

# AMD架构虚拟机性能探索与实践

阿里云 郑翔

# Overview

- AMD虚拟机idle调度优化
- AMD虚拟机支持TLBi
- AMD虚拟机使能AVIC

# Overview

- **AMD虚拟机idle调度优化**
- AMD虚拟机支持TLBi
- AMD虚拟机使能AVIC

# AMD虚拟机idle调度优化——idle介绍

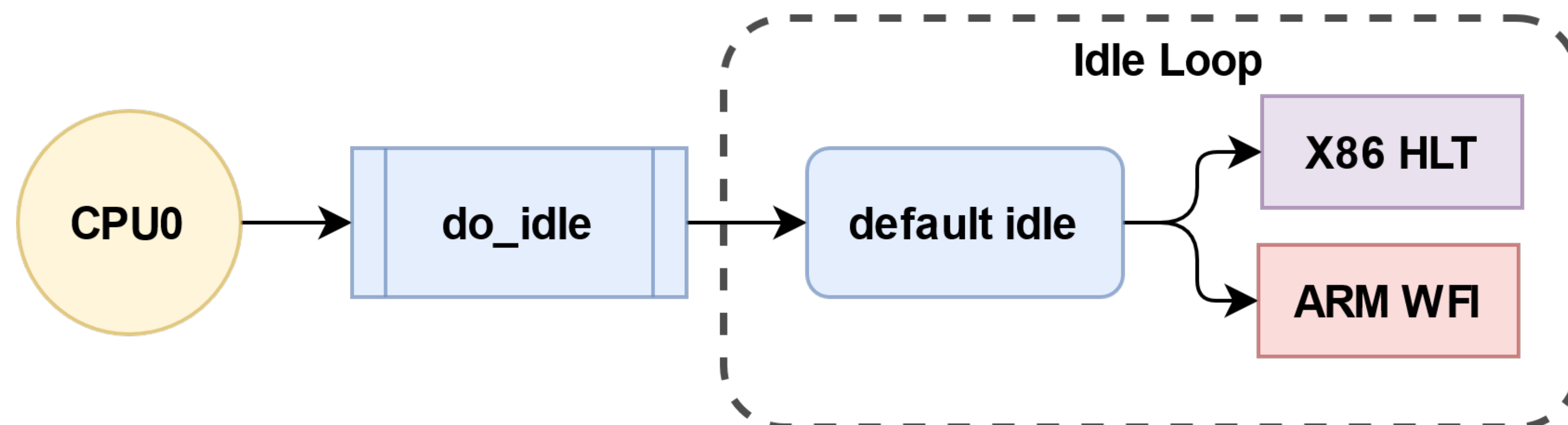
当CPU没有task执行时，即“空闲”的时候，会执行特殊的idle task（熟知的do\_idle）。

不同架构在做do\_idle时，最终调用的指令是不同的：

1. X86调用HLT指令
2. ARM调用WFI（Wait For Interrupt）指令

之后，CPU会进入low-power状态，节省能耗。

当中断（如时钟和IPI）到来时，CPU退出low-power状态，如果有新的需要调度的task，则退出idle loop



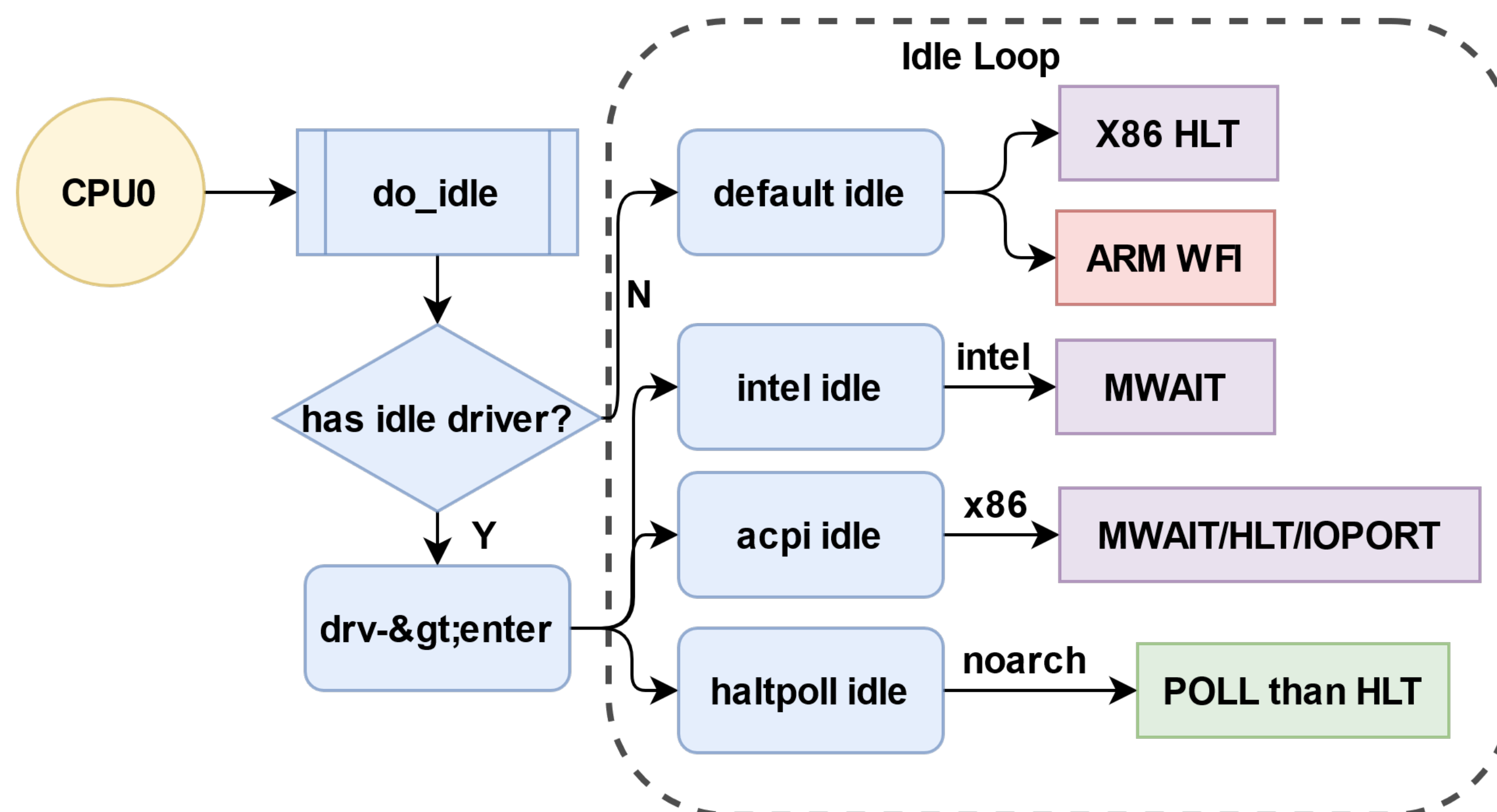
# AMD虚拟机idle调度优化——idle driver介绍

Default idle的实现满足不了越来越复杂的场景，因此内核提供了不同的idle driver：

**1. Intel idle**：专门为Intel CPU实现的idle driver，idle时会执行MWAIT

**2. ACPI idle**：根据ACPI表描述的信息，判断以MWAIT/HLT/IOPORT的其中某种方式进入低功耗状态

**3. Haltpoll idle**：  
guest在执行HLT之前会poll一段时间。



# AMD虚拟机idle调度优化——idle driver介绍

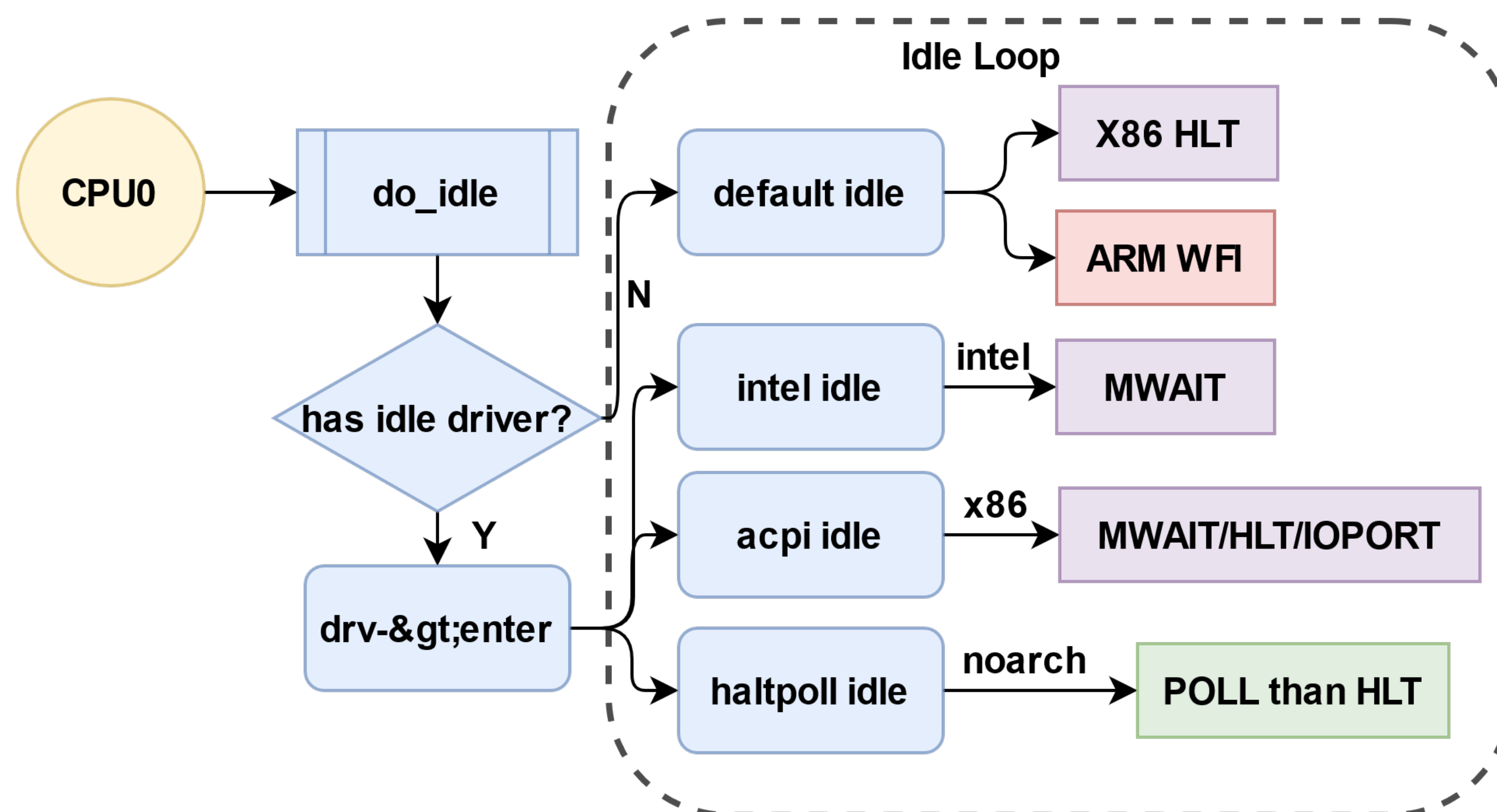
Default idle的实现满足不了越来越复杂的场景，因此内核提供了不同的idle driver：

**1. Intel idle**：专门为Intel CPU实现的idle driver，idle时会执行MWAIT

**2. ACPI idle**：根据ACPI表描述的信息，判断以MWAIT/HLT/IOPORT的其中某种方式进入低功耗状态

**3. Haltpoll idle**：  
guest在执行HLT之前会poll一段时间。

AMD使用的是哪个  
idle driver呢？



# AMD虚拟机idle调度优化——AMD idle driver现状

通过查看Linux提供的sysfs信息可以获取到当前使用的idle driver。

AMD物理机上显示的是acpi idle：

```
#cat /sys/devices/system/cpu/cpuidle/current_driver  
acpi_idle
```

虚拟机内显示的却是none（也就是default idle）：

```
[root@AliYun ~]# cat /sys/devices/system/cpu/cpuidle/current_driver  
none
```



# AMD虚拟机idle调度优化——AMD idle driver现状

通过查看Linux提供的sysfs信息可以获取到当前使用的idle driver。

AMD物理机上显示的是acpi idle：

```
#cat /sys/devices/system/cpu/cpuidle/current_driver  
acpi_idle
```

为什么**AMD**的虚拟机  
没有用**acpi idle**？

虚拟机内显示的却是none（也就是default idle）：

```
[root@AliYun ~]# cat /sys/devices/system/cpu/cpuidle/current_driver  
none
```



# AMD虚拟机idle调度优化——AMD idle driver现状

ACPI idle驱动依赖ACPI表的\_CST objects提供的C-States信息

根据\_CST提供的信息，当CPU在做idle时，会进入到特定的C-States。C-States的级别越高，能耗越低。正常运行时CPU在C0状态，idle时会进入到C1~Cn中的某个状态，支持多少种C-States由硬件实现决定。Linux内核也称之为idle states

然而，AMD虚拟机的ACPI表是  
( **Qemu模拟** ) 没有提供\_CST信息

```
static int acpi_processor_get_cstate_info(struct acpi_processor *pr)
{
>-----unsigned int i;
>-----int result;

>-----/* NOTE: the idle thread may not be running while calling
>----- * this function */

>-----/* Zero initialize all the C-states info. */
>-----memset(pr->power.states, 0, sizeof(pr->power.states));

>-----result = acpi_processor_get_power_info_cst(pr);
>-----if (result == -ENODEV)
>----->-----result = acpi_processor_get_power_info_fadt(pr);

>-----if (result)
>----->-----return result;
```

# AMD虚拟机idle调度优化——AMD idle driver现状

Qemu社区确实有提过对\_CST的支持，但是没有继续推进：

<https://lists.gnu.org/archive/html/qemu-devel/2018-08/msg01267.html>

**From:** Igor Mammedov

**Subject:** [Qemu-devel] [RFC PATCH 0/4] "pc: acpi: \_CST support"

**Date:** Wed, 8 Aug 2018 17:15:45 +0200

---

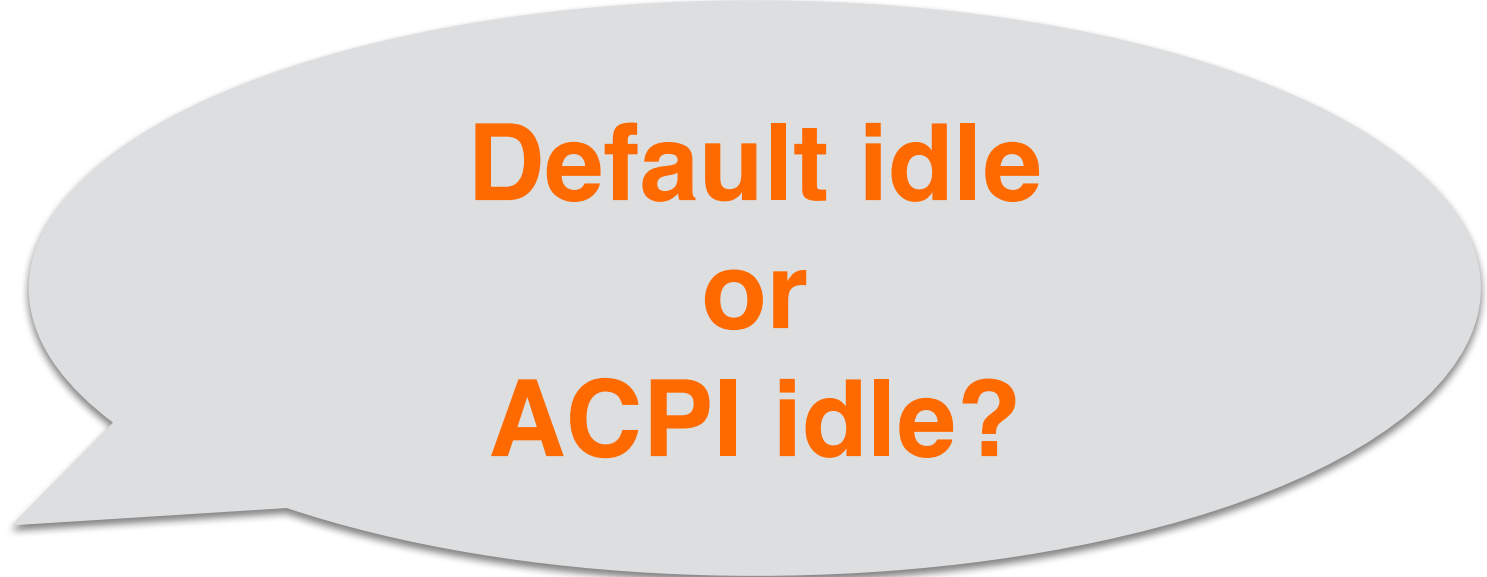
It's an alternative approach to

1) [PATCH hack dontapply v2 0/7] Dynamic \_CST generation which instead of dynamic AML loading uses static AML with dynamic values. It allows us to keep firmware blob static and to avoid split firmware issue (1) in case of cross version migration.

ABI in this case is confined to cpu hotplug IO registers (i.e. do it old school way, like we used to do so far). This way we don't have to add yet another ABI to keep dynamic AML code under control (1).

Tested with: XPsp3 - ws2106 guests.

CC: "Michael S. Tsirkin" <address@hidden>



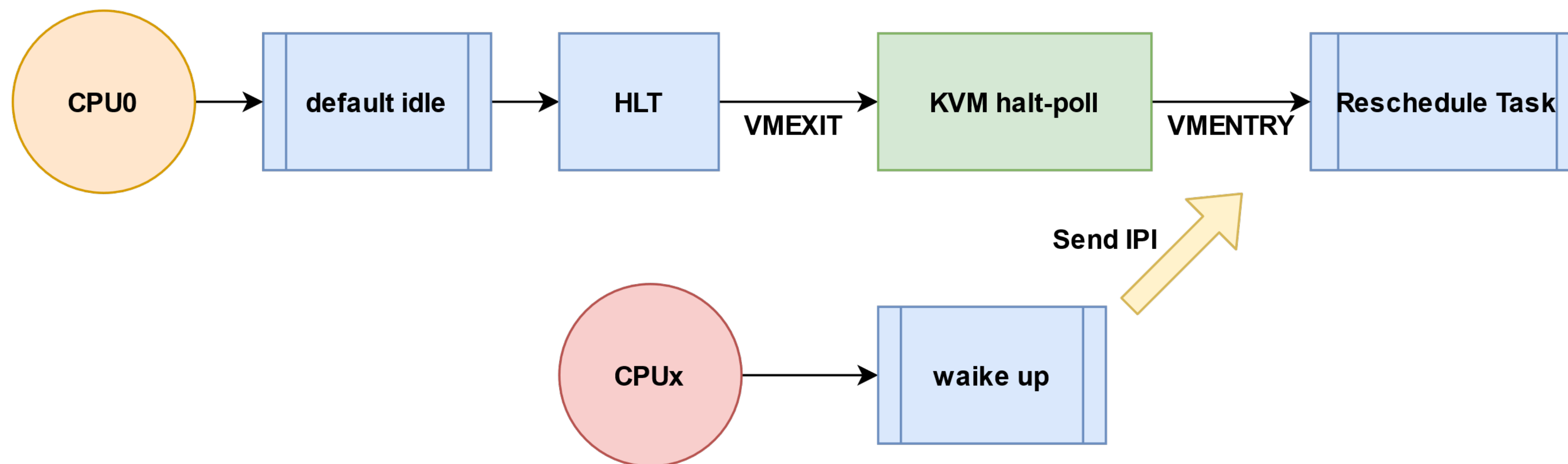
Default idle  
or  
ACPI idle?

# AMD虚拟机idle调度优化——default idle性能问题

使用default idle，AMD CPU idle时会执行HLT指令，然后等待下一次调度

当有新的task需要被调度到该CPU时，其他CPU会发送IPI中断，提醒CPU重新进入调度

**会产生大量的Reschedule IPI ( RES IPI )，虚拟化场景下，导致vCPU频繁退出，增加虚拟化层的开销**



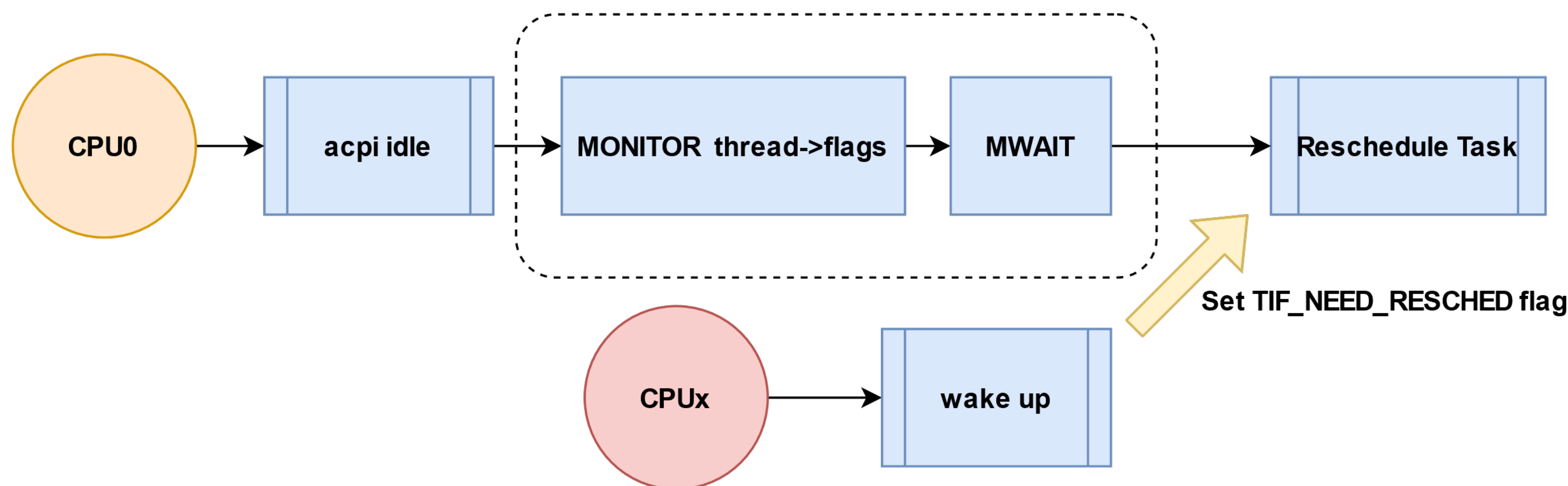
# AMD虚拟机idle调度优化——acpi idle性能优化

使用acpi idle，需要呈现虚拟的ACPI \_CST objects给AMD虚拟机，以及：

1. \_CST objects中设置以MWAIT方式进入C1 State
2. 清除对MONITOR和MWAIT指令的捕获

vCPU在idle时，使用MONITOR监控当前线程flags是否改变，执行MWAIT等待flags变更  
新的task需要调度时，remote vCPU不再发送IPI中断，而是设置线程flags为NEED\_RESCHEDED

**idle调度过程不再需要虚拟化层参与，完全避免了模拟HLT指令和RES IPI中断的开销**

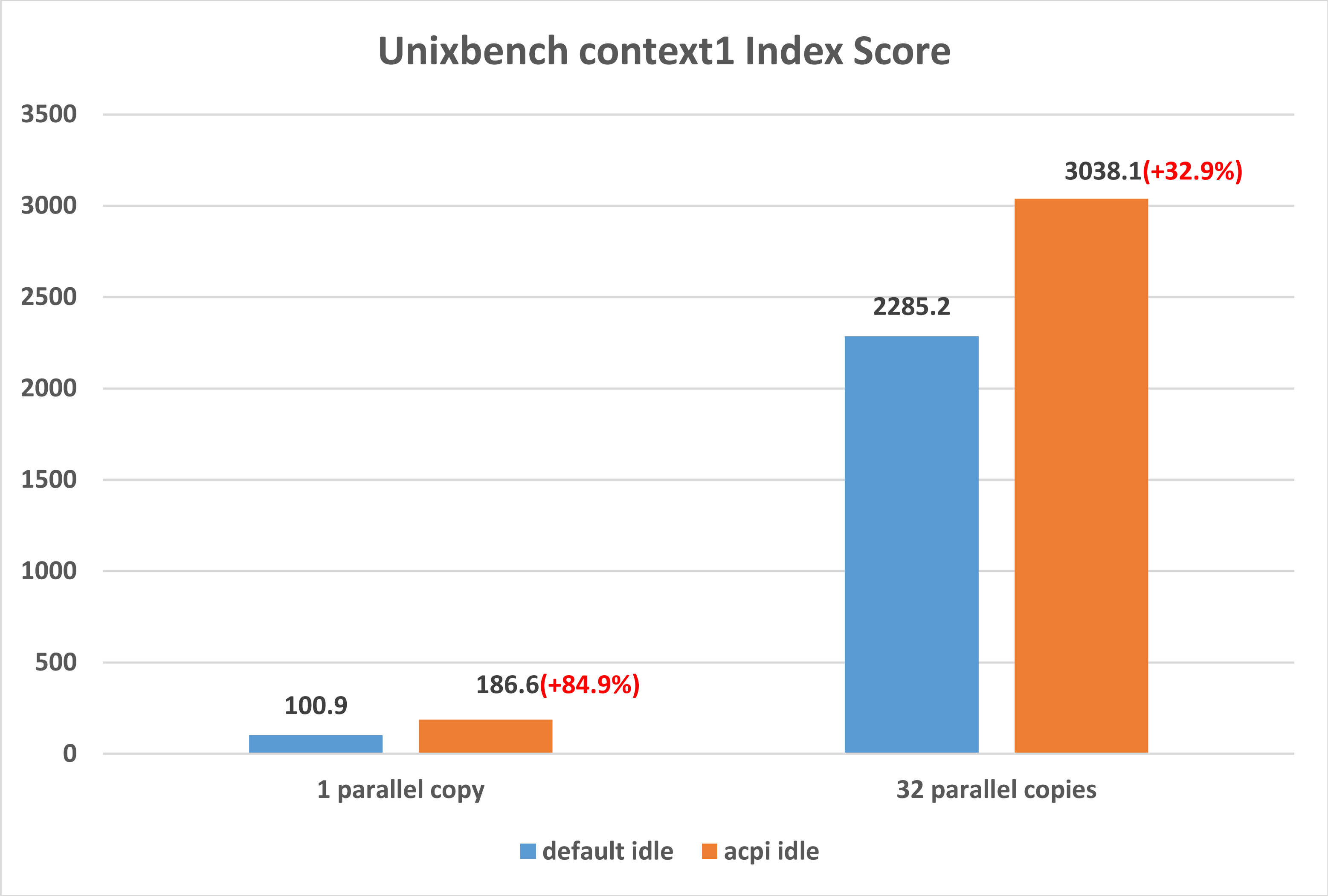


# AMD虚拟机idle调度优化——Unixbench context1测试对比



## AMD虚拟机配置:

- 1. 32 CPUs, 64GB RAM
- 2. GuestOS: CentOS 8.3



# Overview

- AMD虚拟机idle调度优化
- **AMD虚拟机支持TLBi**
- AMD虚拟机使能AVIC

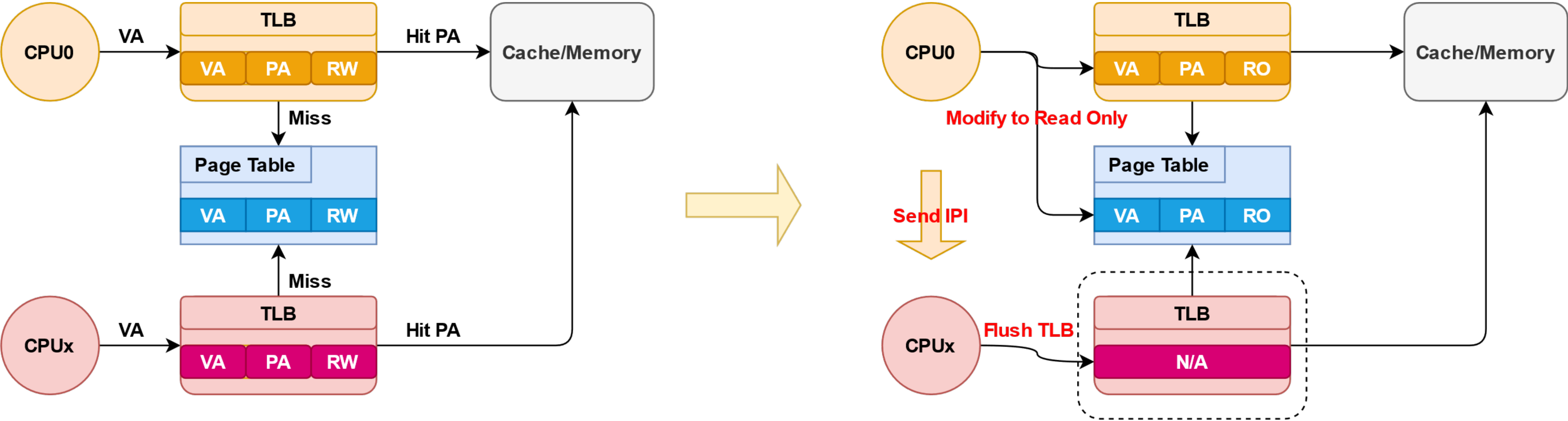


# AMD虚拟机支持TLBi——什么是TLB shutdown

TLB对虚拟地址到物理地址的翻译进行了缓存，减少CPU访存时对页表的遍历。

在SMP系统中，多个CPU会共享同一个页表，但是每个CPU都会拥有独立的TLB。

当其中一个CPU修改了页表（如释放页表、mprotect设置只读等），此时保存在其它CPU的TLB中的翻译就“过时”了，需要通知他们将过时的TLB表项invalidate，这个过程称为TLB shutdown。



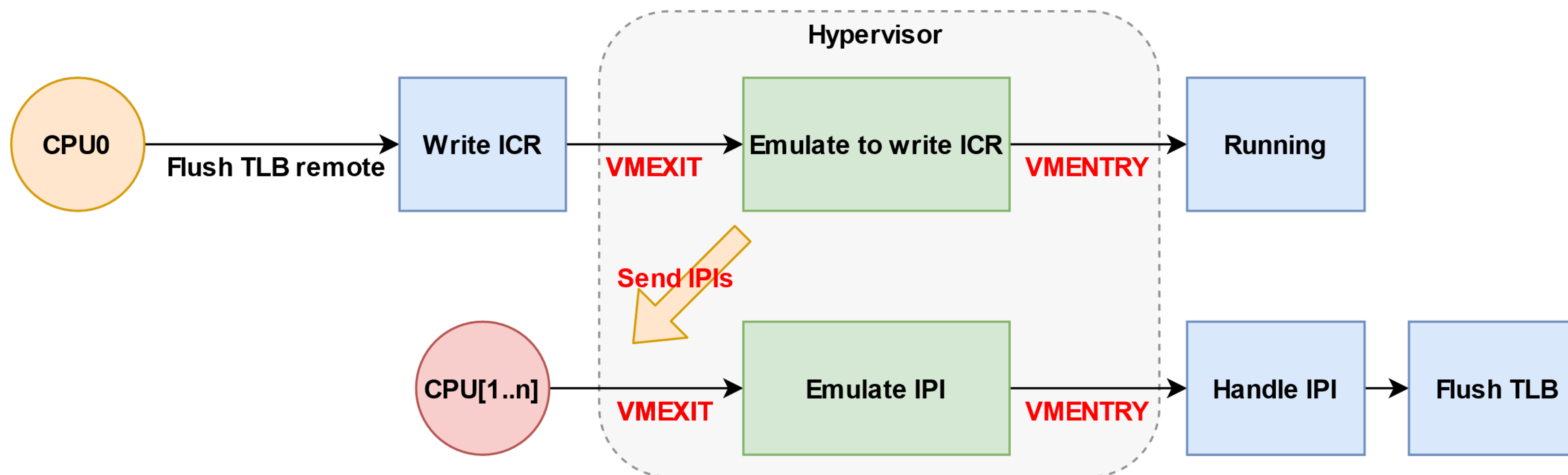


# AMD虚拟机支持TLBi——虚拟化TLB shutdown

不同的CPU架构平台对TLB shutdown有不同的实现。

x86平台通过**发送TLB IPI**来通知其他CPU来刷新TLB；ARM平台通过调用TLBI指令来刷新TLB。

每次发送IPI中断都会导致**发送端和接收端vCPU频繁退出**到Hypervisor进行模拟，极大地增加了虚拟化层的开销。



# AMD虚拟机支持TLBi——业界的一些优化手段

## 虚拟化层对基于IPI方式的TLB shutdown的优化：

### 发送端的优化：

1. IPI Fastpath: <https://patchwork.kernel.org/patch/11255189/>
2. Intel IPIv: <https://lwn.net/Articles/863190/>
3. AMD AVIC硬件加速

### 接收端优化：

1. 基于Intel Posted Interrupt的IPI中断直通
2. AMD AVIC硬件加速

### PV优化：

1. PV IPI: <https://lwn.net/Articles/740363/>
2. PV TLB shutdown: <https://lwn.net/Articles/740363/>
3. Passthrough IPI: <https://www.spinics.net/lists/kvm/msg224093.html>

# AMD虚拟机支持TLBi——业界的一些优化手段

## 虚拟化层对基于IPI方式的TLB shutdown的优化：

### 发送端的优化：

1. IPI Fastpath: <https://patchwork.kernel.org/patch/11255189/>
2. Intel IPIv: <https://lwn.net/Articles/863190/>
3. AMD AVIC硬件加速

### 接收端优化：

1. 基于Intel Posted Interrupt的IPI中断直通
2. AMD AVIC硬件加速

### PV优化：

1. PV IPI: <https://lwn.net/Articles/740363/>
2. PV TLB shutdown: <https://lwn.net/Articles/740363/>
3. Passthrough IPI: <https://www.spinics.net/lists/kvm/msg224093.html>

是否存在类似ARM的  
TLBi指令，而不再需要  
IPI来进行remote  
TLB Flush？

# AMD虚拟机支持TLBi——INVLPGB/TLBSYNC

对比ARM的TLBi，AMD也有了类似技术——INVLPGB/TLBSYNC

INVLPGB ( Invalidate TLB Entries with Broadcast ) 通知其他CPU刷新TLB；  
TLBSYNC ( Synchronize TLB Invalidations ) 等待其他CPU完成刷新TLB。

INVLPGB通过RAX和EDX来传递参数，对指定范围的TLB进行刷新。

rAX	Attributes
0	Valid VA
1	Valid PCID
2	Valid ASID
3	Include Global
4	Final Translation Only
5	Include Nested Translations
11:6	Reserved, MBZ
63:12 or 31:12	VA

EDX	Attributes
15:0	ASID
27:16	PCID
31:28	Reserved, MBZ

# AMD虚拟机支持TLBi——INVLPGB/TLBSYNC

## INVLPGB/TLBSYNC社区状态：


LLVM编译器：<https://reviews.llvm.org/D94134>

Linux还没有任何动态，但是有提到过：<https://lkml.org/lkml/2020/12/5/428>

</> Diffusion > LLVM Github Monorepo > 9386483b7142


### [X86] Add TLBSYNC, INVLPGB and SNP instructions

9386483b7142

 Authored by **GGanesh** on Jan 8 2021, 8:40 AM.

#### Description

[X86] Add TLBSYNC, INVLPGB and SNP instructions

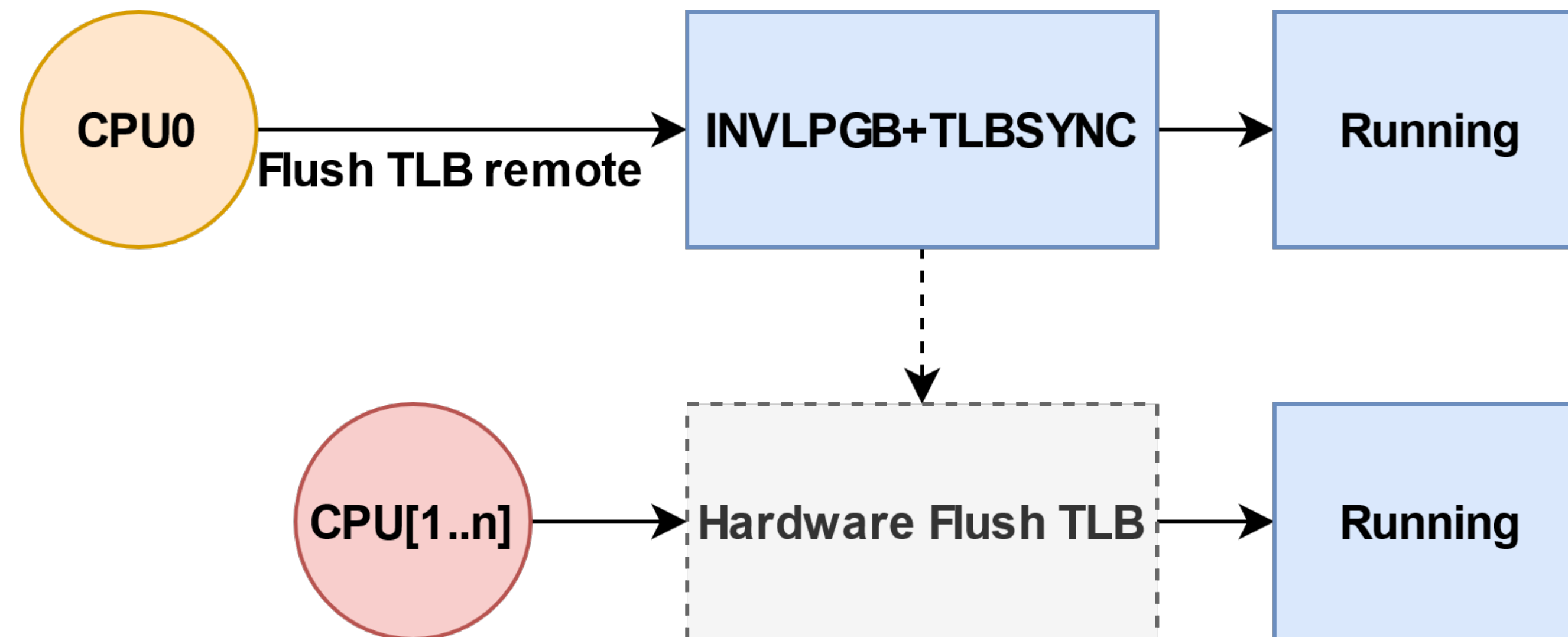
Differential Revision:  <https://reviews.llvm.org/D94134>

# AMD虚拟机支持TLBi——虚拟机支持INVLPGB/TLBSYNC

## 如何在虚拟机中支持INVLPGB/TLBSYNC指令？

1. 将虚拟机的CUID 8000\_000A\_EDX[24]置为1
2. 清除对INVLPGB/TLBSYNC指令的捕获
3. 修改内核，替换发送IPI的方式为调用INVLPGB/TLBSYNC指令

**vCPU在做TLB shutdown时，发送端和接收端都不再退出，TLB的invalidation过程完全由硬件完成。**

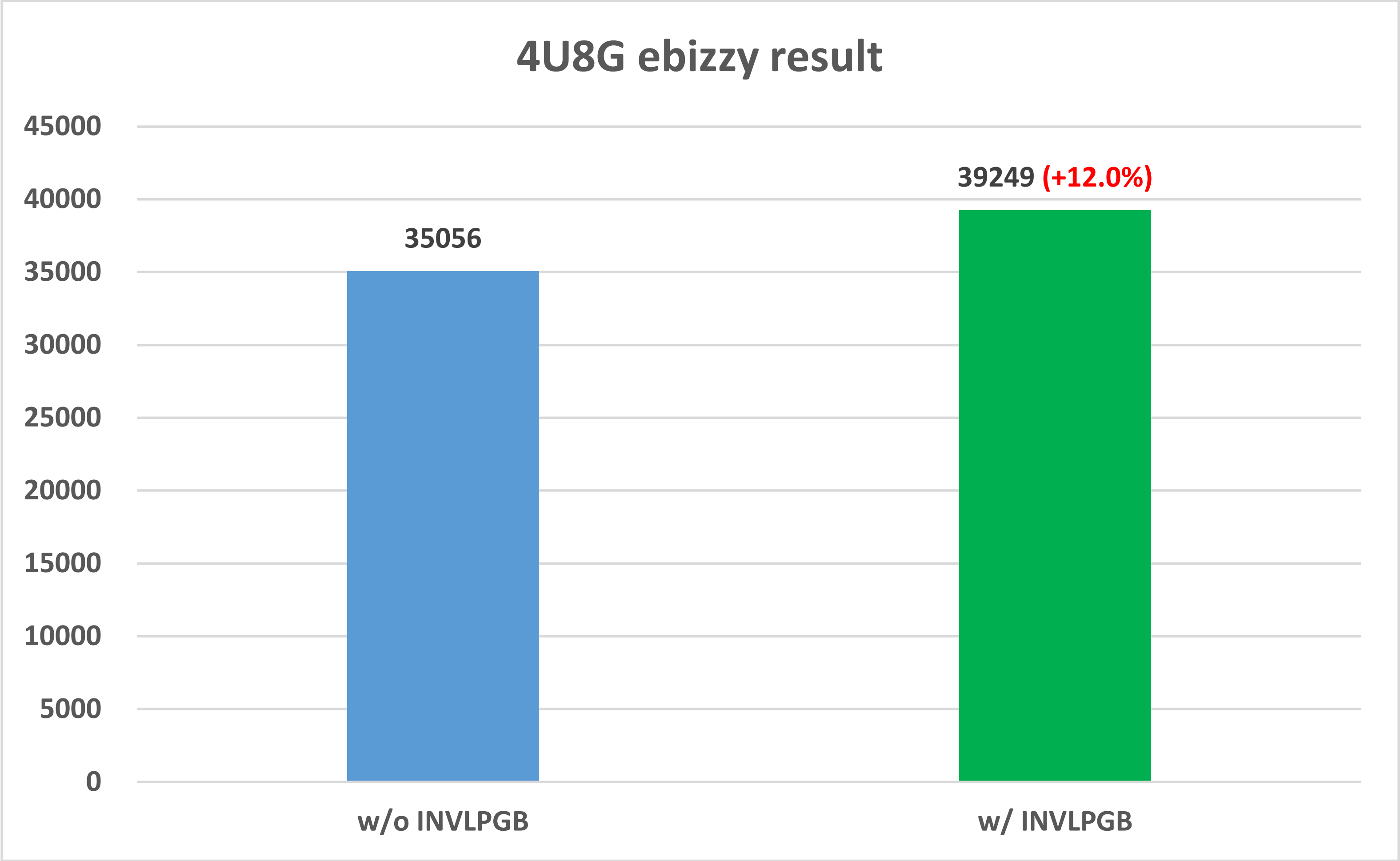




# AMD虚拟机支持TLBi——ebizzy测试结果对比

## AMD虚拟机配置:

- 1. 4 CPUs, 8GB RAM
- 2. GuestOS: CentOS 8.3





# Overview

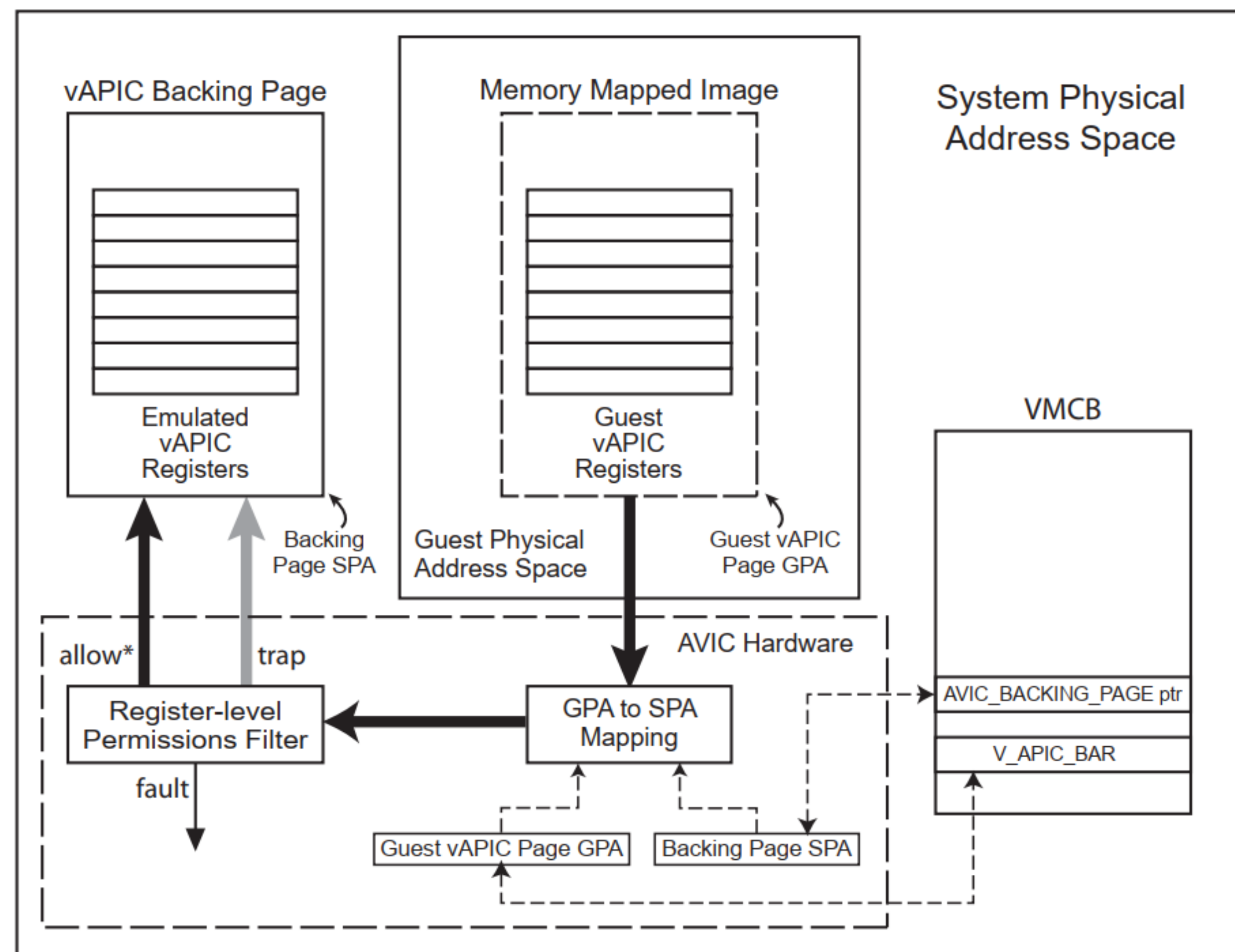
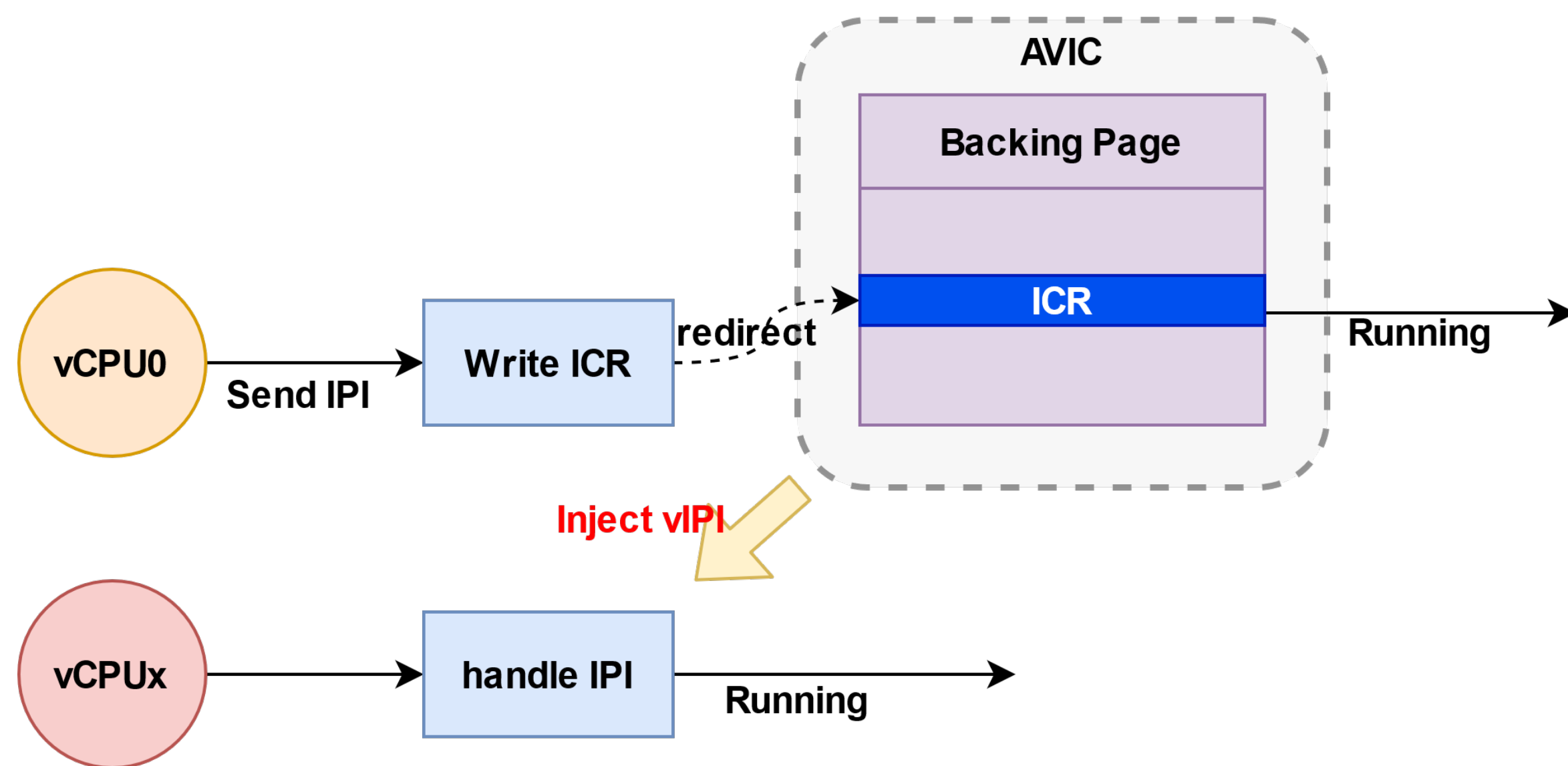
- AMD虚拟机idle调度优化
- AMD虚拟机支持TLBi
- **AMD虚拟机使能AVIC**

# AMD虚拟机使能AVIC——AVIC简介

从RES IPI到TLB IPI，可见虚拟IPI中断的优化确实重要。

AMD提供了AVIC ( Advanced Virtual Interrupt Controller ) 硬件辅助功能。

基于**Backing Page**的机制，AMD的VM拥有一份虚拟的、与Host隔离的硬件vAPIC寄存器。vCPU在发送和接收IPI时，**完全由硬件完成**。

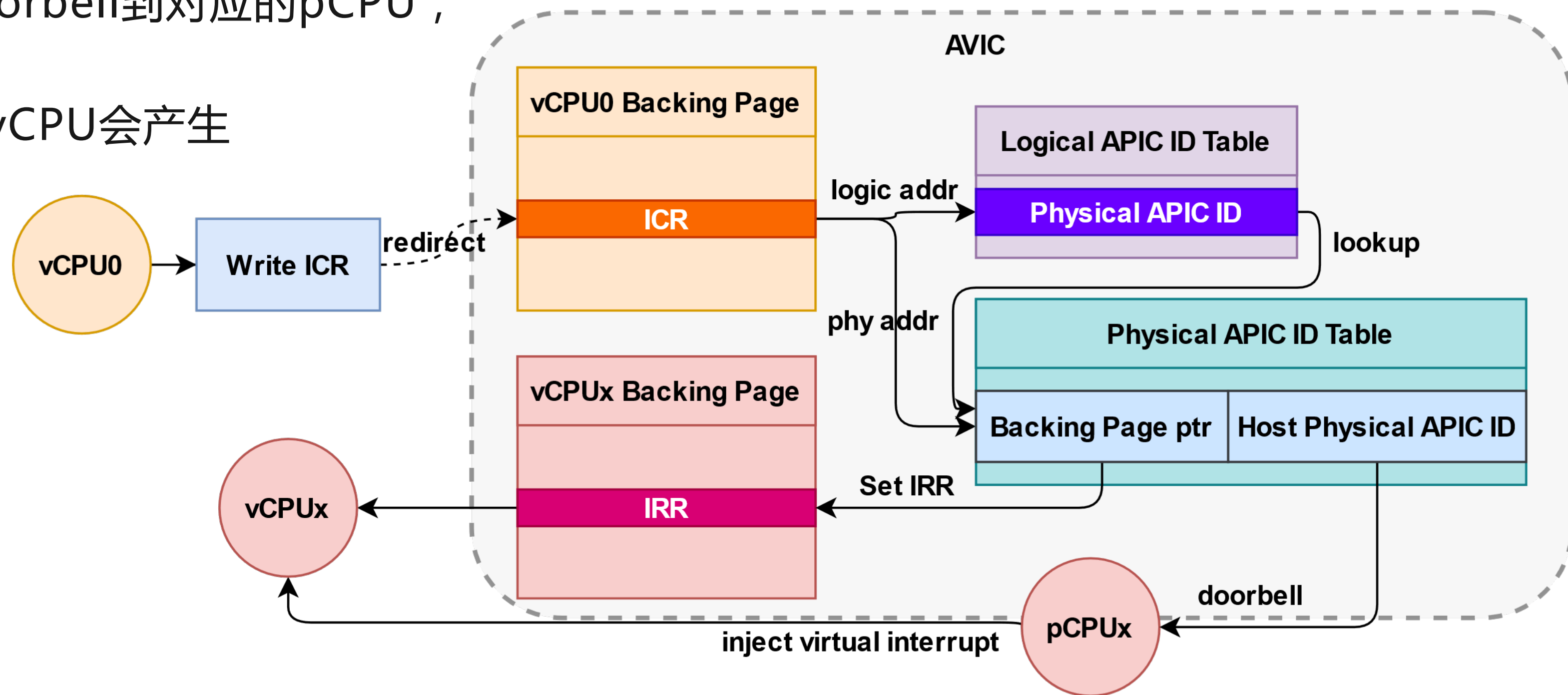


# AMD虚拟机使能AVIC——AVIC简介

vCPU对vAPIC寄存器的访问，都会被重定向到Backing Page（被AVIC捕获）。  
如Guest写ICR寄存器时，AVIC会自动将值写到Backing Page所在的物理内存，同时触发IPI中断。

IPI中断投递到目标vCPU时，根据ICR的内容，做对应的处理，比如：

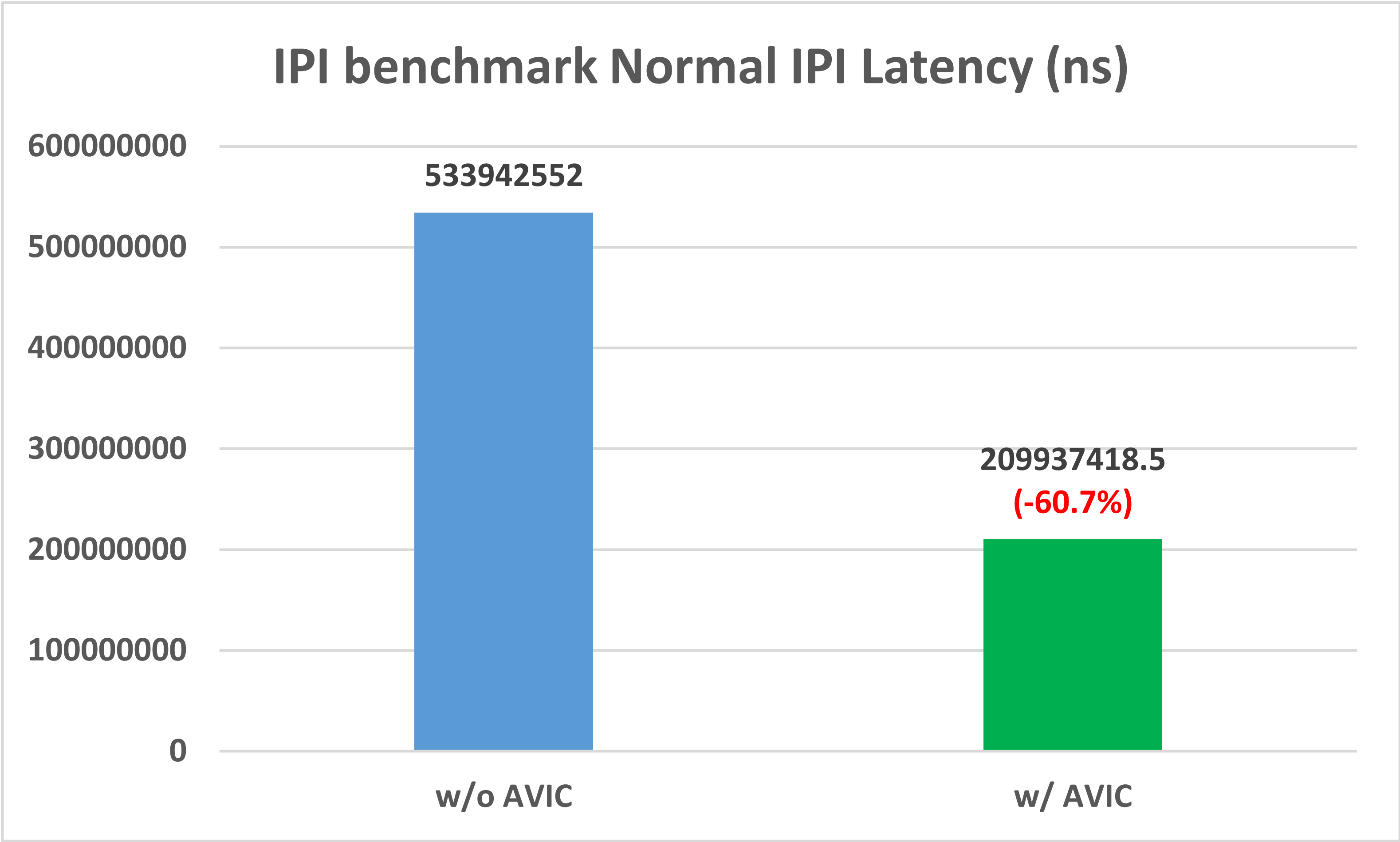
1. 如果destination用的是Physical APIC ID，就会通过Physical APIC table找到对应vCPU的backing page，并将backing page中的IRR置上。
2. 如果目标vCPU在running，发送doorbell到对应的pCPU，通知有中断需要处理；  
如果目标vCPU不在running，当前vCPU会产生AVIC\_INCOMPLETE\_IPI的VMEXIT



# AMD虚拟机使能AVIC——IPI benchmark测试结果对比

## AMD虚拟机配置:

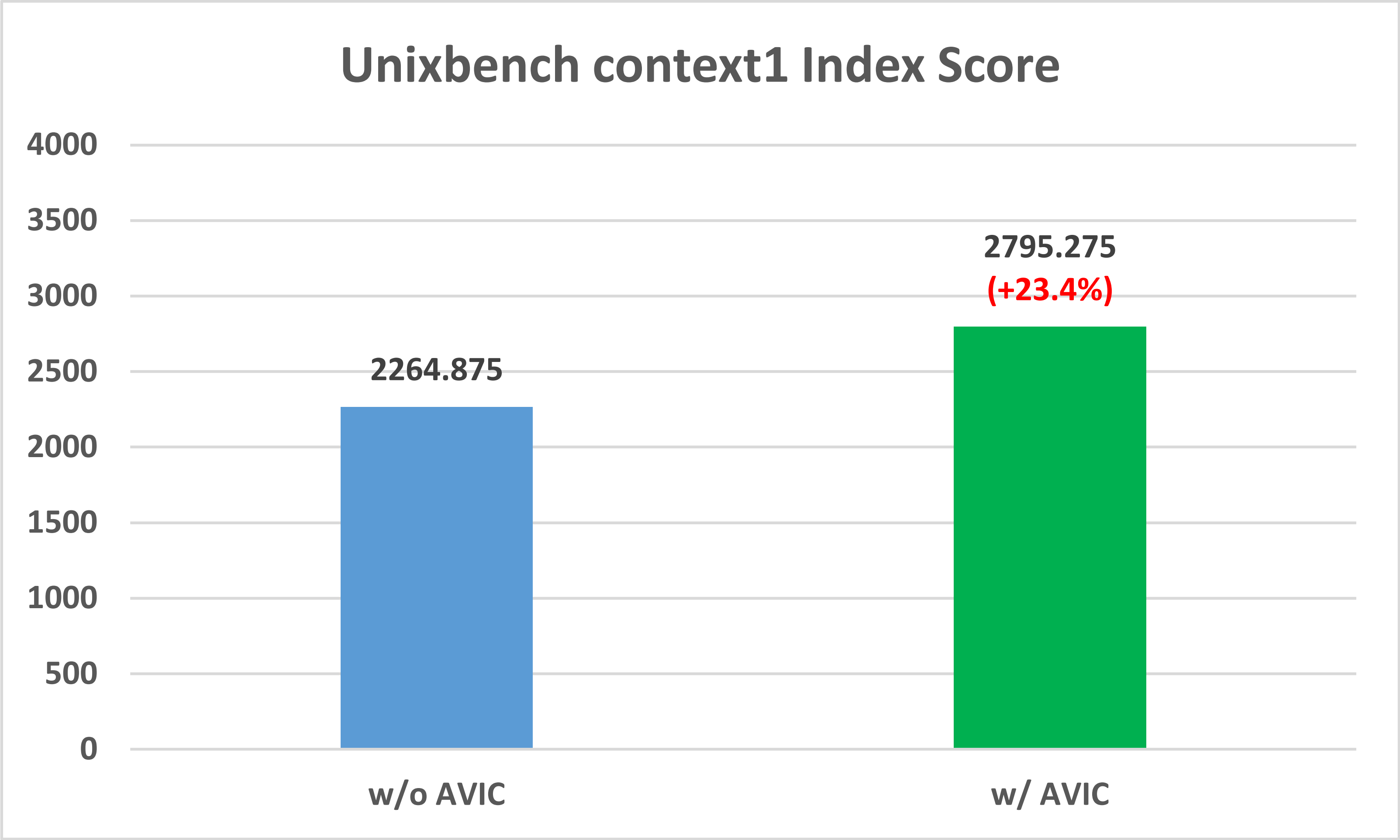
- 1. 32 CPUs, 64GB RAM
- 2. GuestOS: CentOS 8.3



# AMD虚拟机使能AVIC——Unixbench context1测试对比

## AMD虚拟机配置:

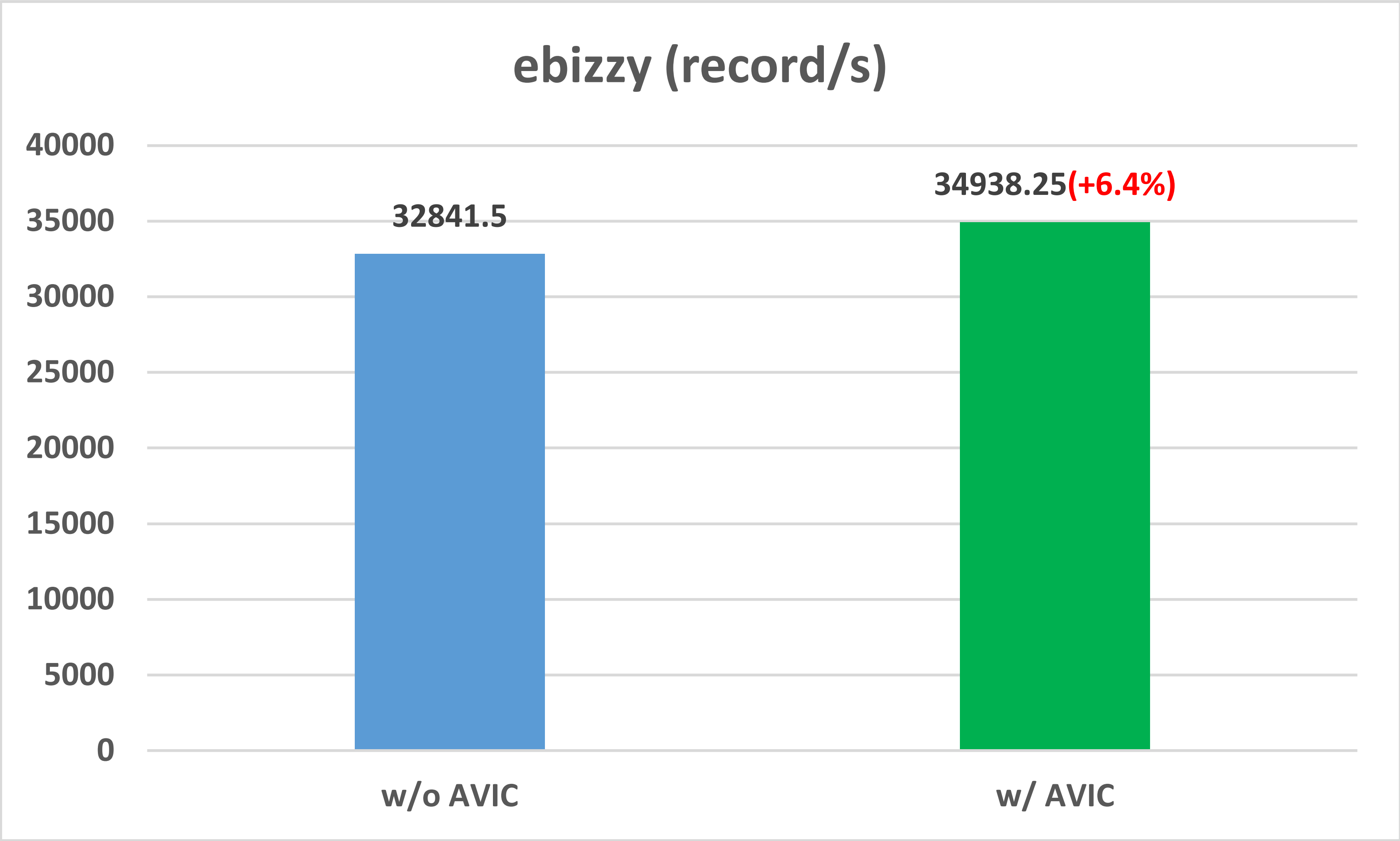
- 1. 32 CPUs, 64GB RAM
- 2. GuestOS: CentOS 8.3



# AMD虚拟机使能AVIC——ebizzy测试对比

## AMD虚拟机配置:

- 1. 32 CPUs, 64GB RAM
- 2. GuestOS: CentOS 8.3



# 总结

- 尽管AMD与Intel同为x86架构体系，可以看到，在具体的硬件和软件的实现上还是有不少差别。比如上述提到的AMD使用acpi\_idle和AVIC，而Intel使用的是intel\_idle和Posted Interrupt。
- 因此，对于AMD虚拟机的性能仍然有很多可以挖掘的优化点，也期望AMD在未来能有更多差异化的竞争点。



**Q&A**  
**Thank you!**



奥运会全球指定云服务商