openEuler

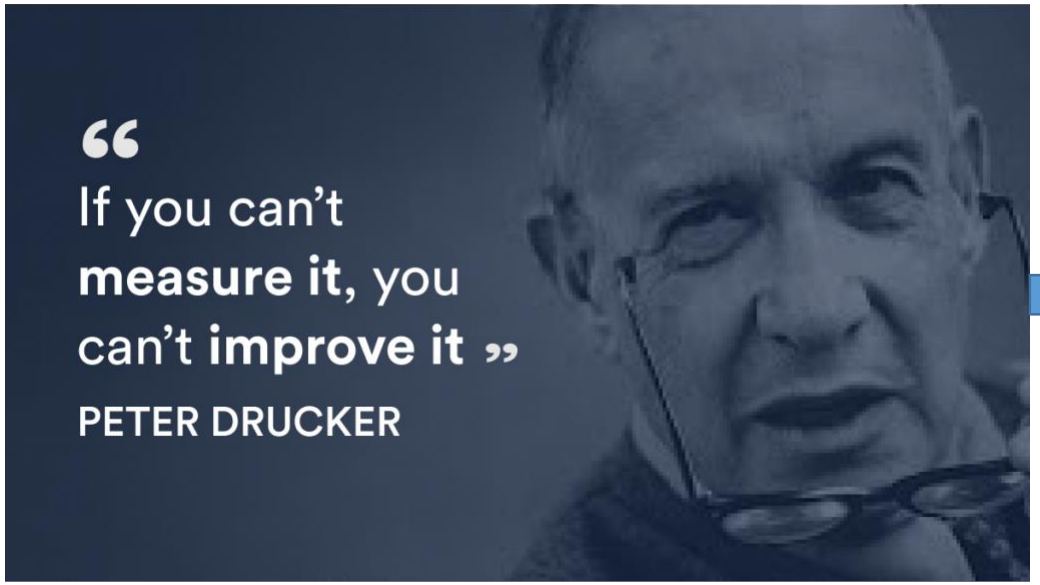
Top-Down性能分析方法介绍

目录

CONTENT

openEuler

背景



- “现代管理学之父”彼得·德鲁克曾说过这样一句话“If you can’t measure it, you can’t improve it”

- 类比到software optimization：”you can only optimize what you can measure”.
- 意味着要提高代码的性能，您必须能够准确地找到性能瓶颈的位置，进行优化并衡量改进，才能达到性能优化的目的。

openEuler

PMU(Performance Monitor Unit)的出现

- Performance Monitor Unit，性能监视单元，其实CPU提供的一个单元，通过访问相关的寄存器能读取到CPU的一些PMU事件；主流的处理器基本上都实现了PMU；
- Linux kernel在2.6.31-rc1时，支持了perf工具，可以在内核态获取硬件事件和用户态解析；

采用传统的PMU event的进行分析的问题

- PMU计数器记录的是发生过某事件的总的统计数量，这个用于判断应用场景的特性，但用于分析关键路径有欠缺。
- 对于乱序发射处理器而言，架构细节非常多，PMU针对的是架构的细节，但这个细节是处理器架构视图的，不能直观的转化为程序员视图。
- PMU计数提供的指标不是正交的，很多计数值之间有交集，导致某些损失被夸大，不能真正的发现性能的瓶颈（bottlenecks）


lwn.net/Articles/338573/

Linux 2.6.31-rc1

From: Linus Torvalds <torvalds@linux-foundation.org>
To: Linux Kernel Mailing List <linux-kernel@vger.kernel.org>
Subject: Linux 2.6.31-rc1
Date: Wed, 24 Jun 2009 17:18:19 -0700 (PDT)
Message-ID: <alpine.LFD.2.01.0906241656030.3605@localhost.localdomain>
Archive-link: Article

We've had the regular two-week (and one day) merge window, and -rc1 is out, and the merge window is closed.

[I suspect I'll still merge the SCore architecture, I wanted to give it a quick peek, but I've been busy with all the other patches so I'm closing the merge window now, but leaving myself the option of merging Score later - last I looked, the only non-SCore file it touched was the MAINTAINERS file, so it's not like it should break anything else]

There's a lot in there, but let me say that as far as the whole merge window has gone, I've seldom felt happier merging stuff from people. I'm really hoping it isn't just a fluke, but people kept their git trees clean, and while there were a couple of times that I said "no, I'm not going to merge that", on the whole this was a really painless merge window for me.

I'm not saying that it was necessarily less bug-free than usual, I'm just saying that on the whole people sent me merge requests that made sense, explained what they did, and when I pulled I saw clear development lines. That just makes it much easier for me.

So thanks to everybody involved.

I hope that doesn't mean that it was really painful for others, or that we'll be chasing down more bugs than usual. And I _really_ hope we can keep things going this way, and it wasn't a one-off "things just happened to work this time".

As to the actual changes - too many to list. As usual, the bulk of the changes are to drivers (70% of the diffs), with drivers/staging being the bulk of that (about 60% of all driver changes - 40% of the total). But wonder of wonder, I think drivers/staging actually _shrank_ this time around, alledgedly due to cleanups. Believe that who will.

On the filesystem front, we had btrfs, ext3 and xfs getting active development (Why xfs? Beats me, but that's what the stats say), and a fair chunk of work on the whole fsnotify unification work. And the VFS layer got some TLC wrt ACL and private namespace handling.

On architectures: ARM, powerpc, mips, sh, x86 are the bulk of it. On ARM, the bulk is new platforms (u300, freescale stmp, whatever), there seems to be no end to crazy new arm platforms. On x86 (and at least some degree on powerpc), a noticeable part is the whole new perf-counter subsystem. Along with lots and lots of other stuff.

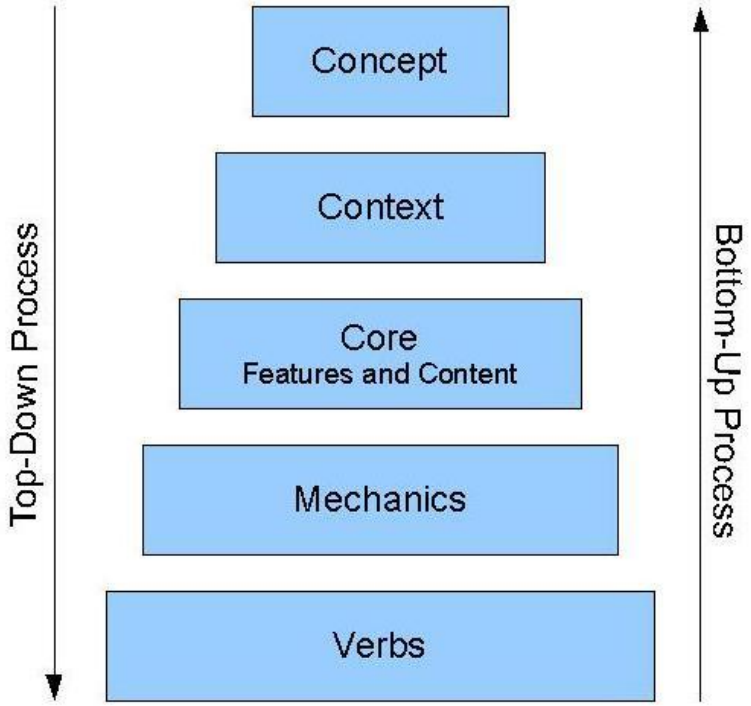
On the whole? Tons of stuff. Let's start testing and stabilizing.

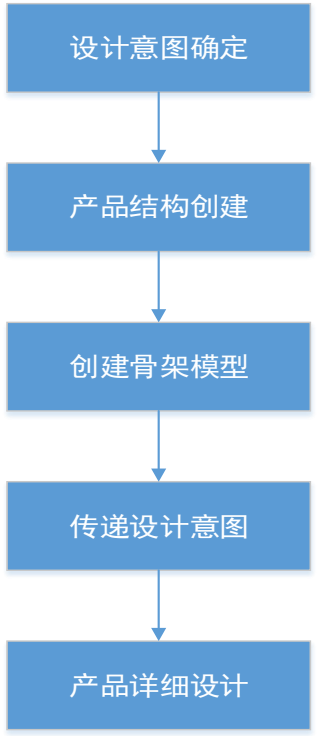
Linus

openEuler

Top-Down VS Bottom-up

- 自上而下和自下而上都是信息处理和知识排序的策略，用于各种领域，包括软件，人文和科学理论，以及管理和组织。在实践中，它们可以被视为一种思维，教学或领导风格。





Top-Down Methodology: Performance Analysis

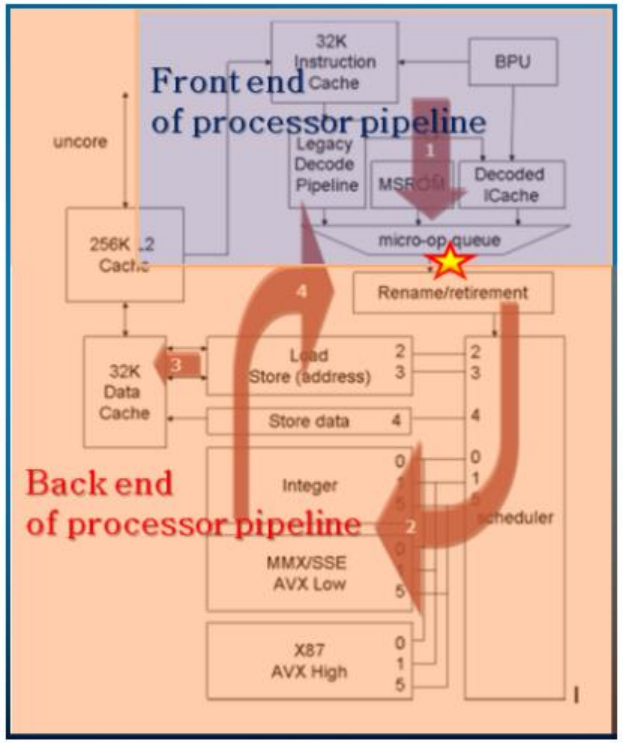
- Performance Analysis:
  - 系统级：
    - CPU 负载；
    - OS层面的影响；
    - 业务的自身模型和算法；
    - 等等；
  - SoC架构及core微架构级：
    - 单核、多核的处理能力；
    - 不同cache level的带宽；
    - Cache miss rate；
    - IPC；
    - 等等；

Top-Down Methodology: background

- Linux下对各种不同的应用及业务的调优越来越重要，但却越来越困难
  - CPU core的复杂度：长pipeline，多发射，乱序；
  - SoC的越来越庞大：多核，多片；
  - 应用及业务的多样性；
  - 等等

Top-Down性能分析模型的提出

- Intel工程师Ahmad Yasin在2014年发表一篇论文《 A Top-Down Method for Performance Analysis and Counters Architecture 》， Top-Down模型相关视图均引自于附件的paper；



A Top-Down for Performance

Figure 1: Out-of-order CPU block diagram - Intel Core™

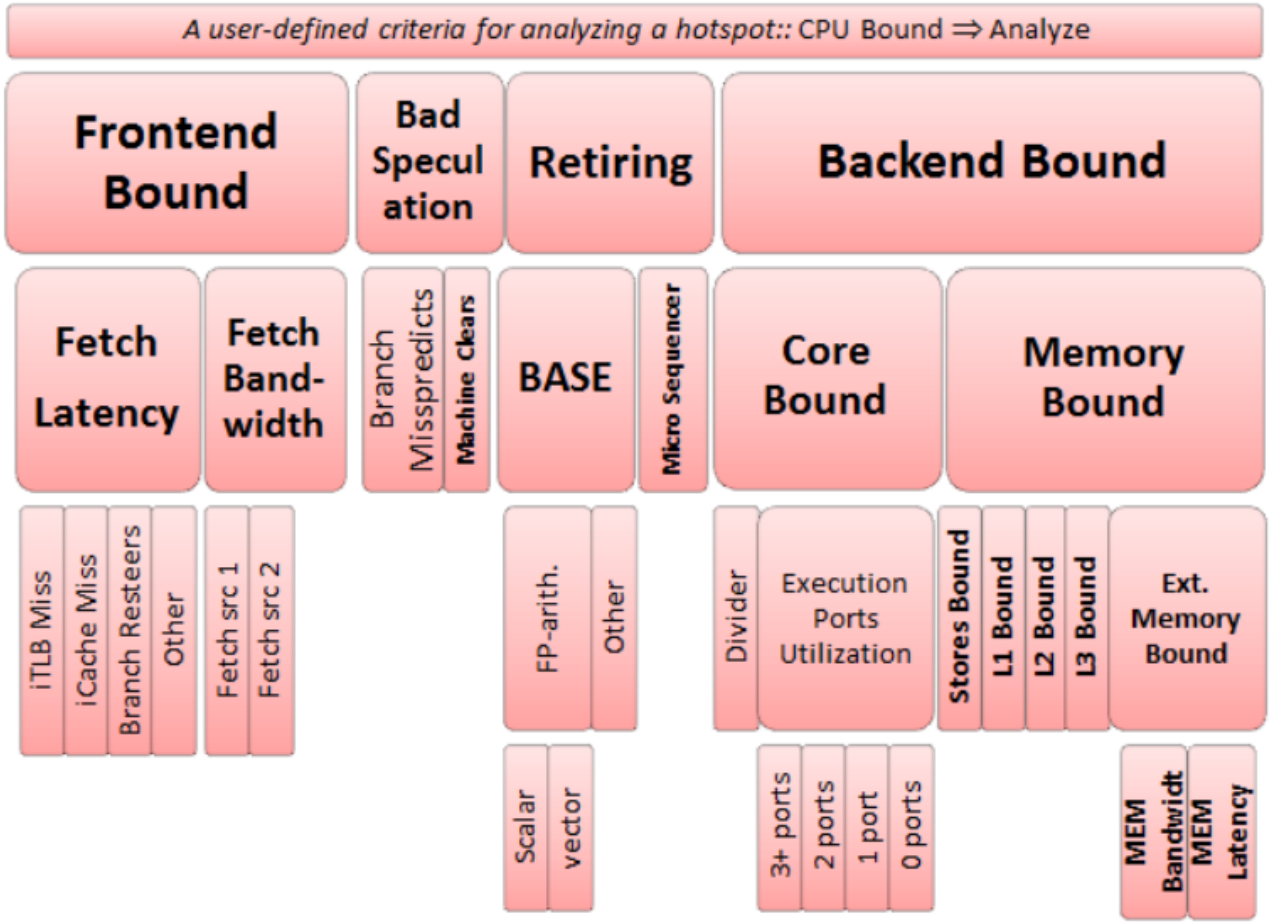
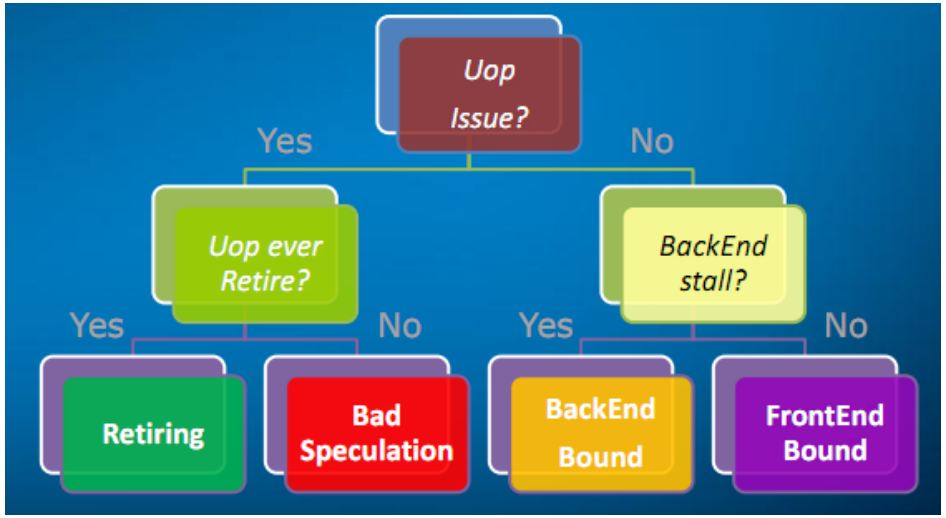


Figure 2: The Top-Down Analysis Hierarchy

Top Level Breakdown – the idea

- 软件指令经过取指和译码变为uop，uop在处理器的pipeline slots执行过程中，是否发生stalled；
- 如果没发生stalled，那么这条uop是正常执行结束(retiring)还是Bad Speclation?
- 如果发生stalled，那么是Frontend Bound还是Backend Bound?



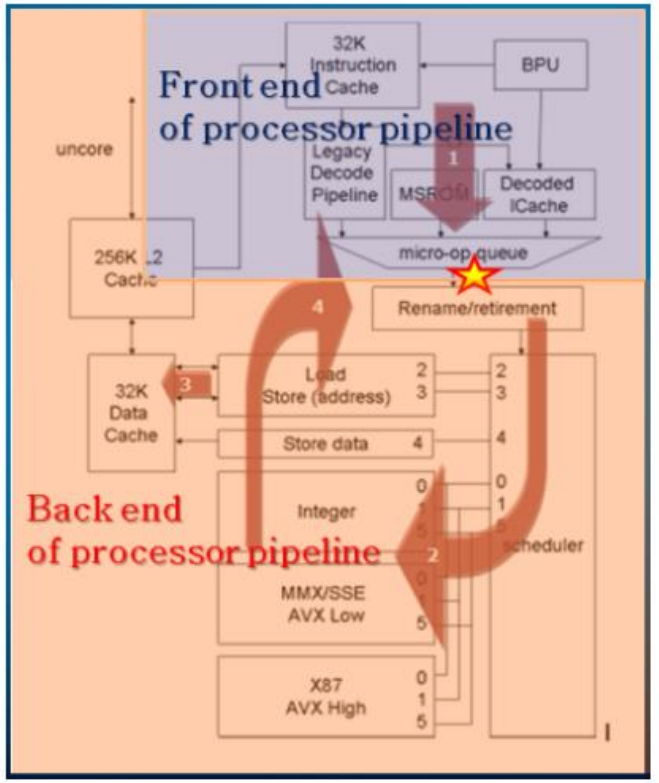


Figure 1: Out-of-order CPU block diagram - Intel Core™

Top Level Breakdown – retiring

- Retiring意味着解码后的uop最终都正常“退休”，没有发生阻塞，一段高效的代码retirng的比例越高；
- 它与 IPC（Instruction Per Cycle）相关，IPC是CPU 性能的一个非常重要的指标。

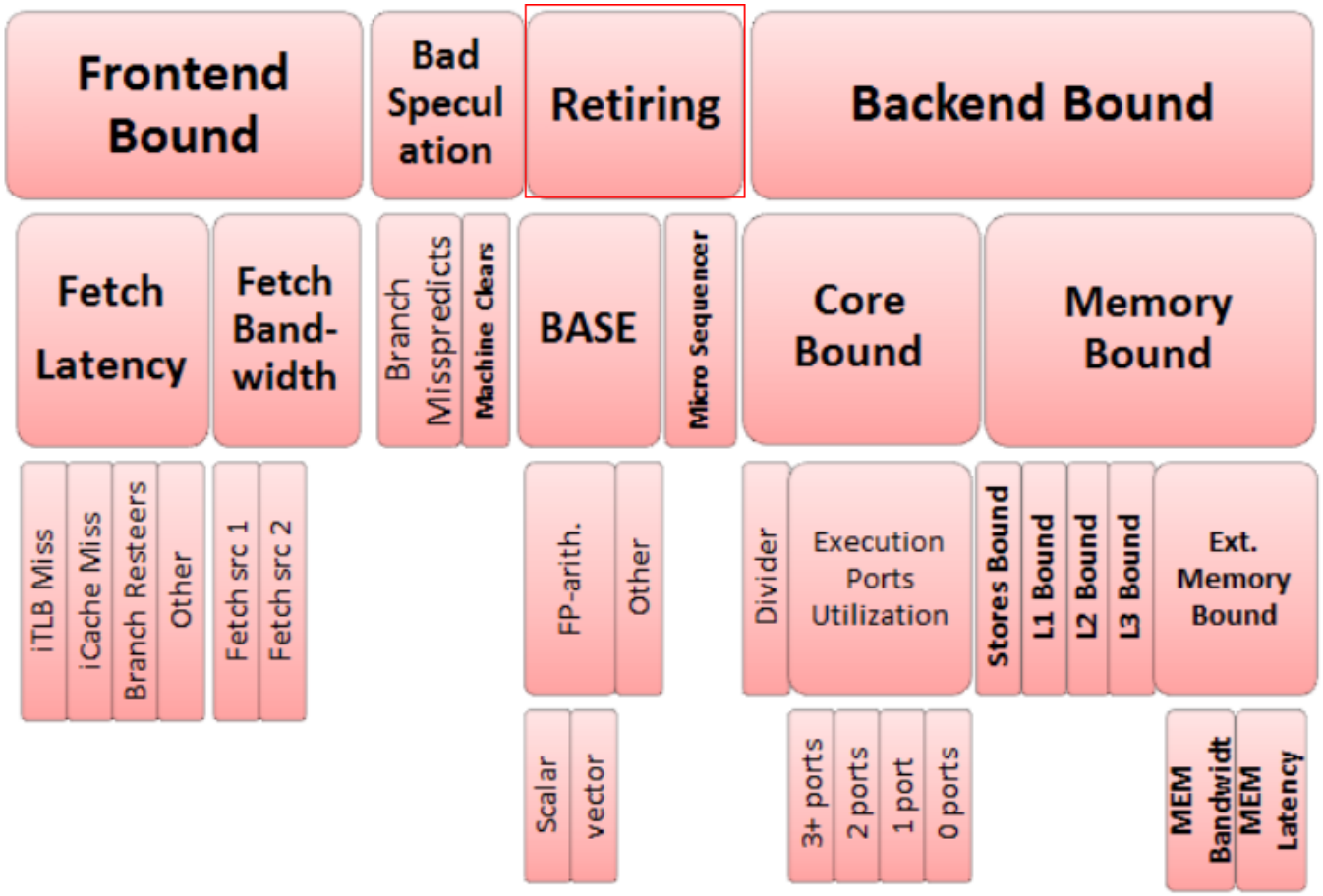
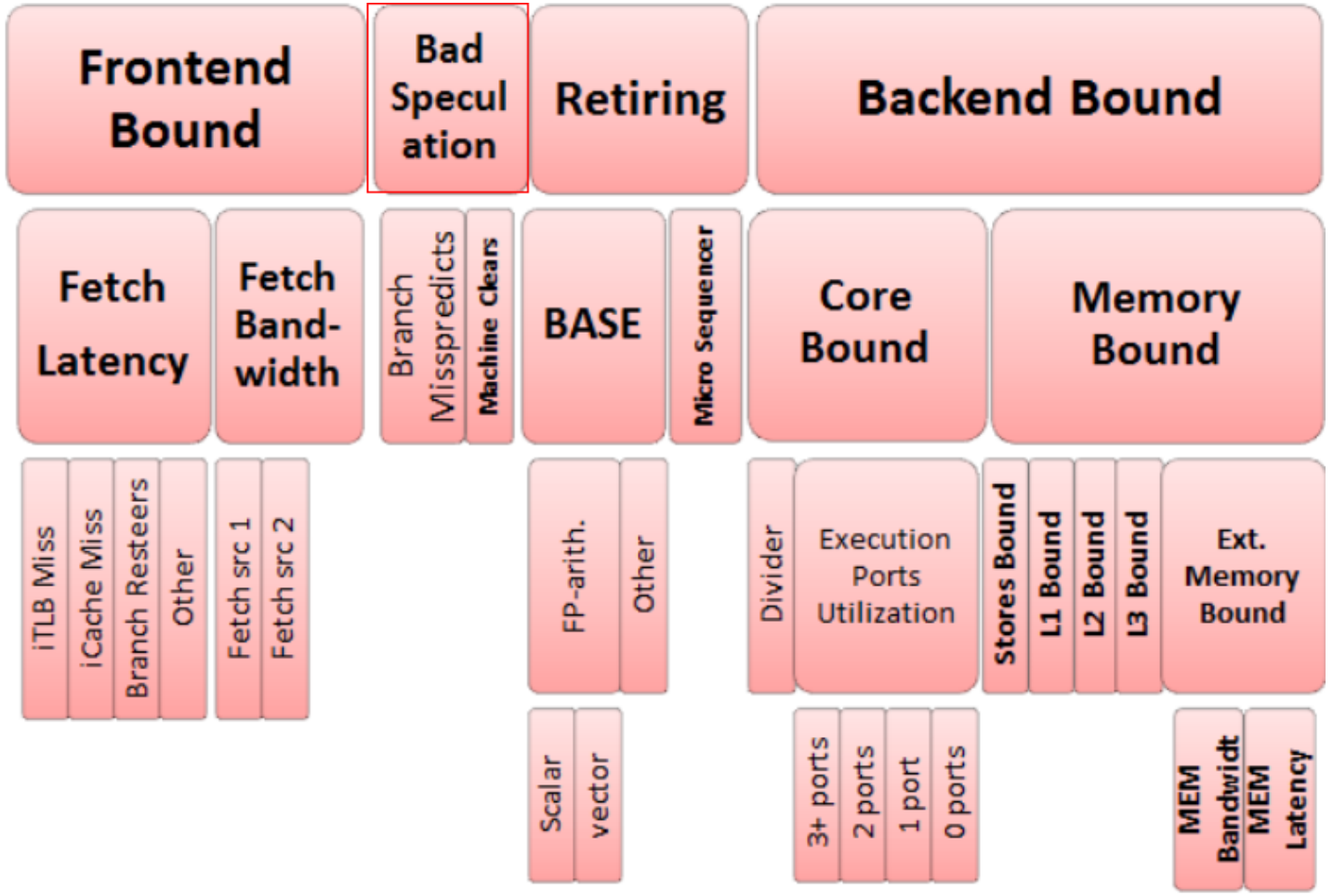


Figure 2: The Top-Down Analysis Hierarchy

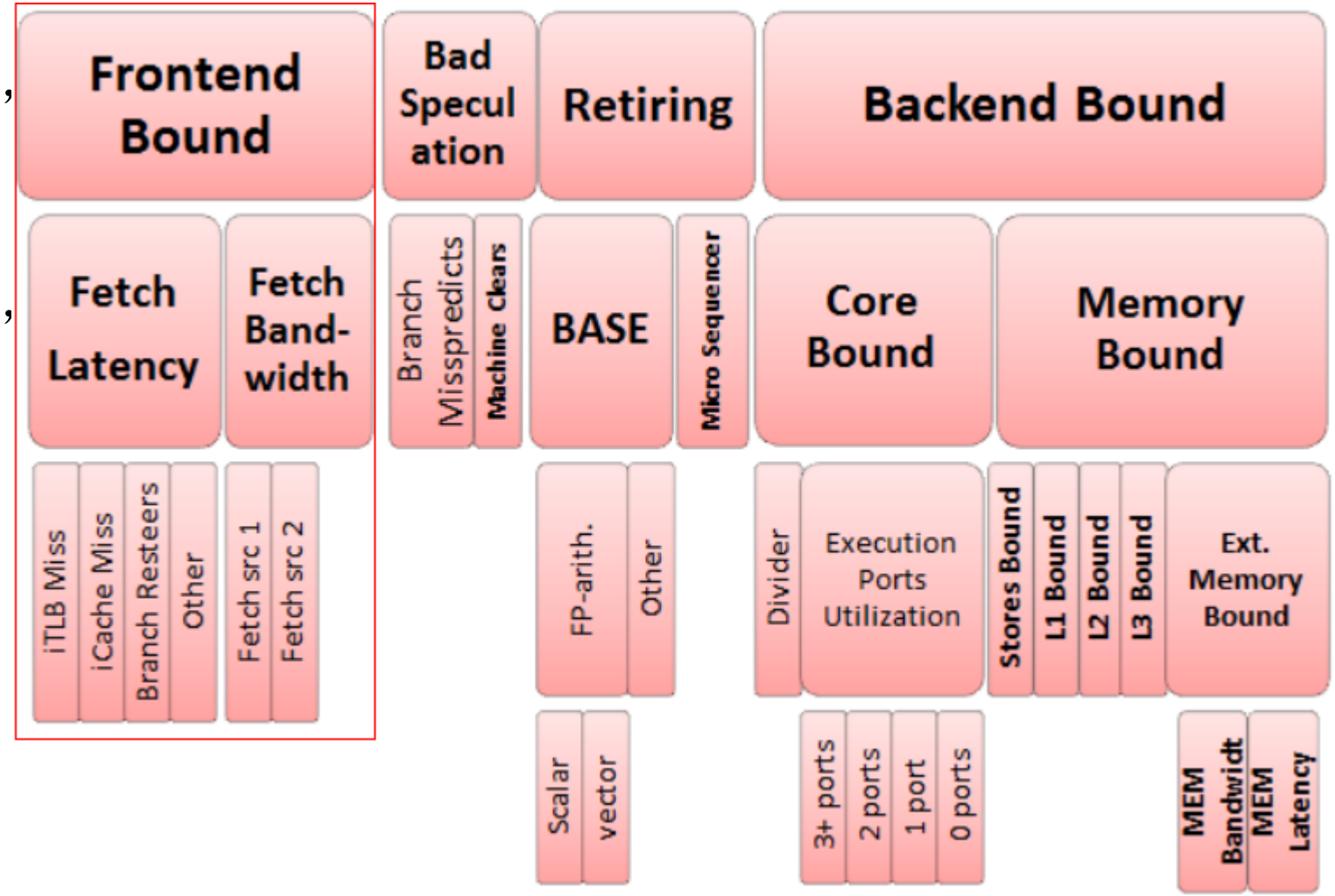
Top Level Breakdown – Bad Speculation

- Bad Speculation是表示因为预测错误导致的pipeline slots被浪费掉，在分支预测错误的情况下解码的指令被归到此类；
- Branch misprediction 和 machine clear



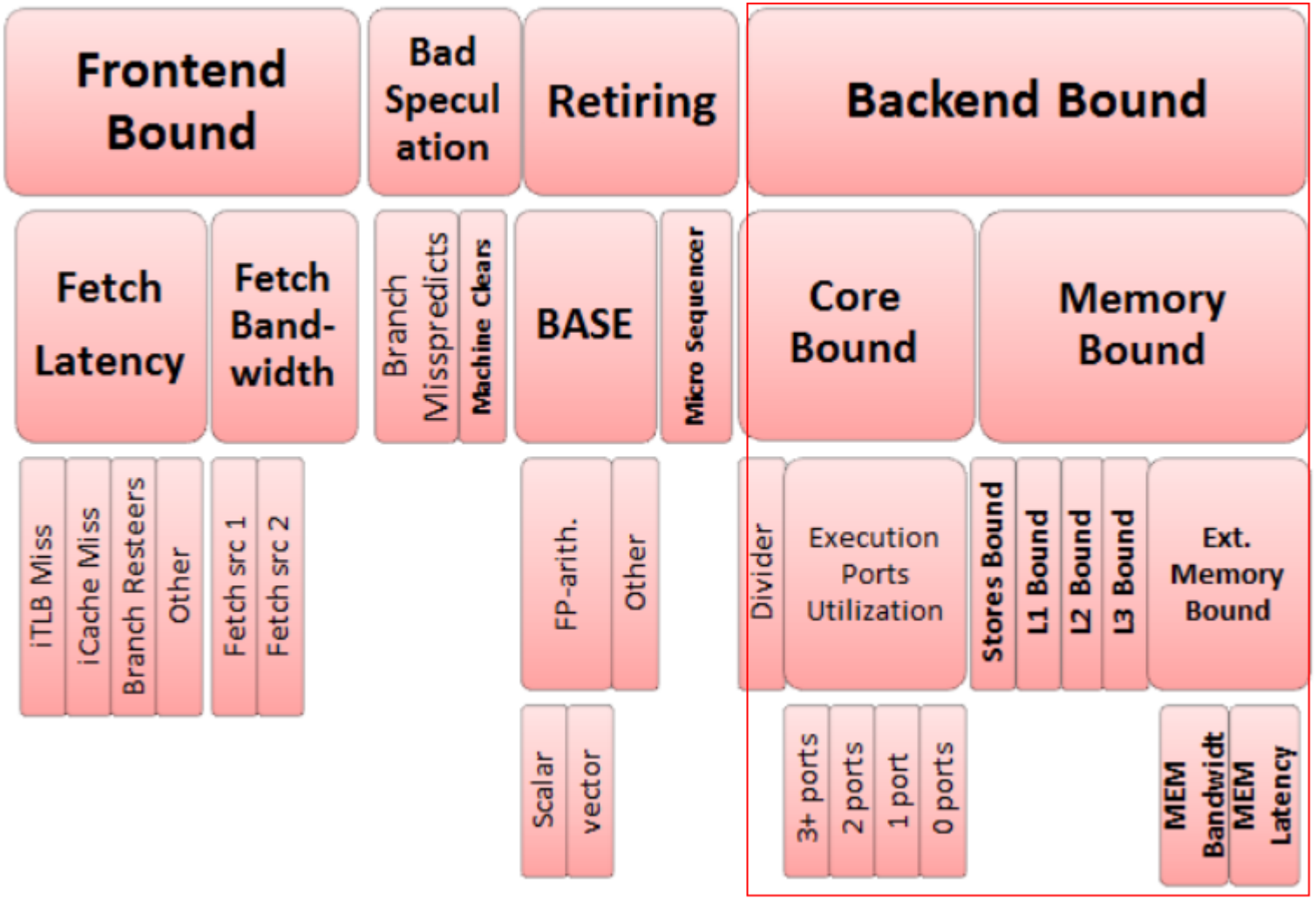
Top Level Breakdown – Frontend Bound

- Frontend Bound说明pipeline前端指令供应不足，导致了Backend执行处于饥饿的等待状态；
- Latency bound是因无法从icache获取指令导致的阻塞，主要包括ITLB miss，icache miss等
- Bandwidth bound意味着获取操作在执行但是传输带宽无法满足；



Top Level Breakdown – Backend Bound

- Backend Bound说明后端缺乏必要资源导致了流水线的停滞;
- Core bound是指计算单元或者指令级别的并行度匮乏，大量指令或者长期使用相同的计算单元
- Memory bound是指cache-memory子系统相关的执行阻碍，通常是因为存储子系统无法提供数据导致流水线的饥饿等待;



Intel Top-Down expected range

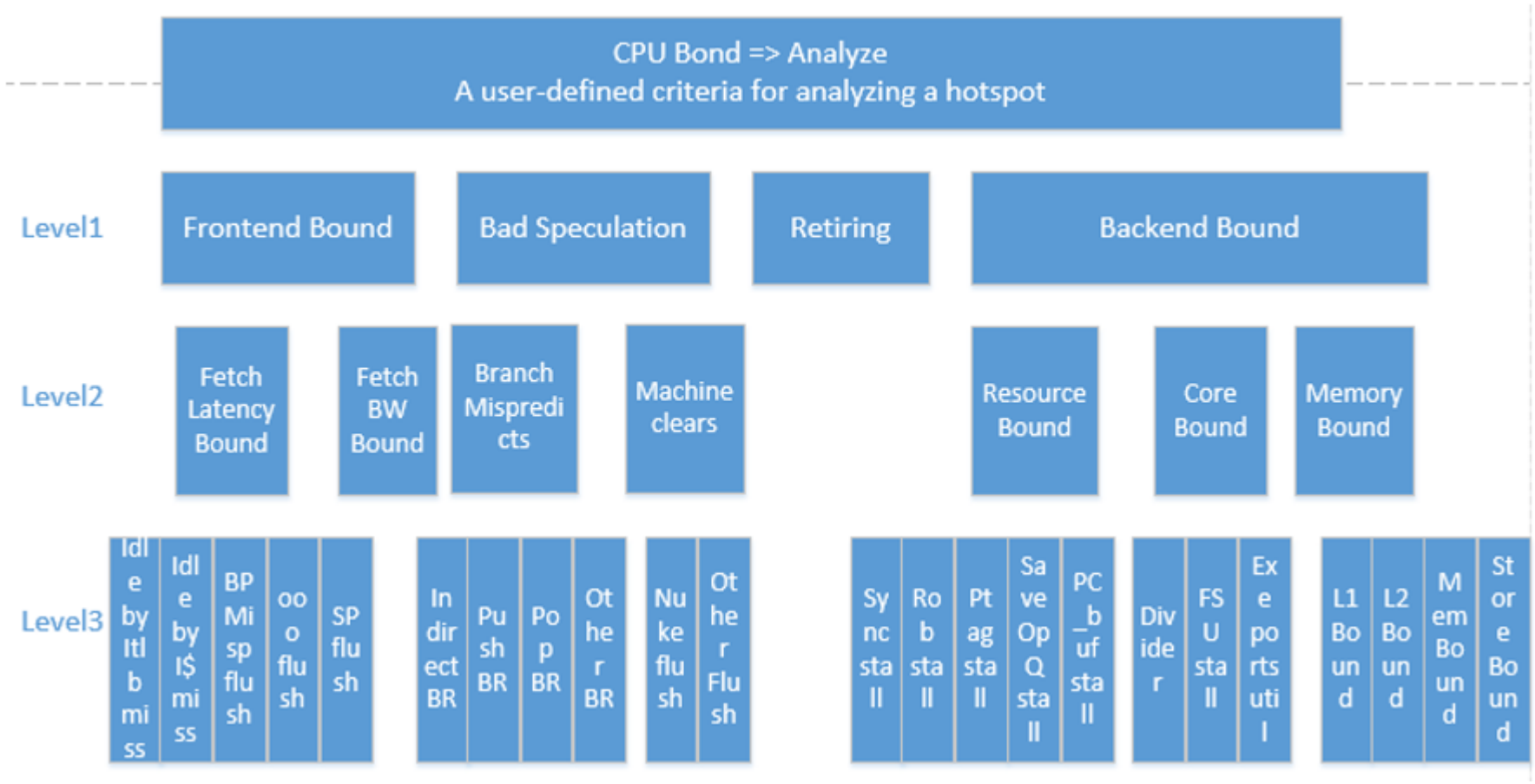
Top-Down的level1一般是正交的，不同的业务形态，四个分类的占比也是不同，Intel披露了一个三种典型业务形态的Top-Down数据；

	Expected Range of Pipeline Slots in This Category, for a Hotspot in a Well-Tuned:		
Category	Client/Desktop Application	Server/Database/Distributed application	High Performance Computing (HPC) application
Retiring	20-50%	10-30%	30-70%
Back-End Bound	20-40%	20-60%	20-40%
Front-End Bound	5-10%	10-25%	5-10%
Bad Speculation	5-10%	5-10%	1-5%

数据来源：https://software.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/methodologies/top-down-microarchitecture-analysis-method.html

openEuler

鲲鹏920 topdown的概览



Top Level Definition and Equations

FrontEnd Bound

- The front end is delivering < 4 uops per cycle while the back end of the pipeline is ready to accept uops
  - Total of FetchBubbles(0x2014) / (4 * ncycles(0x11))

Bad Speculation

- Tracks uops that never retire or allocation slots wasted due to recovery from branch miss-prediction or clears
  - (decoded_insts(0x1b) - retired_insts(0x08)) / (4 * ncycles(0x11))

Retiring

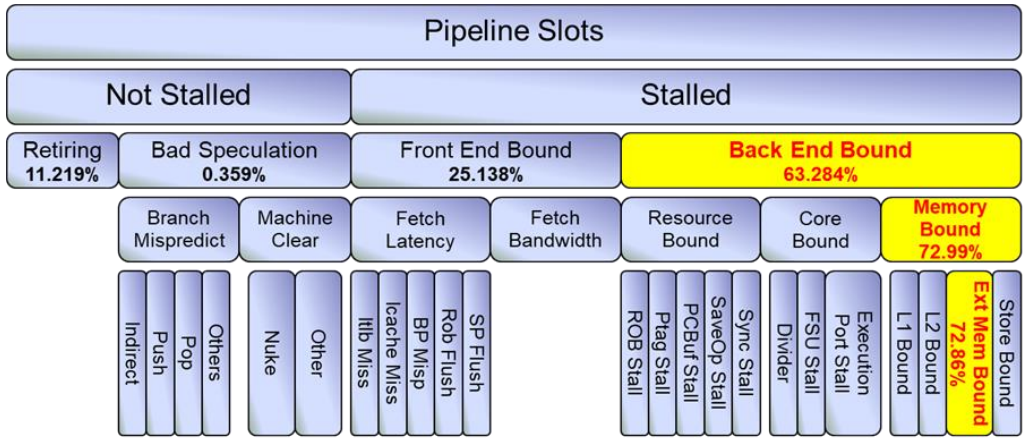
- Successfully delivered uops who eventually do retire
  - retired_insts(0x08) / (4 * ncycles(0x11))

Backend Bound

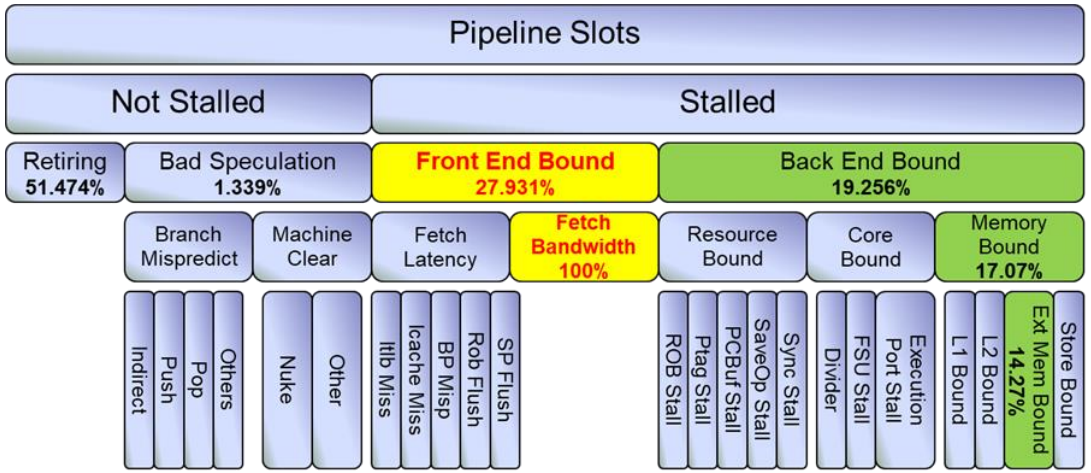
- No uops are delivered due to lack of required resources at the back end of the pipeline
  - 1 - (Frontend_Bound + Bad Speculation + Retiring)

Top-Down模型分析实例

- SpecInt2006的Libquantum测试子项，优化前为Ext Memory bound



- 引入HWP之后，Backend Bound消失，变为Frontend Bound，IPC提升3.5倍



总结

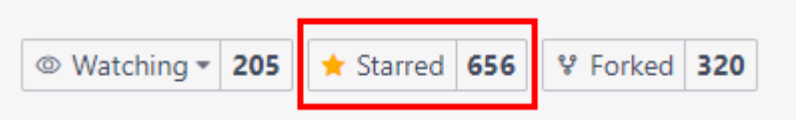
- Top-Down的性能分析方法，按照自上而下的思想，利用PMU（Performance Monitor Unit）提供的能体现CPU pipeline各部分性能的事件（events），从整体上对性能进行分析；
- 通过Top-Down模型，可以实现业务Pattern与CPU微架构相结合，从而快速、准确地识别业务软件和CPU的性能瓶颈的关键点；
- Top-Down没有识别出性能瓶颈点的代码，需要用perf tool进一步根据关键事件来锁定性能瓶颈点的真正原因，然后使用Bottom-up的方法解决性能瓶颈；

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

openEuler

Thank you

openEuler