中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

2024 CCF ChinaSoft 中国软件大会
智能软件创新赋能新质生产力发展

支持多核并行访问的高性能标签缓存设计

宋威、解达、薛子涵
中国科学院信息工程研究所

内容概要

- 标签内存

  标签内存是近年来被广泛研究的内存安全硬件防御机制，正逐渐被Arm和Intel接受并使用，比如Arm的MTE和Morello。

- 标签缓存

  纯硬件管理的标签内存系统一般在内存中预留一个特别分区，来存储内存数据的标签。但是该结构导致内存访问量增长一倍。一般采用一个标签缓存来降低上述的性能代价。

- 并行标签缓存

  最新的标签缓存采用树形结构的压缩存储，但是该结构难以支持并行访问。

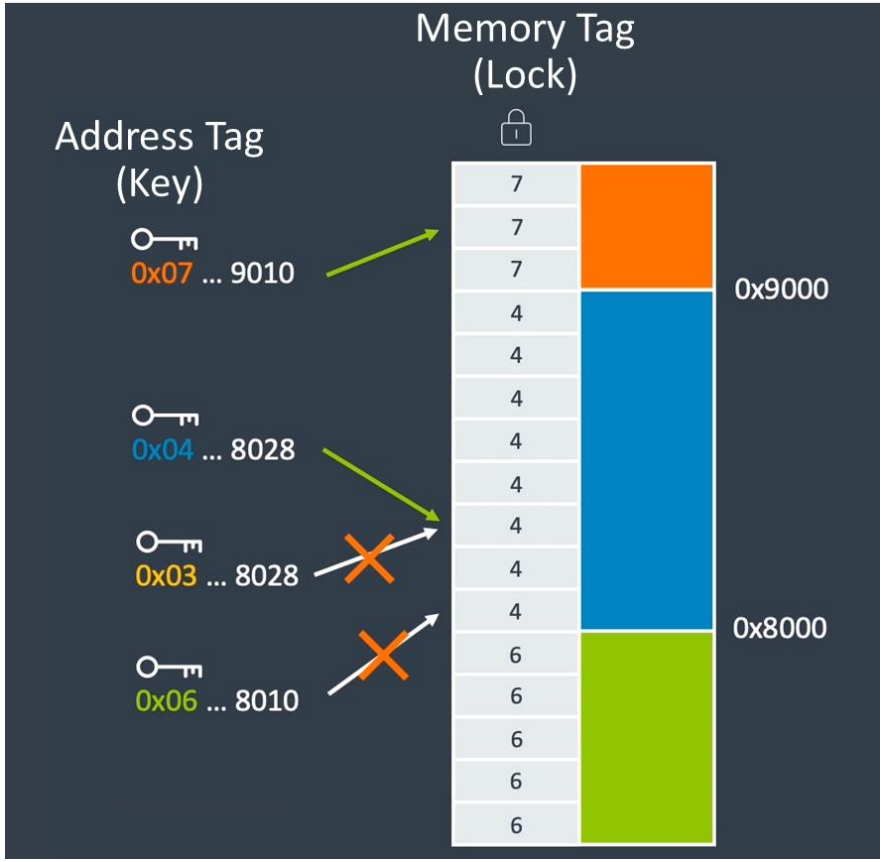
  我们在Rocket-Chip处理器上实现了树形结构标签缓存的并行访问，
  显著提升了标签内存系统的运行性能。

什么是内存标签

标签是依附于数据的元数据，在内存安全防御中用于存储安全相关的信息，供处理器在运行时高效的进行安全检查。

- Arm MTE (Memory Tagging Extension)
  - 每16B内存数据附带一个4-bit的内存标签
  - 内存标签存储数据的color
  - 指针的空闲位携带所指数据的color （key）
  - 指针解引用的时候检查color匹配（**处理器自动完成**）

```
p[17] = 255;
// check p.key == mem[p+17].tag
```



标签内存研究和应用的现状

- 标签内存已被研究领域用于支持大量内存安全防御技术的硬件实现

  边界检查、指针加密、指针完整性、控制流完整性、数据流完整性、内存时间完整性。
  已商用的ISA扩展：Arm PA，Arm MTE，Arm Morello（CHERI，剑桥大学）
  主要支持者：Arm，Intel，Dover，MIT

- 2020年以来的专利不完全统计：

  Arm：9个授权，1个申请
  Intel：16个授权，11个申请
  Dover：4个授权，5个申请
  MIT：3个授权
  合计：36个授权，19个申请中，62%扩展覆盖中国

- 英美资金支持：

  美国2010年以来：至少四个DARPA项目群，CRASH，MRC，SSITH，HARDEN。
  英国2019年以来：11M英镑(UKRI)；70M英镑(UK ISCF)；117M英镑(Google+Microsoft)。

- 国内的相关研究匮乏：

  专利：华为 1个授权，1个申请，都为衍生专利，基于Arm（MTE和PA）
  架构研究：中国科学院计算技术研究所，包云岗研究员，Labeled Architecture（众核内存带宽QoS）
  应用研究：计算技术研究所，武成岗研究员；信息工程研究所，陈李维副研究员

内存标签的硬件管理

现有的标签内存系统一般采用物理地址寻址的纯硬件管理方式。

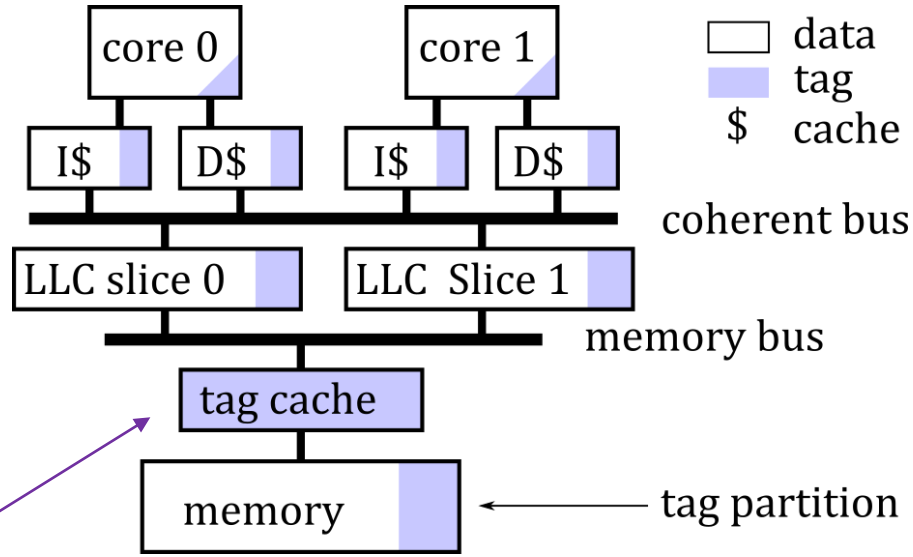
- 处理器内：扩展机器字长

  从处理器核到末级缓存，标签和机器字一起存储
  - 直接支持一致性
  - 支持和内存的直接映射
  - 不需要考虑虚拟页问题

- 内存：独立的标签区域
  - 不需要扩展内存控制器
  - **内存访问量加倍** → **标签缓存**

    专门存储常用标签

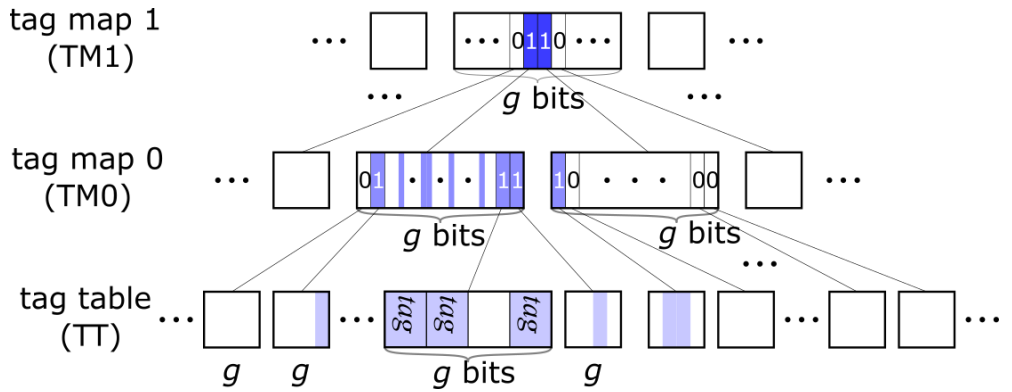
    内存访问量下降90%

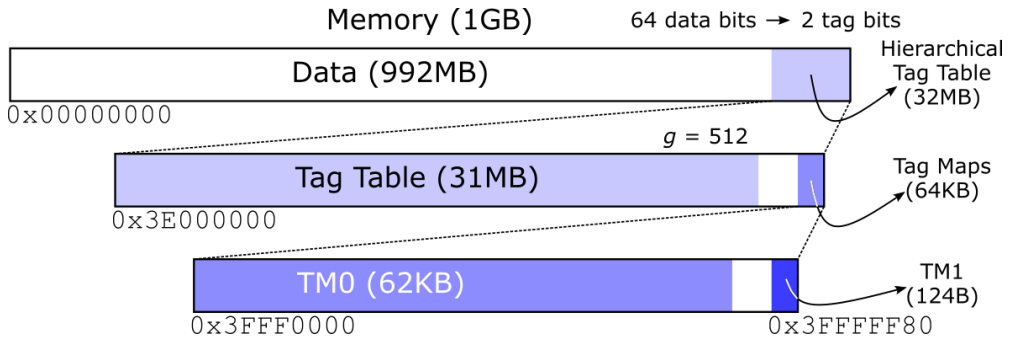


树形压缩的标签缓存

现有最先进的标签缓存架构为CHERI提出的树形压缩结构。

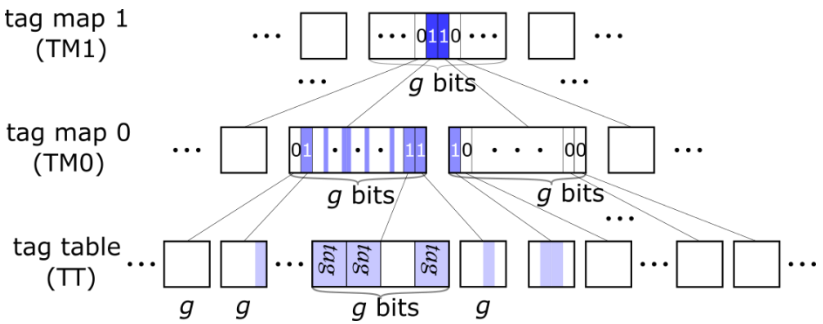
- 大部分的内存数据没有标签
  - 当连续的多个内存数据的标签均为0时，用上层的一个标签位图（tag map）比特标注，则可以避免缓存该全零的标签。
  - 标签位图不需要额外的存储空间。
    内存内的剩余空间已经足够存储标签位图。

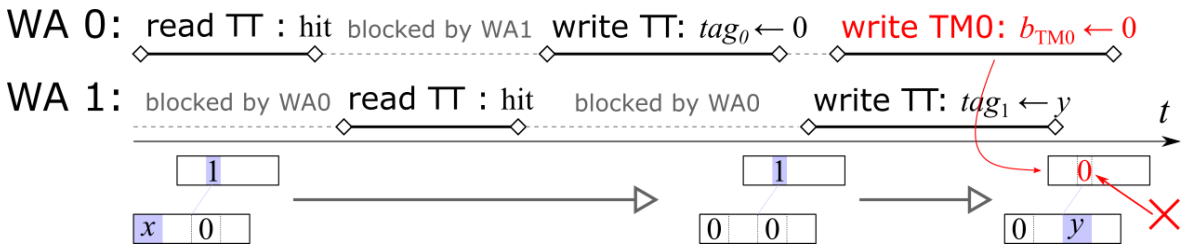




树形压缩的标签缓存难以并行读写（1）

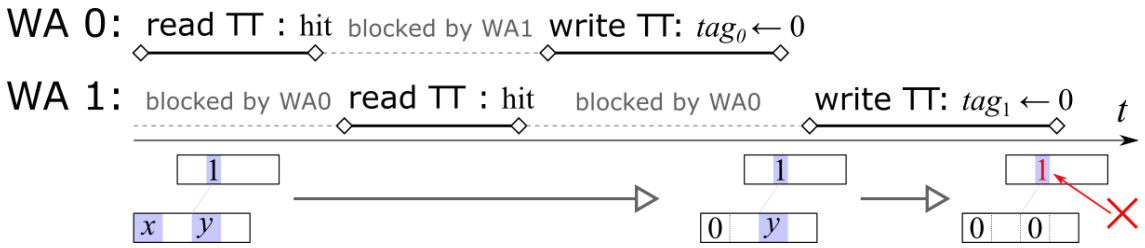
树形压缩结构引入了tag map（TM)和tag（TT)之间的关联，对于标签的改动需要以原子操作的方式修改其位图比特，带来了一致性问题。





WA0清除标签，WA1写新标签y。

并行的WA1写没有发现TM0已经被WA0清除了。



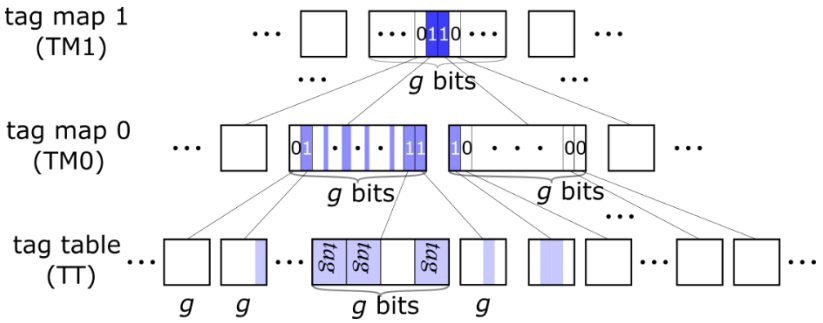
WA0和WA1都是清除标签。

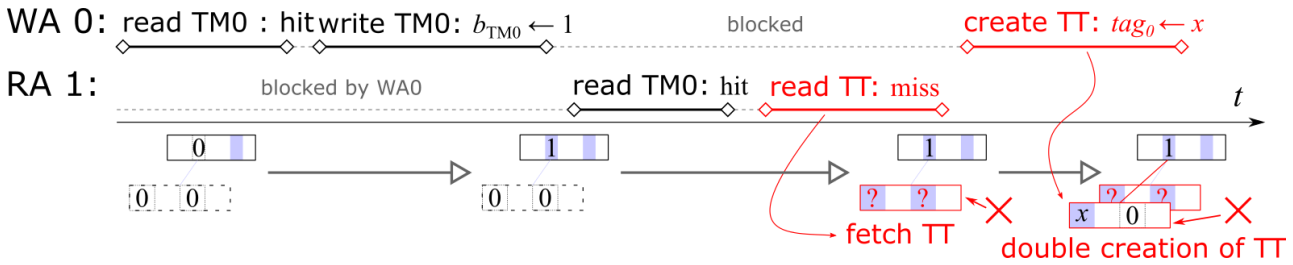
WA0和WA1都不认为自己需要清除TM0比特。

树形压缩的标签缓存难以并行读写（2）

树形压缩结构的优化进一步加大了问题的复杂度。

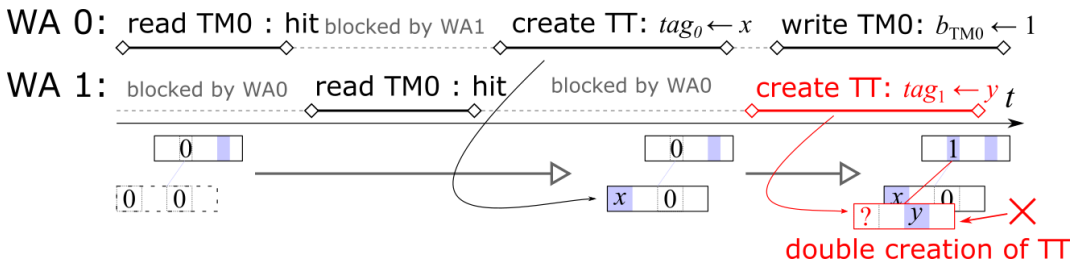
- **写避免**：避免写回空标签缓存块
- **读避免**：缓存直接建立空标签缓存块





WA0创建标签，RA1读空标签。

RA1查完TM0之后，去访问TT时，TT已经被WA0给无效了。WA0还直接建立了一个新的缓存块。



WA0和WA1都各自新建了一个缓存块。

解决树形结构的并行访问问题

树形结构标签缓存的并行问题来源于层级之间的数据依赖问题：

一个事务（transaction）要么完整更新所有层级内的相关数据，要么不做任何修改。

- 一致性问题：

  被一个事务部分修改的数据对其他事务不可见。

  解决方案：**数据库的两阶段封锁协议（two-phase locking）**

- 死锁/活锁问题：

  一旦一个事务开始，一定能够在有限时间内完成。

  解决方案：**读写顺序一致化**

两阶段封锁协议（two-phase locking）

一种较为保守的冲突避免机制，不需要事先知道相关事务之间的依赖关系，保证事务完成后的数据一致性，但是不保证事务能够完成（可能死锁）。

- 假设事务需要访问/修改多个记录（缓存块）
  - 增长阶段

    事务可以不断地申请记录独占访问权限（上锁）
  - 减少阶段

    事务只能释放锁而不能锁新的记录

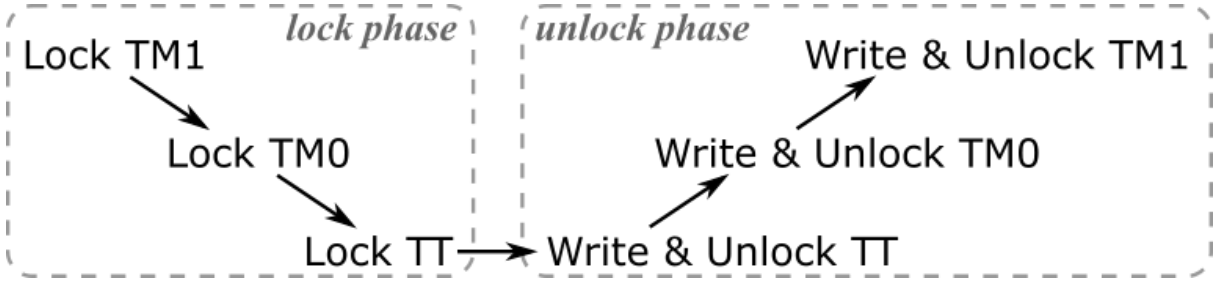
一致性：两阶段的划分保证了一个事务在某一个时刻一定获取了所有的记录访问权限，因而其操作可以原子性的完成。

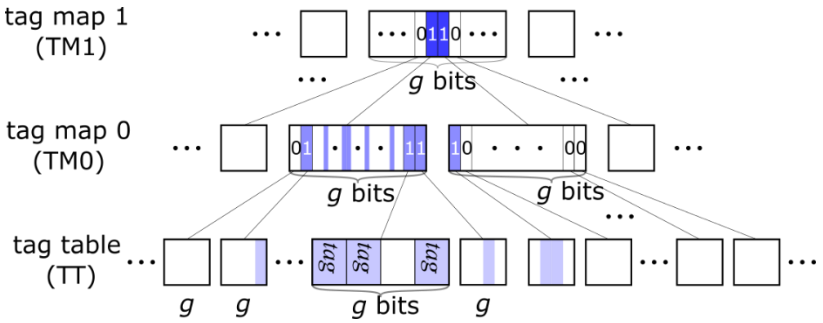
死锁可能性：如果两个冲突事务各自锁住了对方需要的记录，则发生死锁。

其关键是记录访问顺序的问题！

写事务的访问顺序

只有写事务会改变缓存状态，因而需要先确认写顺序。





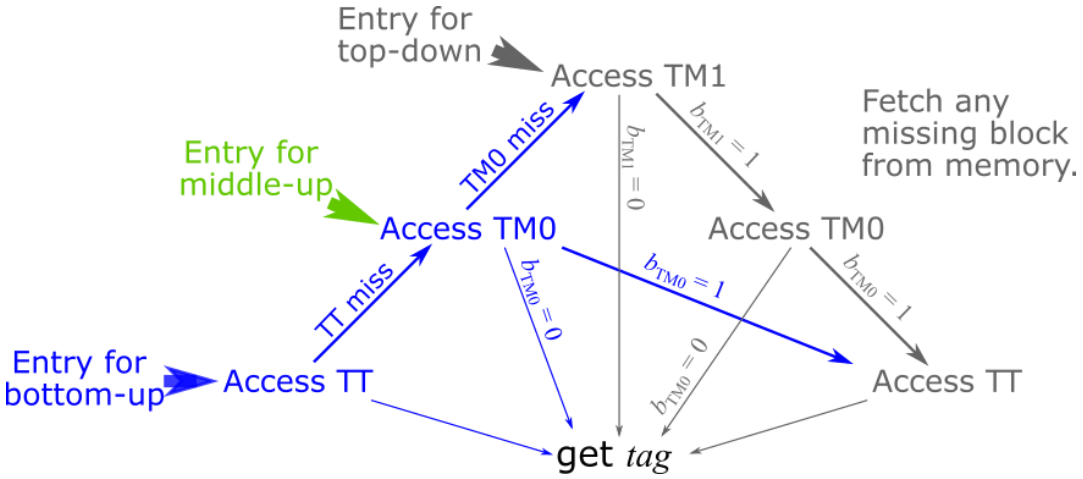
- 先读后写：读加锁，写解锁，强制two-phase locking
- 由于所有相关的写事务共享相同的TM1比特，该顺序杜绝了死锁。

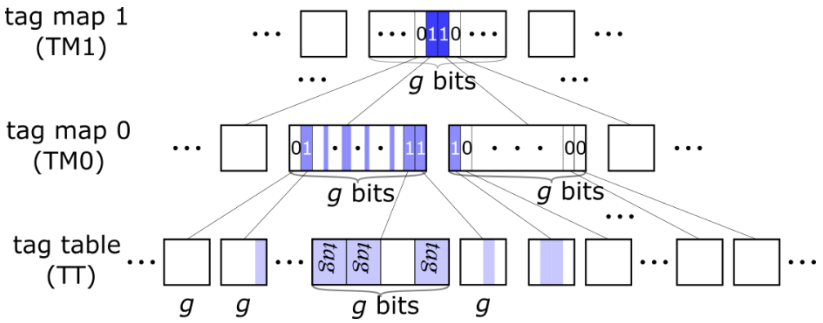
优化：

- 锁只是占位符，当缓存块不存在时，并不强制从内存取回缓存块。（TM1=0，不取回TM0，TT）
- 写的时候如果推断上级缓存块无需改动，写操作退化成解锁操作。
- TM1和TM0锁住的是1bit，而不是缓存块，尽可能降低假冲突。
- TT并不上锁。
- 从根节点开始的顺序，让TT节点先释放，有利于降低对读事务的影响（大部分为读事务，SPEC CPU 2006约为80%）。

带推测的读访问顺序

以三级树形结构为例，存在三种访问顺序：





- 自顶而下：符合常识，必须存储空的TM1和TT节点，适合极少用标签的应用，当TT大量命中时效率低下。
- 自底而上：如果TT命中，则无需访问TM节点，缓存可以踢出空的TM节点节省空间，适合重度使用标签的应用。
- 中间插入：两者的折中方式。

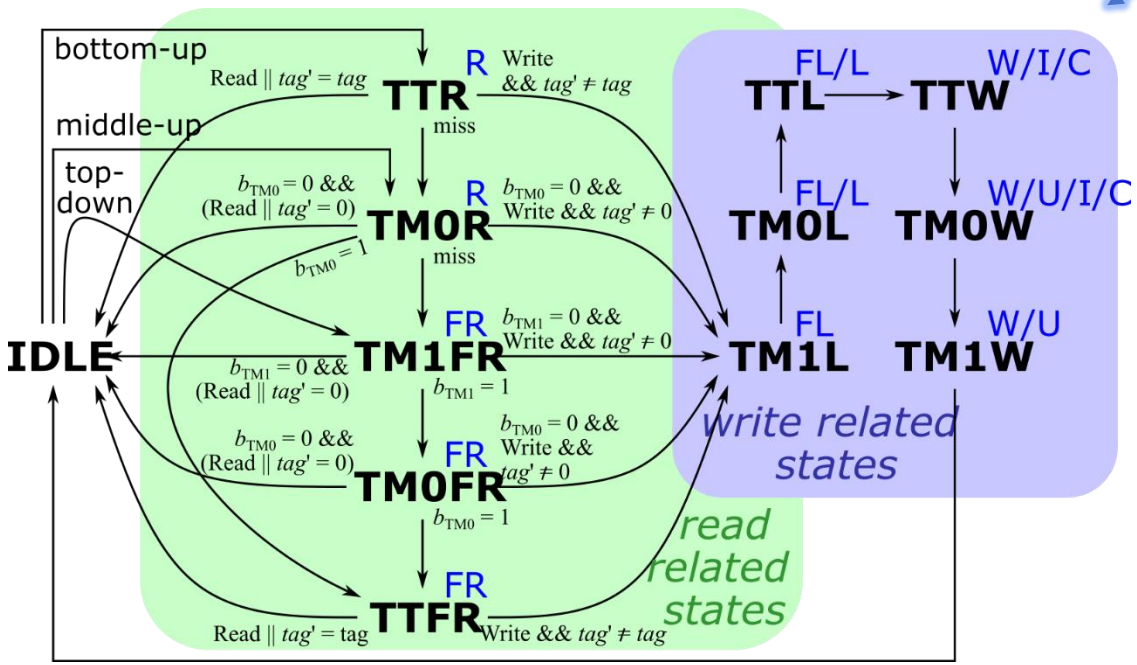
优化：动态顺序选择（假设大部分为读操作，实际测量SPEC CPU 2006读操作接近80%）

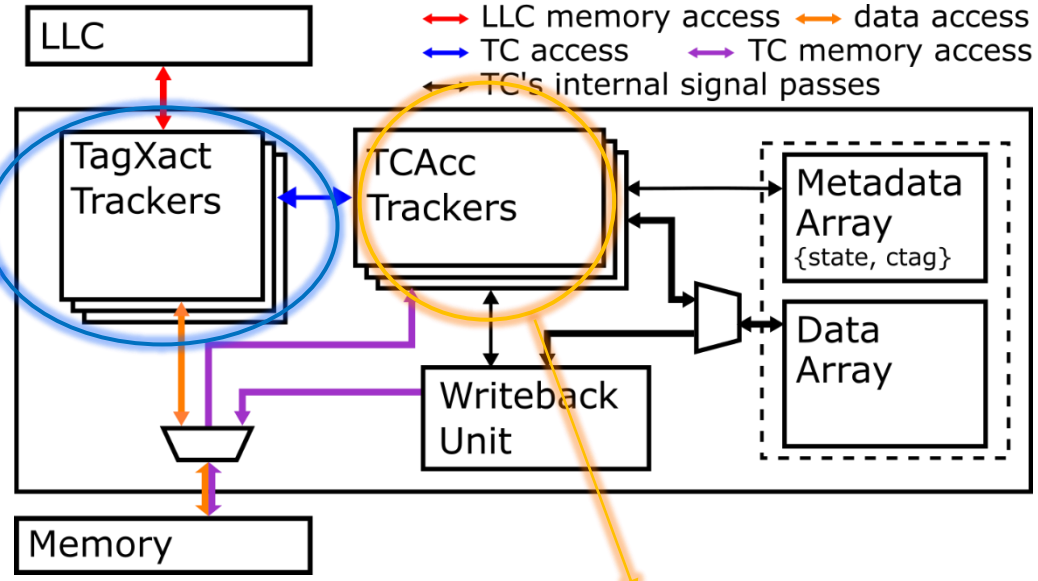
缓存记录最终读取到标签的层级，当TM1的服务占比超过50%选择自顶而下，当TT超过50%选择自底而上，其他选择中间插入，所需访问的缓存块数量最低。

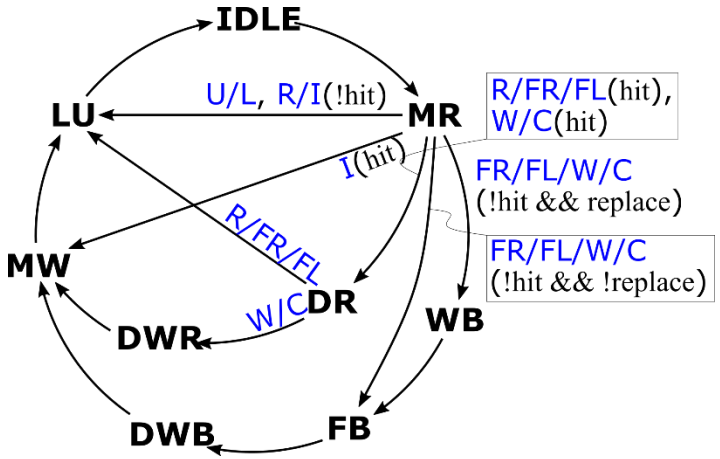
缓存结构

高可配的标签缓存：

支持树形压缩，并行读写，1~3层可配，读顺序可配



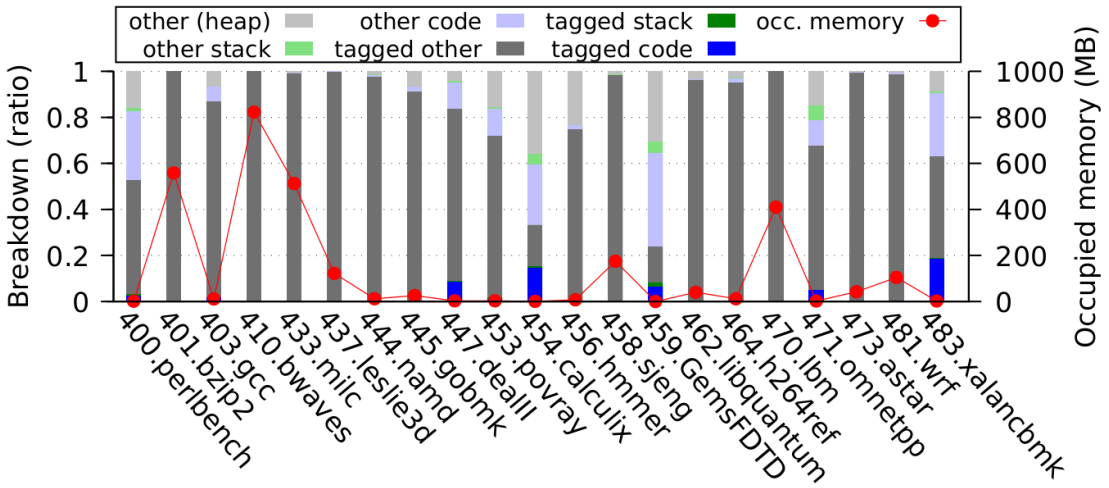




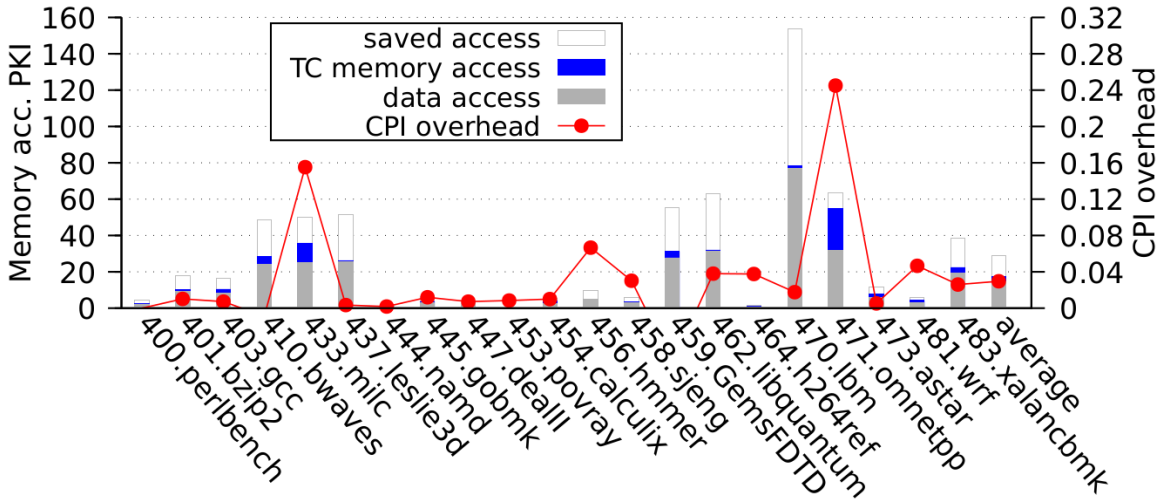
单核标签测试用例

Rocket-Chip，每64-bit机器字2-bit标签，8KB标签缓存，跑SPEC CPU 2006

- 三个基于标签的安全防御：
  1. 标注call/ret指令（指令标签）
  2. 标注返回地址，阻止ROP（栈上标签）
  3. 标注已分配内存，回收机制（堆上标签）



内存占用和占比



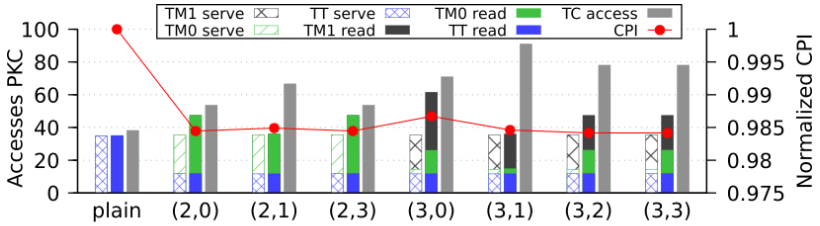
FPGA实测性能

在较重的标签使用下，仅8KB的标签缓存，节省了87.3%内存附加访问，CPI代价2.97%。

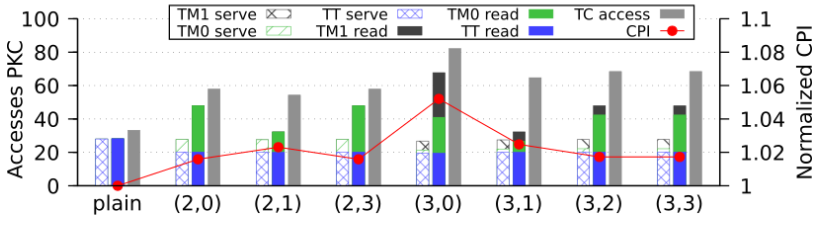
四核并行应用程序（1）

四核测试，每核跑一个SPEC CPU 2006测例，标签缓存扩大至32KB。

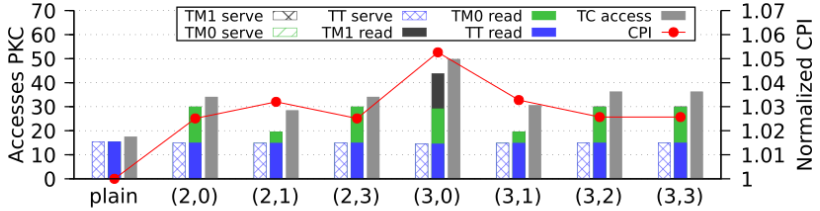
Combination A: 410.bwaves, 437.leslie3d, 459.GemsFDTD, 470.lbm **（大内存访问，但是标复用率高）**
Combination B: 459.GemsFDTD, 462.libquantum, 471.omnetpp, 483.xalancbmk **（大内存访问，一个标签复用率低测例）**
Combination C: 403.gcc.input4, 403.gcc.input6, 433.milc.input0, 471.omnetpp **（轻内存访问，两个标签复用率低测例）**



(a) Combination A



(b) Combination B

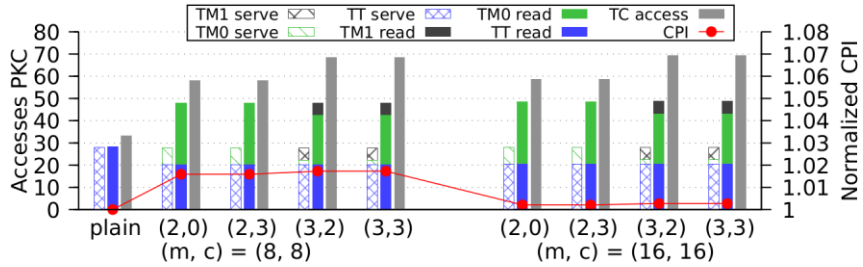


(c) Combination C

当标签复用率高时，多级压缩能有效降低CPI。

当标签复用率低时，使用非压缩结构反而更好。

自动搜索顺序选择总是能达到较好的CPI性能。

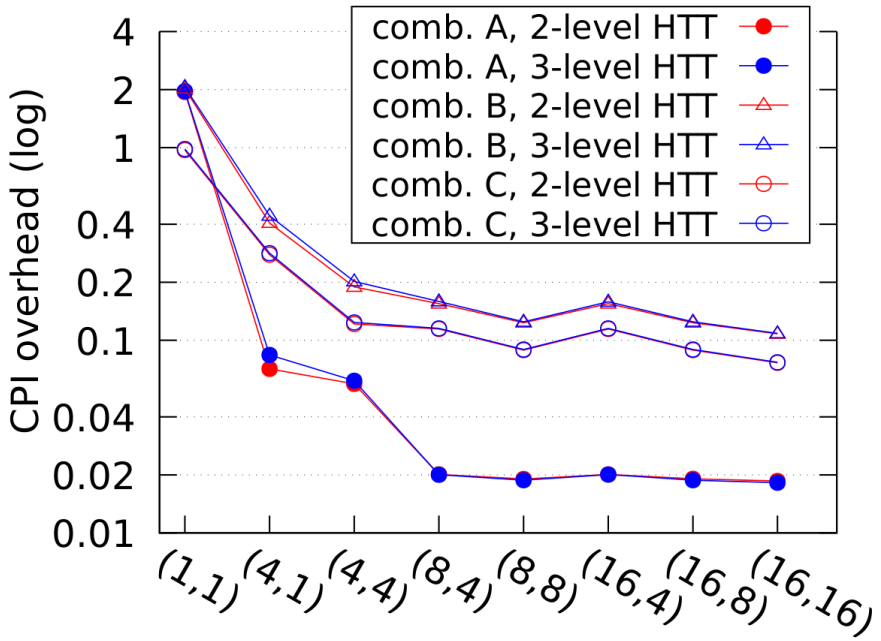


并行8个访问增加到并行16个访问，CPI损失降低值0.27%。
增加MSHR的数量能显著降低CPI损失。

四核并行应用程序（2）

四核测试，每核跑一个SPEC CPU 2006测例，标签缓存扩大至32KB。

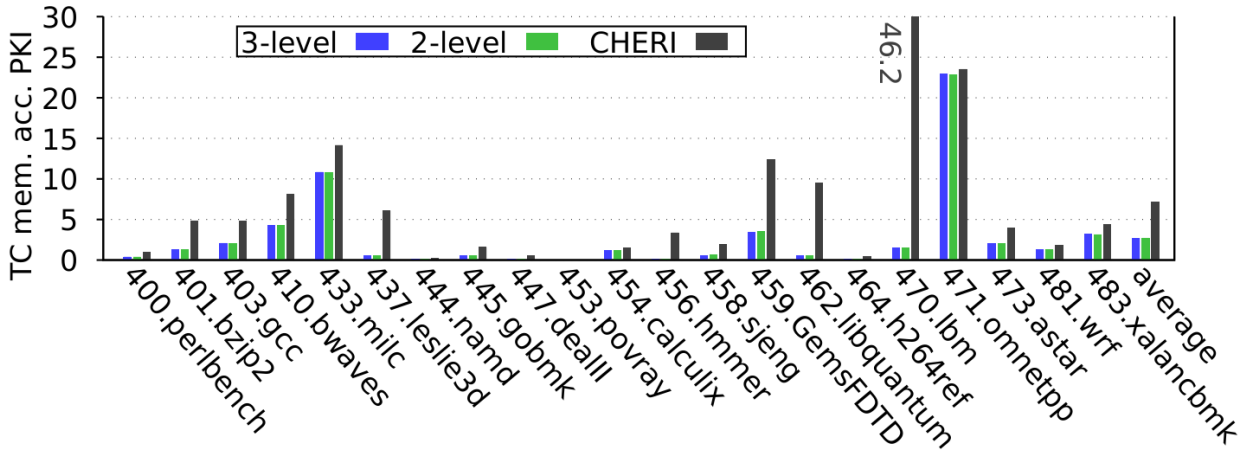
Combination A: 410.bwaves, 437.leslie3d, 459.GemsFDTD, 470.lbm **（大内存访问，但是标复用率高）**
Combination B: 459.GemsFDTD, 462.libquantum, 471.omnetpp, 483.xalancbmk **（大内存访问，一个标签复用率低测例）**
Combination C: 403.gcc.input4, 403.gcc.input6, 433.milc.input0, 471.omnetpp **（轻内存访问，两个标签复用率低测例）**



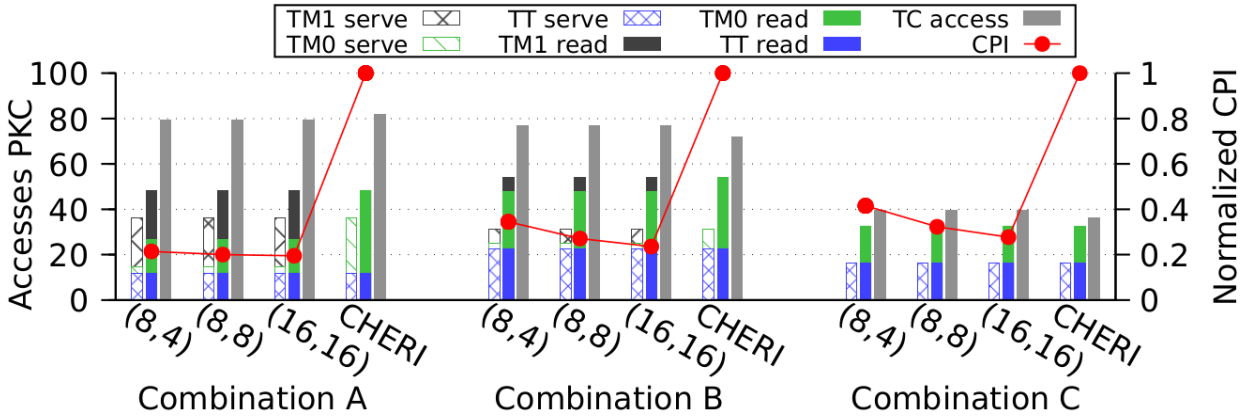
增加并行访问的数量，能显著降低CPI损失，支持8个并行访问大概能满足四核的需求。

和CHERI的标签缓存的对比

CHERI的标签缓存实现在最先进的实现（多级树形压缩，统一存储）



单核性能：
内存访问量下降164%，CPI损失下降1.6%。



四核性能：
支持8个并行访问时，CPI损失下降58~79%；
支持16个并行访问时，CPI损失下降76~81%。

总结

标签内存是众多内存安全机制硬件实现的基础。标签缓存是提升硬件管理的标签内存系统性能重要部件。树形压缩的标签缓存存储效率很高，但是难以并行访问。

我们利用two-phase locking技术实现了属性压缩缓存的并行访问支持。

- 自动搜索顺序选择实现了缓存对不同负载的自适应。
- 增加并行访问量能够邮箱降低多核情况下的CPI损失。
- 相比CHERI，CPI损失降低高达80%。
- 是现在最为先进的标签缓存结构。

http://github.com/comparch-security/lowrisc-tag-v0.4

Wei Song, Da Xie, Zihan Xue, and Peng Liu. “A parallel tag cache for hardware managed tagged memory in multicore processors”. IEEE Transactions on Computers, 2024, 73(11): 2488-2503

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

2024 CCF ChinaSoft 中国软件大会
智能软件创新赋能新质生产力发展

谢谢！