



香山昆明湖架构前端架构 的设计演进

勾凌睿¹ 薛臻^{1,3} **陈国凯**¹ 高泽宇¹ 宋政伟⁴ 傅腾蛟² 郭鸿宇⁵ 满洋⁶

¹中国科学院计算技术研究所

²中国科学院上海光学机械研究所

³鹏城实验室 ⁴西安交通大学

⁵北京航空航天大学 ⁶哈尔滨工业大学 (深圳)

2023 年 8 月 24 日@第三届 RISC-V 中国峰会



香山：开源高性能 RISC-V 处理器

• 第一版：雁栖湖 架构

- 2020/6：代码仓库建立，开始 RTL 实现工作
- 2021/7：完成 28nm 流片，频率 1.3GHz
- 性能：实测 SPEC CPU2006 超过 7 分/GHz

• 第二版：南湖 架构

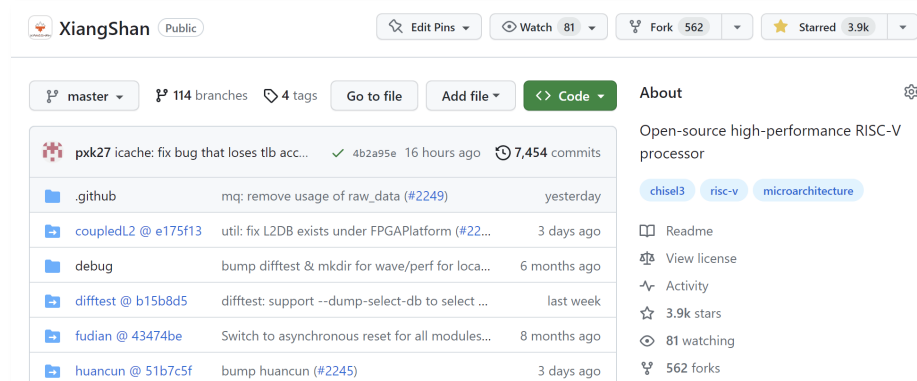
- 2021/5：开始 RTL 实现工作，持续进行设计讨论
- 2023年初：RTL 冻结，物理设计进入收尾阶段
- 即将流片，预估 SPEC CPU2006 约 10 分/GHz，2GHz@14nm

• 第三版：昆明湖 架构

- 性能对标数据中心处理器核 ARM Neoverse N2，目标性能 SPEC CPU2006 ~15分/GHz，~3GHz
- 开发进入中期阶段，与合作企业联合进行研发

• 开源情况

- 开源协议：MulanPSLv2 协议（兼容Apache v2.0）
- 代码托管：GitHub (<https://github.com/OpenXiangShan/XiangShan>)；镜像：Gitee/GitLink/iHub



主仓库在全球最大开源项目托管平台 GitHub
已获得超过 3800 个星标，形成超过 550 分支

第三代香山（昆明湖）规划

• 功能改进

- 支持 RISC-V 向量 (Vector, V) 扩展指令集
- 支持 RISC-V 虚拟化 (Hypervisor, H) 扩展指令集

• 性能探索

- 构建与香山性能校准的体系结构模拟器
- 建立**模拟器设计空间探索** → **RTL 实现与调参**的性能迭代 workflow

• 功能验证

- 覆盖 ST/IT/UT + FPGA 的**多层次验证**
- 工业级的规范验证流程

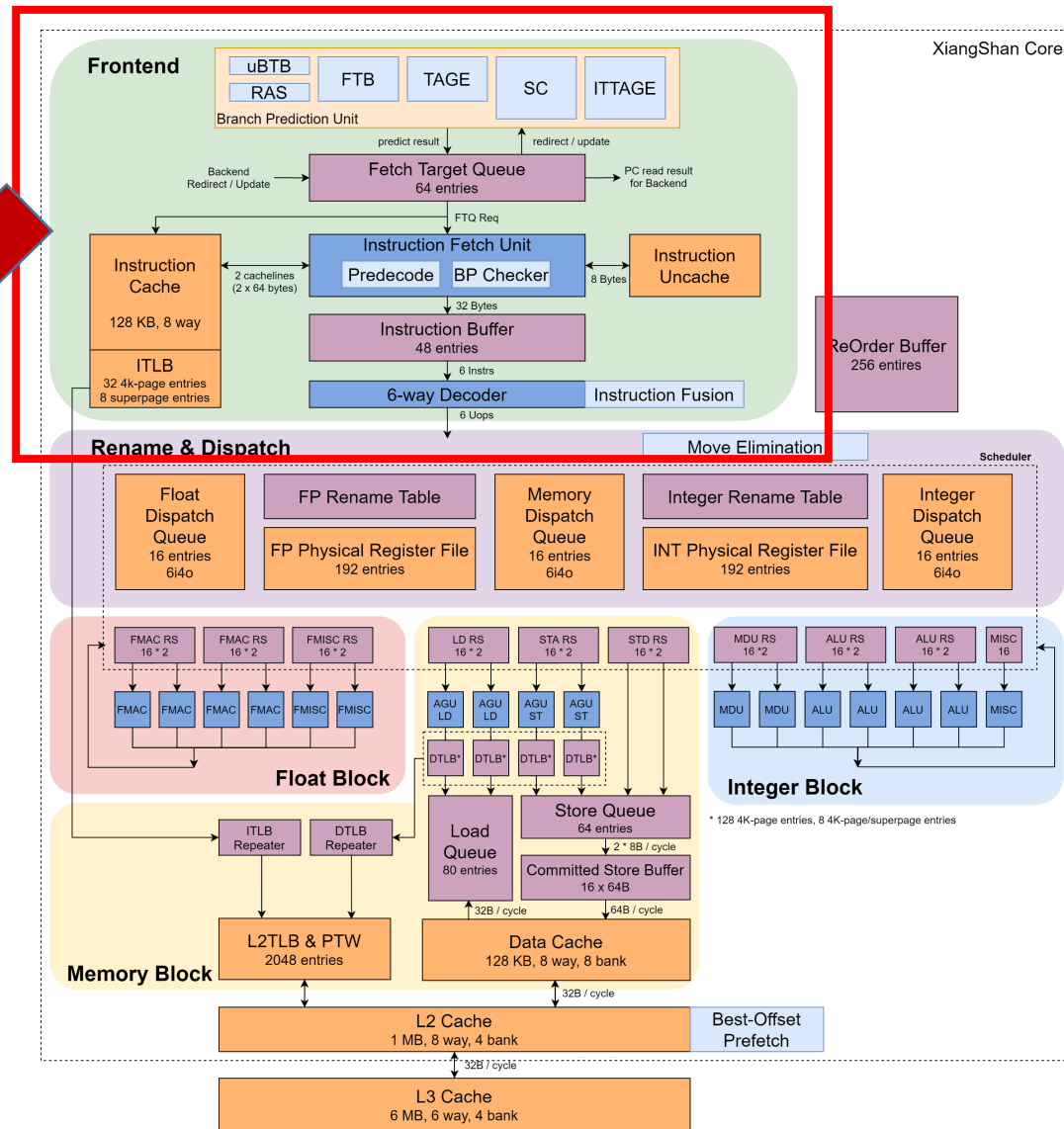
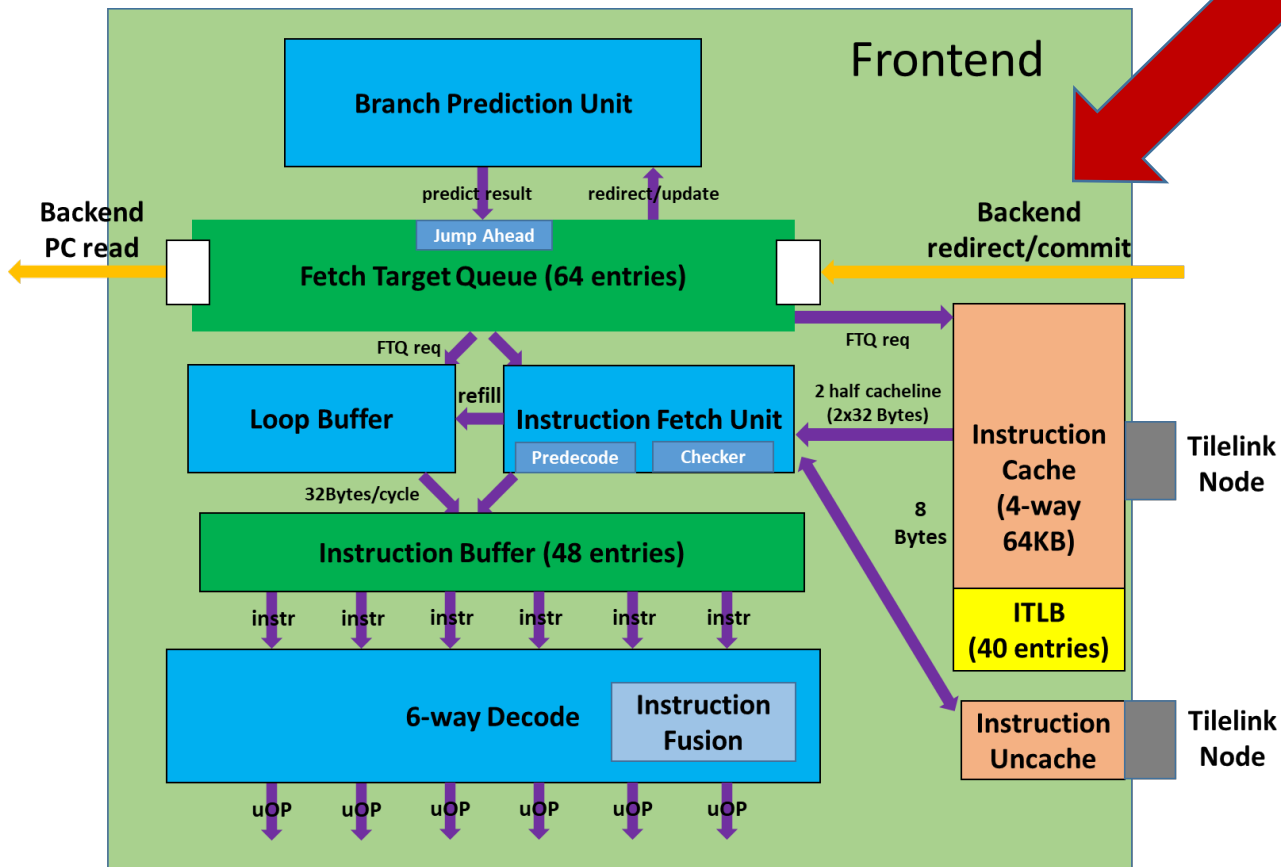
• 物理设计

- 专业的高性能处理器物理后端设计团队
- RTL 修改与物理后端时序评估同步迭代



香山前端架构总览

- 前端主要负责分支预测与取指

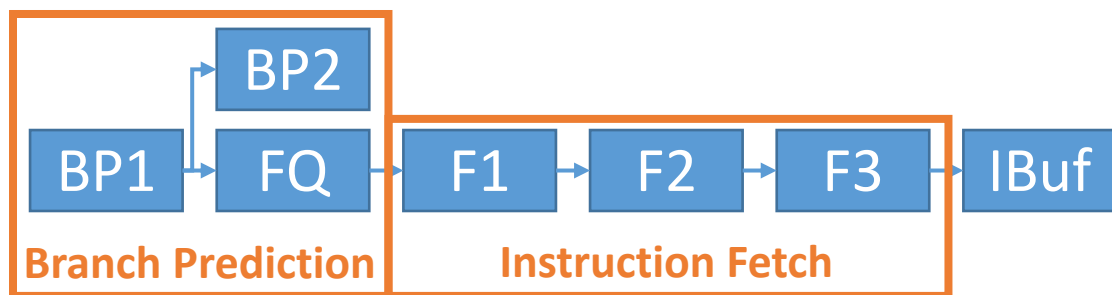


香山整体微架构

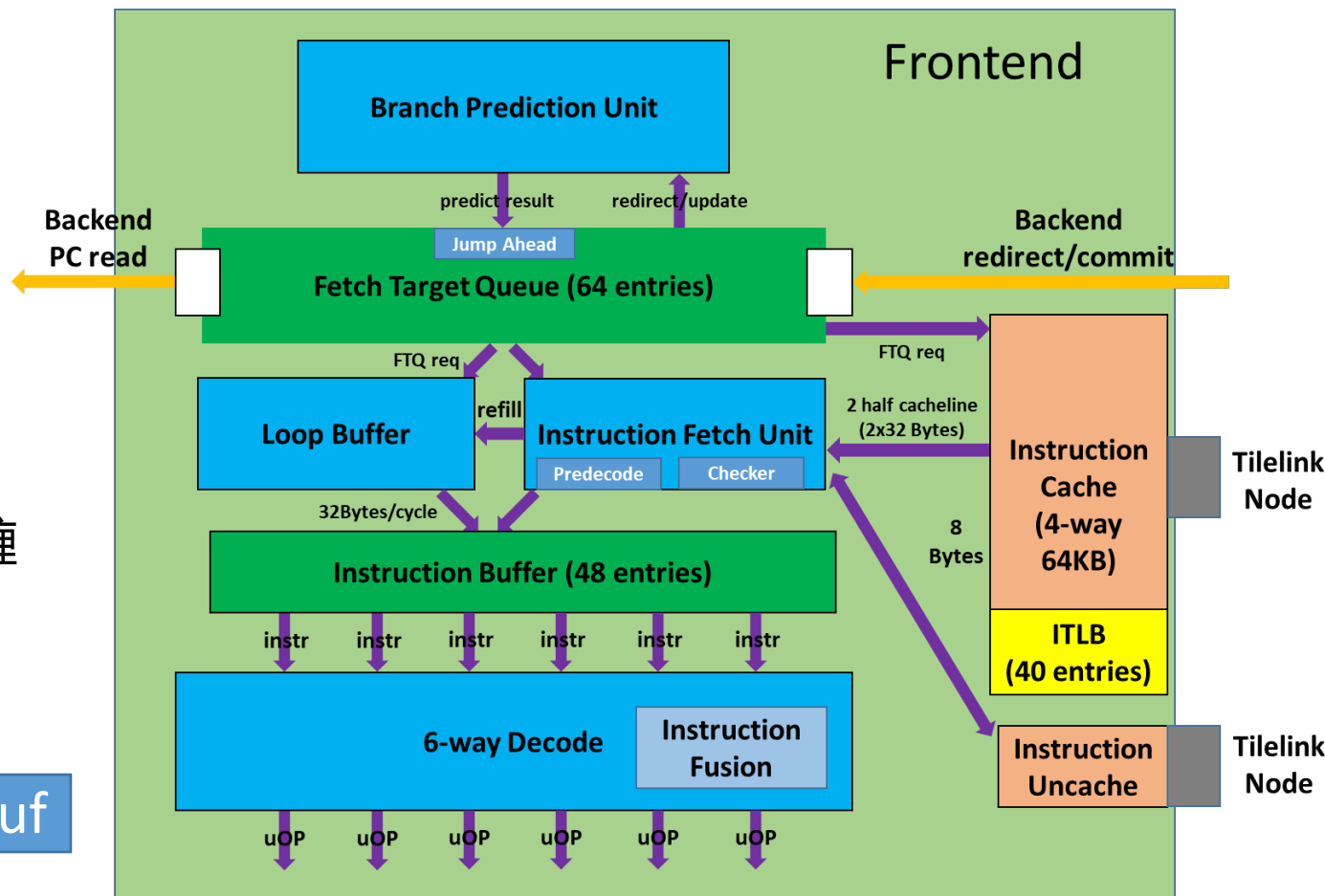


昆明湖前端架构总览

- 继承自南湖架构的解耦架构
 - 取指和分支预测解耦
 - 隐藏分支预测气泡
 - 指导指令预取
 - 避免分支预测和指令缓存路径纠缠



解耦架构设计

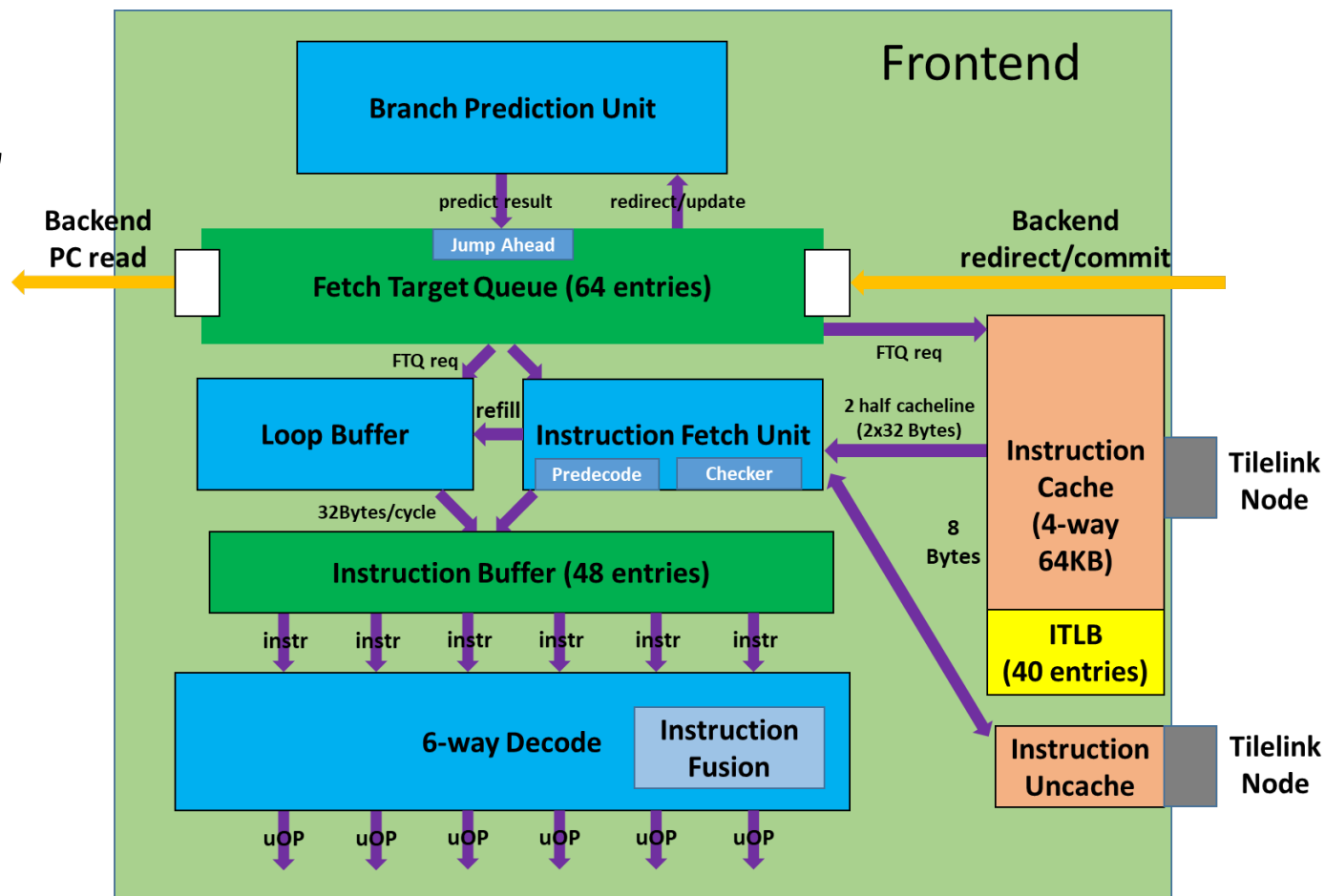




昆明湖前端架构总览

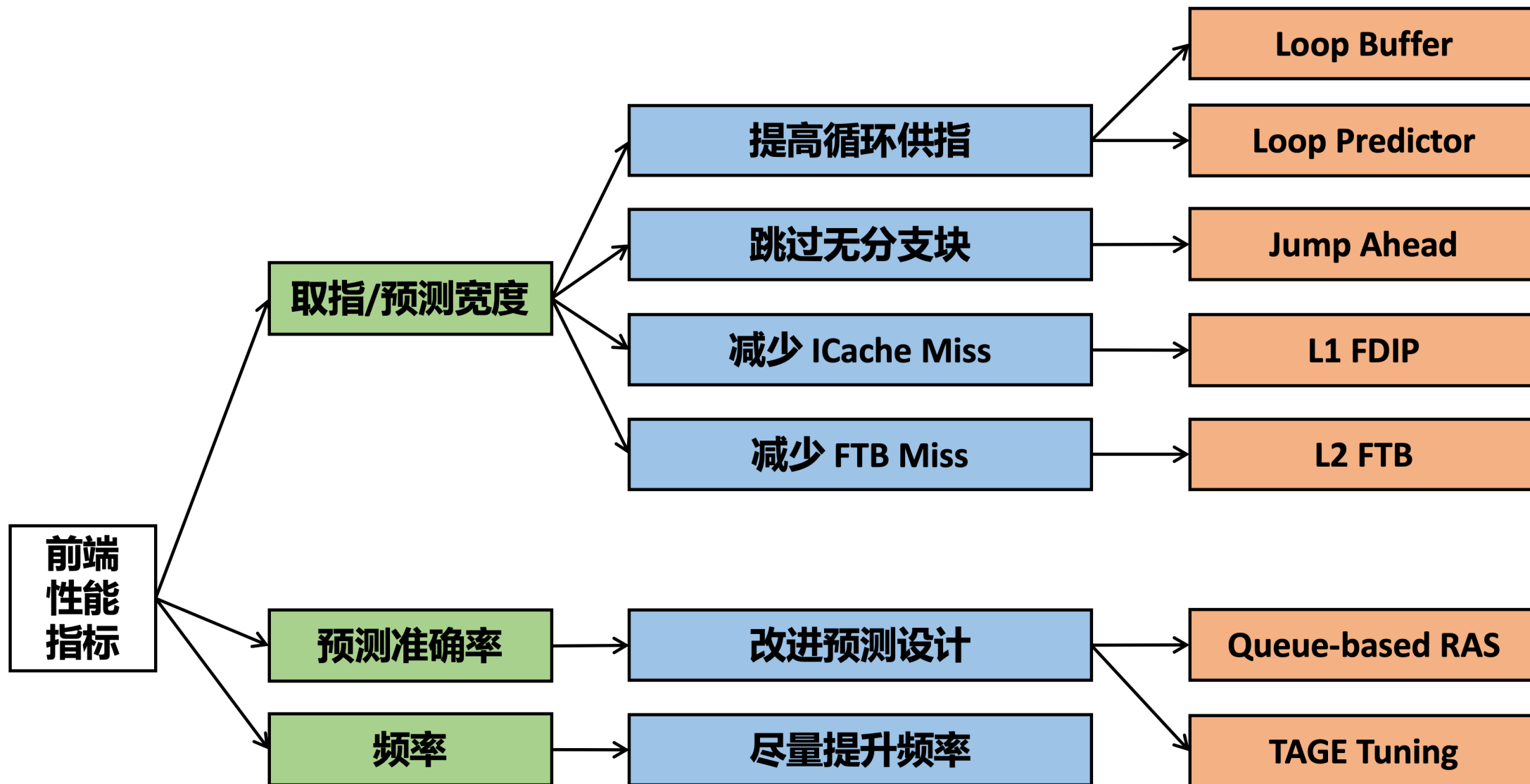
- 取指和分支预测单元迭代优化

- 增大预测宽度
- 提高预测准确率
- 增大取指宽度
- 降低取指延迟
- 提高频率





昆明湖前端迭代方向



方向1：提高循环供指

• 问题

- 长循环体**退出时机**预测不准确
- 小循环体**指令供应**不足
- 循环体**重复访问** ICache 和分支预测器增大功耗

• 解决

- Loop Predictor: 预测**循环退出**
- Loop Buffer: **供应循环体指令**, 一次取指可包含两条跳转的分支指令(**two taken**)

小循环示例

```
for (int i = 0; i < N; i++) {  
    a[i+200] = a[i];  
}
```

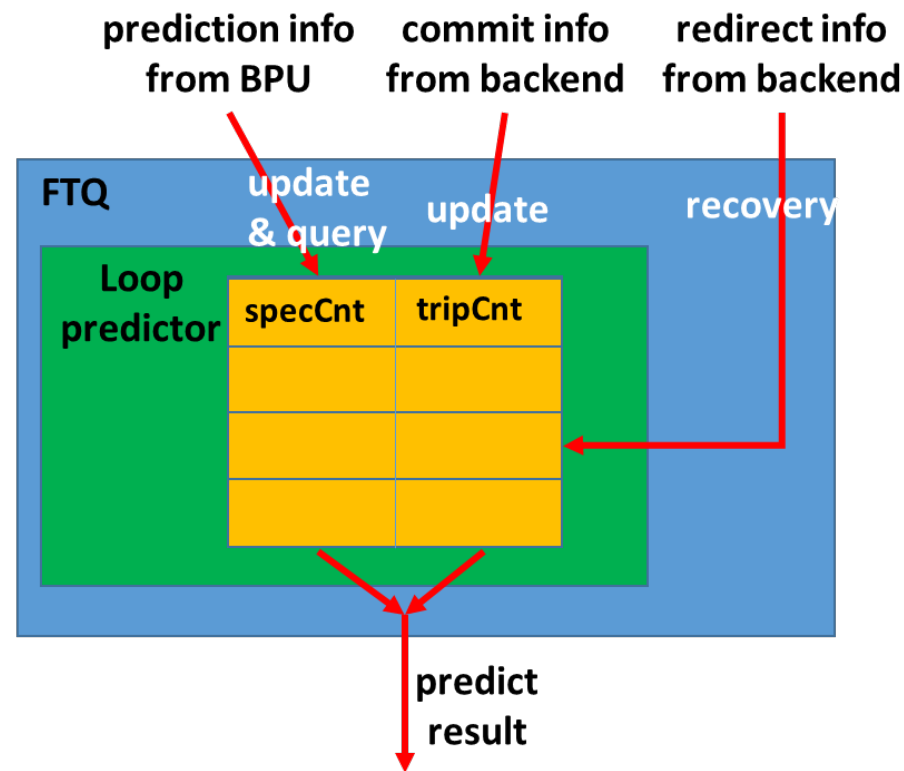
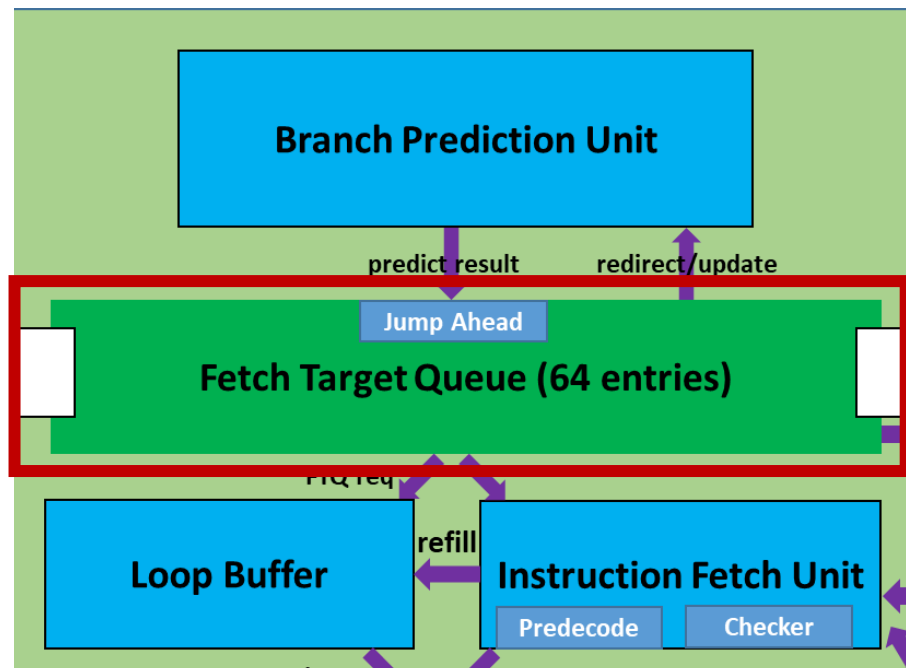


loop:

```
lb t2, 0(t0)  
sb t2, 200(t0)  
addi t0, t0, 1  
bne t0, t1, loop
```

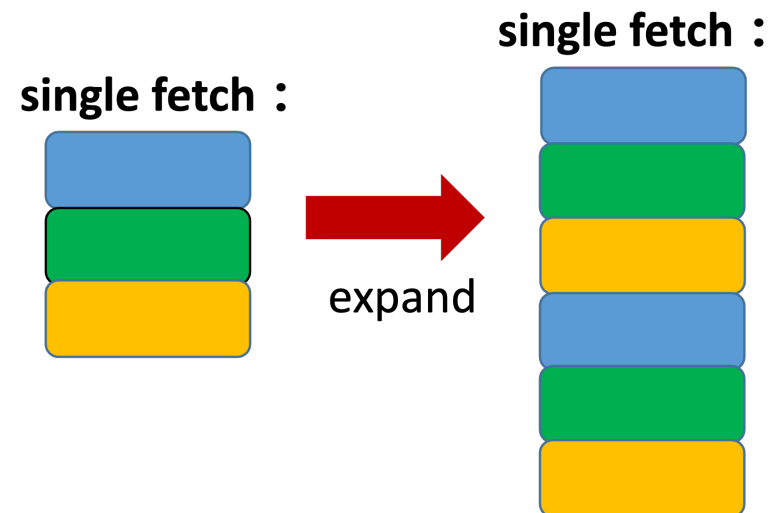
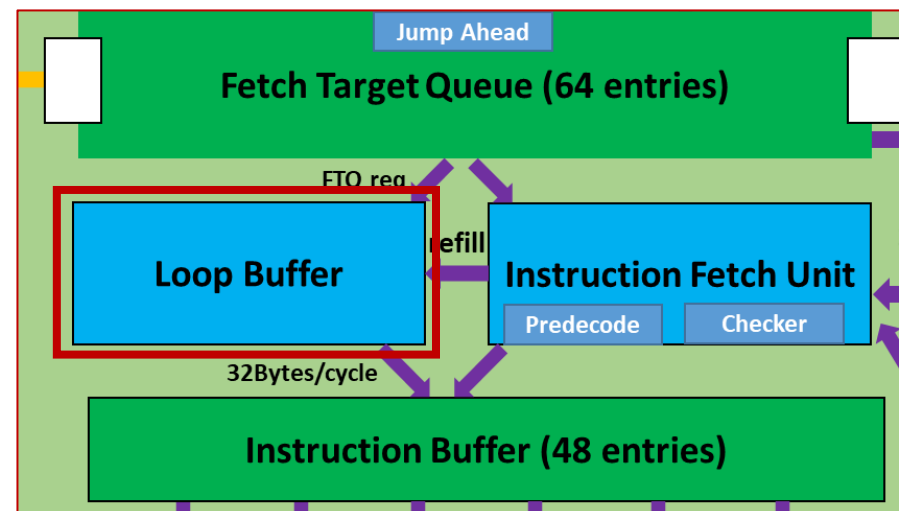

🏔️ 方向1：提高循环供指 – Loop Predictor

- 训练：使用指令提交信息检测循环体并训练**循环体总计数**
- 预测：使用预测结果维护**当前循环计数**



❄️ 方向1：提高循环供指 – Loop Buffer

- 指令填充
 - 提交信息判定循环体
 - 在 lbuffer 入口填充循环体
- 指令供应
 - 运行时关闭 BPU 及 ICache
 - 根据 Loop predictor 预测结果供应指令
 - 硬件循环展开可提供**双倍指令**供应

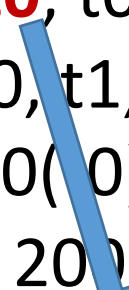


方向1：提高循环供指

- SPEC CPU 2006 性能收益
 - 仅开启 Loop Predictor
平均性能提升 **0.39%**，最大 **2.56%**
 - 额外开启 Loop Buffer
平均性能提升 **0.37%**，最大 **2.54%**
 - 由于循环变量存在**顺序依赖**，Loop Buffer
性能提升有限

寄存器的顺序依赖

```
lb t2, 0(t0)
sb t2, 200(t0)
addi t0, t0, 1
bne t0, t1, loop
lb t2, 0(t0)
sb t2, 200(t0)
addi t0, t0, 1
bne t0, t1, loop
```



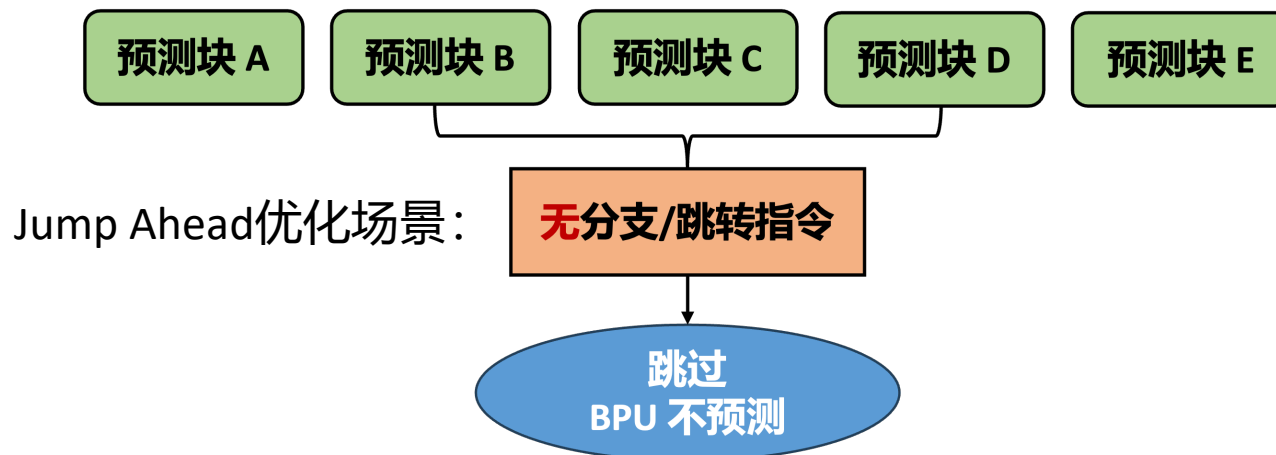
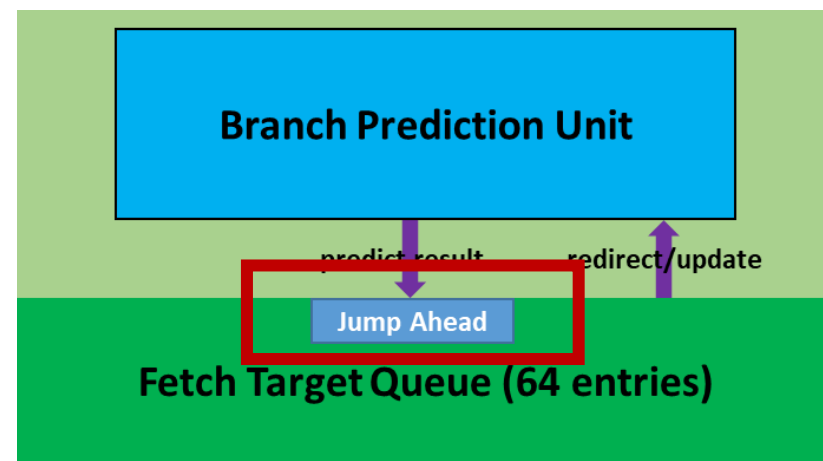
🔥 方向2：跳过无分支块 – Jump Ahead

• 问题

- 非分支跳转指令的预测增大功耗
- 前端吞吐量要求提高

• 解决

- Jump Ahead: 跳过无分支跳转指令预测块





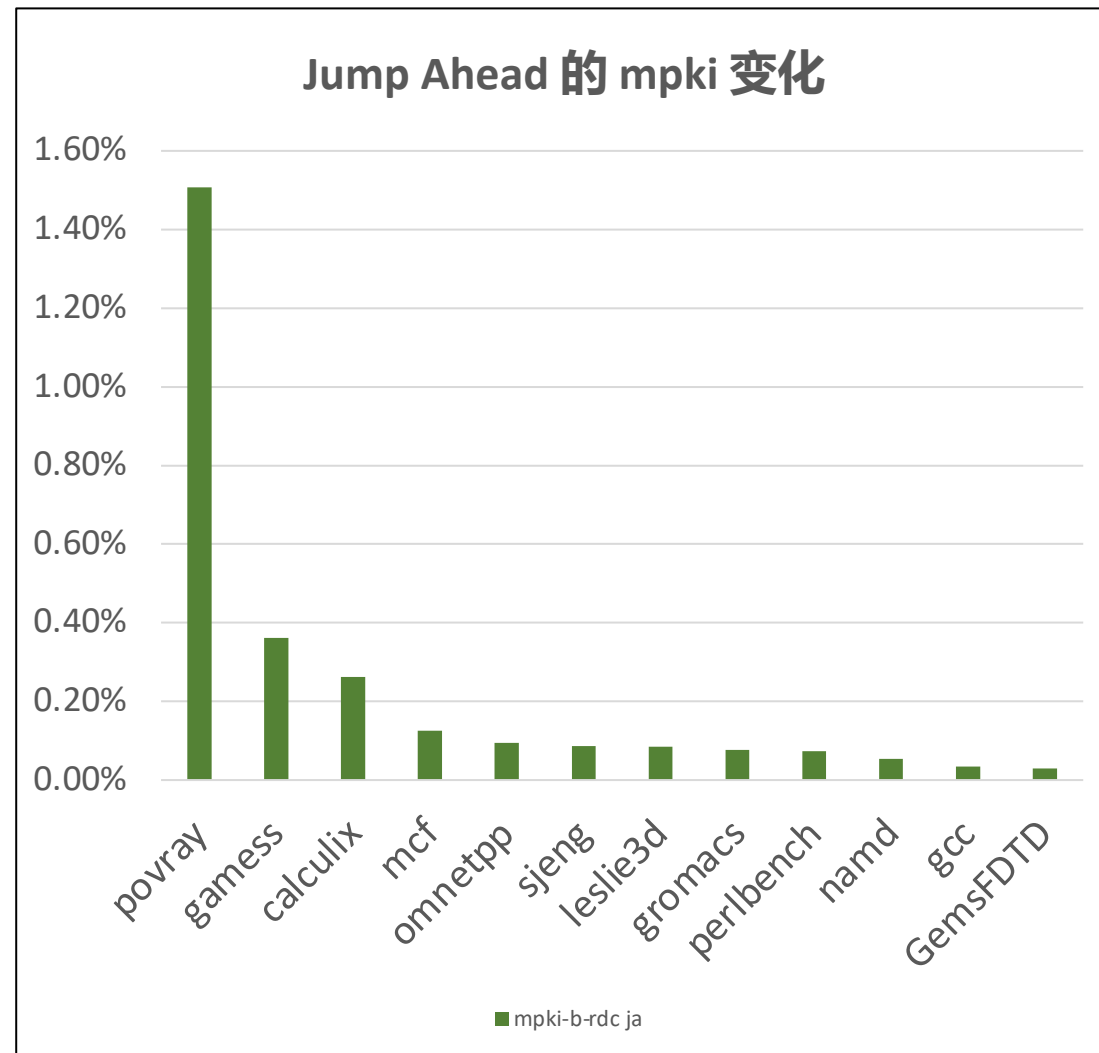
方向2：跳过无分支块 – Jump Ahead

- 训练

- 记录无分支跳转指令的连续**预测块首尾**
- 根据后端 commit 信息训练

- 预测

- 使用 Jump Ahead 结果**跳过**连续无分支跳转预测块





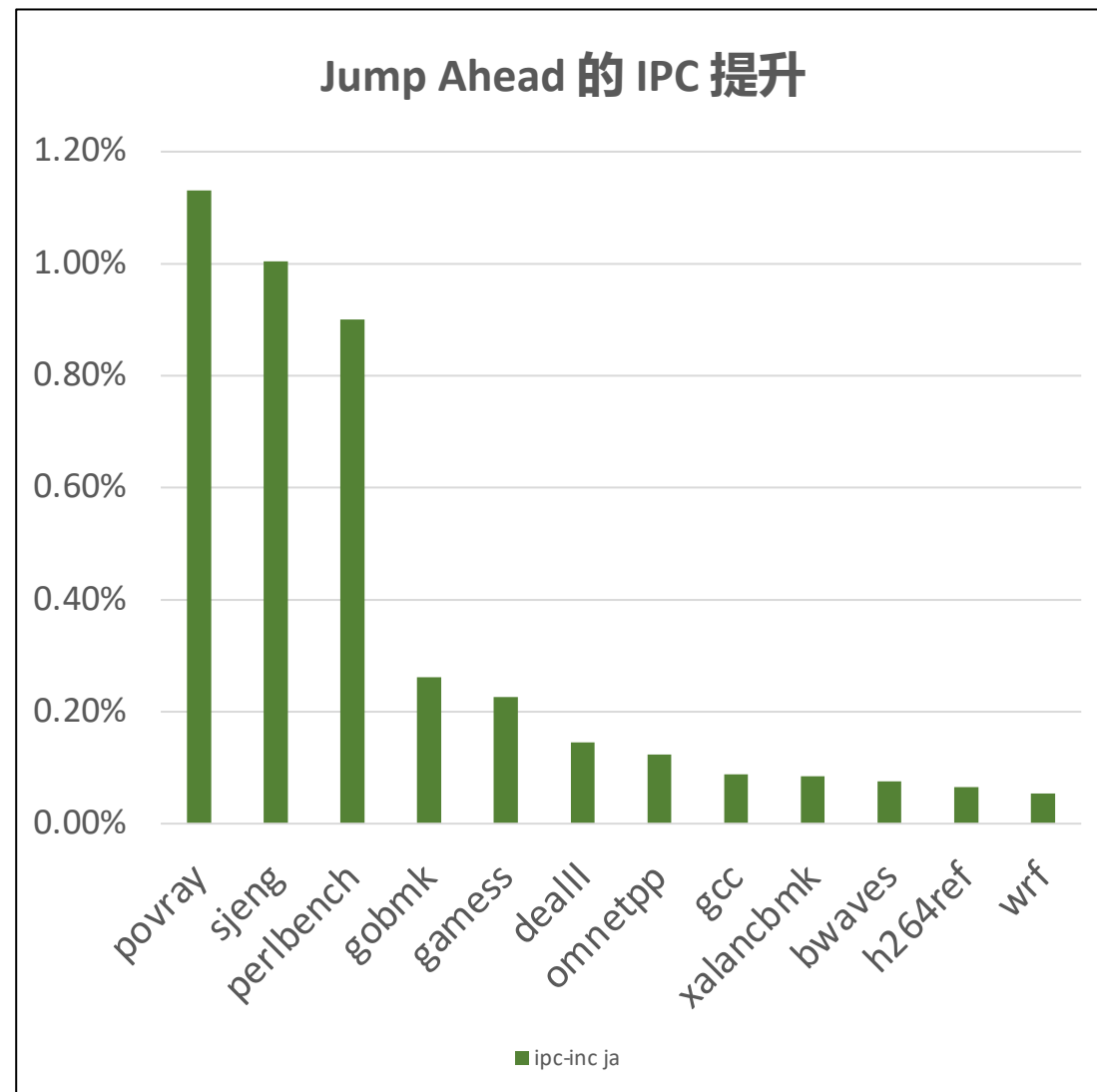
方向2：跳过无分支块 – Jump Ahead

- 训练

- 记录无分支跳转指令的连续**预测块首尾**
- 根据后端 commit 信息训练

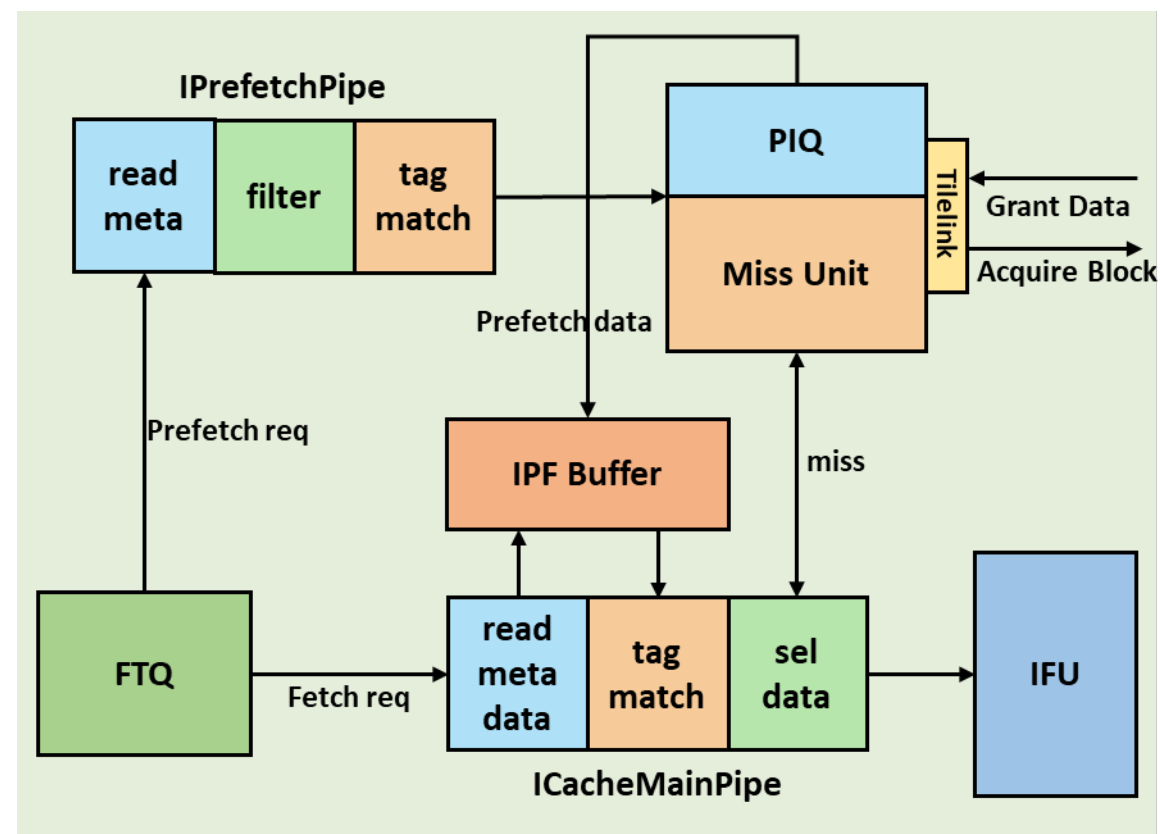
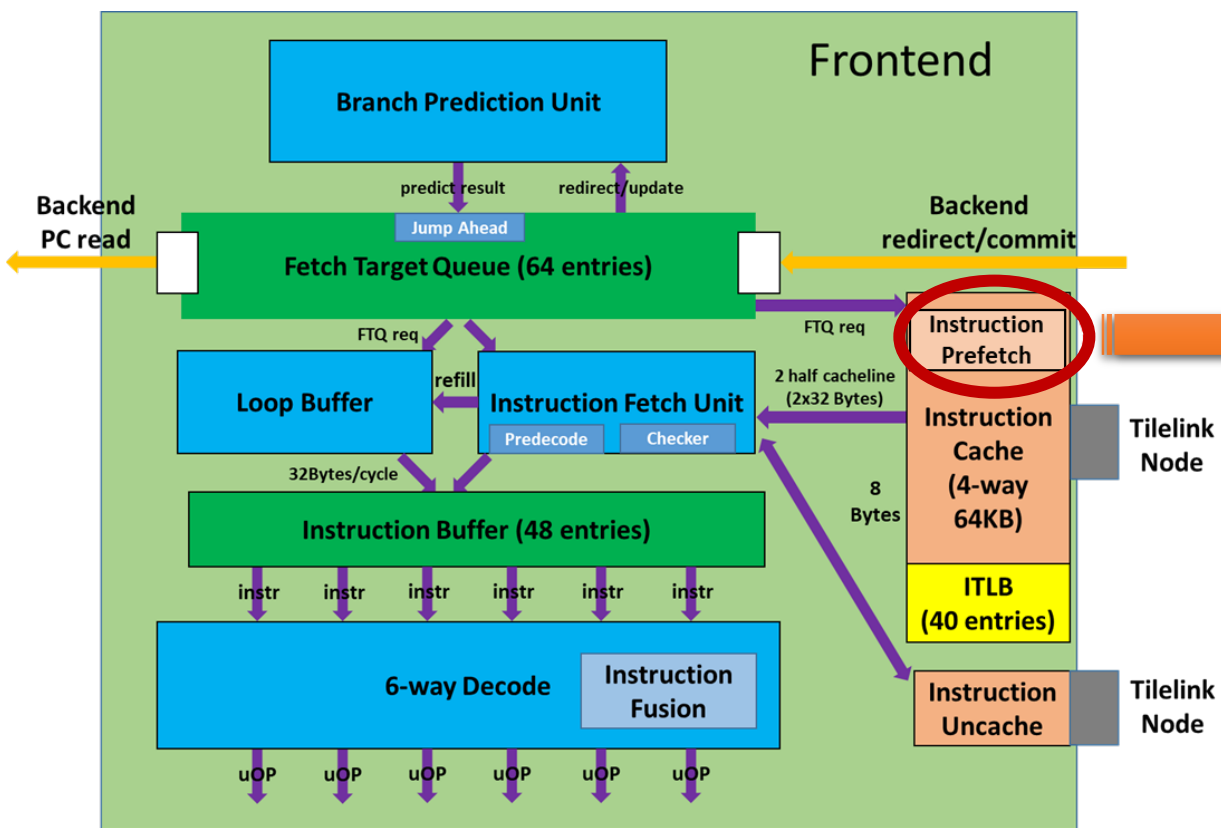
- 预测

- 使用 Jump Ahead 结果**跳过**连续无分支跳转预测块



方向3：减少 ICache Miss

- 基于解耦前端的设计，采用 FDIP 预取算法，根据预测块的信息做预取

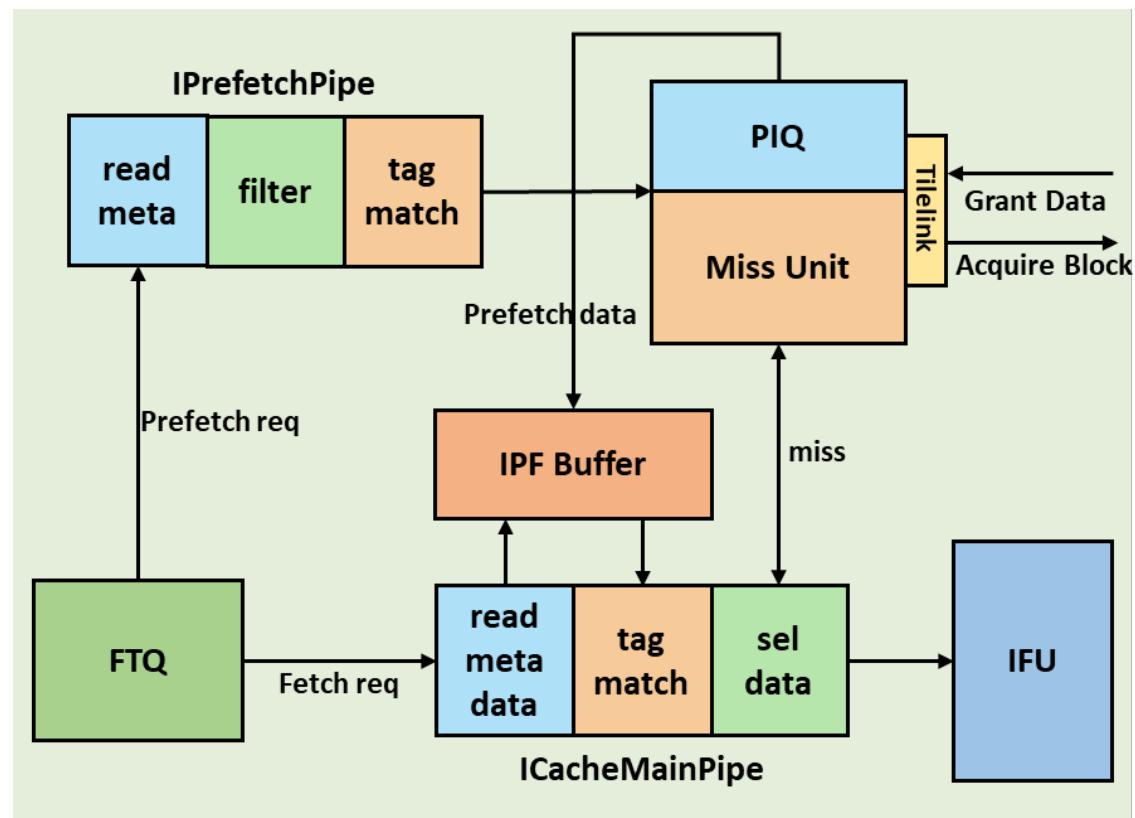


方向3：减少 ICache Miss

- 基于解耦前端的设计，采用 FDIP 预取算法，根据预测块的信息做预取

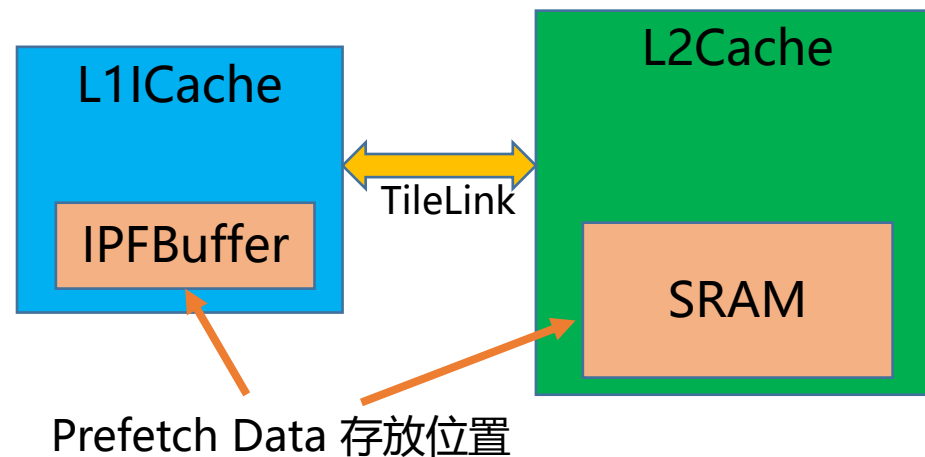
FDIP 预取流程

- 从预测队列中选出预测块发送预取请求
- 预取回数据放入 Prefetch Buffer
- 并行查询 ICache 与 Prefetch Buffer



🏔️ 方向3：减少 ICache Miss – L1 FDIP

- 将预取位置从 L2 Cache 移至 L1 ICache
- 采用两条预取流水线，增加预取带宽
- 细节算法优化

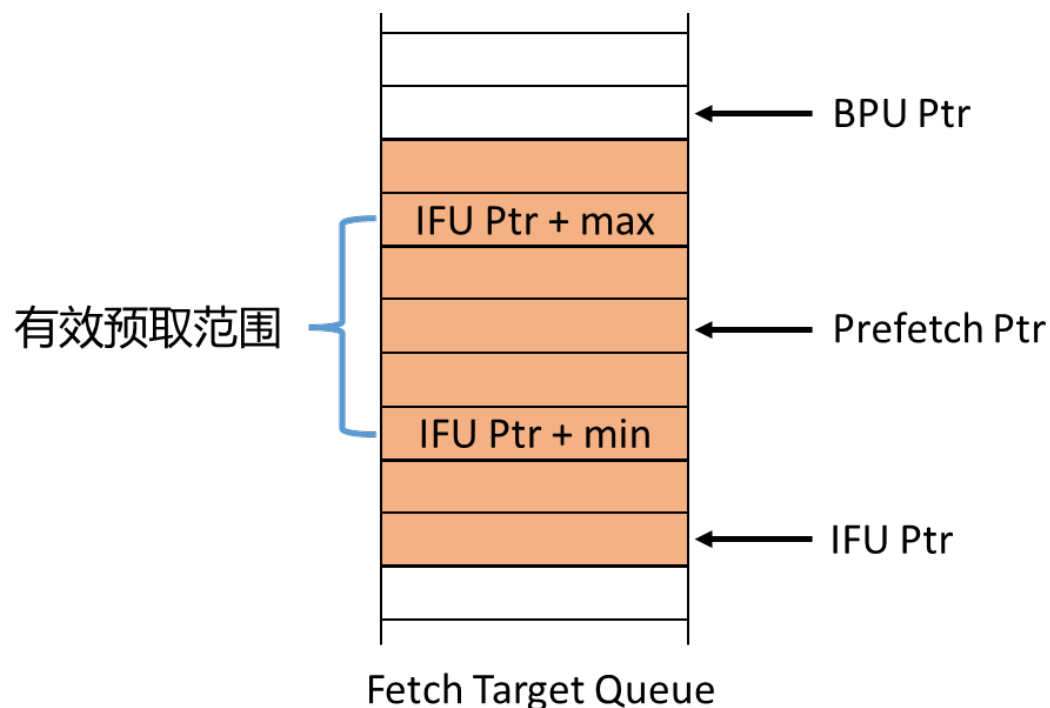


优化手段	优化目的
复制 ICache Meta	防止查询时堵塞 ICache
队列管理 Prefetch data	防止多次写回发生 multi-hit
限定预取范围	增加预取准确率
限制 Prefetch Data 写回 ICache	降低 ICache 污染

❄️ 方向3：减少 ICache Miss – L1 FDIP

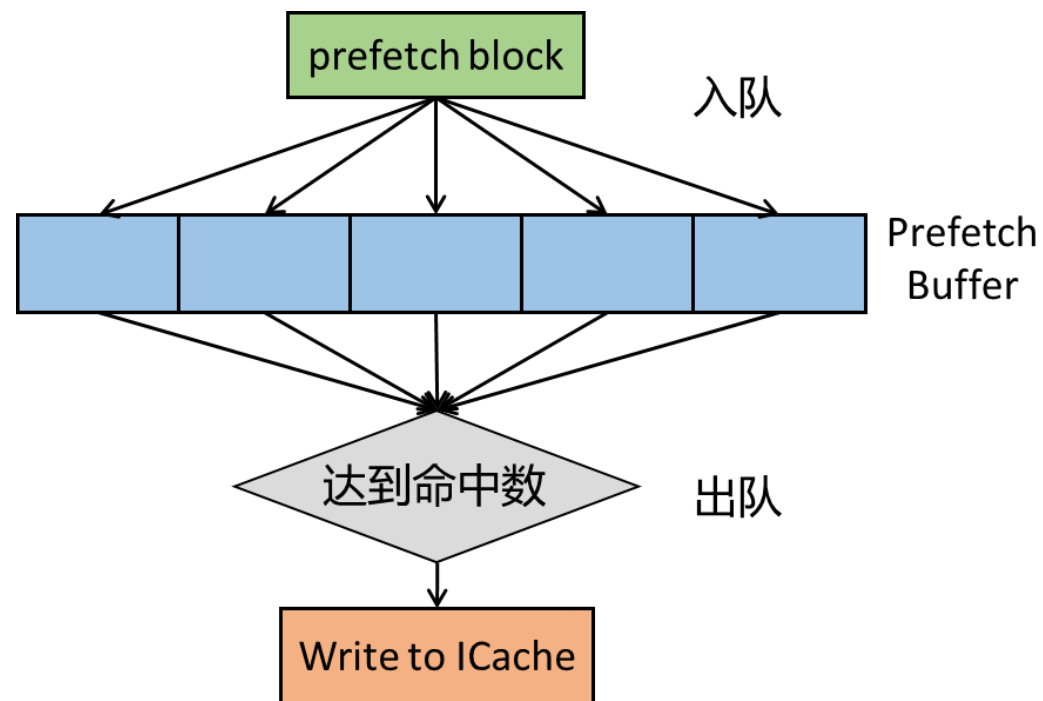
• 预取指针设计

- 限制 Prefetch 指针与取指指针相对距离
防止预取过早或过晚



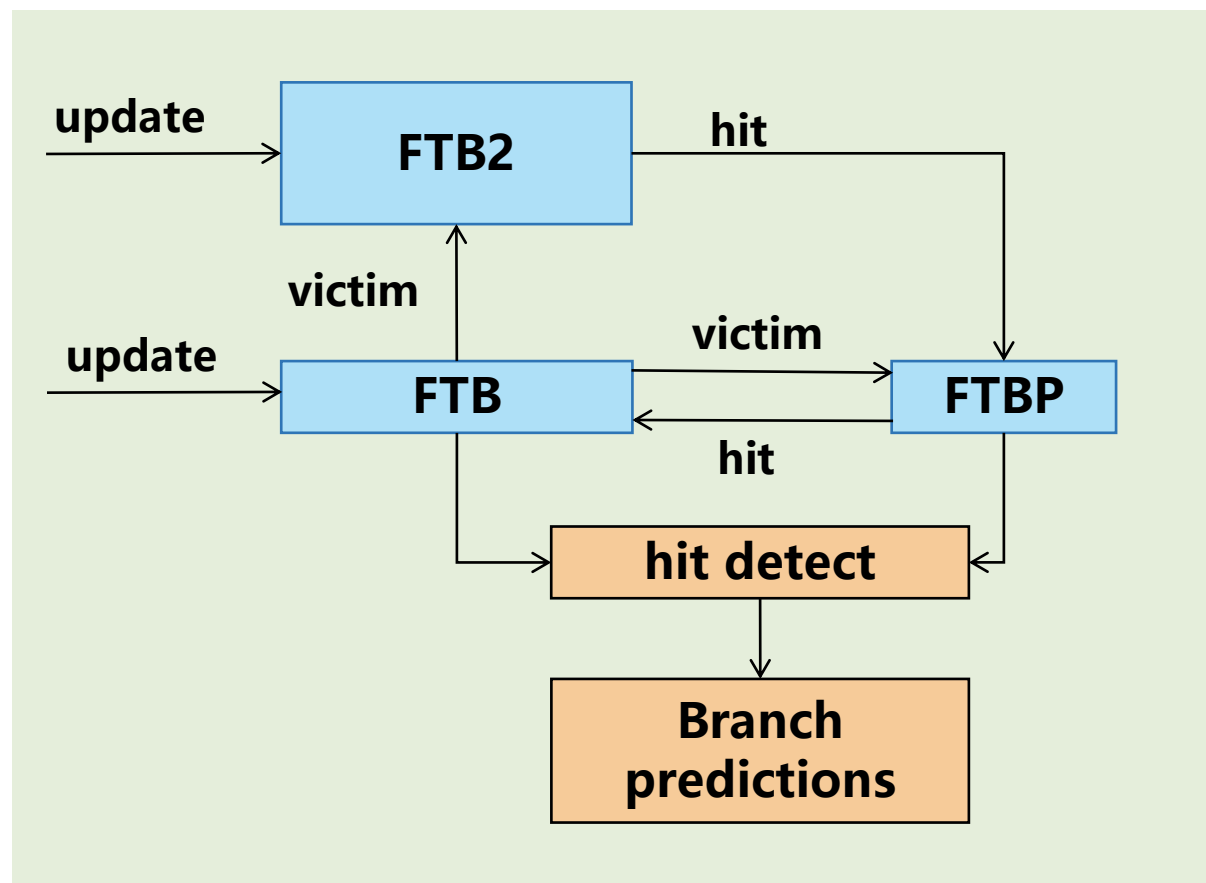
• Prefetch Buffer 设计

- 采用全相联结构，入队采用随机替换策略
- 统计命中数，多次命中才移进 ICache



❄️ 方向4：减少 FTB Miss – L2 FTB

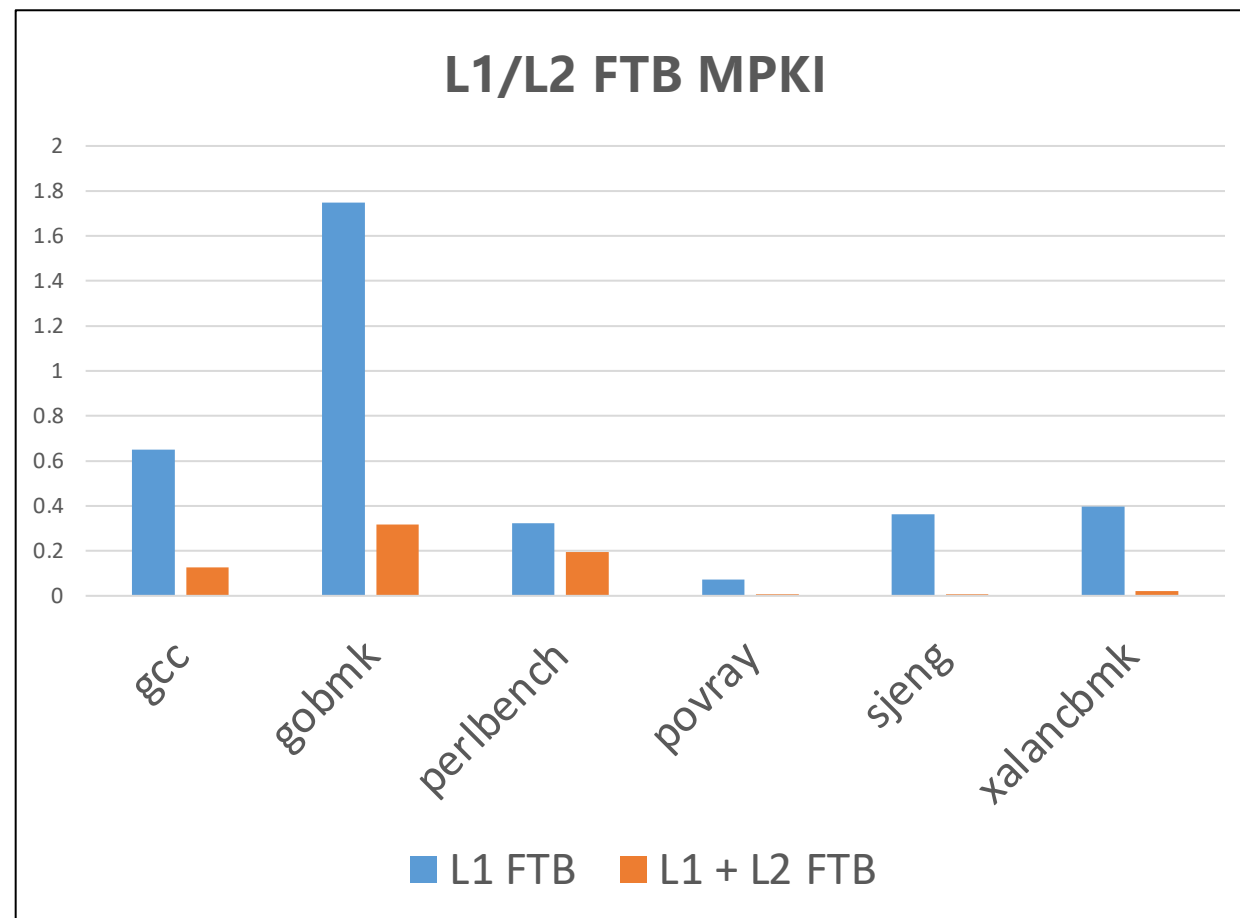
- 扩容 FTB
 - 增加 4K 项二级 FTB
 - 采用 semi-exclusive 结构
 - FTBP 作为 L1 FTB, L2 FTB 之间的缓冲, 存储来自 L1/L2 FTB 的条目



L2 FTB 结构

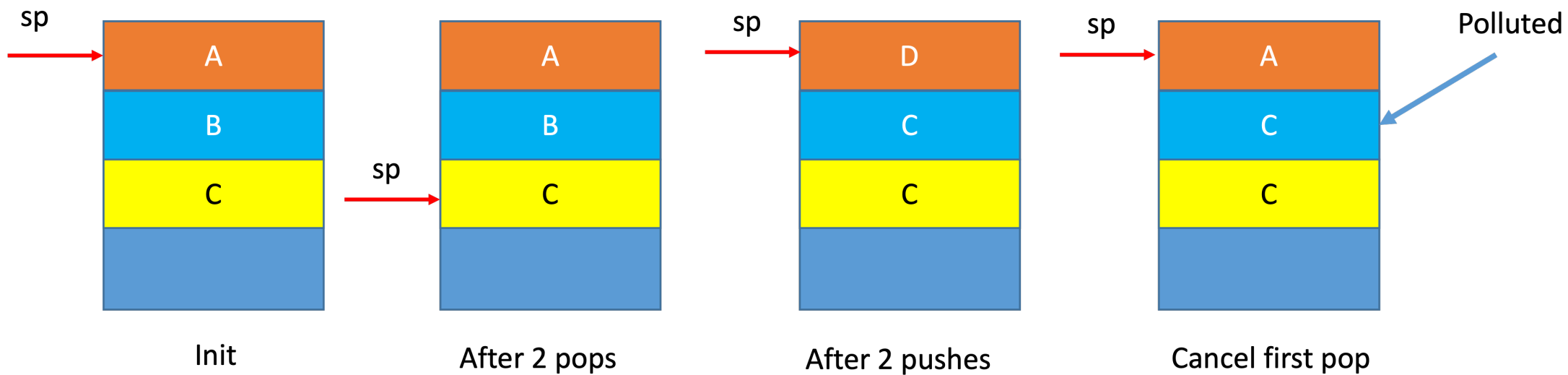
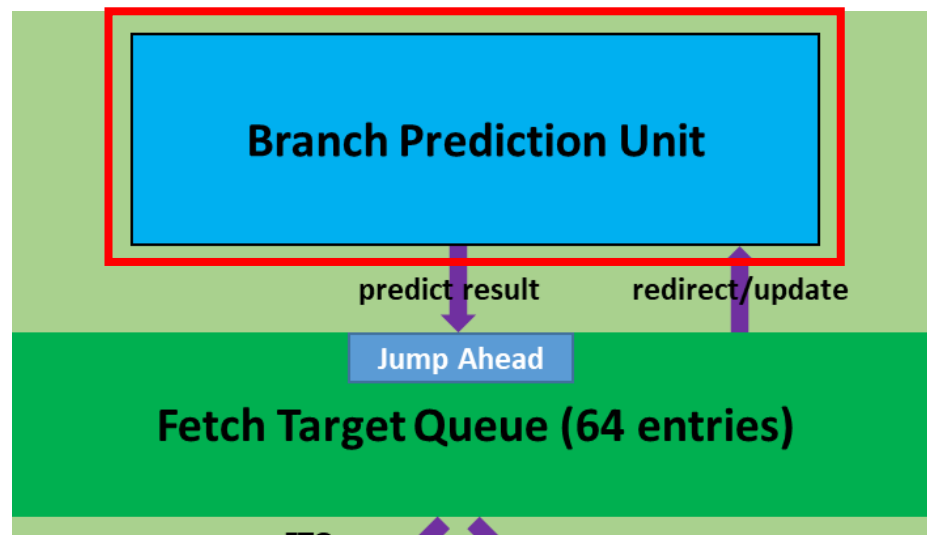
方向4：减少 FTB Miss – L2 FTB

- 扩容 FTB
 - 增加 4K 项二级 FTB
 - 采用 semi-exclusive 结构
 - FTBP 作为 L1 FTB, L2 FTB 之间的缓冲, 存储来自 L1/L2 FTB 的条目



❄️ 方向5：改进预测设计

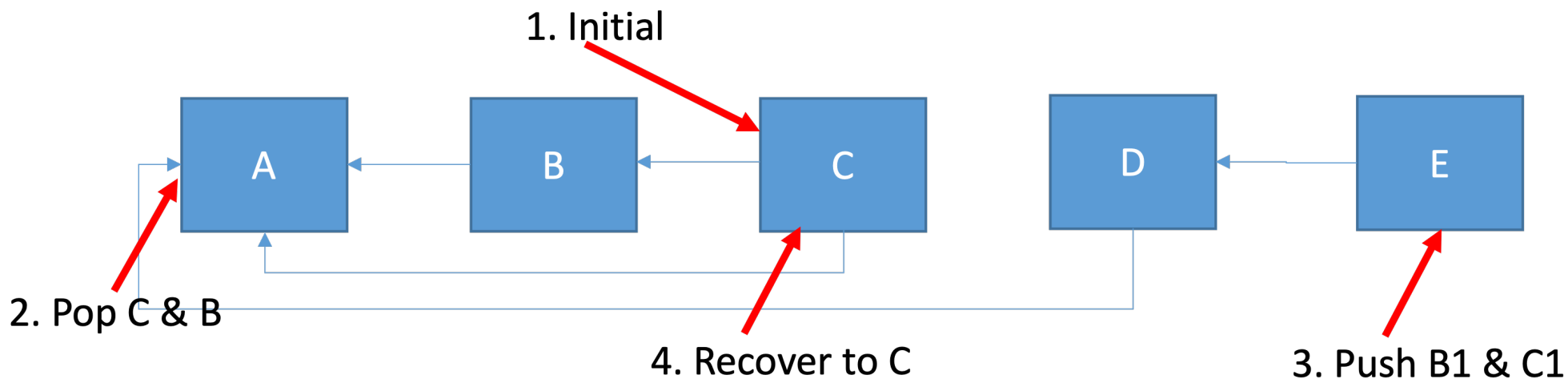
- RAS：预测 return 指令目标地址的栈结构
- 问题：传统 RAS 只恢复栈顶项，无法应对推测路径 pop 后 push 造成的污染。示例：





方向5：改进预测设计 – Queue-based RAS

- 解决：基于持久化队列的 RAS 设计^[1]
 - 传统 RAS 作为提交栈，只存储提交后的确定性信息
 - 推测执行中发生的 push/pop 在持久化队列维护，避免污染，示例：

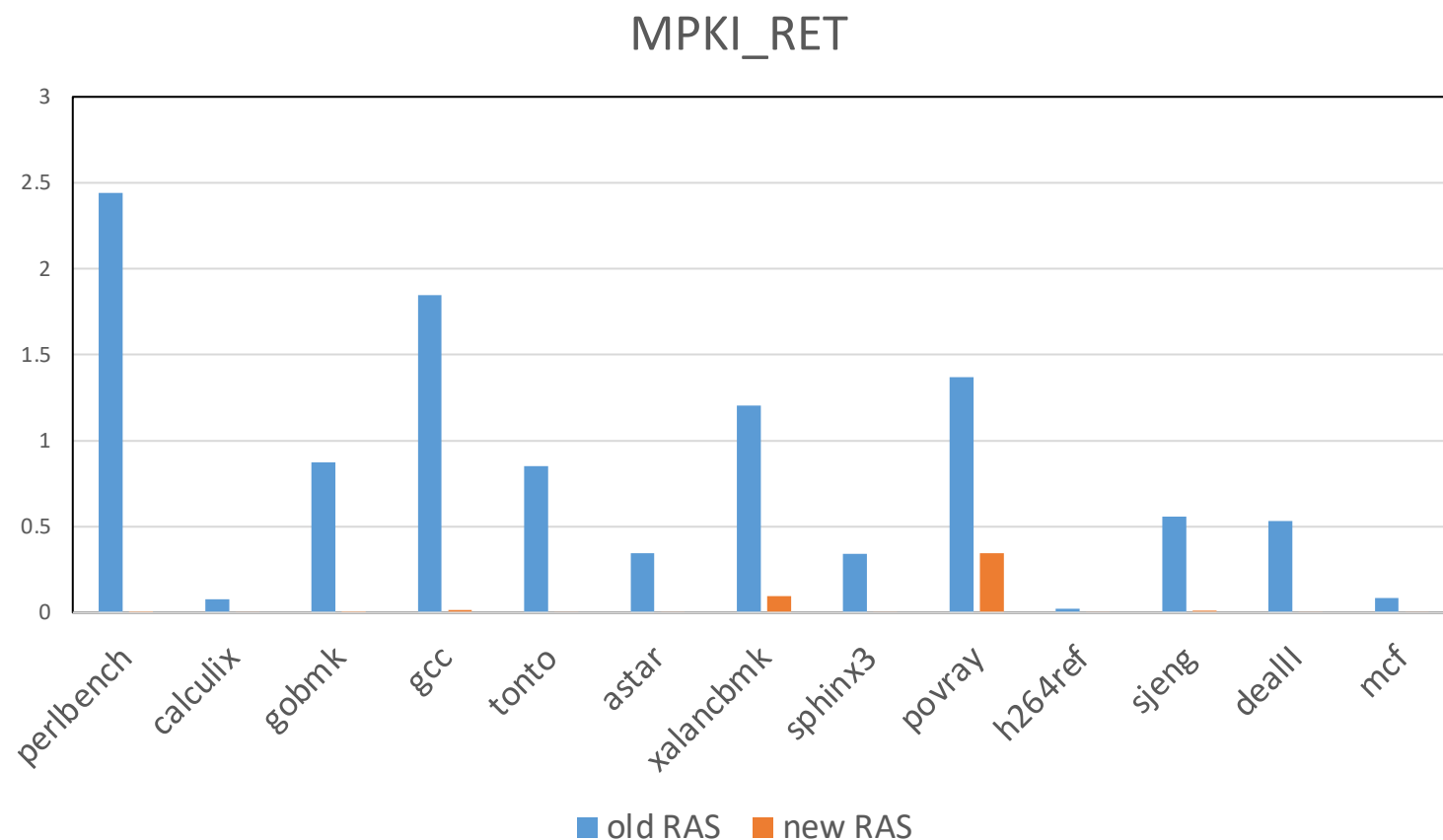


[1] Tan Hongze, Wang Jian. A Return Address Predictor Based on Persistent Stack[J]. Journal of Computer Research and Development, 2023, 60(6): 1337-1345. doi: 10.7544/issn1000-1239.202111274

🏔️ 方向5：改进预测设计 – Queue-based RAS

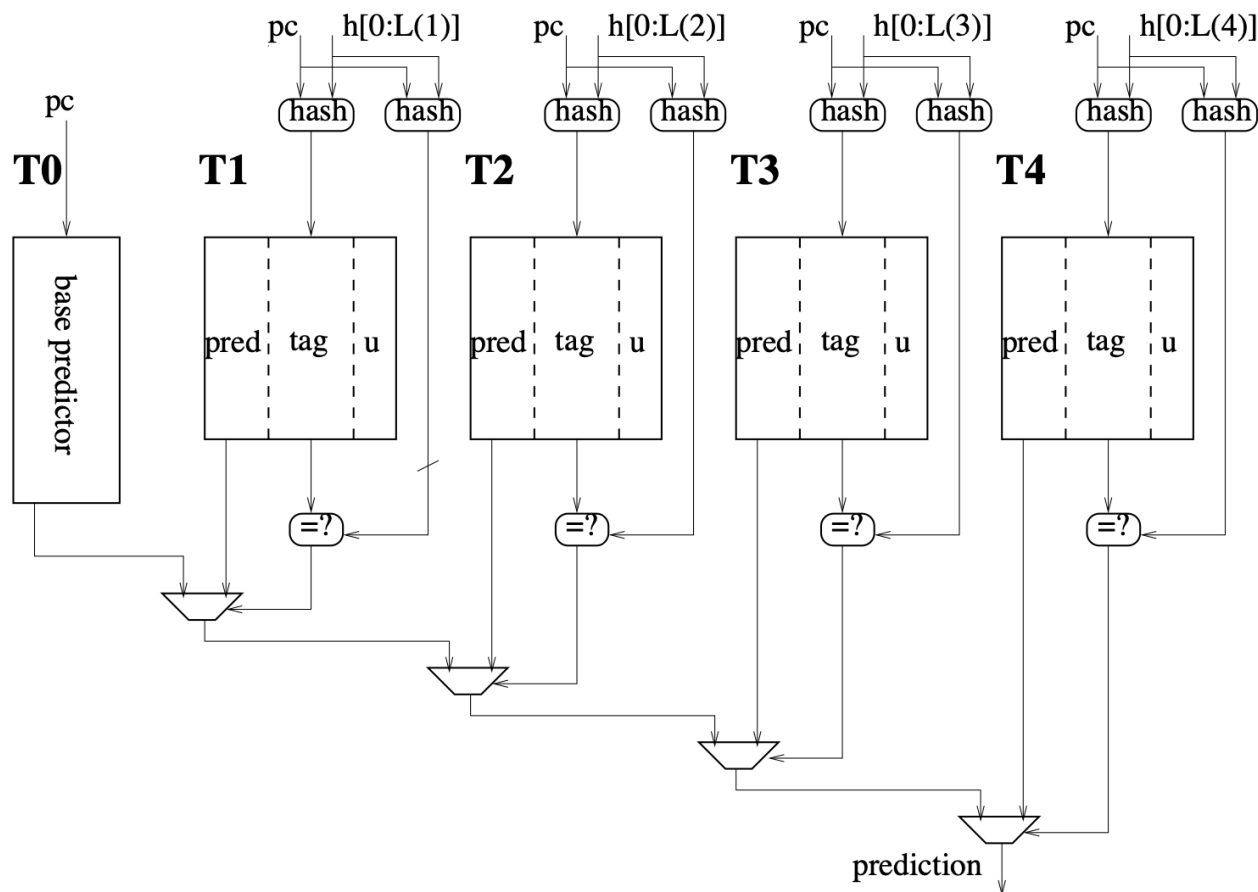
- 性能收益

- 有效降低了 return 指令误预测率



🏔️ 方向5：改进预测设计 –TAGE Tuning

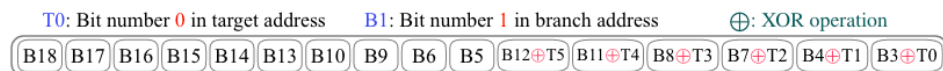
- TAGE: 现代高性能处理器条件分支方向预测的主流算法





方向5：改进预测设计 –TAGE Tuning

• ① 分支信息 Hash



Hash 算法影响性能

昆明湖使用遗传算法、粒子群算法等多种算法，对 TAGE 预测器的参数在模拟器上进行自动调整

南湖参数 -> 较优参数 IPC +1.44%

免费的性能提升!

• ② 表数量和历史长度

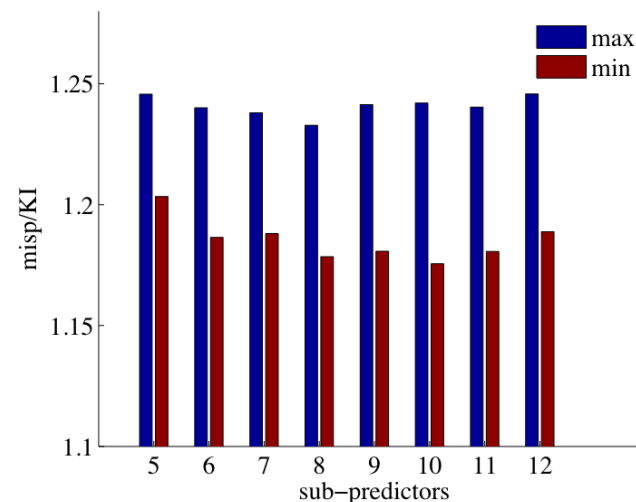


Figure 2: Performance of random parameters

此前研究表明，表数量和历史长度对 TAGE 性能有显著影响，MPKI 最高可相差 5.64%^[1]

[1] C. Zhou, L. Huang, Z. Li, T. Zhang, and Q. Dou, "Design Space Exploration of TAGE Branch Predictor with Ultra-Small RAM," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, Banff Alberta Canada: ACM, May 2017, pp. 281–286.

总结

- 昆明湖架构取指和分支预测单元迭代优化
 - **Loop Predictor, Jump Ahead 及 L2 FTB** 增大**预测宽度**
 - 新 **RAS** 及 **TAGE tuning** 提高**预测准确率**
 - **Loop Buffer** 增大**取指宽度**
 - **L1 FDIP** 降低**取指延迟**
 - 提高**频率**

敬请批评指正！