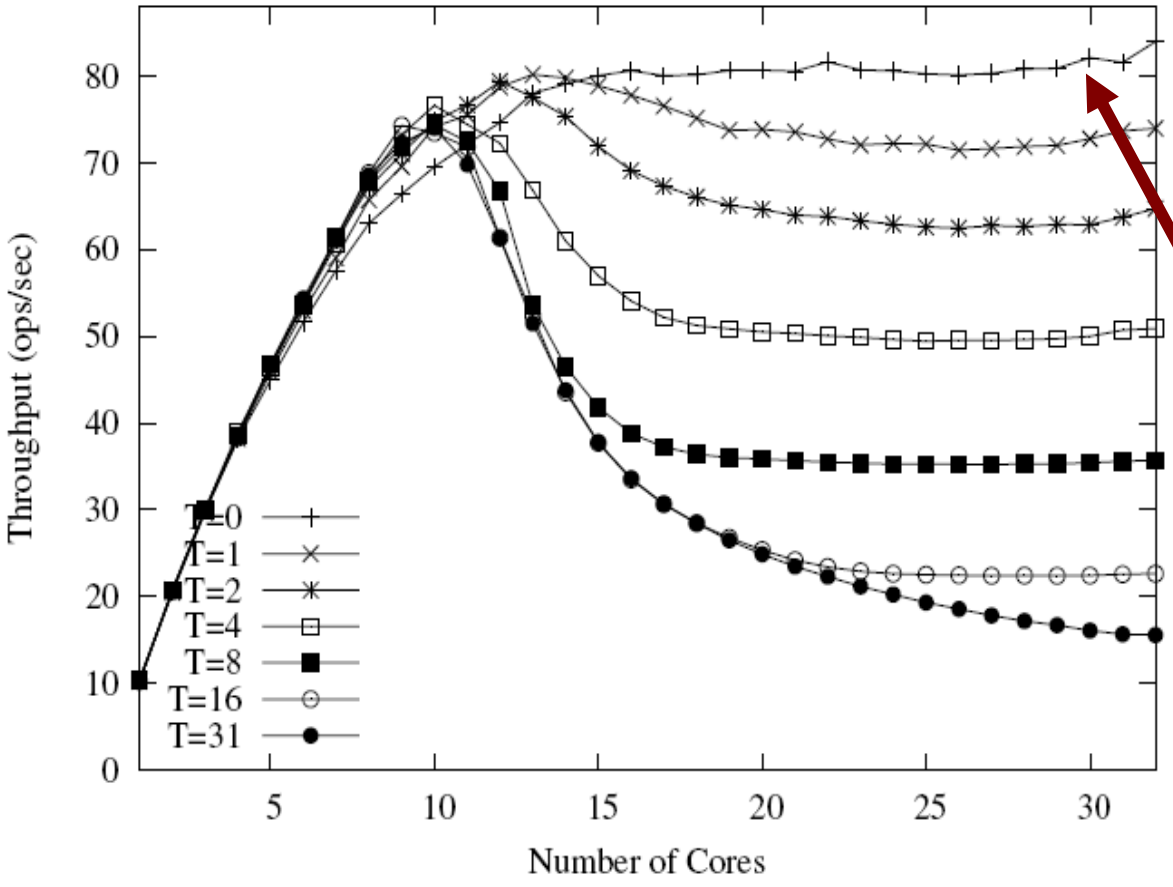
可扩展性优化

1. 轻量级的等锁机制(monitor/mwait)
2. 每CPU等待队列
3. 锁等待者的快速预测
4. 轻量级的锁等待者唤醒方法
5. 避免伪共享



阈值选择

- 阈值 0->1->2->4->8->16->31



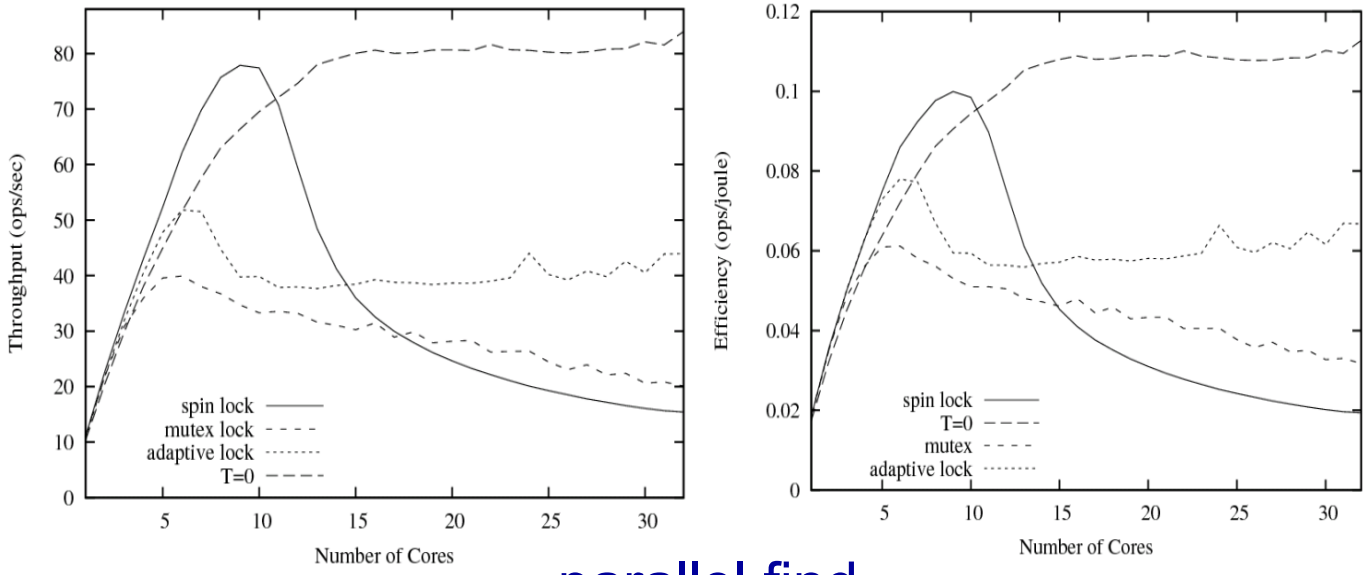
parallel find

阈值设置为0扩展性最好 节能指令量级轻

实验和评价

- 内核: 2.6.29.4、2.6.32
- 平台: AMD NUMA 32、Intel NUMA 40
- 测试程序: mmapbench、sockbench、parallel find、kernbench、parallel postmark
- 功耗测量: 380801 功率仪

与自旋锁、互斥锁、自适应互斥锁比较

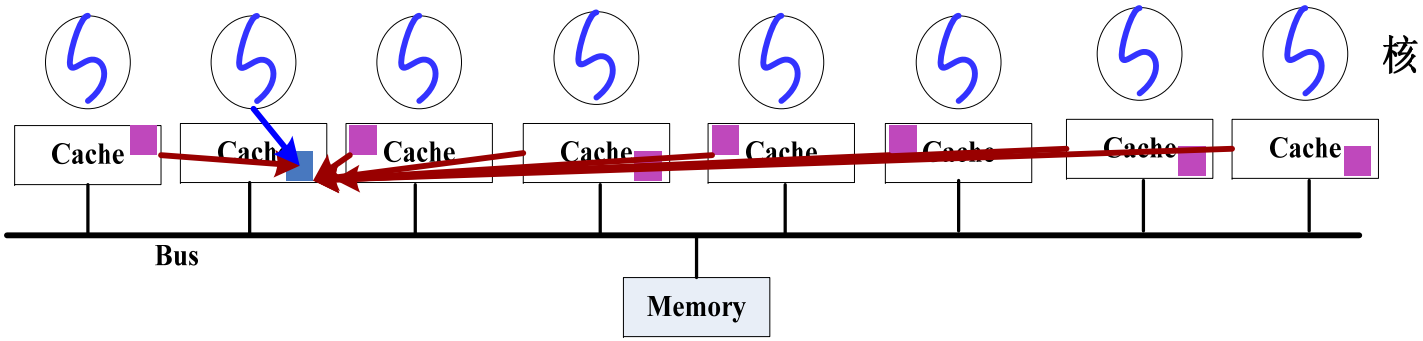


parallel find

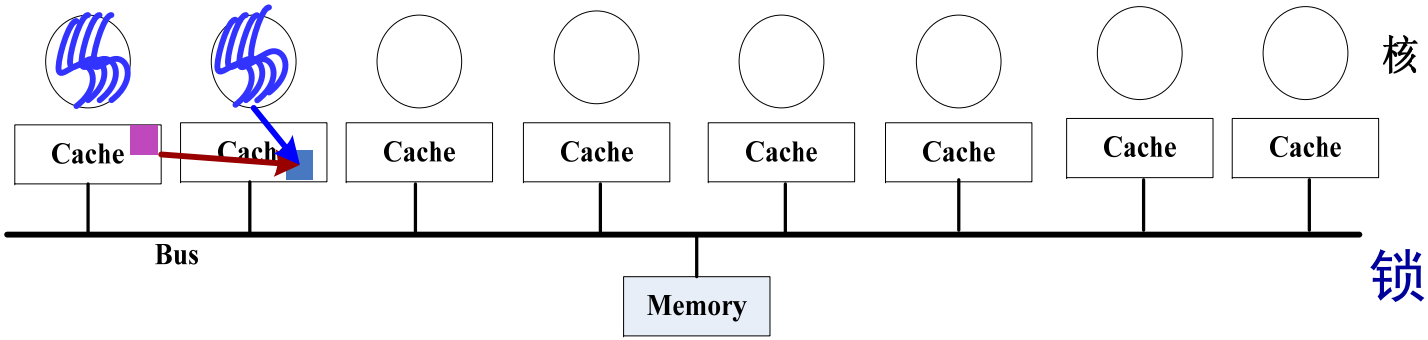
基于等待者数目可扩展锁在可扩展性和能耗有效性上均好于其它方法

锁竞争感知调度算法

- 锁颠簸产生时平均等锁时间百分比大幅度增加
- 基本方法



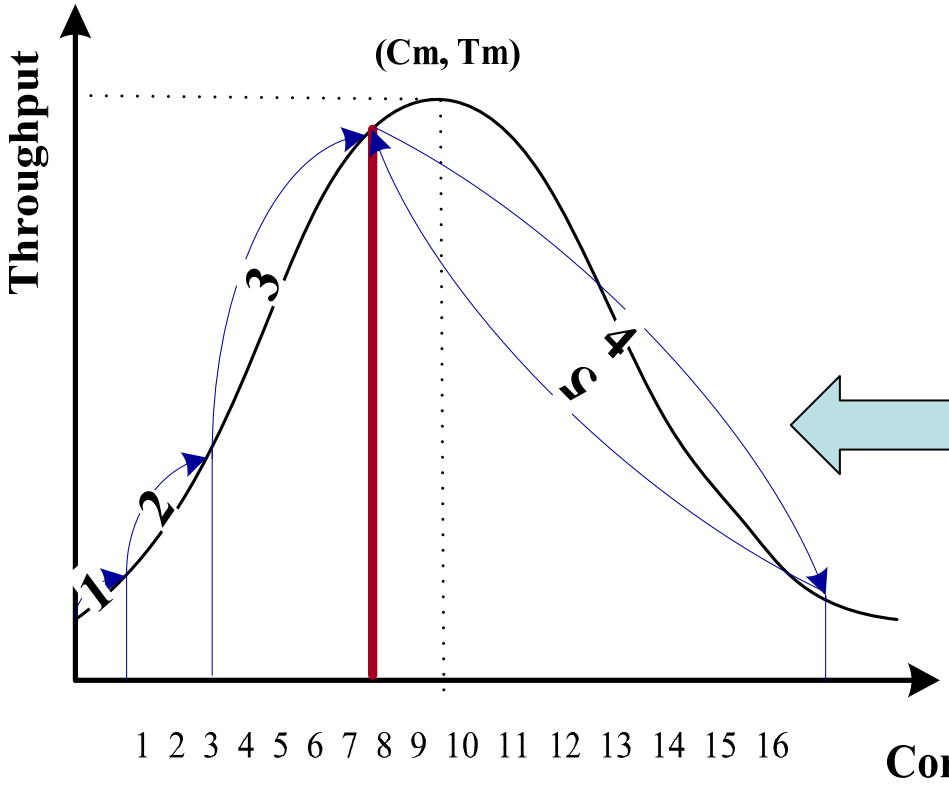
默认调度



锁竞争感知调度

关键问题

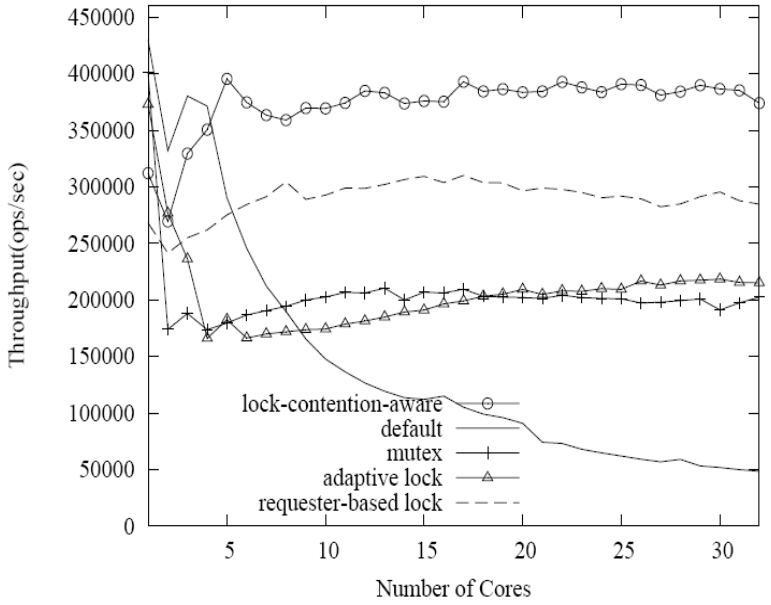
- 如何确定用锁密集类任务使用的核数
  - p(n)通过改写用锁函数获得 同时使用投票机制和迁移状态机避免在线测量误差



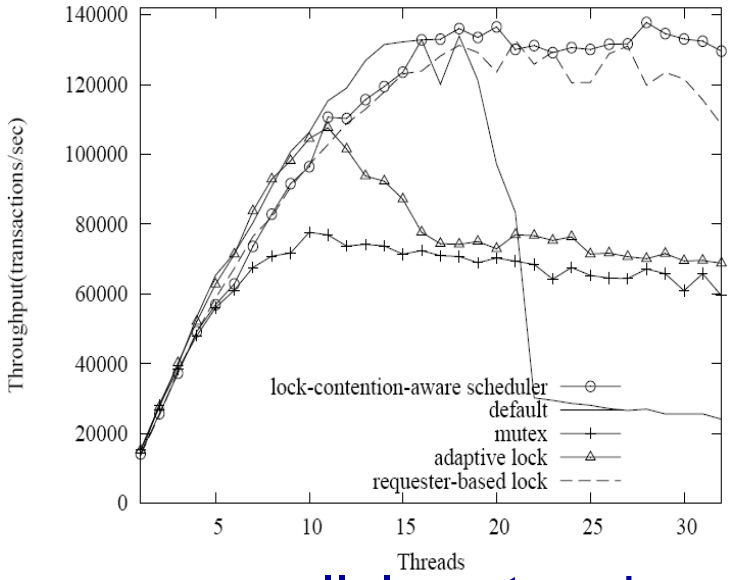
$n^* = \text{argmax}\ \{n.(1-p(n))\}$

实验和评价

与基于等待者数目锁、自旋锁、互斥锁、自适应互斥锁比较可扩展性



sockbench



parallel postmark

锁竞争感知调度策略的可扩展性优于其它各种实现方法，能耗有效性的结果类似

小结

发现制约因素 → 避免锁颠簸 → 降低资源竞争 → 定位扩展瓶颈

- 贡献
  - 锁颠簸现象的模拟和避免
    - 基于离散事件仿真技术的模拟器 准确、快速
    - 基于等待者数目的可扩展锁机制 特殊指令、局部变量自旋
    - 锁竞争感知的调度策略 模型驱动搜索
- 成果
  - ACM Transactions on Architecture and Code Optimization、Concurrency and Computation: Practice and Experience、MASCOTS、ICPADS 发表论文 专利1项 IEEE Transactions on Computers在审一篇
    - **Yan Cui**, Yingxin Wang, Yu Chen, Yuanchun Shi, "Lock Contention Aware Scheduler: A Scalable and Energy Efficient Method for Addressing Scalability Collapse on Multicore Systems", in ACM Transactions on Architecture and Code Optimization (**TACO**) (SCI & EI, **CCF rank A**)
    - **Yan Cui**, Yu Chen, Yuanchun Shi, "Towards Scalability Collapse Behavior", in Concurrency and Computation: Practice and Experience(**CCPE**)(SCI&EI, **CCF rank B**)
    - **Yan Cui**, Yingxin Wang, Yu Chen, Yuanchun Shi, etc, "Reducing Scalability Collapse via Requester-Based Locking on Multicore Systems", in **MASCOTS** 2012(EI, **CCF rank B**)
    - **Yan Cui**, Weida Zhang, etc, "A Scheduling Method for Avoiding Kernel Lock Thrashing on Multicores", in **ICPADS** 2010 (EI, **CCF rank C**)
    - **Yan Cui**, Weiyi Wu, etc, "A Discrete Event Simulation Model for Understanding Kernel Lock Thrashing on Multicores" in **ICPADS** 2010(EI, **CCF rank C**)
    - 秦岭, 陈渝，**崔岩**, 吴谨, 实现自适应锁的方法以及多核处理器系统, 申请号:201110394780, 公开号:CN102566979
    - **Yan Cui,** Yingxin Wang, Yu Chen, Yuanchun Shi, "Requester-Based Lock: A Scalable and Energy Efficient Locking Scheme on Multicore Systems", in IEEE Transactions on Computers, (under review, SCI & EI, **CCF rank A**)

提纲

1. 选题背景
2. 论文工作
   - 分析系统服务接口、发现制约因素
   - 锁颠簸现象的模拟和避免
   - 共享硬件资源竞争降低
   - 可扩展性瓶颈定位方法
3. 总结
4. 研究成果

共享硬件资源竞争降低

- 相关工作
  - 硬件划分: 基于扩展的LRU算法[MICRO'06][HPCA'02]

需要特殊硬件的支持, 缓存划分的粒度过大

  - 软件划分: 基于页着色算法[HPCA'08][WIOSCA'07]

程序的行为频繁发生变化时, 着色的开销过大

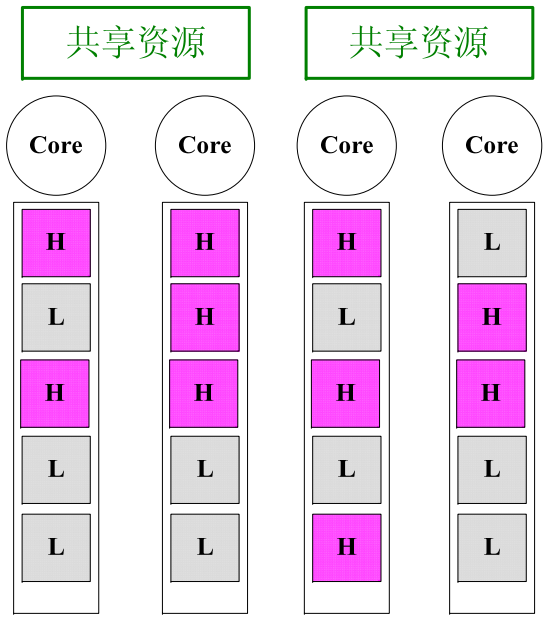
  - 改写调度策略: 不需改动硬件, 开销较低[ASPLOS'10, USENIX'05, IPDPS'05]

1. 任务使用资源的多少缺少准确、稳定的量化指标
2. 只改写上下文切换或者负载均衡逻辑
3. 实际系统整合

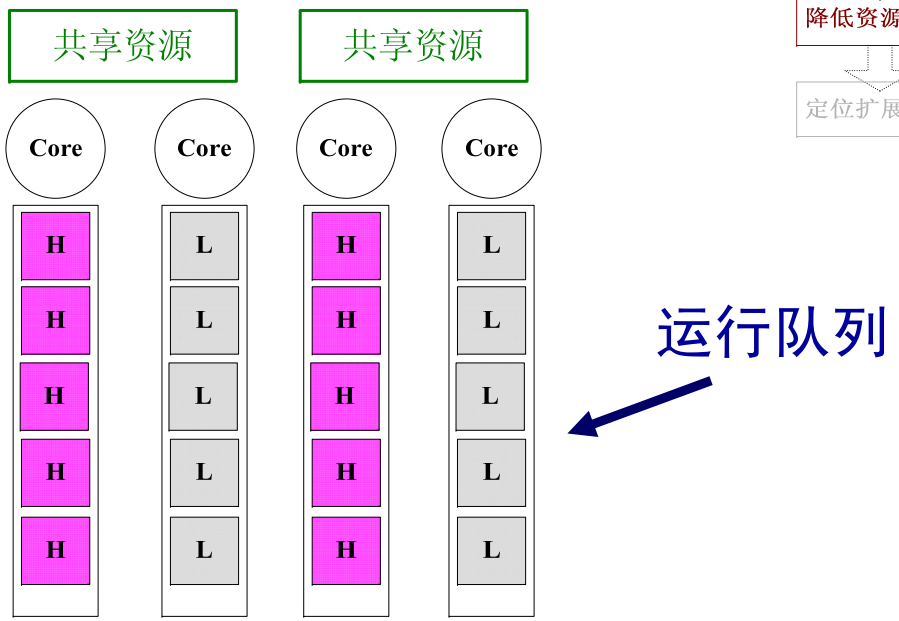
[1].M.K.Qureshi and Y.N.Patt, "Utility-Based Cache Partitioning: A low-overhead, high performance, runtime mechanism to partition shared caches", in MICRO 2006
[2].G.E.Suh, S.Devadas and L. Rudolph, "A New Memory Monitoring Scheme for Memory Aware Scheduling and Partitioning", in HPCA 2002
[3].Jiang Lin, et. al, "Gaining Insights into Multicore Cache Partitioning: Bridging the gap Between Simulation and Real Systems" in HPCA 2008
[4].D.Tam et al, "Managing Shared L2 Cache on Multicore Systems in Software", in WIOSCA 2007
[5].S.Zhuravlev, et al, "Addressing Shared Resource Contention in Multicore Processors via Scheduling", in ASPLOS 2010
[6].Alexandra Fedorova, et al, "Performance of Multithreaded Chip Multiprocessors and Implications for Operating System Design", in USENIX 2005
[7].Robert L.Mcgregor, et al, "Scheduling Algorithms for Effective Thread Pairing on Hybird Multiprocessors", in IPDPS 2005

硬件资源竞争感知调度策略

- 基本方法



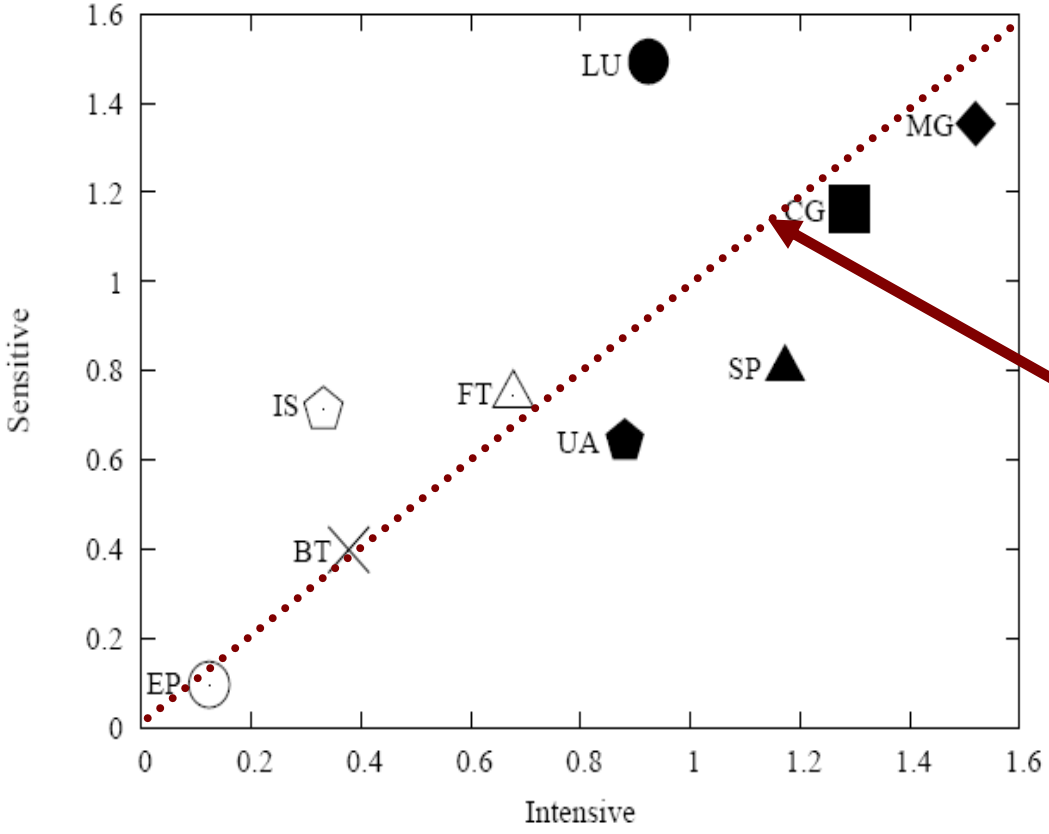
默认调度(资源竞争)



竞争感知调度(资源均衡使用)

关键问题

- 如何确定程序使用共享资源的多少
  - 构造性能降级矩阵 Di,j代表与i运行时, j的性能降级
  - 计算每个程序的强度(矩阵行)、敏感度(矩阵列)



强度和敏感度具有强相关性

关键问题

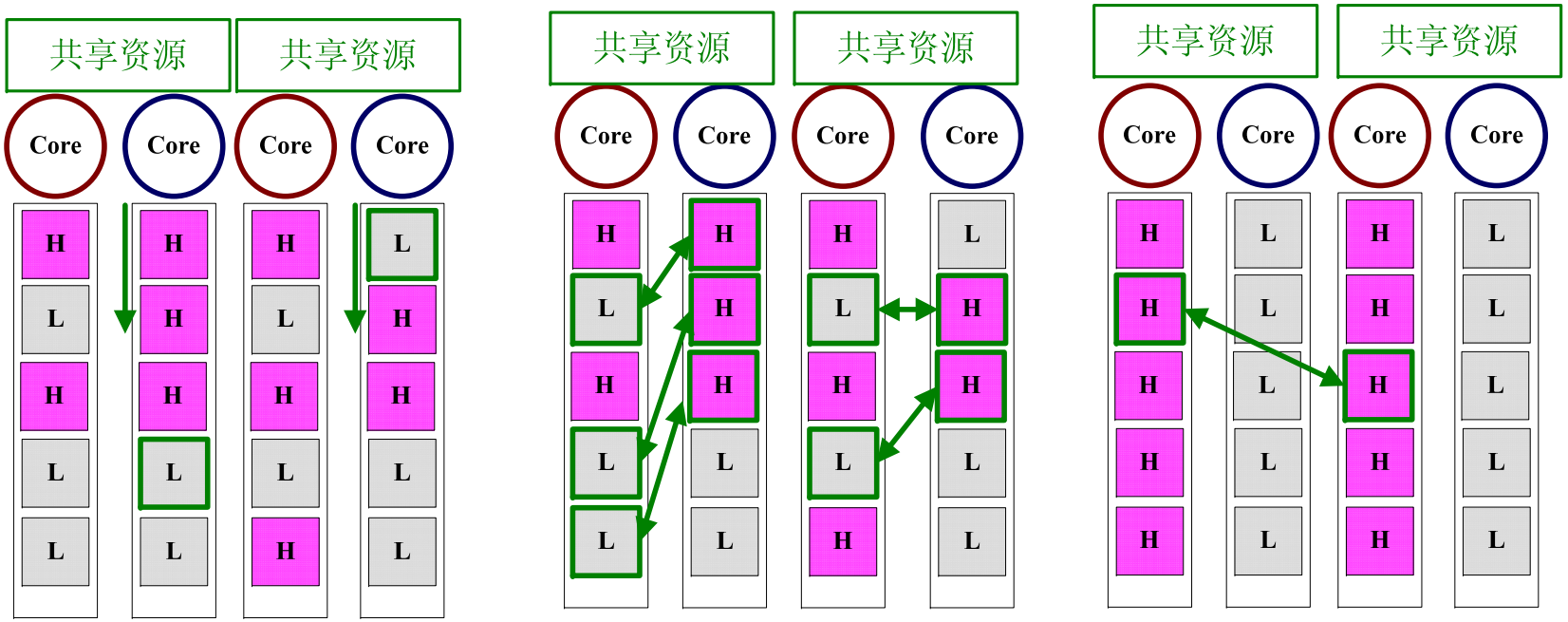
- 如何确定程序使用共享资源的多少
  - 使用强度度量应用使用资源情况 但不能在线获取
  - 计数器获取的启发式指标
  - 准确性(相关系数) 稳定性 列向量的平均长度

选择每条指令的最后级缓存访问率

关键问题

- 如何根据每个任务使用资源多少调度
  - 上下文切换 从核选择合适任务
  - 负载均衡 主从 主主 任务互换

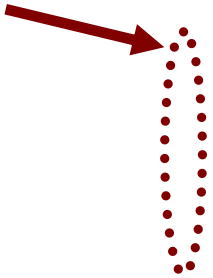
主核　从核



(a) 从核选择合适任务　(b) 主从核任务互换　(c) 主核任务互换

实验和评价

- 内核: 2.6.21.7、2.6.32
- 调度器: CFS、RSDL、O(1)
- 平台: Intel 8核
- 测试程序: NAS-SER中构造8个负载
- 独立任务执行时间减低
  - 近20%性能提升



带有硬件资源竞争感知的CFS调度器

实验和评价

- 可扩展性提升
  - 总体性能提升11.39% 转化为可扩展性12.85%

带有硬件资源竞争感知的CFS调度器

小结

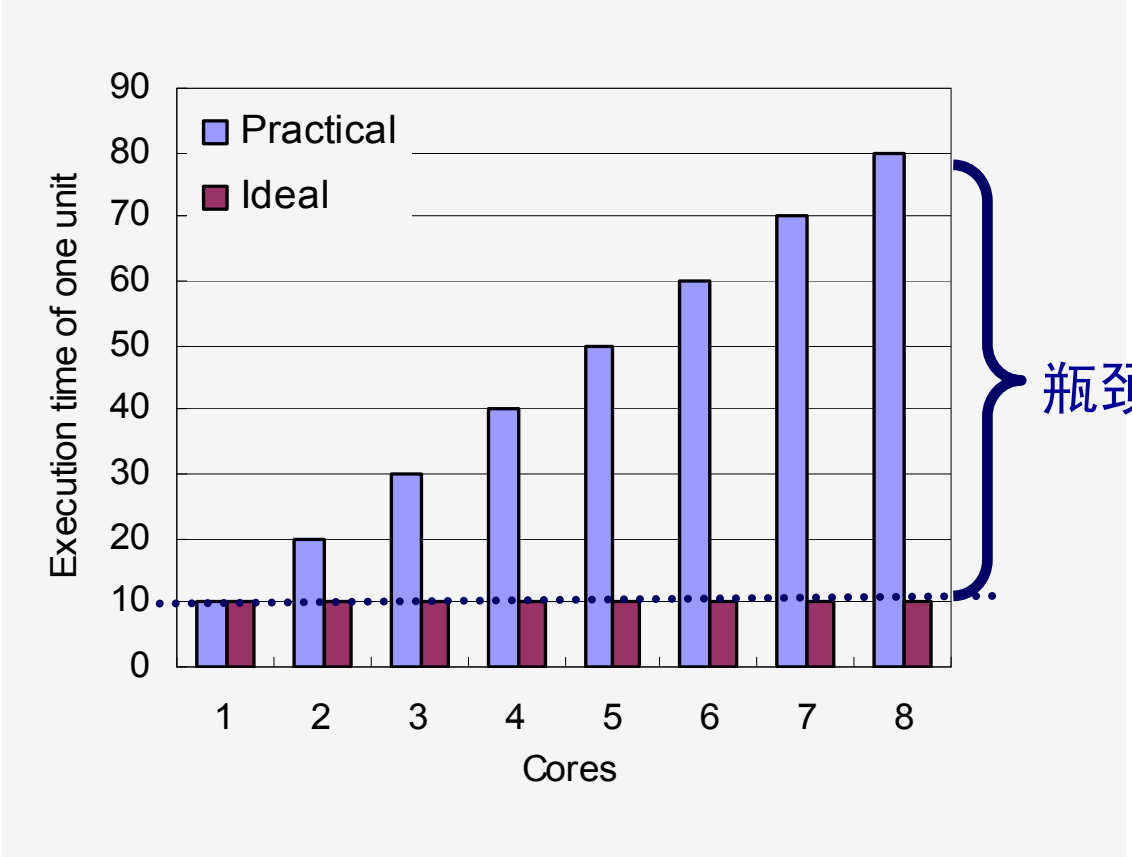
- 贡献
  - 提出了一种资源竞争感知的调度策略
    - 系统地选择启发式指标
    - 同时改进上下文切换时的调度机制和负载均衡机制
    - 实际系统整合
- 成果
  - Oxford Computer Journal 在审一篇(major revision), 专利两项
    - **Yan Cui**, Yingxin Wang, Yu Chen, Yuanchun Shi, “Mitigating Resource Contention on Multicore Systems via Scheduling”, in Oxford Computer Journal(**CJ**)(major revision, SCI&EI, **CCF rank B**)
    - 刘仪阳, 陈渝, 谭玺, **崔岩**, 一种线程调度方法、线程调度装置及多核处理系统, 申请号:201110362773, 公开号:CN102495762
    - 刘仪阳,张知缴, 方帆, 陈渝, **崔岩**, 一种内存分配方法、装置及系统, 申请号:201210176906.X

提纲

1. 选题背景
2. 论文工作
   - 分析系统服务接口、发现制约因素
   - 锁颠簸现象的模拟和避免
   - 共享硬件资源竞争降低
   - 可扩展性瓶颈定位方法
3. 总结
4. 研究成果

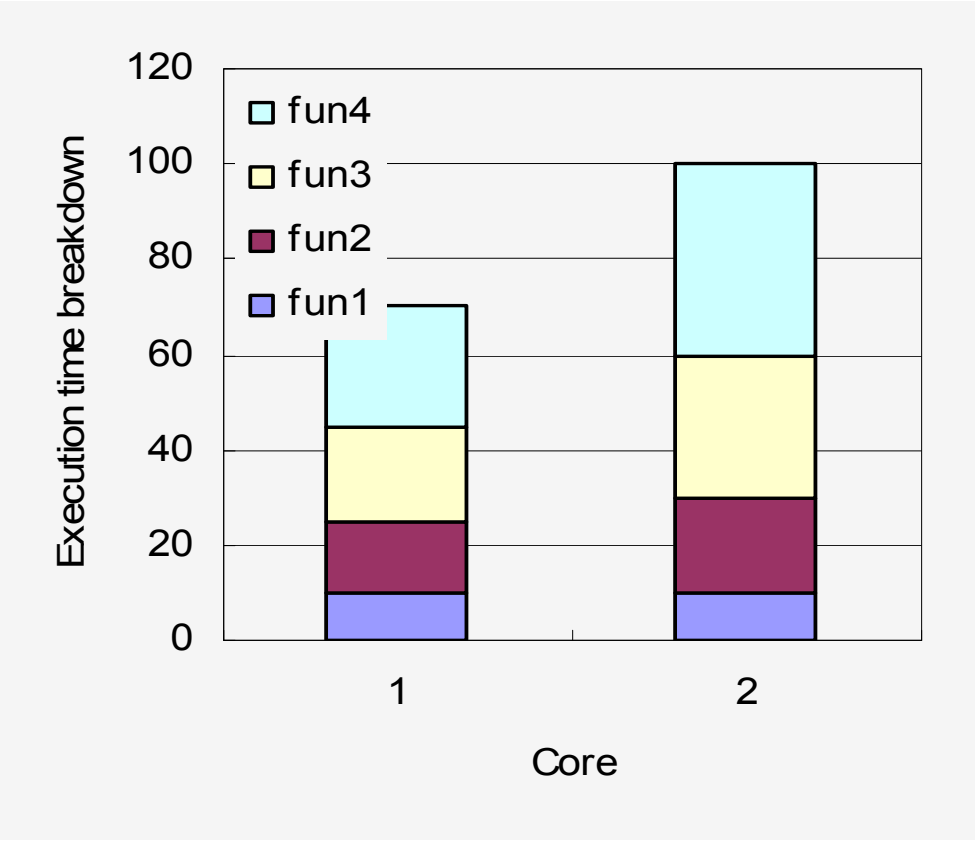
函数可扩展性值

- 若存在可扩展性瓶颈 执行单位工作量的时间在多核比在单核上长 否则不存在瓶颈



函数可扩展性值

- 每单位工作的执行时间分解至每个函数 定义为函数在多核和单核执行时间之差
- 函数包括操作系统、应用、系统库



fun(4) = 40-25=15

fun(3) = 30-20=10

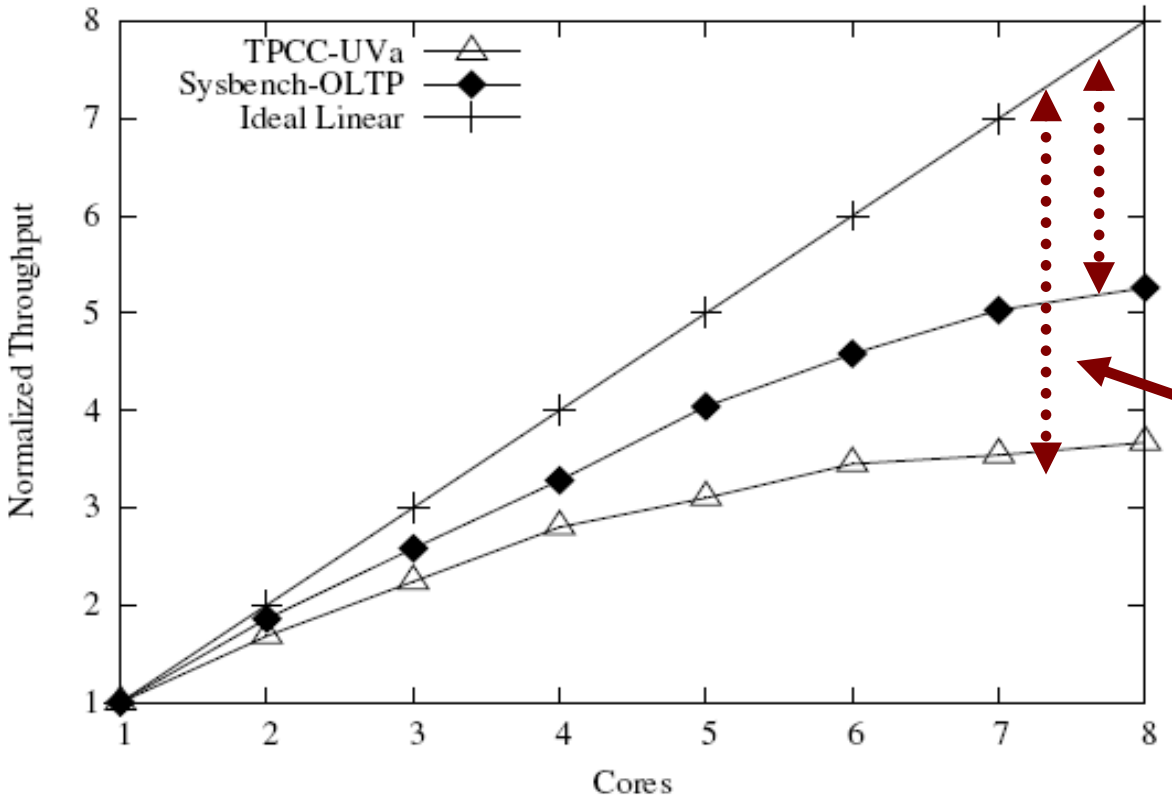
fun(2) = 20-15 =5

fun(1) = 10-10=0

分析函数4以找到最大瓶颈

实验和评价

- 内核: 2.6.25
- 平台: Intel 8核
- 测试程序: OLTP应用 TPCC-UVa (PostgreSQL) 和 Sysbench-OLTP (MySQL)
- 可扩展性



存在可扩展性瓶颈

可扩展性瓶颈定位方法

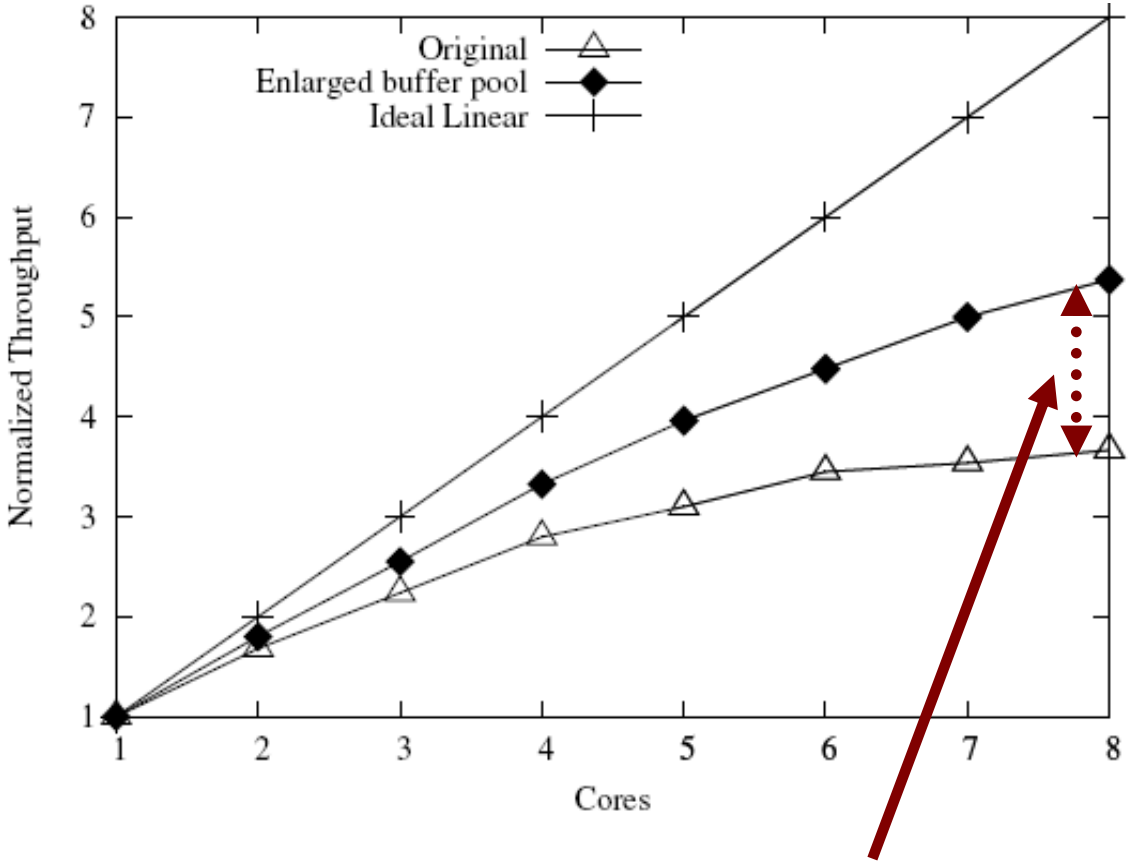
- 利用函数可扩展性值分析

TPCC-UVa

1. copy_user_generic_string() 从内核拷贝数据到用户
2. 表明数据库缓存池竞争
3. 增大缓存池直到该函数不再出现

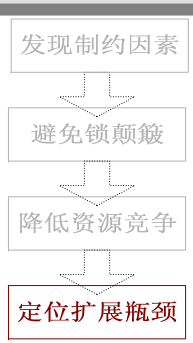
可扩展性瓶颈定位方法

- 利用可扩展性值



可扩展性提升46.52%

可扩展性瓶颈定位方法



- 利用可扩展性值

TPCC-UVa

1. 其他瓶颈: 数据库锁 调度开销 System V IPC锁竞争 可提升2%
2. 操作系统不是最大的瓶颈 继续迭代困难 架构重新设计

提升TPCC-UVa可扩展性达49%, Sysbench-OLTP达15.27%, 去除了操作系统对加速比的限制

小结

- 贡献
  - 提出了一种基于函数可扩展性值的瓶颈定位方法
    - 提升加速比达49%
- 成果
  - IEEE ISPASS发表论文一篇
    - **Yan Cui** , Yu Chen, Yuanchun Shi, “Scaling OLTP Applications on Commodity Multicore Systems”, in ISPASS 2010.

提纲

❶ 选题背景

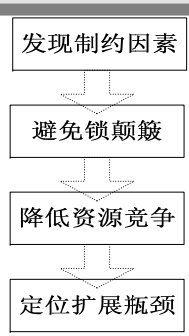
❷ 论文工作

- 分析系统服务接口、发现制约因素
- 锁颠簸现象的模拟和避免
- 共享硬件资源竞争降低
- 可扩展性瓶颈定位方法

❸ 总结

❹ 研究成果

总结



- 分析获得影响可扩展性的关键限制因素
  - 微基准测试程序集
  - 针对重要系统服务接口的深入分析和比较
- 提出了锁颠簸现象的模拟方法及避免机制与策略
  - 基于离散事件仿真技术模拟 准确、快速
  - 基于等待者数目锁 等待者数目选择等锁方式 很好避免锁颠簸
  - 锁竞争感知调度策略 调度控制竞争 比相关工作提升46%
- 提出了一种资源竞争感知的调度策略
  - 提出最佳指标的选择方法 不但考虑准确性同时兼顾稳定性
  - 同时整合任务到核的映射机制和核上任务的选择机制
  - 提升13%加速比 和最优调度接近
- 提出了一种基于函数可扩展性值的瓶颈定位方法
  - 提升加速比49%

提纲

1. 选题背景
2. 论文工作
   - 分析系统服务接口、发现制约因素
   - 锁颠簸现象的模拟和避免
   - 共享硬件资源竞争降低
   - 可扩展性瓶颈定位方法
3. 总结
4. 研究成果

论文发表(期刊)

第一作者发表的期刊

1. **Yan Cui** , Yingxin Wang, Yu Chen, Yuanchun Shi, “Lock Contention Aware Scheduler: A Scalable and Energy Efficient Method for Addressing Scalability Collapse on Multicore Systems”, in ACM Transactions on Architecture and Code Optimization. (**TACO**) (SCI&EI, **CCF rank A**).
2. **Yan Cui**, Yu Chen Yuanchun Shi, “Towards Scalability Collapse Behavior”, in Concurrency and Computation: Practice and Experience (**CCPE**) (SCI&EI,**CCF rank B**)
3. **Yan Cui**, Yu Chen, Yuanchun Shi, “Comparing Operating System Scalability by Microbenchmarking”, in IEICE Transactions on Information and Systems (**IEICE Transactions**) (SCI & EI)

第一作者发表的在审期刊

1. **Yan Cui**, Yingxin Wang, Yu Chen, Yuanchun Shi, “Mitigating Resource Contention on Multicore Systems via Scheduling ”, in Oxford Computer Journal (**CJ**) (major revision, SCI & EI, **CCF rank B**)
2. **Yan Cui**, Yingxin Wang, Yu Chen, Yuanchun Shi, “Requester-Based Lock:A Scalable and Energy Efficient Locking Scheme on Multicore Systems”, in IEEE Transactions on Computers (**TC**) (under review, SCI & EI, **CCF rank A**)

论文发表(会议)

第一作者发表的会议论文(oral)

1. **Yan Cui**, Yingxin Wang, Yu Chen, Yuanchun Shi, etc, “Reducing Scalability Collapse via Requester-Based Locking on Multicore Systems”, in **MASCOTS** 2012 (EI, **CCF rank B**)
2. **Yan Cui** , Yu Chen, Yuanchun Shi, “Experience on Comparison of Operating System Scalability on the Multicore Architecture”, in **CLUSTER** 2011 (EI, **CCF rank B**).
3. **Yan Cui**, Yu Chen, Yuanchun Shi, “Comparison of Lock Thrashing Avoidance Methods and Its Performance Implications for Lock Design”, in **HPDC workshops** 2011(EI)
4. **Yan Cui**, Yu Chen, Yuanchun Shi, “Scaling OLTP applications on Commodity Multicore Systems”, in **ISPASS** 2010 (EI)
5. **Yan Cui**, Weida Zhang, Yu Chen, Yuanchun Shi, “A Scheduling Method for Avoiding Kernel Lock Thrashing on Multicores”, in **ICPADS** 2010 (EI, **CCF rank C**)
6. **Yan Cui**, Weiyi Wu, Yingxin Wang, etc, “A Discrete Event Simulation Model for Understanding Kernel Lock Thrashing on Multicore Architecture”, in **ICPADS** 2010 (EI, **CCF rank C**)
7. **Yan Cui**, Yu Chen, Yuanchun Shi, “Parallel Scalability Comparison of Commodity Operating Systems on Many Cores”, in **ISCA workshops** 2009

论文发表(会议)

第一作者发表的张贴报告(poster)

1. **Yan Cui**, Weiyi Wu, Yu Chen, Yuanchun Shi, etc, “Reinventing Lock Modeling for Multicore Systems”, in **MASCOTS** 2010 (EI)
2. **Yan Cui**, Yu Chen, Yuanchun Shi, “Scalability Comparison of Commodity Operating Systems on Multicores”, in **ISPASS** 2010 (EI)
3. **Yan Cui**, Yu Chen, Yuanchun Shi, “Improving Kernel Scalability by Lock-Aware Thread Migration”, in **PACT** 2009

非第一作者发表的文章

1. Yingxin Wang, **Yan Cui**, Pin Tao, Yuanchun Shi, “Reducing Shared Cache Contention by Scheduling Order Adjustment on Commodity Multicores”, in **IPDPS workshops** (EI)
2. Wei Jiang, Yisu Zhou, **Yan Cui**, Yu Chen, etc, “CFS Optimization to KVM Threads on Multicore Environment”, in **ICPADS** 2009, (EI, **CCF rank C**)
3. Shen Wang, Yu Chen, Wei Jiang, Peng Li, Ting Dai, **Yan Cui**, “Fairness and Interactity of Three CPU schedulers in Linux,” in **RTCSA** 2009 (EI).

项目和专利

- 项目

1. 国家自然科学基金项目(NSFC)，“面向众核体系结构的操作系统并行优化关键技术”
2. 清华-Intel国际合作项目, “面向众核架构最后级缓存软件优化”
3. 清华-华为科技合作项目, “基于ccNUMA的性能优化项目”

- 专利

1. 秦岭，陈渝， **崔岩**，吴瑾，实现自适应锁的方法以及多核处理器系统，申请号: 201110394780 公开号: CN102566979
2. 刘仪阳，陈渝，谭玺， **崔岩**，一种线程调度方法、线程调度装置及多核处理器系统，申请号: 201110362773 公开号:CN102495762
3. 刘仪阳, 张知缴, 方帆, 陈渝，**崔岩**, 一种内存分配方法、装置及系统，申请号: 201210176906.X

致谢

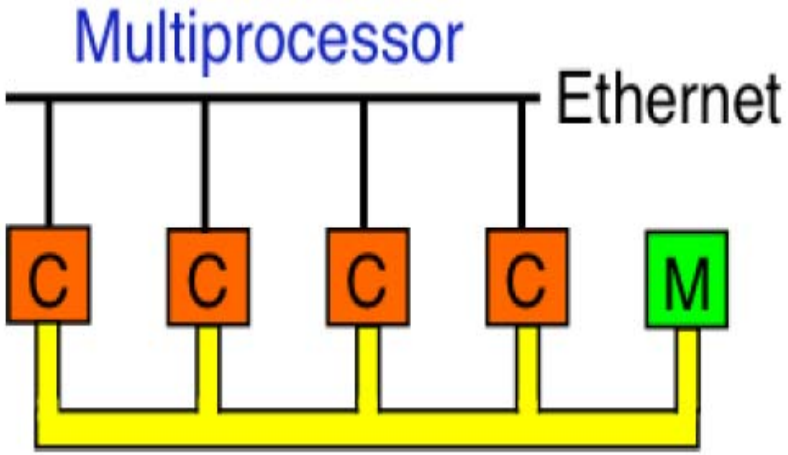
感谢各位评阅人的建设性意见!

感谢各位老师莅临指导!

欢迎大家提问!

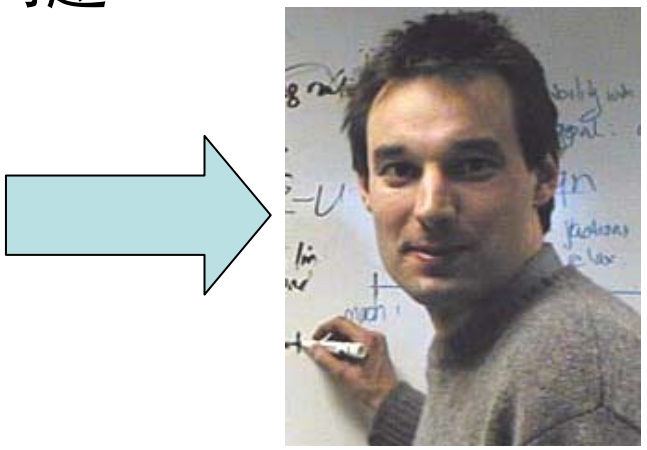
提问相关

- 多核系统与传统多CPU系统区别？
  - 并行计算系统
    - VU(1987)
    - Shared-memory cluster
    - 16 68030s (16MHZ)



提问相关

- 多核系统与传统多CPU系统区别？
  - 1987: 并行计算解决的不是紧迫的问题[APSys'12]
    - 存在更简单的性能提升方式
    - 时钟频率每18个月翻一番
    - 16*16MHZ = 256MHZ
  - 现在: 必须通过并行获得性能提升
    - 更快的单核处理器已经买不到了
    - 新时代的并行计算
  - 将来: 核数指数形式增长、更多关键硬件资源共享



[1].Frans Kaashoek. The Multicore Evolution and Operating Systems. In Keynote of APsys 2012.

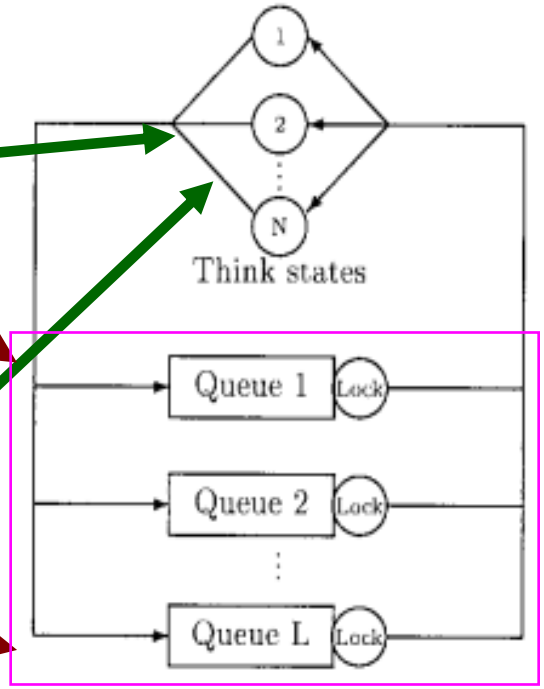
提问相关

- 锁竞争为何导致吞吐量下降？

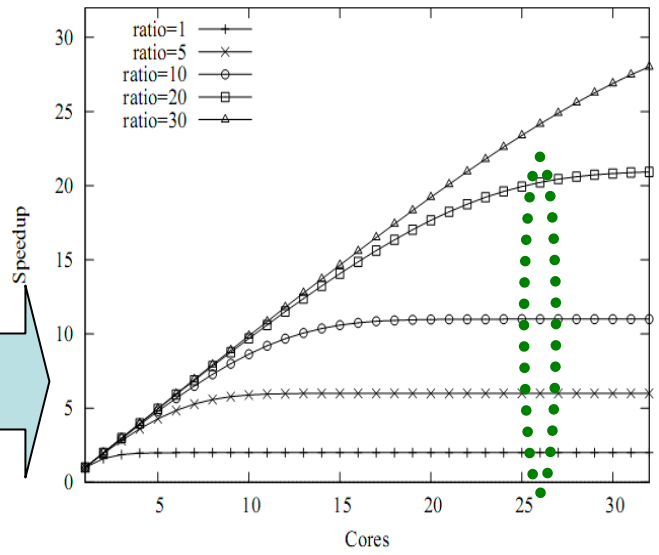
Each thread ‘s code:

```
thread (void) {
  while (1) {
      some other code
      spin_lock(&lock1);
      critical section1
      spin_unlock(&lock1);
      some other code
      …
       spin_lock(&lockL);
      critical sectionL
      spin_unlock(&lockL);
}}
```

延时服务



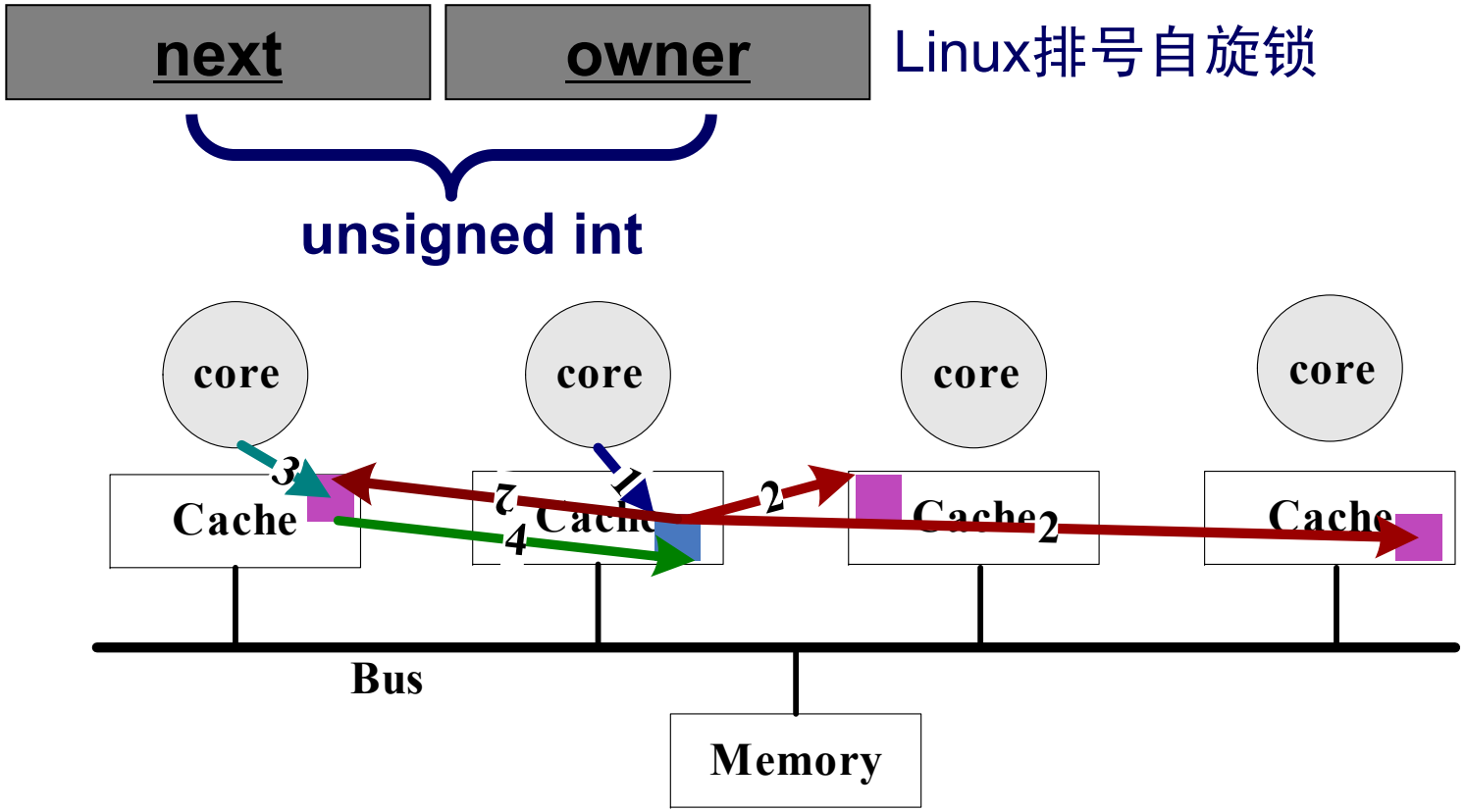
排队服务



$$ratio = \frac{T_{NCS}}{T_{CS}}$$

提问相关

- 锁竞争为何导致吞吐量下降？



提问相关

- 内核锁竞争的概率有多大？