针对x86混合架构优化软件

白皮书

2021 10月

修订版1.0

文件编号：348851-001US

注意：本文件包含开发设计阶段的产品信息。此处信息如有更改，恕不另行通知。不要使用此信息完成设计。

性能因使用、配置和其他因素而异。有关详细信息，请访问www.Intel。com/PerformanceIndex。

性能结果基于截至配置中所示日期的测试，可能无法反映所有公开可用的更新。有关配置详细信息，请参阅备份。没有任何产品或组件是绝对安全的。

所有产品计划和路线图如有更改，恕不另行通知。

英特尔使用代码名来识别正在开发但尚未公开的产品、技术或服务。这些名称不是"商业"名称，也不打算用作商标。

您的成本和结果可能会有所不同。

英特尔技术可能需要启用硬件、软件或服务激活。

您不得将本文档用于或促进与本文所述英特尔产品相关的任何侵权或其他法律分析。您同意向英特尔授予此后起草的任何专利申请（包括本文披露的主题）的非独占、免版税许可。

本文件未授予任何知识产权许可（明示或默示、禁止反悔或其他方式）。所描述的产品可能包含称为勘误表的设计缺陷或错误，这可能导致产品偏离发布的规范。当前特征勘误表可根据要求提供。

所描述的产品可能包含称为勘误表的设计缺陷或错误，这可能导致产品偏离发布的规范。当前特征勘误表可根据要求提供。

英特尔否认所有明示和默示保证，包括但不限于适销性、特定用途适用性和非侵权的默示保证，以及因履约过程、交易过程或贸易使用而产生的任何保证。

英特尔不控制或审计第三方基准数据或本文档中引用的网站。您应该访问引用的网站并确认引用的数据是否准确。

目录

修订历史记录

修订号	描述	日期
1.0	• 文件的首次发布。	2021 10月

引言

本技术文档提供了有关为英特尔®核心优化软件的信息™ 处理器
支持混合架构。

该文档概述了x86混合体系结构、与Windows的混合核心使用，并详细介绍了软件应用程序和驱动程序如何确保最佳核心使用。

本文还介绍了可在支持x86混合体系结构的英特尔核心处理器上使用的关键Windows处理器电源管理设置（PPM设置），以满足系统性能与功耗目标。

1 X86混合架构概述

1.1 12真实航向 第二代INTEL® 核心™ 支持性能混合架构的处理器

12第支持性能混合架构的第二代Intel Core处理器由多达八个性能核（P核）和八个高效核（E核）组成。这些处理器还包括每个IDI模块的3MB末级缓存（LLC），其中一个模块是一个P核或四个E核。它具有对称ISA，有多种配置。

P核提供单线程或有限线程性能，而E核有助于提高可扩展性和多线程效率。这些处理器上的P核还可以启用Intel®超线程技术。当操作系统决定在所有处理器上调度时，所有内核可以同时处于活动状态。

实现混合的关键OSV需求是性能混合架构中不同核心类型的对称ISA。在12第二代Intel Core处理器支持性能混合架构，ISA收敛到P核和E核之间的公共基线。

1.2 11真实航向 第二代INTEL® 核心™ 支持混合架构的处理器

11第支持混合架构的第二代Intel Core处理器由一个P核和一个四个E核群集组成，通过共享4MB LLC连接到CCF。E核模块和P核通过两个IDI通道连接到CCF和共享4MB LLC。LLC包括在内，所有核都完全一致。

P-core是禁用Intel超线程技术的单物理核。操作系统枚举并查看五个物理核。当操作系统决定调度所有五个内核时，所有内核可以同时处于活动状态。

实现混合的关键OSV需求是混合架构中不同核心类型的对称ISA。

下表提供了有关11的其他详细信息第支持混合架构的第二代Intel核心处理器。

	细节
平台目标	• 新的双屏设备类别和体验。 • 移动外形和电池寿命。 • 工作负载：光生产率、浏览、媒体消耗、连接、双屏。
混合配置和ISA	• 四个E核+一个P核，4M LLC，在P核上禁用SMT。 • 收敛到统一/公共ISA。 • 将混合ISA（Perfmon，MCA）公开给调试/性能调优工具。
混合调度	• 向操作系统公开所有核心（性能和效率）。 • 并行P核和E核执行和调度。 • 使用新的Windows混合调度来适应IA混合（即硬件引导调度）。

2 混合调度

2.1 硬件引导调度

通过硬件引导调度，硬件以以下形式向操作系统（也称为硬件反馈接口）提供动态反馈：

- 基于功率/热极限的P核和E核的动态性能和能效能力。
- 功率和热量受限时的空转提示。

在支持混合架构的处理器中，所有内核都暴露于操作系统。操作系统调度程序负责确定应在哪种内核类型上调度哪些软件线程。

硬件支持通过CPUID指令枚举，并由操作系统通过写入配置MSR来启用。

相关CPUID：

- CPUID[6]。EAX[19]–表示支持硬件反馈。
- CPUID[6]。EDX[7:0]–支持功能的位图。
- CPUID[6]。EDX[11:8]——HGS表的尺寸。
- CPUID[6]。EDX[31:16]–HGS表中逻辑处理器行的索引。配

置MSR：

- IA32\u HW\u FEEDBACK\u CONFIG MSR（17D1H）
  -位0–启用HGS。

有关这项技术的更多详细信息，请参阅以下“英特尔64和IA-32体系结构软件开发人员手册”：www.Intel。通信/传感和诊断模块。

2.2 英特尔® 线程控制器

使用Intel Thread Director，硬件基于各种IPC性能特征，以以下形式向每个线程的操作系统提供运行时反馈（即增强的硬件反馈）：

- 基于功率/热极限的P核和E核的动态性能和能效能力。
- 功率和热量受限时的空转提示。

Intel Thread Director在支持从12开始的性能混合架构的Intel Core处理器上可用第第二代Intel Core处理器系列，帮助操作系统为正确的线程选择正确的内核。

线程特定的硬件支持通过CPUID指令枚举，并由操作系统通过写入配置MSR来启用。

相关CPUID：

- CPUID[6]。EAX[23]–表示支持Intel增强硬件反馈。
- CPUID[6]。ECX[11:8]–英特尔线程控制器类的数量。

- CPUID[0x20，ECX=0][EBX[0]]–HRESET指令支持。配置

MSR：

- IA32\u HW\u FEEDBACK\u CONFIG MSR（17D1H）
  - 位1–启用“英特尔线程控制器”多类支持。
- IA32\u HW\u FEEDBACK\u THREAD\u CONFIG MSR（17D4H）
  - 位1–启用“英特尔线程控制器”多类支持。

有关这项技术的更多详细信息，请参阅以下“英特尔64和IA-32体系结构软件开发人员手册”：www.Intel。通信/传感和诊断模块。

2.3 使用INTEL进行调度® 支持X86混合架构的处理器上支持超线程技术

在硬件同级超线程繁忙的情况下，E核的设计可以提供比逻辑P核更好的性能。

2.4 具有多E核模块的调度

空闲模块中的E核有助于提供比繁忙模块中的E核更好的性能。

2.5 在X86混合体系结构上调度后台线程

在大多数情况下，后台线程可以利用E-Core的可扩展性和多线程效率。

3 WINDOWS的主要功能和性能

3.1 线程调度概述

支持x86混合架构的处理器根据其在硬件操作系统共享内存区域中列举的性能和效率能力进行分类。该内存区域由操作系统设置并枚举到硬件，在本文档中称为硬件反馈内存表。然后，内存区域由硬件填充，并在更新时通知操作系统。

支持x86混合架构的处理器上的调度是QoS，然后是优先级驱动的，本质上是抢占式的。在大多数情况下，在调度约束范围内，最高的QoS和优先级线程可以在性能最高的内核上运行。

开发人员还可以选择线程以高效的频率在高效的内核上运行，以优化功率和性能。此操作是通过使用QoS API来定义线程的功率调节来完成的。（API详情见第6.4节。）

硬处理器亲和力可能会扰乱操作系统决策。虽然亲和力可能用于引导线程实现性能或效率，但给定的性能/效率是动态功能，而不是基于核心类型。因此，硬亲和力可能不会产生预期的结果。利用功率调节有助于引导工作朝着正确的核心方向发展，以实现性能/效率。如果绝对需要亲和力，则可以利用CPUSets API。（详见第6.3节。）

在12第支持性能混合架构的第二代Intel核心处理器，硬件还向Windows提供线程类信息，以帮助在处理器上放置性能更高的线程。此提示用于相同或更低的QoS/优先级线程。

参见第5节中的具体调度示例。

3.2 WINDOWS CORE驻车引擎概述

通常，Windows Core Parking Engine会对工作负载做出全局可扩展性决策，并确定用于执行的最佳计算核心集。在支持x86混合架构的处理器中，它还通过确定最佳的P核和E核集来提供帮助。

性能和效率来自硬件反馈内存表。在performant Core中，性能最高的Core首先被解列。在高效内核中，最高效的内核首先被解列。

支持x86混合架构的处理器上可以使用两种高级停车配置：

- 第一，停车决策仅基于性能能力，即标准停车或偏好的核心停车配置。

- 第二，停车决策基于效率和绩效能力；这就是所谓的异驻车配置。

这些配置可以通过HeteroPolicy PPM注册表和相关停车PPM阈值调整进行控制。

请参阅第5节中的具体示例和配置。第7.2节提供了其他PPM详细信息。

3.3 性能状态控制引擎

与同类Intel系统类似，Windows利用Intel®Speed Shift技术（即HWP）指示需要以高效性能水平运行的线程的性能约束，或调整不同滑块位置的能量性能偏好，以满足系统范围内的性能与电池寿命目标。

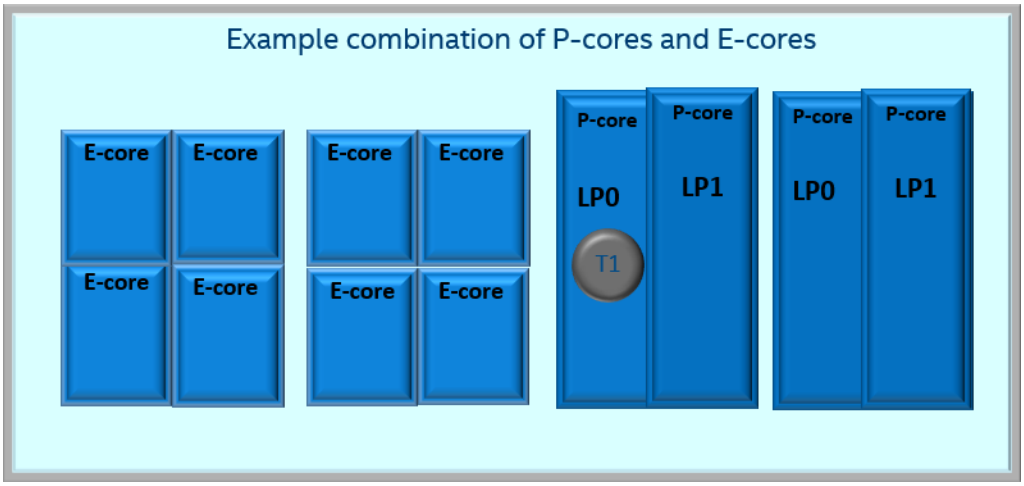
各种PPM设置–EnergyPerfPreference、MinimumPerformance、FrequencyCap、MaxPerformance –由窗口在不同的窗口滑块位置、配置文件中动态更改，以满足不同线程的QoS需求。

请参阅第7.3节中的相关PPM设置详细信息。

4 第12代INTEL® 核心™ 处理器窗口调度/停车示例

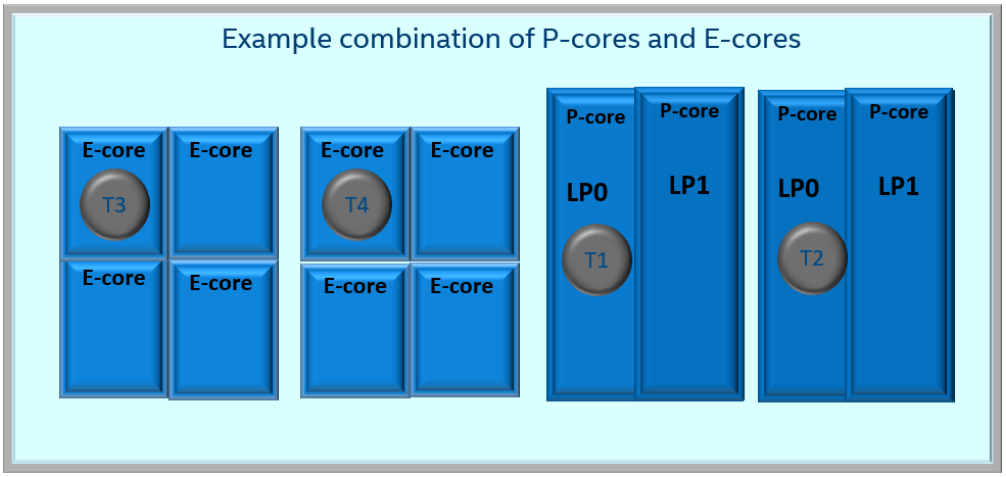
4.1 单线程场景

以下示例显示了Windows利用Intel内核实现单线程性能。当逻辑处理器（LP）0具有最高性能时，动态实现此行为。



4.2 有限线程场景示例1

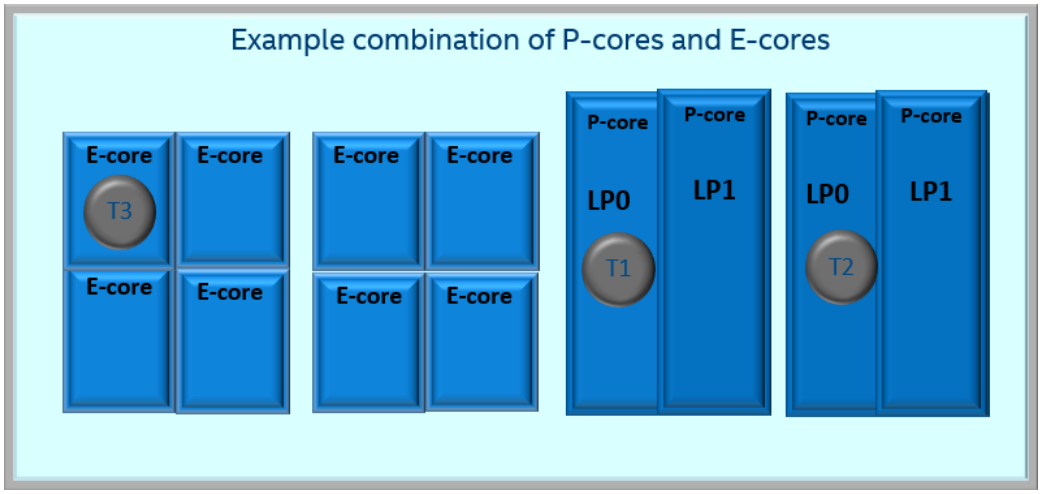
以下示例显示了有限软件线程场景中的调度行为示例。当P核比E核性能更好时，此行为由Windows调度程序/停车引擎动态实现。E核比繁忙核的SMT同级更具性能。当功能动态变化时，Windows会自动对此进行解释，以实现最佳调度。



4.3 有限线程场景示例2

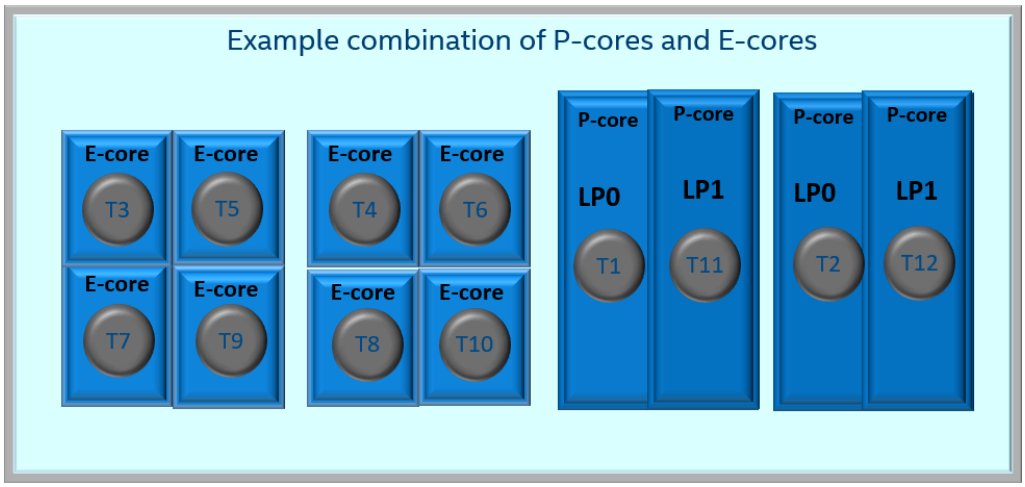
以下示例显示T1和T2放置在P核上，T3放置在E核上。在本例中，T3的核心性能分别低于T1和T2。

当功能动态变化时，Windows会自动考虑这一点以实现最佳调度。



4.4 多线程场景

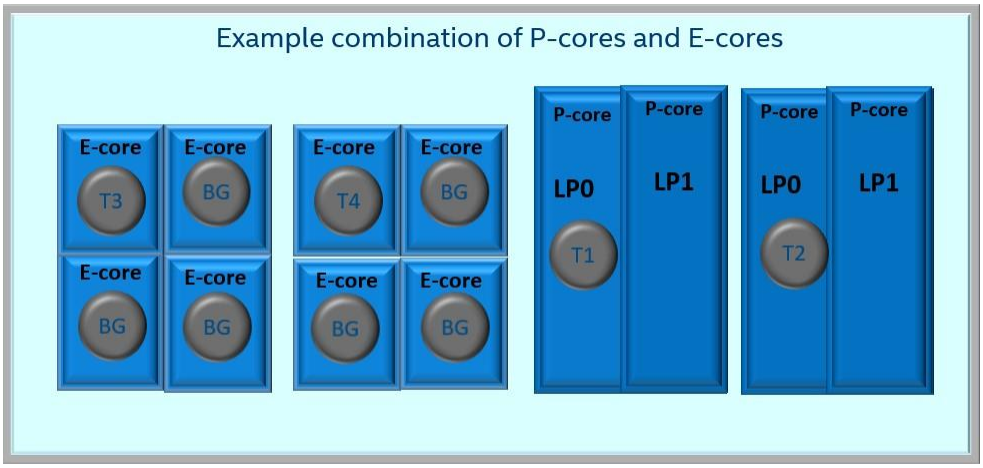
Windows在多线程场景中使用所有内核。



在功率/热约束场景中，有时可能无法使用所有核心来实现最佳系统性能/效率。该行为是通过硬件向Windows提供反馈来动态实现的，Windows会自动对该反馈进行操作。

4.5 后台线程

以下示例显示Windows在高效内核上动态放置背景线程。在本例中，E-core具有最高的多线程效率能力。



此选项可能在某些面向性能的配置上未启用，但可以通过PPM包启用，用于OEM特定的调优。

如果在CurrentScheme上看到PPM设置，则默认情况下启用该行为：

a. *后台->调度策略设置为3或4。*
b. 如果后台->调度策略不存在，则检查默认->调度策略是否存在设置为5。

在某些情况下，当P核空闲时，在P核上运行后台活动可能更为理想。这是通过Windows Core Parking Engine对调度程序提供哪些P核和E核进行优化决策来实现的。

4.6 多媒体线程

与背景线程类似，多媒体线程也动态放置在高效内核上。

此选项可能在某些配置上未启用，可以通过PPM包启用，以进行OEM特定的调整。

如果在CurrentScheme上看到以下PPM设置，则默认情况下启用该行为。

a. *MultiMediaQoS->SchedulingPolicy 设置为3或4。*
b. 如果没有MultiMediaQoS->SchedulingPolicy，则检查默认->SchedulingPolicy是否设置为5。

在某些情况下，当核心不在时，在核心上运行后台活动可能更有效
前台或更高QoS线程需要。

4.7 ECO QOS线程

与后台线程类似，eco线程也动态地放置在高效内核上。

此选项可能在某些配置上未启用，可以通过PPM包启用，以进行OEM特定的调整。

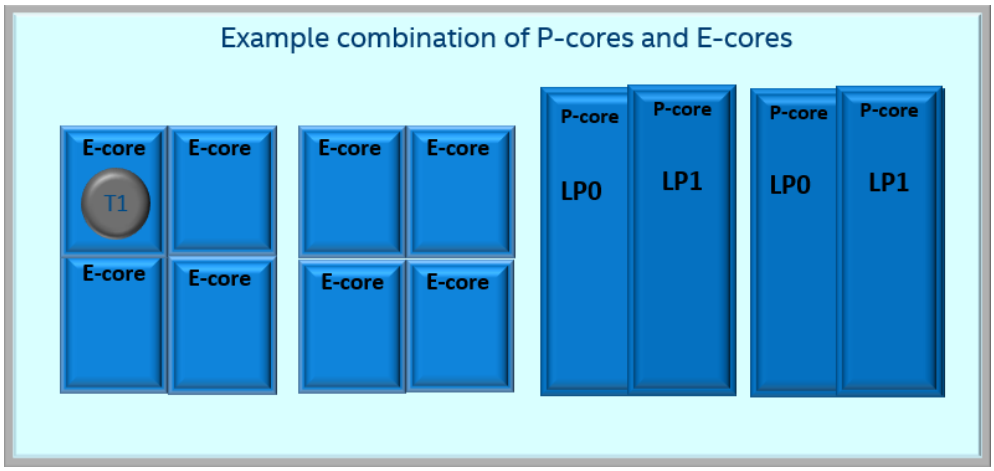
如果PPM设置如下所示，则默认情况下启用该行为：

a. *EcoQoS->SchedulingPolicy 设置为3 或4*。
b. 如果EcoQoS->SchedulingPolicy不存在，则检查默认->SchedulingPolicy是否存在设置为5。

在某些情况下，在核心不存在的情况下，在核心上运行后台活动可能更有效
前台或更高QoS线程需要。

4.8 低利用率线程

在某些低功耗信封配置中，更希望在E核上运行低利用率线程。



电池滑块设置在一些12[第]默认情况下，在Windows上，Generation Intel Core处理器可能会启用此行为。或者，可以通过OEM利用PPM设置-异种停车策略来实现这种行为。

注意：低利用率的重要/关键线程可能会带来性能成本；启用此Windows Core停车功能时请考虑这一点。

4.9 WINDOWS事件跟踪

4.9.1 是Intel® 系统上是否启用线程控制器？

从Windows的事件跟踪中，可以成功确认Windows是否已成功启用并正在利用Intel Thread Director。这是通过检查跟踪日志中“microsoft windows 内核处理器电源”事件下的“AdvanceHgsEnabled”字段来实现的。

4.9.2 如何检查进程/线程的QoS？

从Windows的事件跟踪中，在CPU使用率->新线程BAMQoSLevel下，可以识别各种进程或线程的QoS级别。有关Windows QoS的更多详细信息，请参阅第7.6节。

停车示例

Windows使用硬件引导调度来实现各种目标：

1. 主要目标是提高ST主导工作负载的突发性能。当P核与E核相比具有更高的性能（P核不受功率/热约束）时，操作系统调度器会识别性能要求较高的线程，并在P核上对其进行调度。
2. 第二个目标是在IA性能不提供任何附加值且能效/电池寿命是主要目标的情况下，将电子内核用于所有场景工作负载和后台活动。期望这些工作负载以节能频率执行，其中对于大多数工作负载，E核比P核更高效。
3. 第三个目标是实现GFX应用的节能性能。在这些应用中，重要的是最大限度地提高GFX功率预算，同时提供足够的IA性能，为GFX设备供电。在大多数GFX应用中，E核提供了足够的IA性能和增强的能效，最大限度地提高了GFX的功率预算。这些应用程序需要安排在E核上。然而，对于一些GFX工作负载，最好在P-core上执行，因为这些工作负载需要更高的IA性能，或者E-core需要在P-core更高效的频率下执行。在这些情况下，P核上的调度满足了应用程序性能需求，同时也增加了GFX的功率预算。

下表提供了具有调度期望的各种场景的快照。

脚本	调度程序行为
视频播放应用程序发布（型材支撑）	P-core未经切割，始终可立即用于重要螺纹。
视频播放（减去应用程序启动）（半活跃情景）	P-core始终驻车。在E-core上计划的工作利用率<60%（未分析阈值*）。不重要的线程不使用P核。
ADK应用程序发布（型材支撑）	P-core未经切割，始终可立即用于重要螺纹。
最小化CINEBENCH	最小化时，CINEBENCH线程仅在E核模块上运行。
wPrime 1T（前景/高QoS）	wPrime单线程在P核上运行。
Geekbench MT公司（前景/高QoS MT）	由于五核（一个P核+四个E核），机器翻译性能高于ST-核心）与四核（四个E核）的比较。
双屏幕	调度不同于高QoS的MediumQoS*的机会。
游戏/图形	基于功率/热量，基于硬件的P核反馈
屏蔽	P-core始终驻车。只有E-core可用于调度。

6 软件应用：英特尔常见问题和建议® 核心™ 支持X86混合架构的处理器

6.1 软件可见的ISA差异是什么？

软件可以看到一些ISA差异。示例包括：

1. P核和E核之间的结构差异（TLB、缓存、拓扑）。
2. 性能监视、英特尔处理器跟踪、最后一个分支记录和MCA ISA的某些方面。
3. 一些特定于模型的MSR。

期望这些差异不会影响软件应用程序。

6.2 亲和性可以用于支持X86混合架构的处理器吗？

一般来说，软件应避免设置亲和力。在支持x86混合体系结构的处理器上，设置处理器相关性可能会导致性能或效率不理想。性能和效率决策是根据硬件到Windows枚举的当前性能和效率能力以及其他因素（如QoS、优先级等）动态做出的。

亲和力可用于某些场景。例如，在特定处理器上读取计数器时。

或者，如果必须使用相关性，则利用CPUSets API–SetProcessorDefaultCpuSets、SetThreadSelectedCpuSets以与操作系统电源管理兼容的“软”方式声明应用程序相关性。

这同样适用于中断。驱动程序应避免特定处理器上的仿射中断，以便在支持x86混合体系结构的Intel Core处理器上通过Windows将中断优化为性能或高效内核。

6.3 什么是WINDOWS POWER THROTTLING API？应用程序开发人员如何影响混合调度？

开发人员可以通过使用Windows API SetProcessInformation、SetThreadInformation，以最佳方式引导工作实现高性能或高效的核心。

如果这些API没有与ProcessPowerThrottling参数一起使用，那么默认情况下会触发Windows的自动机制。这可能会导致Windows错误识别可以利用高效核心的线程（例如，EcoQoS）和可以利用性能核心的线程（例如，高QoS）。
开发人员可以通过使用这些API来帮助改进Windows分类，这可能有助于提高效率/电池寿命，减少风扇噪音，提高系统性能。

SetProcessInformation函数定义为：

```
布尔SetProcessInformation(
手柄                          H过程，
PROCESS\u INFORMATION\u CLASS ProcessInformationClass，
LPVOID                        处理信息，
```

```
        德沃德                                处理信息大小
        );
```

以下示例显示如何使用ProcessPowerThrottling调用SetProcessInformation，以在当前进程上启用限制策略。

```
PROCESS\u POWER\u THROTTLING\u STATE PowerThrottling；RtlZeroMemory
（&PowerThrottling，sizeof（PowerThrottling））；功率节流。版本=PROCESS\u
POWER\u THROTTLING\u CURRENT\u版本；

//
//打开ExecutionSpeed节流。ControlMask选择机构和
//StateMask声明应打开或关闭哪个机制。
//

功率节流。ControlMask=PROCESS\u POWER\u THROTTLING\u EXECUTION\u SPEED；
功率节流。StateMask=PROCESS\u POWER\u THROTTLING\u EXECUTION\u SPEED；

SetProcessInformation（GetCurrentProcess（），
                      ProcessPowerThrottling，
                      &功率节流，
                      sizeof（功率节流）；

//
//关闭ExecutionSpeed节流。ControlMask选择机构和
//状态掩码设置为零，因为应关闭机制。
//

功率节流。ControlMask=PROCESS\u POWER\u THROTTLING\u EXECUTION\u SPEED；
功率节流。状态掩码=0；

SetProcessInformation（GetCurrentProcess（），
                      ProcessPowerThrottling，
                      &功率节流，
                      sizeof（功率节流）；

//
//让系统管理所有功率节流。ControlMask设置为0，因为我们不希望
//控制任何机制。
//

功率节流。控制掩码=0；
功率节流。状态掩码=0；

SetProcessInformation（GetCurrentProcess（），
                      ProcessPowerThrottling，
                      &功率节流，
                      sizeof（功率节流）；
```

6.4 线程数

基于处理器计数的线程创建可能并不总是最优的。一些内核的性能可能不如其他内核，甚至处于空闲/停放状态。在某些场景中，多线程扩展是一个潜在的问题，不仅仅与混合相关。

了解使用不同内核数的权衡对于确定应用程序的理想线程数可能很重要。

6.5 活动旋转

执行活动旋转等待的线程可能会影响重要或关键线程的性能。相反，软件可以选择用包含UMWAIT、TPAUSE或PAUSE的较轻重量的旋转来代替旋转等待。

6.6 驱动程序开发人员可以为优化中断控制做出贡献？

6.6.1 不配置中断策略

驱动程序应使用IrqPolicyMachineDefault作为中断策略，选择Windows中断引导。默认情况下，这将使Windows能够将中断引导到生成中断的核心，并与核心驻车引擎一起工作。通过在同一个内核上寻址中断，您可以避免唤醒空闲内核的延迟，或者避免将元数据移动到另一个内核以寻址中断的延迟，这基本上有助于提供延迟/性能优势。此外，避免中断空闲核心也会对电池寿命产生积极影响。

6.6.2 建筑关系

在Windows上，软件应用程序可以使用用户模式API GetLogicalProcessorInformationEx（）或内核模式API KequeryLogicProcessorRelationship（）来了解处理器关系。

如果驱动程序正在分配每个处理器队列，则驱动程序可以使用这些API来优化队列分配。

7 混合密钥PPM设置和设置值

7.1 WINDOWS电源管理设置

从Windows 10 October update'18（RS5）开始，这些设置有一个额外的增强功能，以利用混合功能：

a. 性能状态引擎设置。
b. 核心停车引擎设置。
c. 平台特定控制。

7.2 堆芯停车发动机设置

Core parking engine对工作负载做出全局可扩展性决策，并确定要使用的最佳计算核心集。在支持x86混合架构的处理器中，它还决定了性能核心与高效核心的最佳组合。

7.2.1 异质政策

通用\电源\策略\设置\处理器\异质策略

7.2.1.1 设定值：0（即标准停车或优惠核心停车）

在这种配置中，从性能最高的核心开始，对计算核心的最佳集进行分解。

用于确定要断开的最佳内核数的相关PPM设置：

- *CPMinCores：指定在任何给定时间，从性能最高的内核开始，可以处于未分离状态的逻辑处理器的最小百分比。逻辑处理器是指每个NUMA 节点内系统上启用的所有逻辑处理器。*
- *CPMaxCores：指定在任何给定时间，从性能最高的内核开始，可以处于未分离状态的逻辑处理器的最大百分比。逻辑处理器是指每个NUMA 节点内系统上启用的所有逻辑处理器。*
- *CPIncreaseTime：指定在其他已停驻的性能最好的逻辑处理器可以从已停驻状态转换为未停驻状态之前必须经过的最短时间。时间以处理器性能时间检查间隔的数量为单位指定。*
- *CPDecreaseTime：指定在其他未分离的性能最低的逻辑处理器可以从未分离状态转换到已分离状态之前必须经过的最短时间。时间以处理器性能时间检查间隔的数量为单位指定。*
- *CPConcurrency：指定用于确定节点并发性的阈值。*
- *CPDistribution：指定在并发分布中使用的利用率（百分比），以选择将实用程序分发到的性能最佳的逻辑处理器的数量。这*

该数量可以小于但决不能大于被选择为未被解析的逻辑处理器的数量。

- *CPHeadroom：指定利用率的值，如果未配对的处理器集中使用最少的处理器具有更高的利用率，则会导致核心驻车引擎重新配对另一个性能最高的驻车逻辑处理器。此设置允许检测到并发性的增加。*
- *CpLatencyHintUnpark：指定当检测到系统低延迟提示时，未分离的内核的最小数量（从性能最高的内核开始）。*

7.2.1.2 设置值：4（即异质停车）

在这种配置中，基于利用率，首先对性能最高或效率最高的内核的组合进行分离。

在某些情况下，如低功耗封装SKU或更好的电池寿命目标，可以更高效地在具有更高效率能力的核上以高效频率运行低利用率工作。此策略在这些场景中与最佳性能状态引擎设置结合使用。

用于确定要断开的最佳内核数的相关PPM设置：

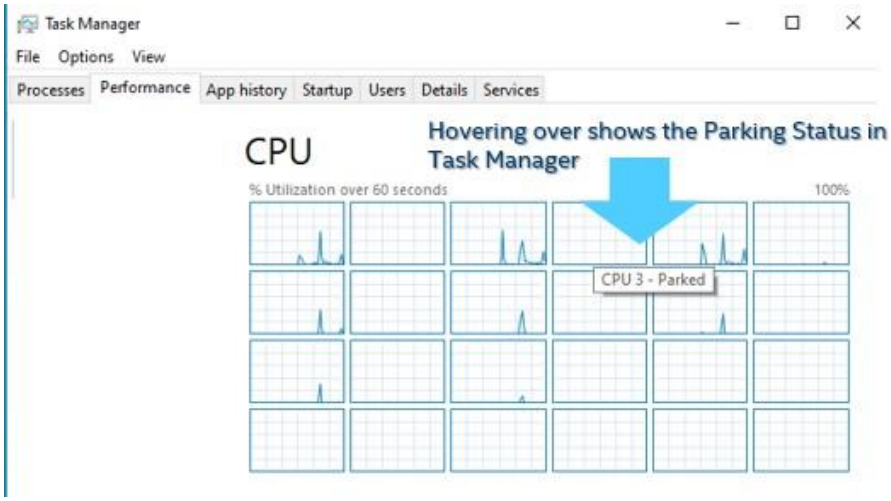
- *CPMinCores：指定在任何给定时间可以处于未分离状态的逻辑处理器的最小百分比。逻辑处理器是指每个NUMA节点内系统上启用的所有逻辑处理器。*
- *CPMaxCores：指定在任何给定时间可以处于未分离状态的逻辑处理器的最大百分比。逻辑处理器是指每个NUMA节点内系统上启用的所有逻辑处理器。*
- *CPIncreaseTime：指定处于已停驻状态的其他已停驻最高效逻辑处理器可以从已停驻状态转换为未停驻状态之前必须经过的最短时间。时间以处理器性能时间检查间隔的数量为单位指定。*
- *CPDecreaseTime：指定在将其他未分离的效率最低的逻辑处理器从未分离状态转换为已驻留状态之前必须经过的最短时间。时间以处理器性能时间检查间隔的数量为单位指定。*
- *CPConcurrency：指定用于确定节点并发性的阈值。*
- *CPDistribution：指定在并发分布中使用的利用率（百分比），以选择将实用程序分发到的性能最佳的逻辑处理器的数量。这个数字可能会比选择要取消连接的逻辑处理器的数量更少，但永远不会更大。*
- *CPHeadroom：指定利用率的值，如果未配对的处理器集中使用最少的处理器具有更高的利用率，则会导致核心驻车引擎重新配对另一个性能最高的驻车逻辑处理器。此设置允许检测到并发性的增加。*
- *CpLatencyHintUnpark：指定当检测到系统低延迟提示时，未分离的内核的最小数量（从性能最高的内核开始）。*

- *HeteroIncreaseThreshold：指定要在上面跨越的阈值，这是解析第n个性能最高的已驻留核心所必需的。每个核心索引都有单独的值。*
- *heterodecreasethrown：指定要在其上交叉的阈值，这是驻车第n个性能最低的未切割内核所必需的。每个核心索引都有单独的值。*
- *HeteroIncremaseTime：指定在断开额外的性能内核（从性能最高的处理器开始）之前必须经过的最短时间。*
- *HeteroDecreaseTime：指定在其他处理器（从性能最低的未处理处理器开始）可以从未处理状态转换为驻留状态之前必须经过的最小时间量。*
- *HETEROCLASSINITALPERF：指定未进行性能测试的处理器的初始性能百分比。*
- HeteroClass0FloorPerf：指定在至少一个处理器未进行性能分析时，高效处理器的最低性能百分比。

7.2.2 检查处理器的驻车状态

可以通过各种工具查看处理器的驻车状态。

7.2.2.1 示例工具#1：任务管理器



7.2.2.2 示例工具#2：通过Windows Performance Analyzer进行Windows跟踪的事件

Processor Parking State Parking State by Processor *

Line #	CPU	State
1	0	Unpark
2	1	SoftPark
3	2	Unpark
4	3	SoftPark
5	4	Unpark
6	5	SoftPark
7	6	Unpark
8	7	SoftPark
9	8	Unpark
10	9	SoftPark
11	10	Unpark
12	11	SoftPark

7.3 性能状态设置

在支持HWP的系统上，性能状态通过硬件控制的性能状态（即HWP）进行协调。存在各种设置，可以帮助HWP硬件引擎做出最佳性能状态决策。

7.3.1 性能参考

指定要在E核的能量性能首选项（EPP）寄存器中编程的值。该值以百分比形式指定，并由寄存器中的窗口以0–255的比例填充。

在HWP系统上，对于单个逻辑处理器，EPP寄存器为774H[位31:24]。

7.3.2 性能参考1

指定要在P核的能量性能首选项（EPP）寄存器中编程的值。该值以百分比形式指定，并由寄存器中的窗口以0–255的比例填充。

在HWP系统上，对于单个逻辑处理器，EPP寄存器为774H[位31:24]。

7.3.3 最小性能

指定要在E核的最低性能寄存器中编程的值。该值以百分比形式指定，并由寄存器中的窗口以0–255的比例填充。

在HWP系统上，对于单个逻辑处理器，EPP寄存器为774H[位7:0]。

7.3.4 最低性能1

指定要在P核的最小性能寄存器中编程的值。该值以百分比形式指定，并由寄存器中的窗口以0–255的比例填充。

在HWP系统上，对于单个逻辑处理器，EPP寄存器为774H[位7:0]。

7.3.5 最大性能

指定要在E核的最大性能寄存器中编程的值。该值以百分比形式指定，并由寄存器中的窗口以0–255的比例填充。

在HWP系统上，对于单个逻辑处理器，EPP寄存器为774H[位15:8]。

7.3.6 最大性能1

指定要在P核的最大性能寄存器中编程的值。该值以百分比形式指定，并由寄存器中的窗口以0–255的比例填充。

在HWP系统上，对于单个逻辑处理器，EPP寄存器为774H[位15:8]。

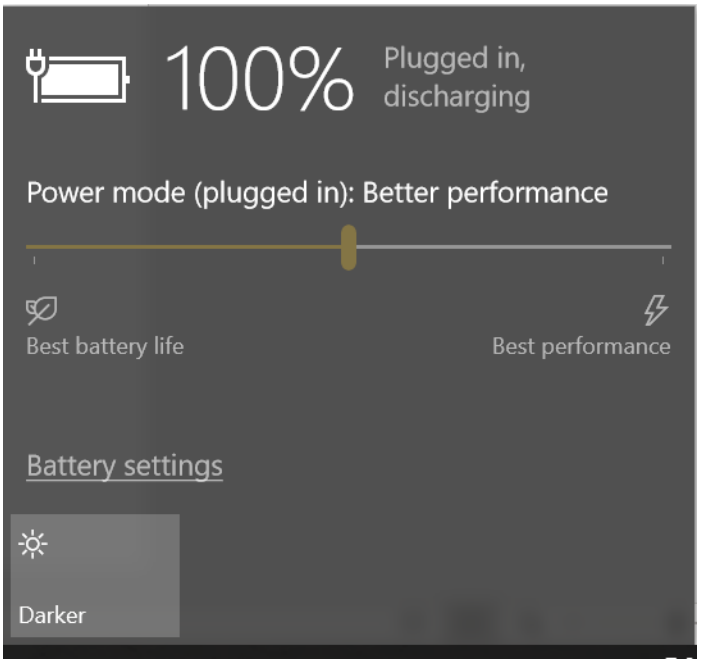
7.4 WINDOWS性能增强滑块

Windows performance power slider使最终客户能够快速、智能地用系统性能换取更长的电池寿命。当客户在四种滑块模式之间切换以牺牲性能换取电池寿命（反之亦然）时，Windows电源设置会在后台进行。

可用的不同滑块位置：

a. 更好的电池。
b. 更好的性能。
c. 最佳性能。

下图显示了Windows performance power滑块的快照，用户可以在各种滑块位置之间切换。



在以下情况下，不同滑块位置下的PPM设置会发生变化：

a. PerfEnergyPreference在每个滑块位置上都不同，在更面向电池的滑块位置上具有更高的值。
b. QoS、配置文件设置不同，针对电池寿命的节流设置（例如，更高的性能参考）或面向功率的QoS/配置文件。

7.5 关键功率配置文件

与Windows性能电源滑块一样，可以为不同的电源配置文件配置不同的Windows电源设置。以下部分列出了Windows支持的各种配置文件和建议的配置。

7.5.1 默认配置文件

默认配置文件是大多数时间处于活动状态的配置集。这些设置与当前电源方案的设置相同。

7.5.2 低延迟

低延迟是在引导和应用程序启动期间激活的配置文件。

7.5.3 低功耗

低功耗是在媒体播放场景的缓冲阶段激活的配置文件。

7.5.4 屏蔽

屏蔽是现代备用系统上使用的一种配置文件。当系统进入长期睡眠阶段时，该配置文件被激活；所有系统静默行为已完成，没有音频播放，也没有移动热点参与。当系统从睡眠中醒来时，此配置文件被断开。

7.6 QOS、HWP和混合调度

7.6.1 违约

该行为专门针对默认和高QoS（或重要线程）进行了优化。并且，如果没有定义其他QoS，那么也会影响其他QoS线程。

- 示例：Default->EnergyPerfPreference可以在不同的滑块位置上具有不同的值，以帮助实现不同的性能和电池寿命目标。

7.6.2 遮挡窗口

PPM设置允许将阻塞窗口HWP和调度策略配置为不同于默认/高QoS线程。

7.6.3 后台线程

默认情况下，使用以下PPM设置专门为后台线程优化行为：

a. 背景->EnergyPerfPreference与默认->EnergyPerference的值不同。
b. 背景->MaximumPerformance与默认->MaximumPerformance具有不同的值。
c. 默认->频率上限设置为非零值。
d. 背景->调度策略与默认->调度策略具有不同的值。
e. 后台->调度策略不存在，默认->调度策略设置为5。

在某些配置上（例如，推荐最佳性能滑块位置），背景设置与默认设置相同可能更为理想，以避免限制背景线程。对于此行为，设置将填充为：

a. 背景->EnergyPerfPreference=默认->EnergyPerference。
b. 背景->MaximumPerformance=默认->MaximumPerformance。
c. 默认->未设置频率上限。
d. 后台->调度策略=默认->调度策略。
e. 默认->SchedulingPolicy设置为2。

7.6.4 多媒体线程

默认情况下，使用以下PPM设置专门针对多媒体线程优化行为：

a. MMQoS->EnergyPerfPreference与默认->EnergyPerference具有不同的值。
b. MMQoS->MaximumPerformance与默认->MaximumPerformance具有不同的值。
c. MMQoS->调度策略与默认->调度策略具有不同的值。
d. MMQoS->SchedulingPolicy不存在，默认->SchedulingPolicy设置为5。

注意：与背景线程类似，在某些配置（例如最佳性能滑块位置）上，多媒体设置与默认设置相同可能更为理想，以避免多媒体线程的节流。对于此行为，设置将填充为（例如最佳性能滑块位置下）：

a. 背景->EnergyPerfPreference=默认->EnergyPerference。
b. 背景->MaximumPerformance=默认->MaximumPerformance。
c. 后台->调度策略=默认->调度策略。
d. 默认->SchedulingPolicy设置为2。

7.6.5 Eco线程

开发人员可以利用Windows power throttling API SetProcessInformation、SetThreadInformation来选择退出对开发人员应用程序线程或进程的性能或效率目标的Windows动态检测。开发人员标记为高效运行的线程称为Eco QoS线程。

从PPM的角度来看，使用以下设置专门针对EcoQoS线程优化了行为：

a. 与默认->EnergyPerReference相比，EcoQoS->EnergyPerfPreference具有更高的值。
b. 与默认值->MaximumPerformance相比，EcoQoS->MaximumPerformance的值较低。
c. EcoQoS->SchedulingPolicy与默认->SchedulingPolicy的值不同。
d. EcoQoS->SchedulingPolicy不存在，默认->SchedulingPolicy设置为5。

7.6.6