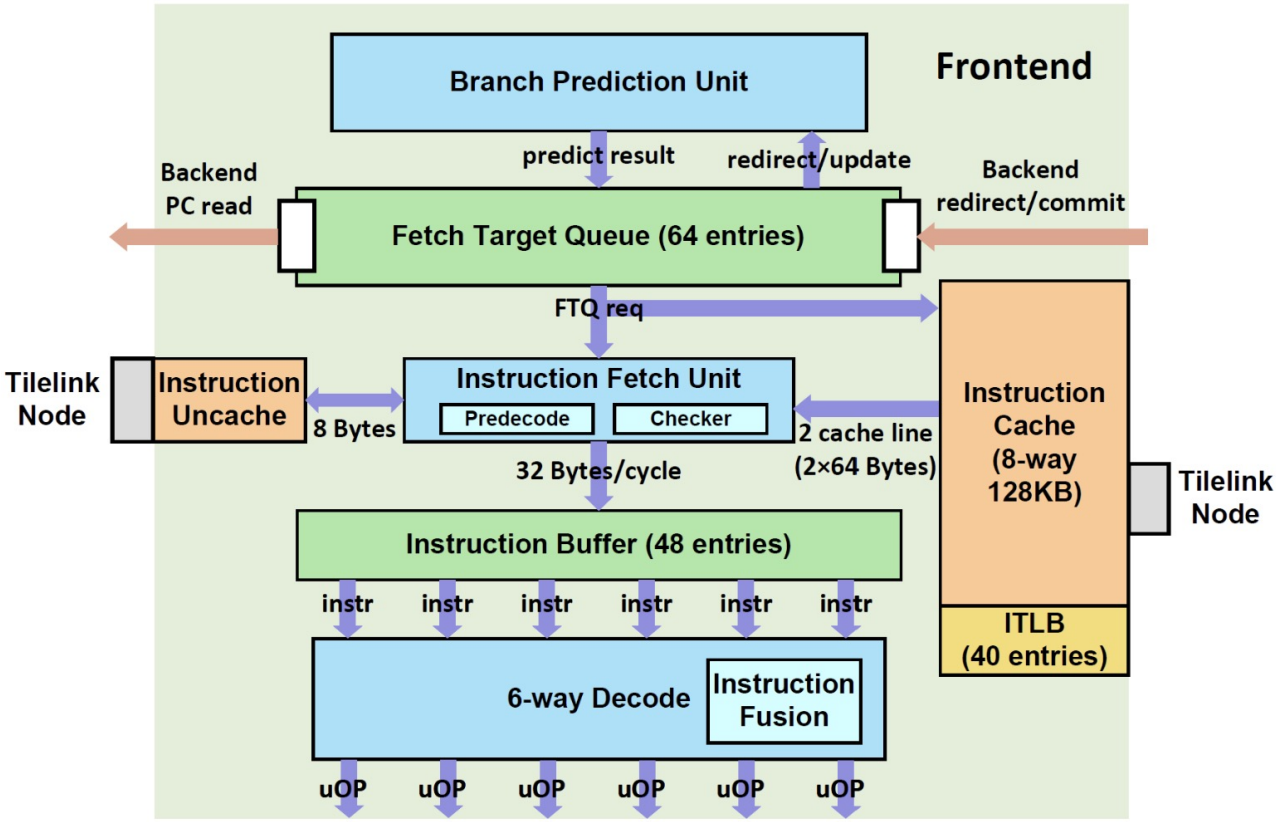
# 香山处理器前端取指架构演进

**勾凌睿**[1]、**金越**[1]、邹江瑞[2]、陈国凯[1]
[1]中科院计算所
[2]深圳大学

2022 年 8 月 25 日

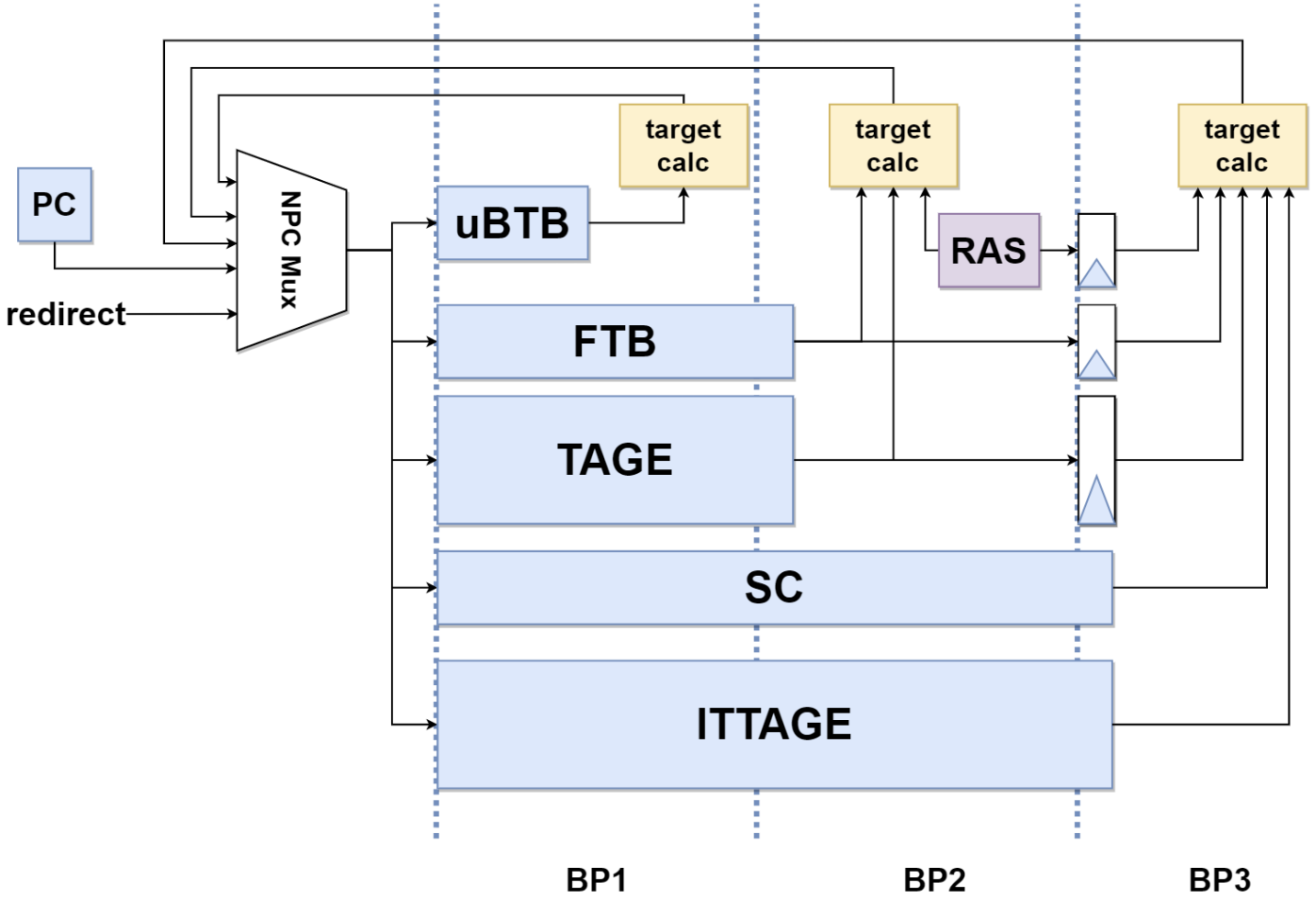
# 南湖前端取指架构总览

- 取指和分支预测解耦
  - 性能
    - 隐藏分支预测气泡
    - 指导指令预取
  - 时序
    - 避免分支预测和指令缓存的路径纠缠
- 分支预测和取指单元细节优化



# 南湖分支预测总体架构

- 三级覆盖预测
  - 256 项 uBTB
  - 2K 项 4 路组相联 FTB
  - 28KB TAGE，3KB SC
  - 12.5KB ITTAGE
  - 32 项 RAS



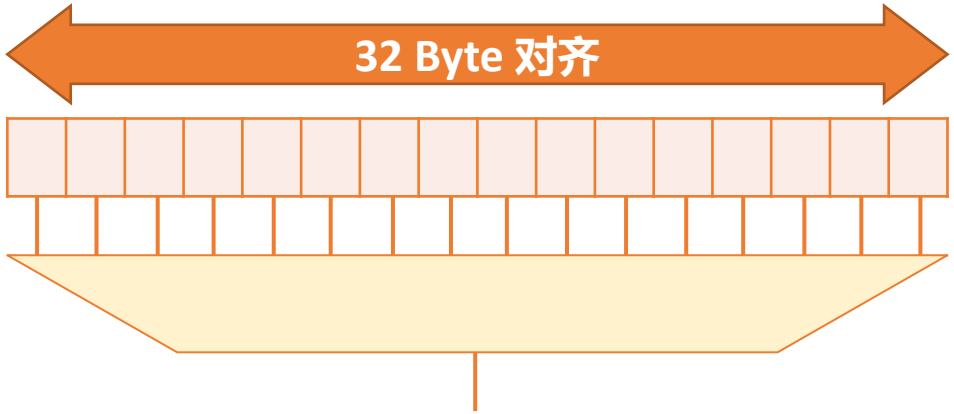
# 南湖分支预测提升汇总

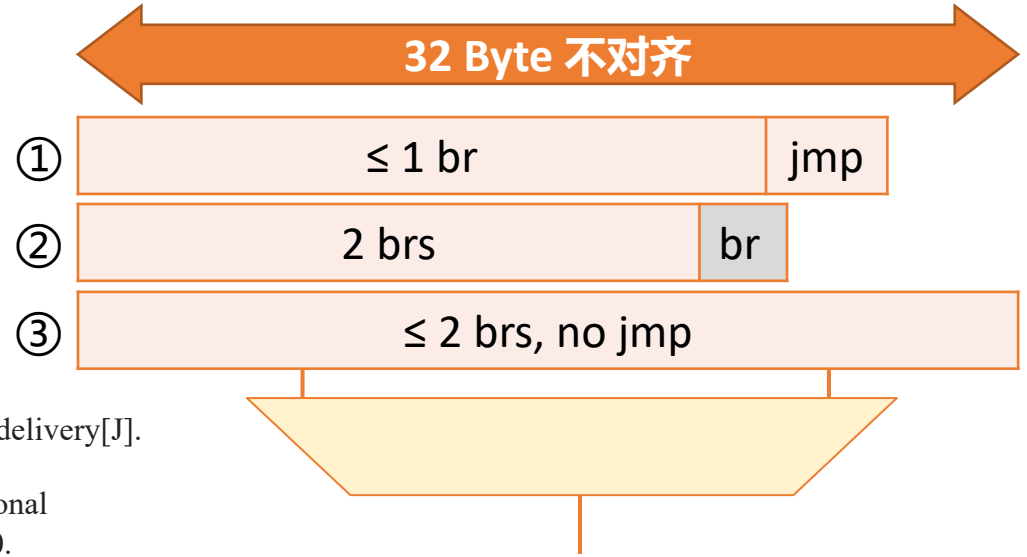
- 更大的规模
- 更低的延迟
- 更准确的预测

| | 雁栖湖 | 南湖 | 提升 |
|---|---|---|---|
| BTB（FTB） | 2K 条分支，2路 | 最多4K条分支，4路 | 2x |
| TAGE 延迟 | 3拍 | 2拍 | 1.5x |
| 分支历史 | 压缩，64bit | 准确，119bit | 1.85x |
| 间接跳转预测 | 无 | ITTAGE | New |
| RAS | 16项，3拍 | 32项，2拍 | 2x，1.5x |
| FTQ | 48项 | 64项 | 1.33x |

# 核心预测机制：BTB → FTB[1]

- 雁栖湖 BTB
  - 并行预测（16个结果）
  - 2K 项，每项对应一条分支
  - 2K 条分支
- 南湖 FTB
  - 最多2条分支[2]
  - 2K 项，每项对应一个预测块
  - 最多 4K 条分支

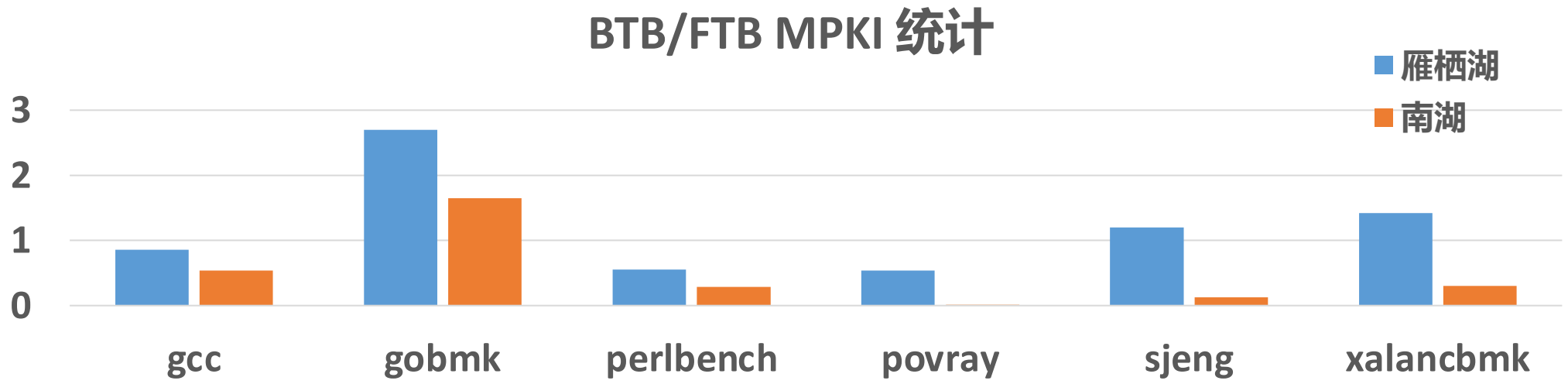



[1] Reinman G, Austin T, Calder B. A scalable front-end architecture for fast instruction delivery[J]. ACM SIGARCH Computer Architecture News, 1999, 27(2): 234-245
[2] Perais A, Sheikh R, Yen L, et al. Elastic instruction fetching[C]//2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2019: 478-490.

# 核心预测机制：BTB → FTB

- 雁栖湖 BTB
  - 并行预测（16个结果）
  - 2K 项，每项对应一条分支
  - 2K 条分支
- 南湖 FTB
  - 最多2条分支
  - 2K 项，每项对应一个预测块
  - 最多 4K 条分支



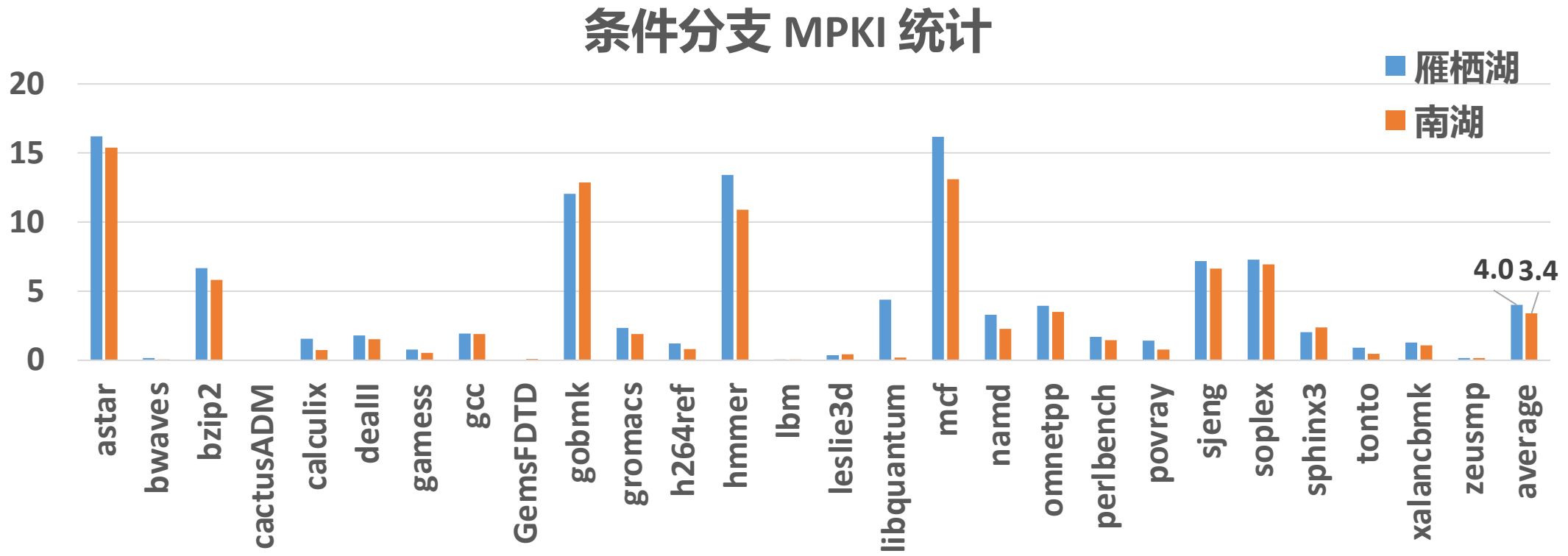
# 条件分支预测——TAGE

- 减少一拍延迟（IPC **+6%**）
- 总项数不变（16K），更长分支历史（**64→119**），遗传算法调参
- 细节算法优化
  - USE_ALT_ON_NA 处理冷启动问题[1]
  - bank interleaving，减少读写冲突
  - 训练算法调优

[1] Seznec A. A 256 kbits l-tage branch predictor[J]. Journal of Instruction-Level Parallelism (JILP) Special Issue: The Second Championship Branch Prediction Competition (CBP-2), 2007, 9: 1-6.

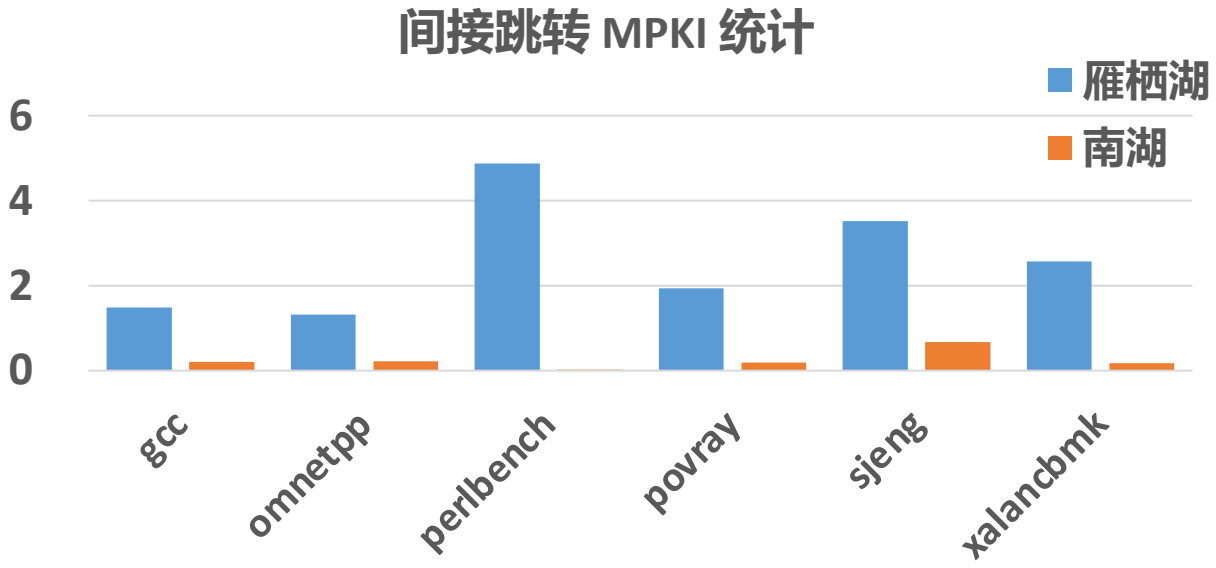
# 条件分支预测数据对比

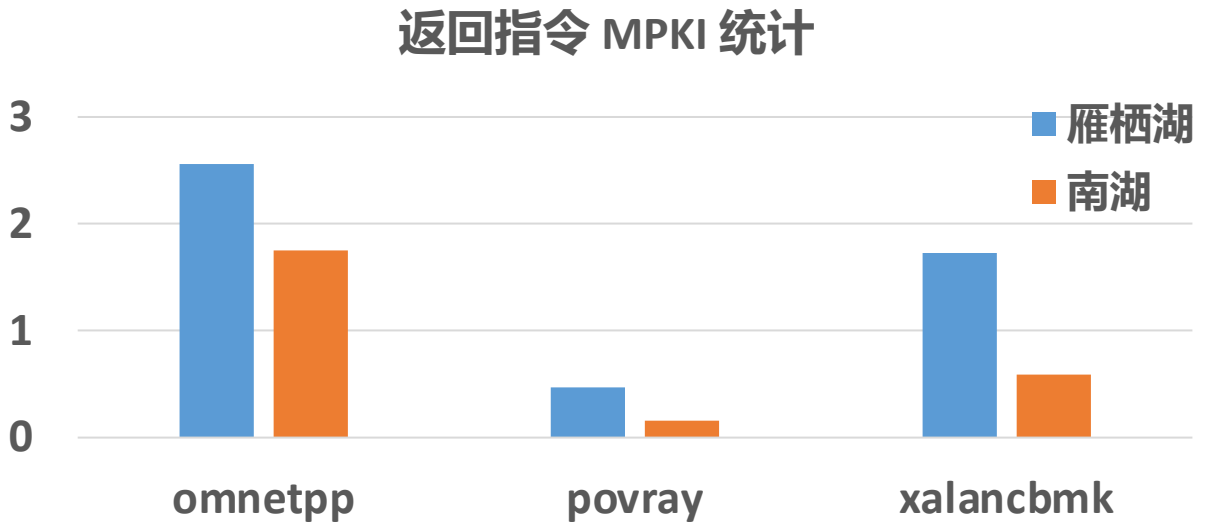
- 误预测**下降 15%**
  - SPEC CPU 2006 平均 MPKI 由 **4.0 → 3.4**



# 间接跳转预测

- **ITTAGE**[1]
  - 目前最佳算法
  - 2K 项，5 张表，最大历史 32
  - MPKI 大幅降低
- **RAS**
  - 项数翻倍
  - 减少一拍延迟



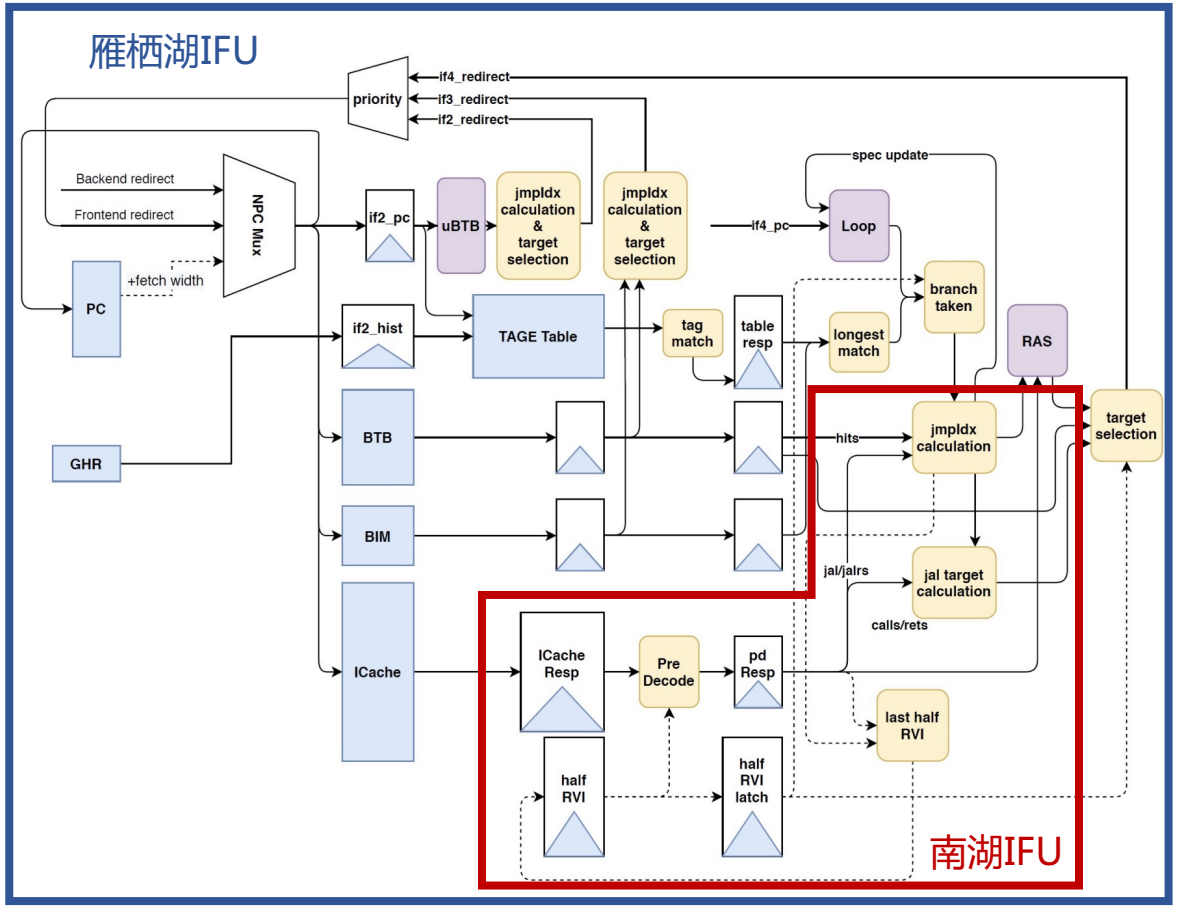




[1] Seznec A. A 64-Kbytes ITTAGE indirect branch predictor[C]//JWAC-2: Championship Branch Prediction. 2011.

# 取指令单元（IFU）

- **南湖架构相较于雁栖湖的变动**
  - + 添加基于预测块的取指令逻辑
  - - 将分支预测器之间的前向覆盖逻辑搬到 BPU 内部流水线

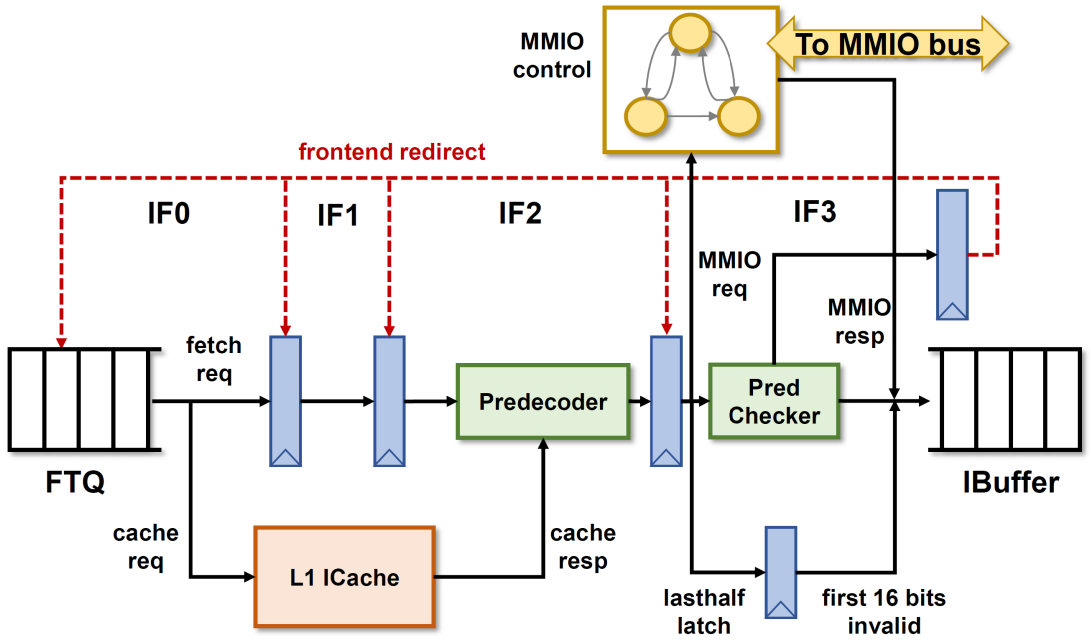
- **主要功能概述**
  - 四级流水线（IF0 – IF3）
  - **基于预测块的取指令方式**
  - **基于预译码的分支预测错误检查**
  - MMIO 取指令



# 取指令单元（IFU）

- **南湖架构相较于雁栖湖的变动**
  - + 添加基于预测块的取指令逻辑
  - - 将分支预测器之间的前向覆盖逻辑搬到 BPU 内部流水线

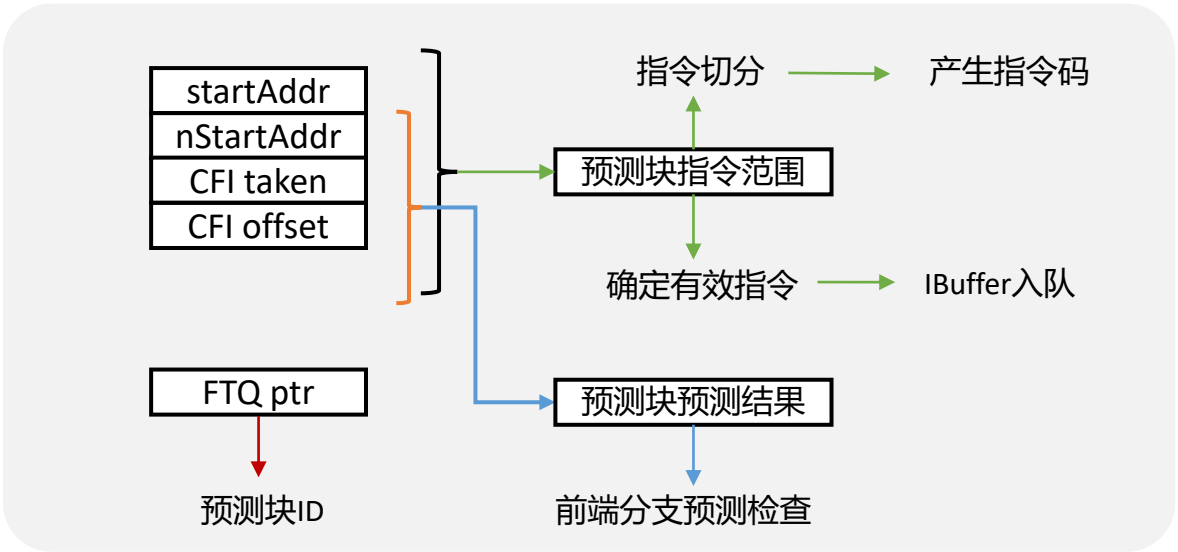
- **主要功能概述**
  - 四级流水线（IF0 – IF3）
  - **基于预测块的取指令方式**
  - **基于预译码的分支预测错误检查**
  - MMIO 取指令

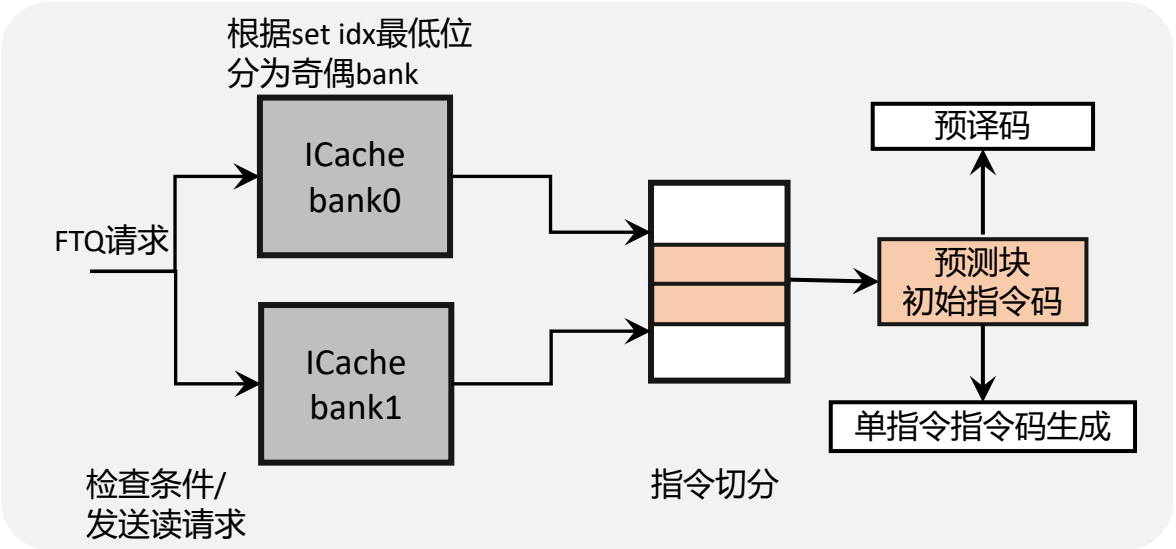


# 基于预测块的取指令

- **FTQ发出预测块信息：**
  - 预测块起始地址（startAddr）
  - 下一个预测块的起始地址（nStartAddr）
  - 该预测块在 FTQ 里的队列指针 (FTQ ptr)
  - 该预测块有无跳转的 CFI 指令 (CFI taken)
  - 该跳转的 CFI 指令在预测块里的位置（以2字节单位的偏移）(CFI offset)

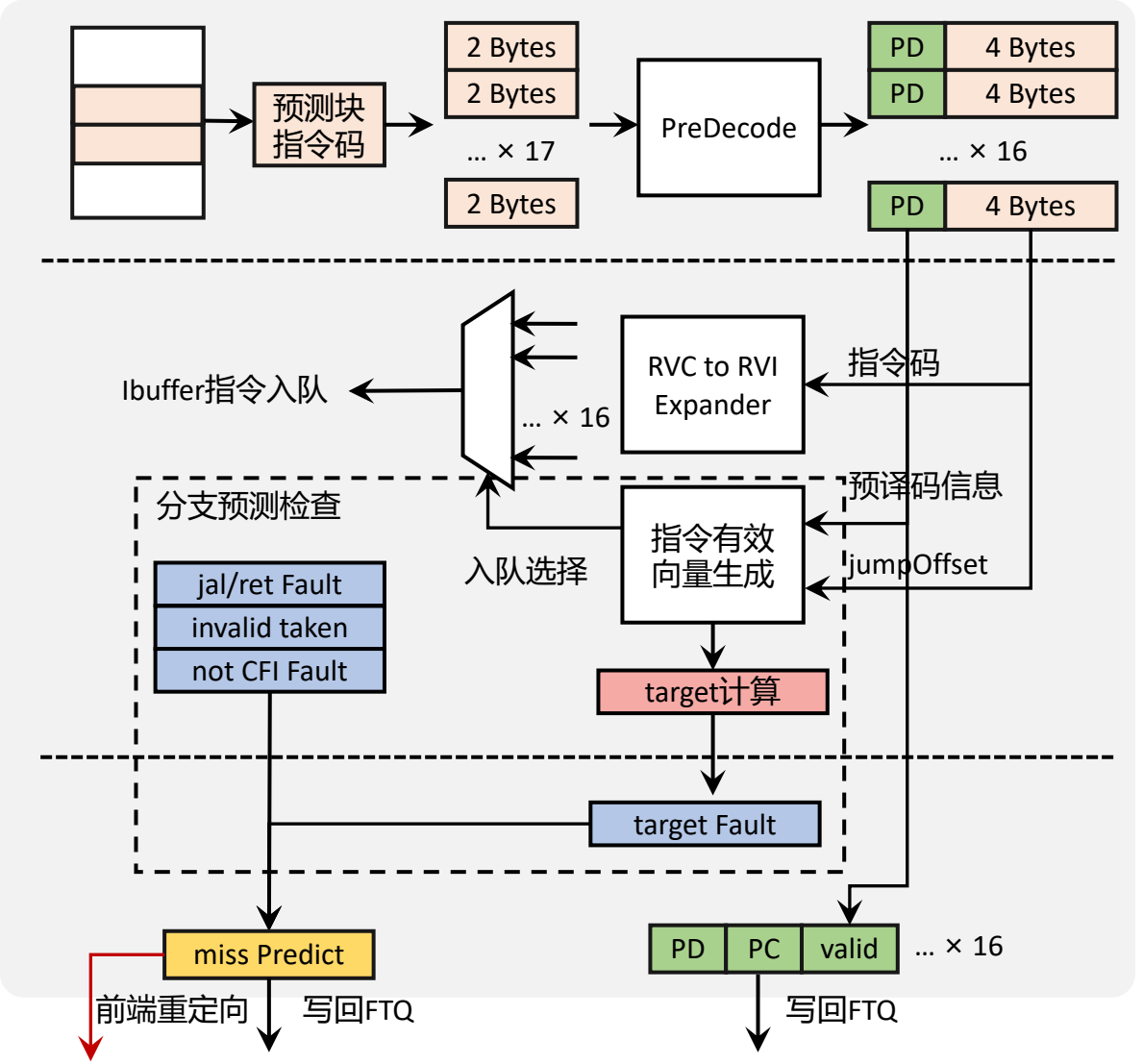
- **跨行预测块的判断**
  - Bank interleaving 实现每周期最多读两个缓存行
  - 起始地址位于一个缓存行（64B）的后半部分
  - 从指令缓存里取地址相邻的两个缓存行
  - 优势：增加指令带宽
  - 劣势：指令缓存的功耗会比较高



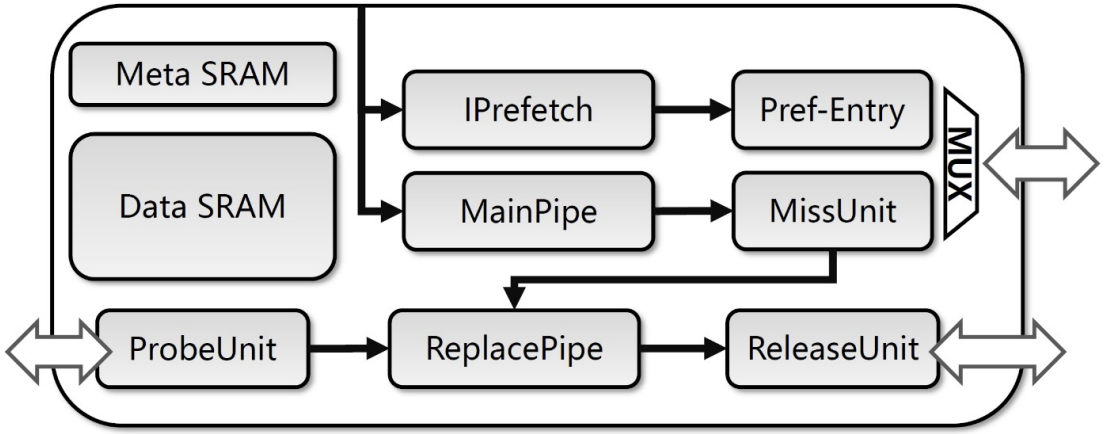
# 基于预译码的分支预测错误检查

- **IF2接受来自指令缓存的最多两个缓存行**
  - 指令切分，产生17×2字节的初始指令码
  - 预译码得到16条指令码和预译码信息
  - 提取指令码中的 jumpOffset
- **IF3根据预译码信息进行分支预测检查**
  - 检查3种类型的分支预测错误
  - 计算跳转目标地址
  - 选择入队 Ibuffer 的指令
- **WB（IF4）将信息写回给 FTQ**
  - 检查目标地址错误
  - 指令信息写回 FTQ
  - 发现错误则发起前端重定向，冲刷流水线，重新取指



# 指令缓存

- **主要功能概述**
  - 128KB 8-w VIPT blocking cache
  - 奇偶 bank 实现双行读取
  - **支持 Tilelink 一致性协议**
  - PLRU 替换算法
  - 简单的 FDP 指令预取

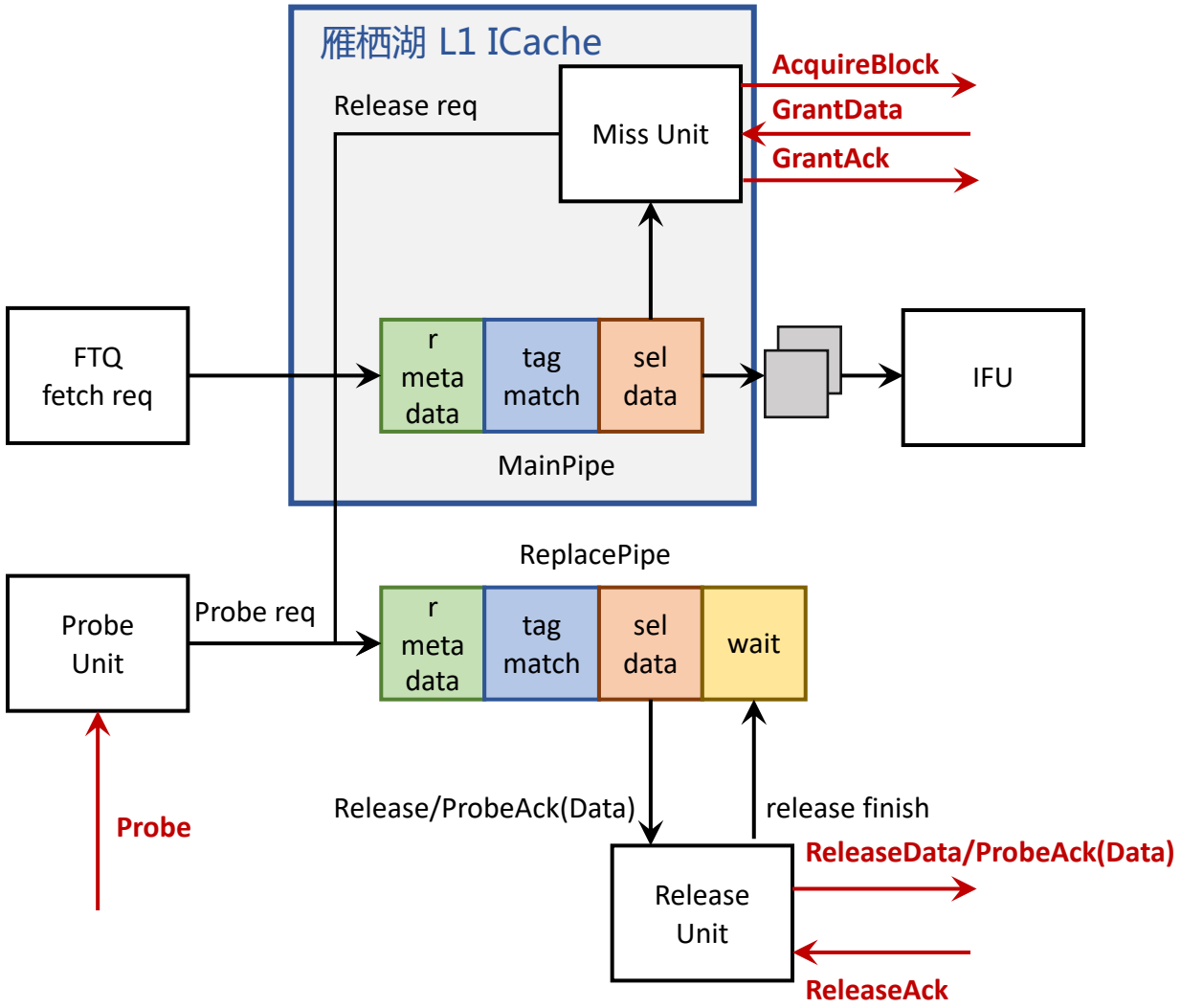



| | 雁栖湖 | 南湖 | 对比 |
|---|---|---|---|
| 缓存层级 | 两级 | 一级 | Reduce |
| L1I大小 | 16KB, 4-w | 128KB, 8-w | 8×, 2× |
| L1plus缓存 | 128KB, 8-w | 无 | Reduce |
| 读缓存带宽 | 64 B | 128 B | 2× |
| 支持Tilelink一致性协议 | 否 | 是 | New |
| 指令预取 | L1 plus Stream预取 | L2 FDP | New |

# 指令缓存一致性修改

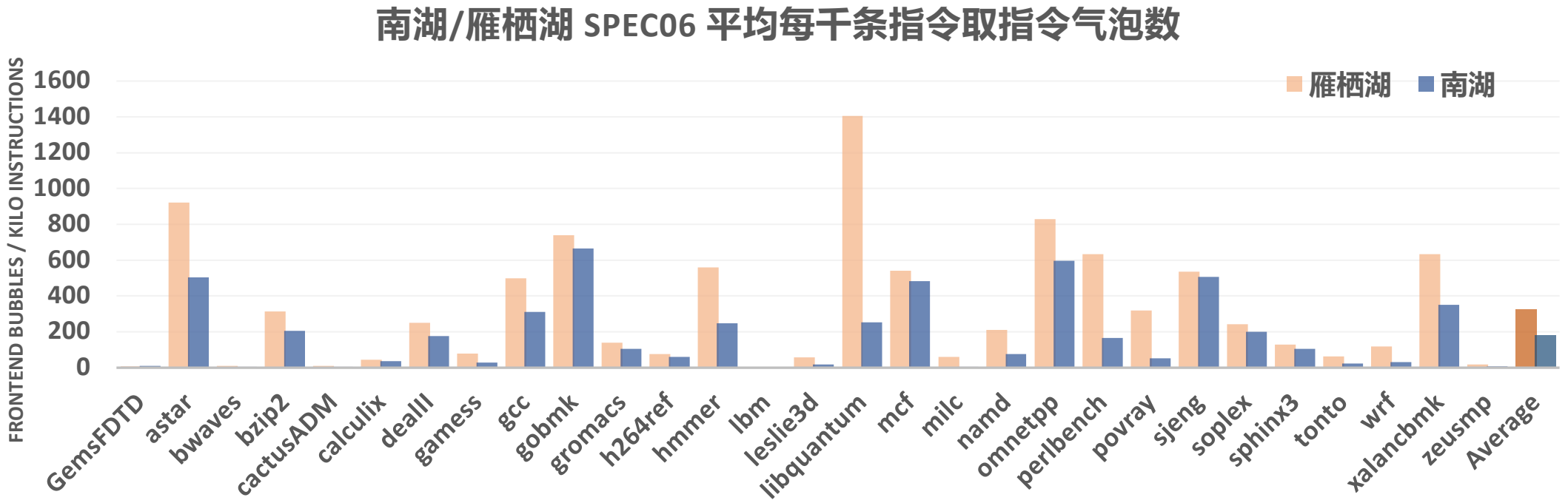
- **设计考虑：**
  - 软件维护方式需要无效掉全部指令缓存的内容
  - 南湖 L1I 容量增加导致 RISC-V 软件维护一致性的性能代价比较大
  - 基于 Cache 一致性协议的硬件维护

- **主要修改和实现实现方式：**
  - 添加 ProbeUnit 处理 Probe 请求
  - 添加 ReleaseUnit 处理 Release 和 ProbeAck 请求
  - 添加 Replace Pipe 用于处理替换请求和 ProbeUnit 查询请求
  - 缓存行 Meta 信息扩展：有效/无效 → 四种状态



# 南湖架构前端取指性能提升

- **取指令气泡**：定义为 Ibuffer 到 Decode 六条指令，Decode 可以接收但是 Ibuffer 不能给出指令的总次数。气泡数量越少说明处理器的供指能力相对来说越强。
- 南湖相较于雁栖湖架构在 SPEC 06 在绝大多数 workload 上都有明显的平均取指令气泡数（每千条指令）的减少
- 最高减少**90%**（433.milc），整体平均减少**44%**
- **南湖前端的性能相较雁栖湖有明显的提升**



# 谢谢！
# 敬请批评指正！