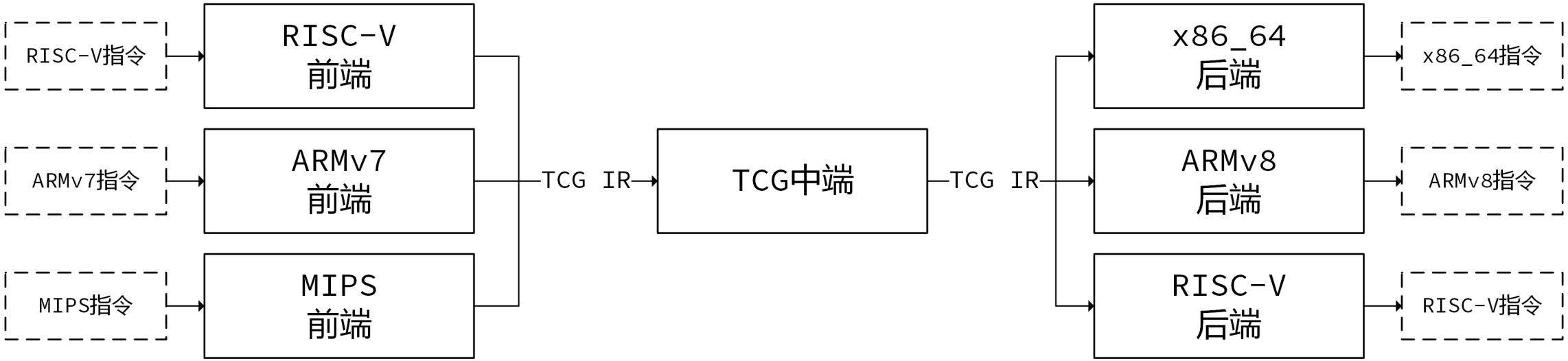
基于地址空间标识符的QEMU动态跳转优化

位金弈、梁洪亮
北京邮电大学计算机学院

目录

QEMU-TCG简介

- 开源仿真器QEMU-TCG
  - 基于动态二进制翻译技术完成指令集架构的功能仿真
    - Tiny Code Generator



  - 包含用户级仿真和系统级仿真两种模式

研究动机

- QEMU动态跳转通过运行时翻译块查找完成，且依赖地址翻译
  - 间接跳转、跨页直接跳转
  - 指令未编码目标地址、内存映射可变等导致跳转目标不确定

- **间接跳转**使用的缓存需要频繁刷新，导致反复的地址翻译

- **跨页直接跳转**保守地视为动态跳转
  - 开销在数十到数千条指令不等，不跨页则开销仅为单条汇编指令
  - 常见于程序模块间的函数调用，且变化可能性小

动态跳转优化 – 观察与讨论

- 翻译块查找无需使用完整物理地址
  - 只需要匹配虚拟地址并区分不同的地址空间

- TLB 刷新本身不涉及内存映射内部的改变
  - 监测页目录*可以区分地址空间的整体切换与局部修改

- 对页目录的监测只需关注被使用过的部分
  - 消除主动探测页目录完整结构所带来的开销

* 假设地址空间可由一个页目录完全决定

动态跳转优化 – QASID的定义与维护

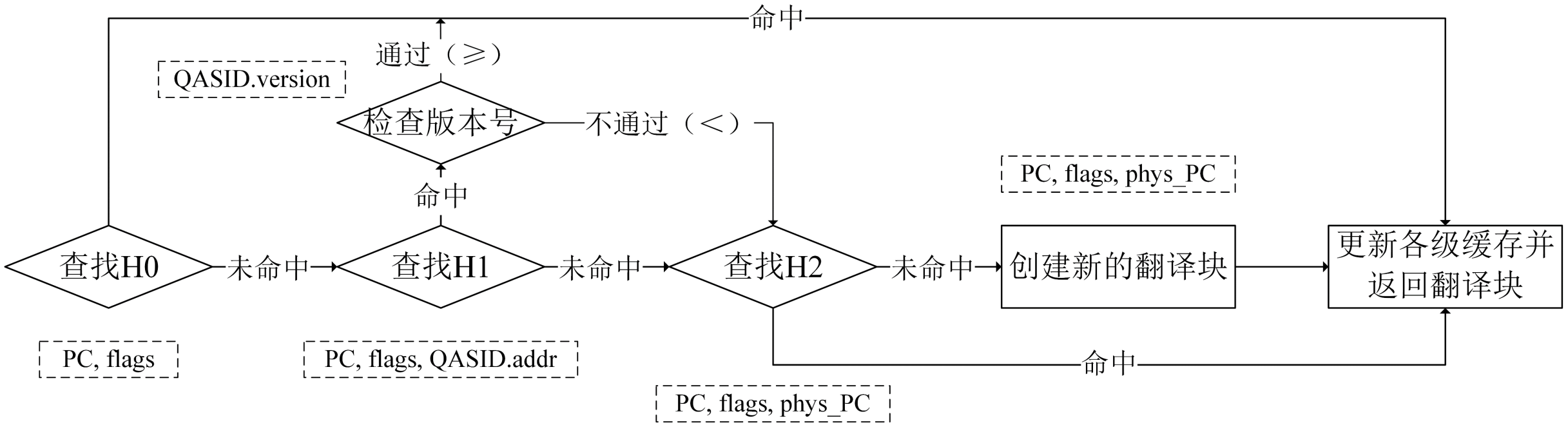
- 使用页目录地址与单调递增版本号唯一描述一个地址空间
  - 通过比较QASID可以判断内存映射是否发生变化

- 检测到页目录被修改时更新页目录版本号
  - 作指令地址翻译时监视相关页表页，类比DIRTY_MEMORY_CODE
  - 剪枝优化以减少无用的版本更新：对无效页表项、过时页表页的写操作均可忽略

- 页表切换或TLB刷新时vCPU获取当前生效的QASID
  - TLB与页表的coherence本身就是由软件维护的

动态跳转优化－间接跳转

- 仿照H2引入哈希表H1用于索引TranslationBlockWrapper，并以QASID.addr替换键phys_PC
- H1命中后还需检查Wrapper的版本号QASID.version

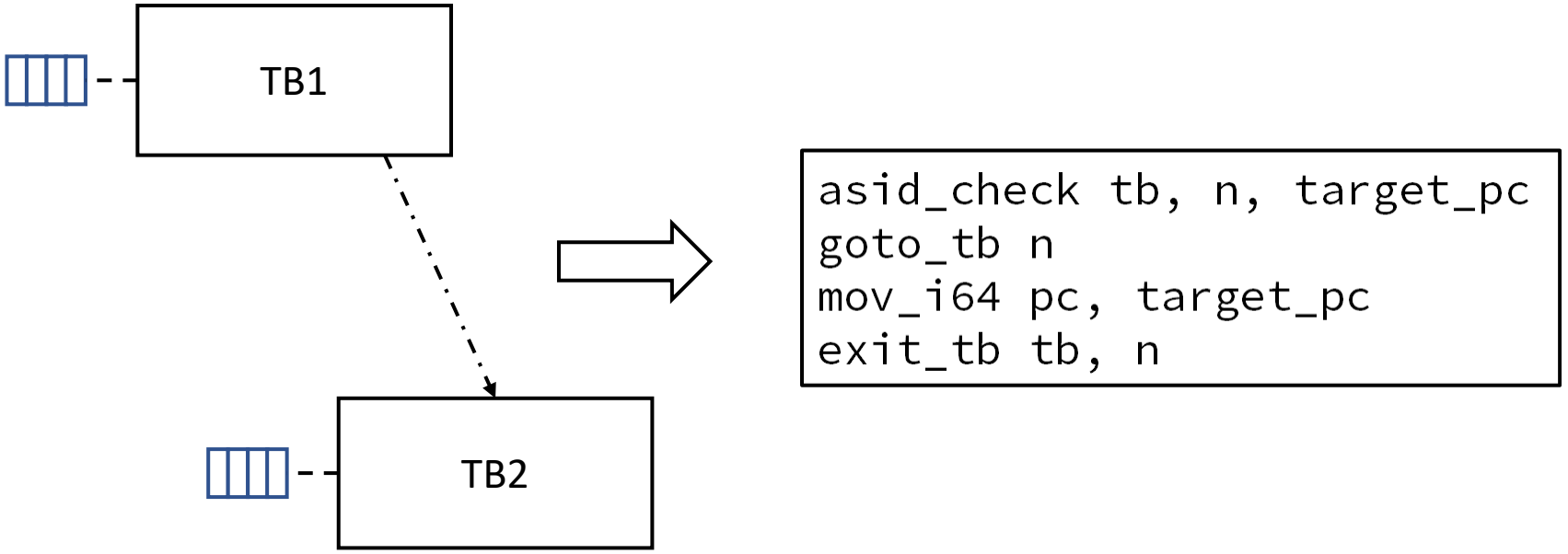


动态跳转优化 – 间接跳转

- Wrapper版本号小于 vCPU 版本号， H2 中使用物理地址查找到相同的翻译块
  - 内存映射改变但未影响当前翻译块，更新Wrapper版本号即可
- Wrapper版本号小于 vCPU 版本号， H2 中使用物理地址未查找到或查找到不相同的翻译块
  - 内存映射改变且影响当前翻译块，创建新的Wrapper记录翻译块和版本号
- Wrapper版本号大于 vCPU 版本号
  - 其他vCPU线程更新了Wrapper的版本号，可视为命中

动态跳转优化－跨页直接跳转

- 原本的困境在于QEMU缺少合适的手段验证跳转目标的正确性
- QASID提供了一种检测内存映射改变的高效手段
  - 在跨页直接跳转前插入运行时检查，确保跳转目标的正确性



动态跳转优化－跨页直接跳转

- 在翻译块尾部分配QASID缓存，在跳转前与vCPU的QASID比较
- 不相等则进入slow path调用asid_check_helper做进一步处理
  - 使用当前QASID和跳转目标地址查询H1

```
asid_check tb, n, target_pc
# 读取vCPU的编号
movl -0xc(%rbp), %edi

# 读取vCPU的地址空间标识符
movq -0x8(%rbp), %rsi

# 与翻译块缓存的地址空间标识符做比较
leaq 0x..(%rip), %rdx
cmpq (%rdx, %rdi, 8), %rsi

# 标识符不匹配则跳转
jne slow_path

# 后续指令
next_instr:
```

```
slow_path:
# 传递参数并通过尾调用跳转至辅助函数
movq %rbp, %rdi
leaq $(tb | n), %rsi
movq $target_pc, %rdx
leaq $next_instr, %rax
push %rax
jmpq asid_check_helper
# 尾调用直接返回
```

动态跳转优化－跨页直接跳转

- 若H1命中、版本号匹配且对应翻译块与goto_tb链接状态相符，则
  - 更新翻译块的QASID缓存并返回
  - 可看作QASID缓存冲突

- 否则断开后续goto_tb的链接并返回
  - 执行流通过exit_tb返回QEMU并尝试重新链接
  - 可能是内存映射发生改变或同时存在多个目标TB

- 若不考虑线程间翻译块共享，则方案可以更加简化
  - 翻译块包装QASID变化时，断开翻译块已有的跨页链接

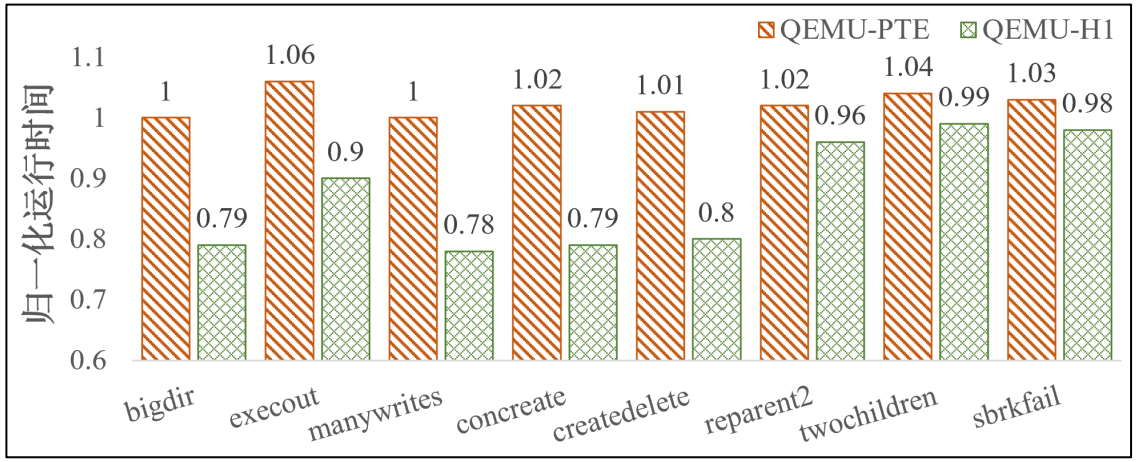
实验评估

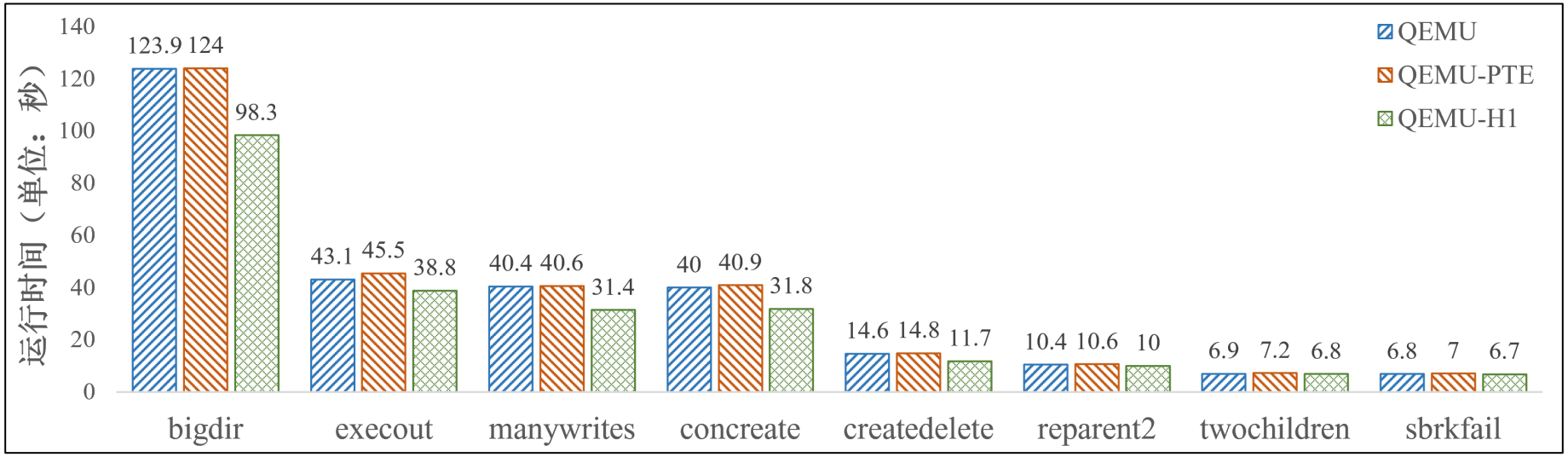
- 使用RISC-V架构教学用UNIX系统xv6-riscv的自带测试程序集
  - 共包含60余测试程序，涵盖了进程管理、内存分配、文件系统等方面
  - 去除原本的-mno-relax编译选项，以便将auipc+jalr尽可能转化为jal指令

名称	程序逻辑描述
bigdir	在当前目录下创建文件，并为其创建 500 个硬链接，随后删除所有硬链接
execout	创建子进程分配系统的全部内存，释放其中的少部分后加载新程序，测试 exec 的错误恢复功能
manywrites	创建若干子进程并发创建文件并执行写入操作
concreate	创建若干子进程对相同文件并发执行创建、删除和链接操作
createdelete	创建若干子进程在相同目录中并发创建和删除文件
reparent2	创建若干子进程，子进程继续执行 fork 后不经 wait 直接退出
twochildren	创建两个子进程，并使其同时退出，循环 1000 次
sbrkfail	创建子进程测试内存分配失败后的清理机制

实验评估

- 文件操作涉及较多跨模块操作
  - 文件系统日志、inode管理与查找
  - 磁盘块读写、用户空间数据交换
  - xv6-riscv未作小函数内联优化
- 跨页直接跳转不再争用查找缓存





不足与展望－不足

- 不能处理实际场景中复杂的虚拟内存系统
  - RISC-V物理内存保护机制PMP——pmpcfg, pmpaddr
  - RISC-V虚拟机监视器扩展H-ext——vsatp, hgatp两阶段地址翻译
- 破坏了QEMU原本的breakpoint功能
  - 6.1.0版本后中程序断点的触发需要翻译块与QEMU的配合
- 仅对翻译块查找过程中的地址翻译做了改动
  - qemu_ld, qemu_st

不足与展望 – 展望

- 跨指令集架构仿真的硬件支持
  - BTMMU[1]：使用对偶硬件TLB，依然需要依靠软件模拟页表查询过程
  - DAISY[2]：为PowerPC动态二进制翻译做专门支持的VLIW处理器

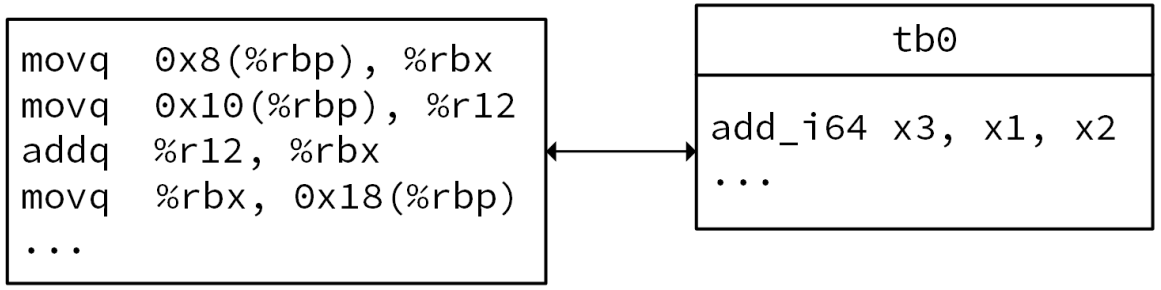
For efficient binary translation support, the DAISY architecture contains a “load VLIW instruction address (LVIA)” operation. **The LVIA operation loads the VLIW instructions corresponding to a section of PowerPC code.** If the code is already translated and in memory, the LVIA operation returns the address of the appropriate VLIW starting instruction, otherwise it loads the VMM translator, which translates the PowerPC code before executing the corresponding VLIW instructions.

[1] Huang K, Zhang F, Li C, et al. BTMMU: An Efficient and Versatile Cross-ISA Memory Virtualization.
[2] Ebcioglu K, Fritts J, Kosonocky S, et al. An Eight-Issue Tree-VLIW Processor for Dynamic Binary Translation.

感谢大家的聆听

TCG Internal - 1

- 以翻译块（TranslationBlock）为单位进行运行时指令翻译和缓存



≈

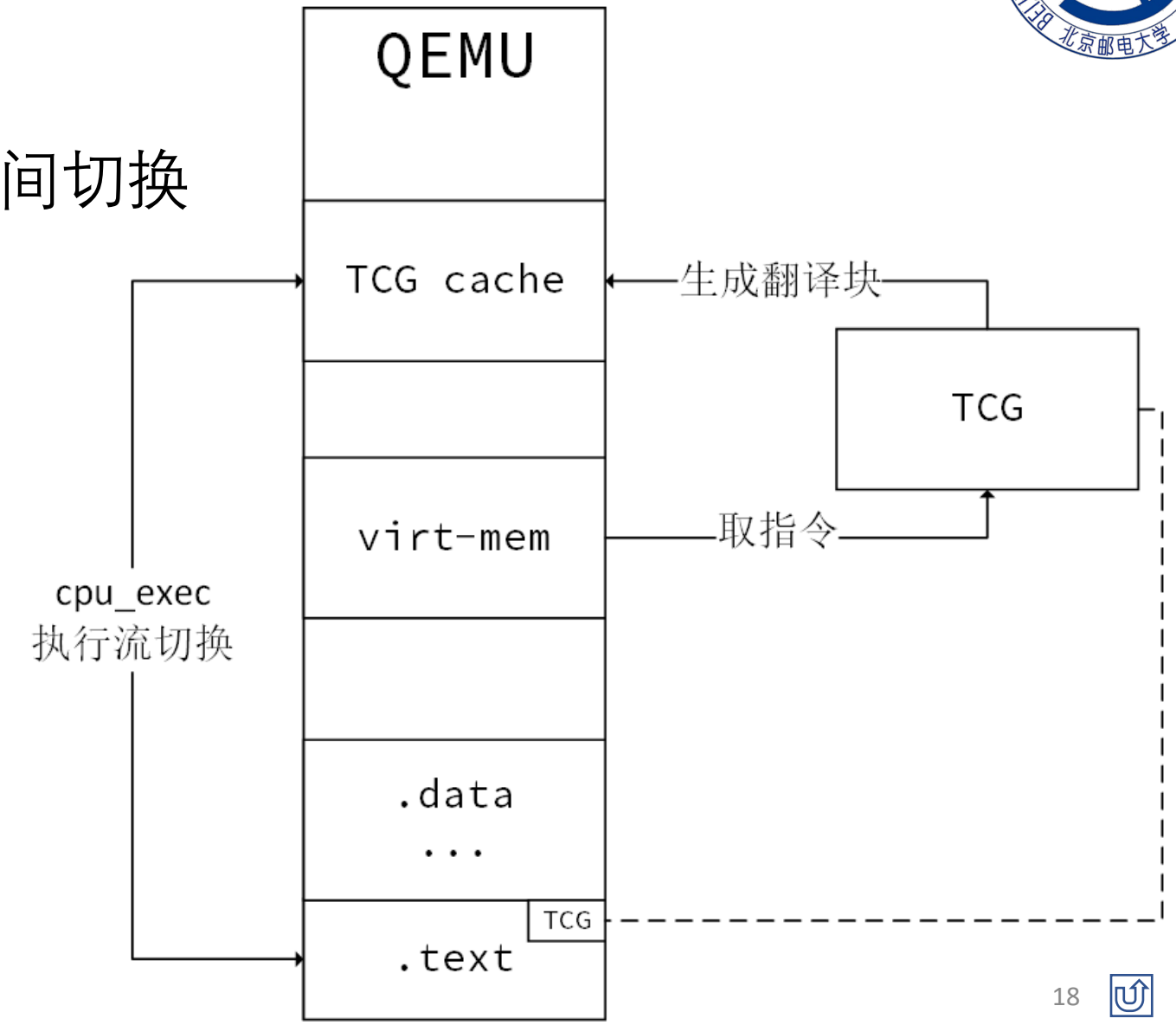
```
void *tb0_function(CPUArchState *env)
{
    uint64_t x1, x2, x3;

    x1 = env->gpr[1];
    x2 = env->gpr[2];
    x3 = x1 + x2;
    env->gpr[3] = x3;

    ...
}
```

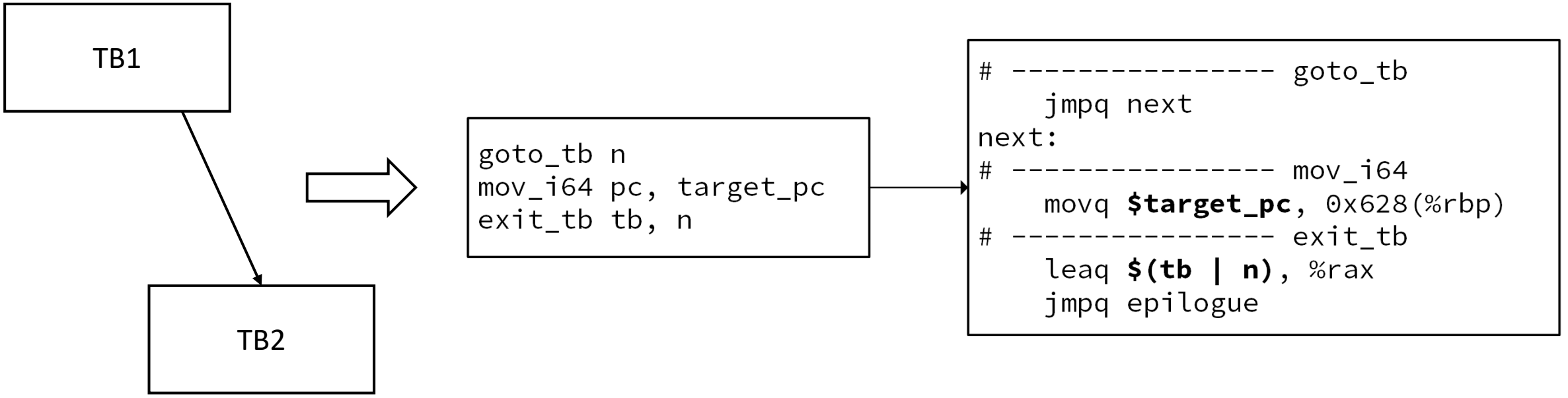
TCG Internal - 2

- 执行流在QEMU与生成代码间切换
  - QEMU - 翻译块
  - 翻译块 - 翻译块
  - 翻译块 - QEMU



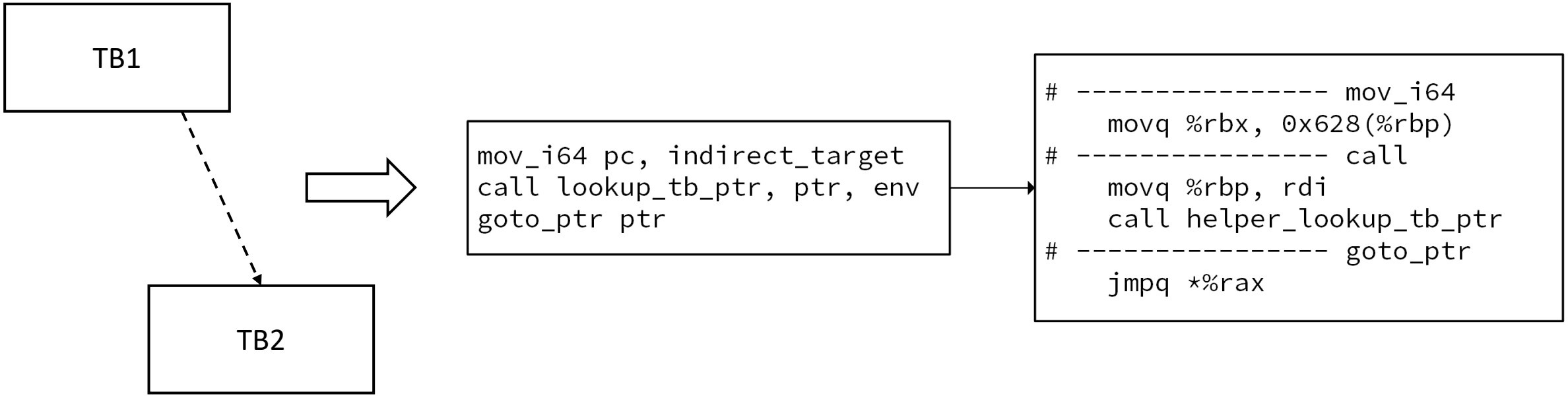
TCG Internal - 3

- 翻译块直接链接:
  - 页内直接跳转



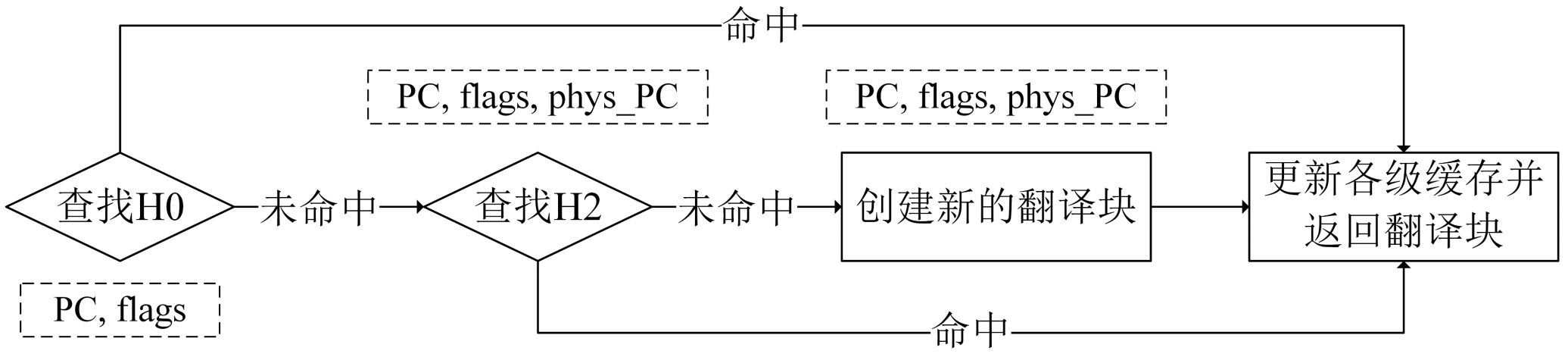
TCG Internal - 4

- 翻译块动态跳转：
  - 间接跳转、跨页直接跳转



TCG Internal - 5

- 运行时翻译块查找
  - 包含两级翻译块查找缓存：H0（vCPU私有）、H2（vCPU间共享）
  - H0通过虚拟地址（PC）哈希直接映射索引，易冲突且需要频繁刷新
  - H2通过物理地址（phys_PC）和其他标志索引，需要进行地址翻译



TCG Internal - 6

```
qemu_ld t_ret(%ebx), t_addr(%ebx), leul, idx(0)

# 计算TLB表项位置
movl    %ebx,%edi
shrl    $0x7,%edi
andl    -0x20(%rbp),%edi
addq    -0x18(%rbp),%rdi

# 匹配内存地址的tag位
leal    0x3(%rbx),%esi
andl    $0xfffff000,%esi
cmpl    (%rdi),%esi

# 未命中则跳转
movl    %ebx,%esi
jne     slow_path

# 命中则获取对应宿主机地址并进行访问
addq    0x10(%rdi),%rsi
movl    (%rsi),%ebx

# 后续指令
next_instr:
```

```
slow_path:
# 传递参数并跳转至load_helper函数进行地址翻译
movq    %rbp,%rdi
movl    $0x20,%edx
leaq    $next_instr,%rcx
callq   helper_le_ldul_mmu
movl    %eax,%ebx
# 返回正常指令流
jmpq    next_instr
```