使用DCI和RTIT技术进行精准调优

张银奎

2020/6/20 上海

张银奎，Raymond Zhang，格蠹老雷，《软件调试》和《格蠹汇编》作者
http://advdbg.org http://weibo.com/dbgger 格友公众号


RTIT技术背景
通过DCI会话启用RTIT
追踪包
演示
应用

大名PT

- Processor Trace
- 处理器追踪技术
- 给CPU的执行经过写日志

CHAPTER 35
INTEL® PROCESSOR TRACE

35.1 OVERVIEW

Intel® Processor Trace (**Intel PT**) is an extension of Intel® Architecture that captures information about software execution using dedicated hardware facilities that cause only minimal performance perturbation to the software being traced. This information is collected in **data packets**. The initial implementations of Intel PT offer **control flow tracing**, which generates a variety of packets to be processed by a software decoder. The packets include timing, program flow information (e.g. branch targets, branch taken/not taken indications) and program-induced mode related information (e.g. Intel TSX state transitions, CR3 changes). These packets may be buffered internally before being sent to the memory subsystem or other output mechanism available in the platform. Debug software can process the trace data and reconstruct the program flow.

Intel Processor Trace was first introduced in Intel® processors based on Broadwell microarchitecture and Intel Atom® processors based on Goldmont microarchitecture. Later generations include additional trace sources, including software trace instrumentation using PTWRITE, and Power Event tracing.

小名RTIT

- Real Time Instruction Trace
- 实时指令追踪
- MSR寄存器仍保留这个名字

```
#define MSR_IA32_RTIT_OUTPUT_BASE 0x00000560
#define MSR_IA32_RTIT_OUTPUT_MASK_PTRS    0x00000561
#define MSR_IA32_RTIT_CTL          0x00000570
#define MSR_IA32_ADDR0_START       0x00000580
#define MSR_IA32_ADDR0_END         0x00000581
#define MSR_IA32_ADDR1_START       0x00000582
#define MSR_IA32_ADDR1_END         0x00000583
```

Design and Test Technology Conference

EAX = 0x14, ECX = 0

EBX

- Bit 00: IA32_RTIT_CTL.CR3Filter can be set to 1
- IA32_RTIT_CR3_MATCH MSR can be accessed.
- Bit 01: Configurable PSB and Cycle-Accurate Mode.
- Bit 02: IP Filtering, TraceStop filtering, and preservation of Intel PT MSRs across warm reset.
- Bit 03: MTC timing packet and suppression of COFI-based packets.

ECX

- Bit 00: Tracing can be enabled with IA32_RTIT_CTL.ToPA = 1 utilizing the ToPA output scheme
- IA32_RTIT_OUTPUT_BASE and IA32_RTIT_OUTPUT_MASK_PTRS MSRs can be accessed.
- Bit 01: ToPA tables can hold any number of output entries
- Maximum specified by the MaskOrTableOffset field of IA32_RTIT_OUTPUT_MASK_PTRS.
- Bit 02: Single-Range Output scheme.
- Bit 03: Output to Trace Transport subsystem.
- Bit 31: Generated packets which contain IP payloads have LIP values
- Includes the CS base component

EAX = 0x14, ECX = 1

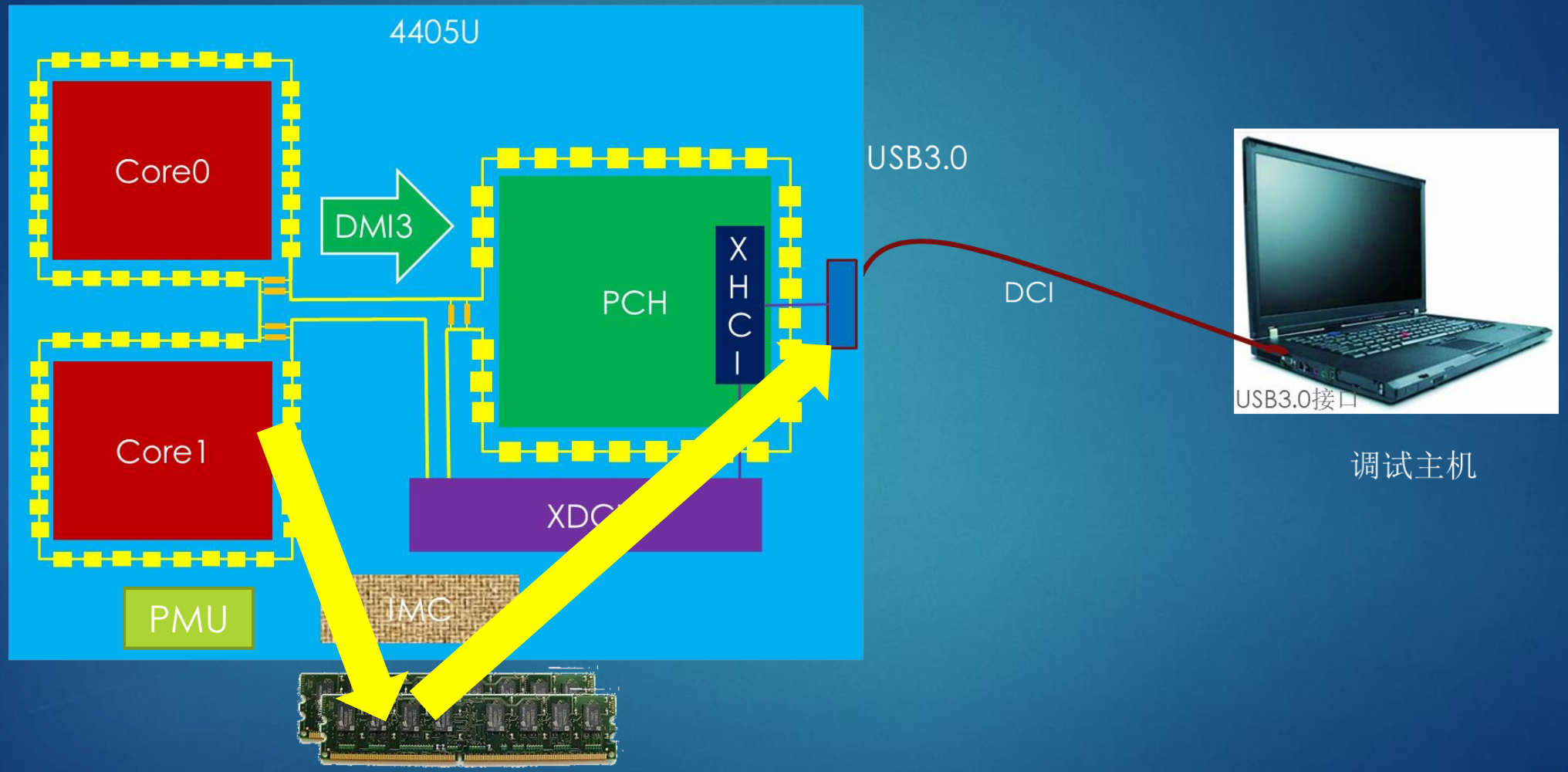
Packet Generation

EAX

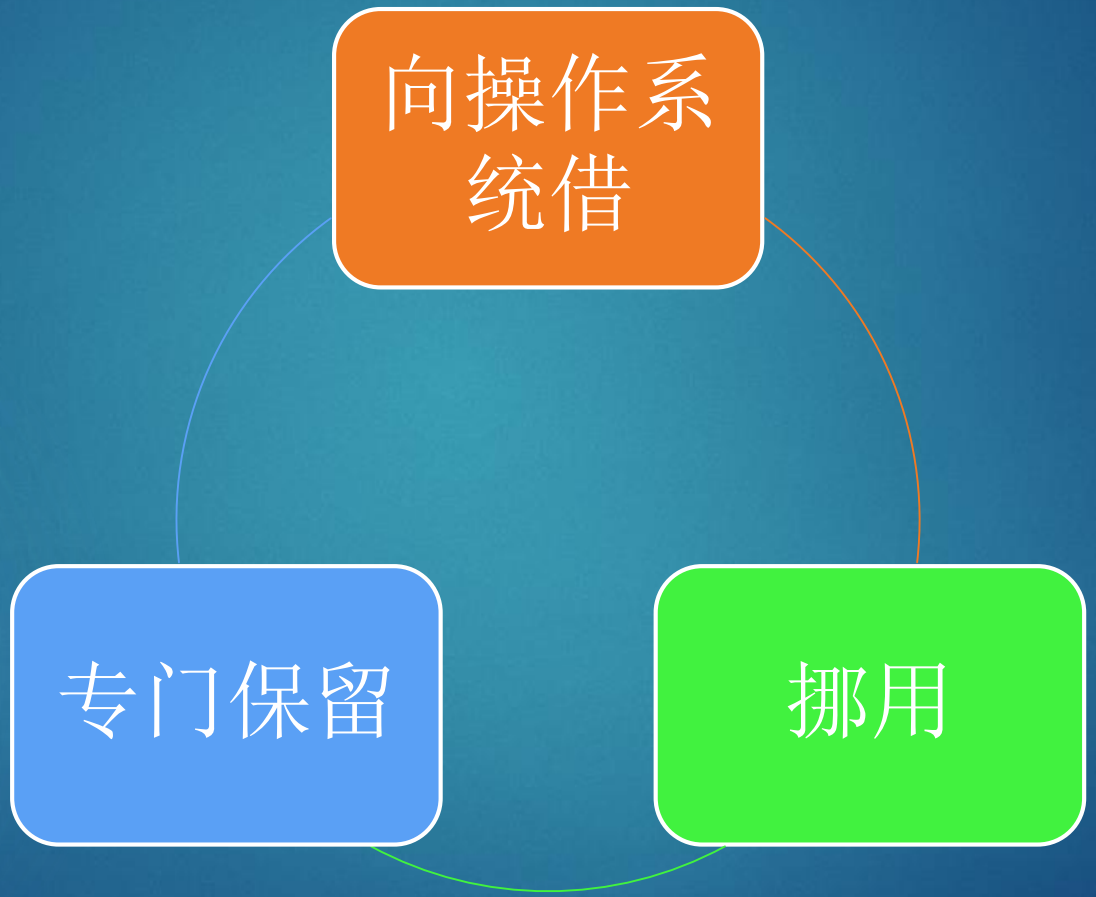
- Bits 2:0: Number of configurable Address Ranges for filtering.
- Bit 31:16: Bitmap of supported MTC period encodings

EBX

- Bits 15-0: Bitmap of supported Cycle Threshold value encodings
- Bit 31:16: Bitmap of supported Configurable PSB frequency encodings


4405U
Core0
Core1
DMI3
PCH
XHCI
XDCI
IMC
PMU
USB3.0
DCI
USB3.0接口
调试主机

预留内存



向Windows借用

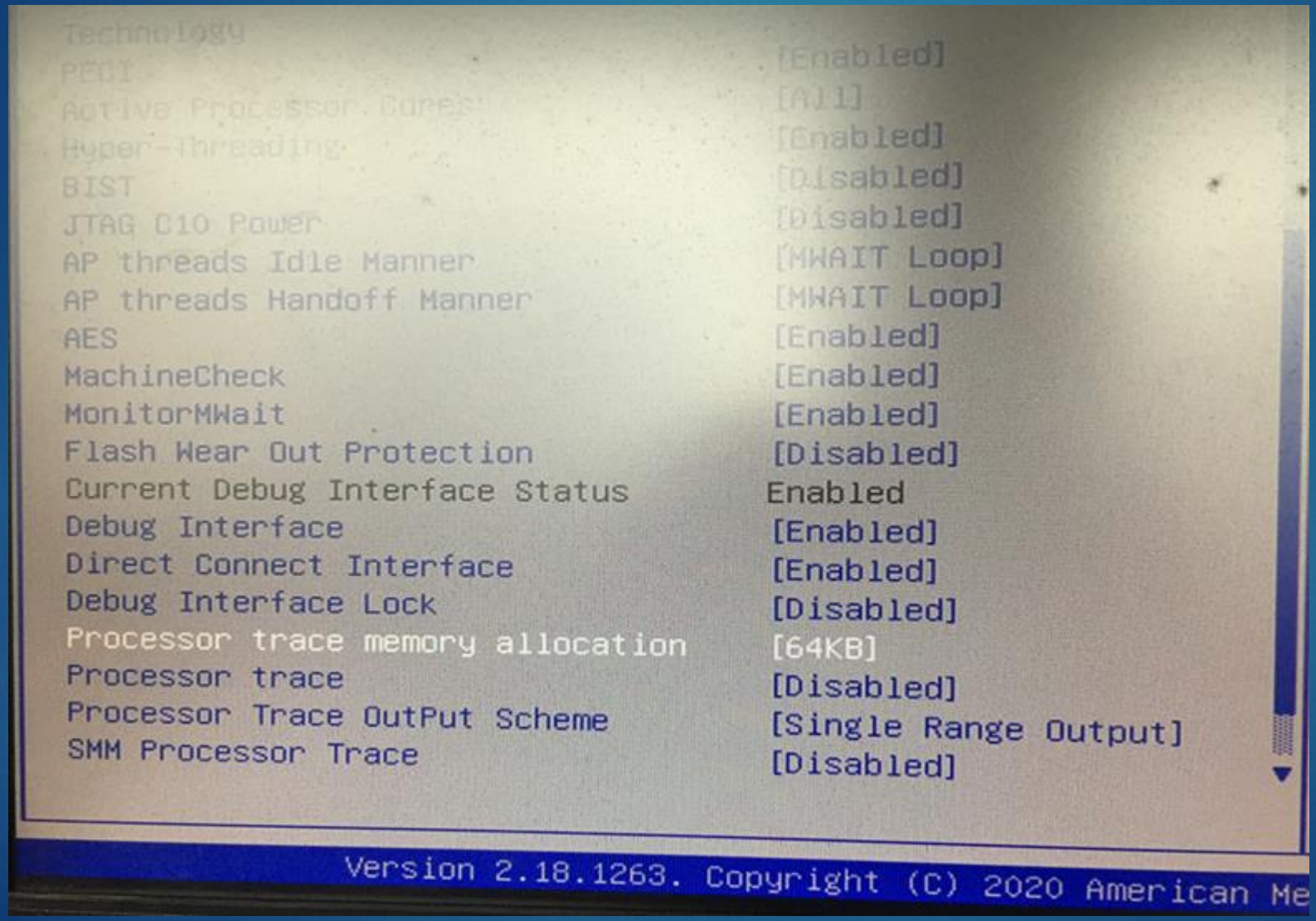
bcdedit /set badmemoryaccess no
bcdedit /set badmemorylist <list of PFNs, space-delimited>

设置给扩展

!setoutputconfig [/here] /start <expr> /size <expr>
/start <expr> - Start address of available buffer
/size <expr> - Size of available buffer
/here - Create and set output configuration on the currently selected processor only

!setoutputconfig /start 0x1C0000000 /size 0x80000

GDK7的设置界面


Technology
PECI [Enabled]
Active Processor Cores [All]
Hyper-threading [Enabled]
BIST [Disabled]
JTAG C10 Power [Disabled]
AP threads Idle Manner [MWAIT Loop]
AP threads Handoff Manner [MWAIT Loop]
AES [Enabled]
MachineCheck [Enabled]
MonitorMWait [Enabled]
Flash Wear Out Protection [Disabled]
Current Debug Interface Status Enabled
Debug Interface [Enabled]
Direct Connect Interface [Enabled]
Debug Interface Lock [Disabled]
Processor trace memory allocation [64KB]
Processor trace [Disabled]
Processor Trace OutPut Scheme [Single Range Output]
SMM Processor Trace [Disabled]
Version 2.18.1263. Copyright (C) 2020 American Me

使用ptext64观察

启用之前

```
<0> Output buffer at 0000000085000000 - 0000000085ffffff (16MB), current offset 00000000.
<1> Output buffer at 0000000083000000 - 0000000083ffffff (16MB), current offset 00000000.
<2> Output buffer at 000000007f000000 - 000000007fffffff (16MB), current offset 00000000.
<3> Output buffer at 0000000081000000 - 0000000081ffffff (16MB), current offset 00000000.
```

```
<0> Output buffer at 0000000085000000 - 0000000085ffffff (16MB), current offset 00000000.
<1> Output buffer at 0000000083000000 - 0000000083ffffff (16MB), current offset 00000000.
<2> Output buffer at 0000000081000000 - 0000000081ffffff (16MB), current offset 00000000.
<3> Output buffer at 0000000081000000 - 0000000081ffffff (16MB), current offset 00000000.
```

```
<0> Output buffer at 0000000086a60000 - 0000000086a6ffff (64KB), current offset 00000000.
<1> Output buffer at 0000000086a40000 - 0000000086a4ffff (64KB), current offset 00000000.
<2> Output buffer at 0000000086a00000 - 0000000086a0ffff (64KB), current offset 00000000.
<3> Output buffer at 0000000086a20000 - 0000000086a2ffff (64KB), current offset 00000000.
```

启用之后，运行一段时间后再中断观察

```
<0> Output buffer at 0000000085000000 - 0000000085ffffff (16MB), current offset 0007e640.
<1> Output buffer at 0000000083000000 - 0000000083ffffff (16MB), current offset 0007a660.
<2> Output buffer at 000000007f000000 - 000000007fffffff (16MB), current offset 00155d70.
<3> Output buffer at 0000000081000000 - 0000000081ffffff (16MB), current offset 00076450.
```

实例

```
<0> Output buffer at 0000000085000000 - 0000000085ffffff (16MB), current offset 00329840.
<1> Output buffer at 0000000083000000 - 0000000083ffffff (16MB), current offset 007c08c0.
<2> Output buffer at 000000007f000000 - 000000007fffffff (16MB), current offset 005eb340.
<3> Output buffer at 0000000081000000 - 0000000081ffffff (16MB), current offset 00212b80.
```

msr[571] = 00009790`00000003

msr[570] = 00000000`0000200d

追踪区的原始数据

```
#85000000 82028202 82028202 82028202 82028202
#85000010 00002099 43020000 15d40e00 00000000
#85000020 00040302 00002302 00150302 43020000
#85000030 162d1600 00000000 00000000 43020000
#85000040 162d1780 00000000 00000000 43020000
#85000050 162d1600 00000000 00000000 43020000
#85000060 162d1780 00000000 00000000 43020000
#85000070 162d1600 00000000 00000000 43020000
```

***CPUID EAX=0x14-ECX=0x0:
EAX=0x1 EBX=0xf ECX=0x7
EDX=0x0

- GDK7支持：CR3过滤、精准时钟模式、IP过滤和追踪停止区、MTC包
- 不支持PTWRITE和电源事件追踪

Table 35-11. CPUID Leaf 14H Enumeration of Intel Processor Trace Capabilities

CPUID.(EAX=14H,ECX=0)		Name	Description Behavior
Register	Bits		
EAX	31:0	Maximum valid sub-leaf Index	Specifies the index of the maximum valid sub-leaf for this CPUID leaf
EBX	0	CR3 Filtering Support	1: Indicates that IA32_RTIT_CTL.CR3Filter can be set to 1, and that IA32_RTIT_CR3_MATCH MSR can be accessed. See Section 35.2.7. 0: Indicates that writes that set IA32_RTIT_CTL.CR3Filter to 1, or any access to IA32_RTIT_CR3_MATCH, will #GP fault.
	1	Configurable PSB and Cycle-Accurate Mode Supported	1: (a) IA32_RTIT_CTL.PSBFreq can be set to a non-zero value, in order to select the preferred PSB frequency (see below for allowed values). (b) IA32_RTIT_STATUS.PacketByteCnt can be set to a non-zero value, and will be incremented by the processor when tracing to indicate progress towards the next PSB. If trace packet generation is enabled by setting TraceEn, a PSB will only be generated if PacketByteCnt=0. (c) IA32_RTIT_CTL.CYCEn can be set to 1 to enable Cycle-Accurate Mode. See Section 35.2.7. 0: (a) Any attempt to set IA32_RTIT_CTL.PSBFreq, to set IA32_RTIT_CTL.CYCEn, or write a non-zero value to IA32_RTIT_STATUS.PacketByteCnt any access to IA32_RTIT_CR3_MATCH, will #GP fault. (b) If trace packet generation is enabled by setting TraceEn, a PSB is always generated. (c) Any attempt to set IA32_RTIT_CTL.CYCEn will #GP fault.
	2	IP Filtering and TraceStop supported, and Preserve Intel PT MSRs across warm reset	1: (a) IA32_RTIT_CTL provides at one or more ADDRn_CFG field to configure the corresponding address range MSRs for IP Filtering or IP TraceStop. Each ADDRn_CFG field accepts a value in the range of 0:2 inclusive. The number of ADDRn_CFG fields is reported by CPUID.(EAX=14H, ECX=1):EAX.RANGECNT[2:0]. (b) At least one register pair IA32_RTIT_ADDRn_A and IA32_RTIT_ADDRn_B are provided to configure address ranges for IP filtering or IP TraceStop. (c) On warm reset, all Intel PT MSRs will retain their pre-reset values, though IA32_RTIT_CTL.TraceEn will be cleared. The Intel PT MSRs are listed in Section 35.2.7. 0: (a) An Attempt to write IA32_RTIT_CTL.ADDRn_CFG with non-zero encoding values will cause #GP. (b) Any access to IA32_RTIT_ADDRn_A and IA32_RTIT_ADDRn_B, will #GP fault. (c) On warm reset, all Intel PT MSRs will be cleared.
	3	MTC Supported	1: IA32_RTIT_CTL.MTCEn can be set to 1, and MTC packets will be generated. See Section 35.2.7. 0: An attempt to set IA32_RTIT_CTL.MTCEn or IA32_RTIT_CTL.MTCFreq to a non-zero value will #GP fault.
	4	PTWRITE Supported	1: Writes can set IA32_RTIT_CTL[12] (PTWEn) and IA32_RTIT_CTL[5] (FUPonPTW), and PTWRITE can generate packets. 0: Writes that set IA32_RTIT_CTL[12] or IA32_RTIT_CTL[5] will #GP, and PTWRITE will #UD fault.
	5	Power Event Trace Supported	1: Writes can set IA32_RTIT_CTL[4] (PwrEvtEn), enabling Power Event Trace packet generation. 0: Writes that set IA32_RTIT_CTL[4] will #GP.
	31:6	Reserved	


RTIT技术背景
通过DCI会话启用RTIT
追踪包
演示
应用

多种方式使用PT

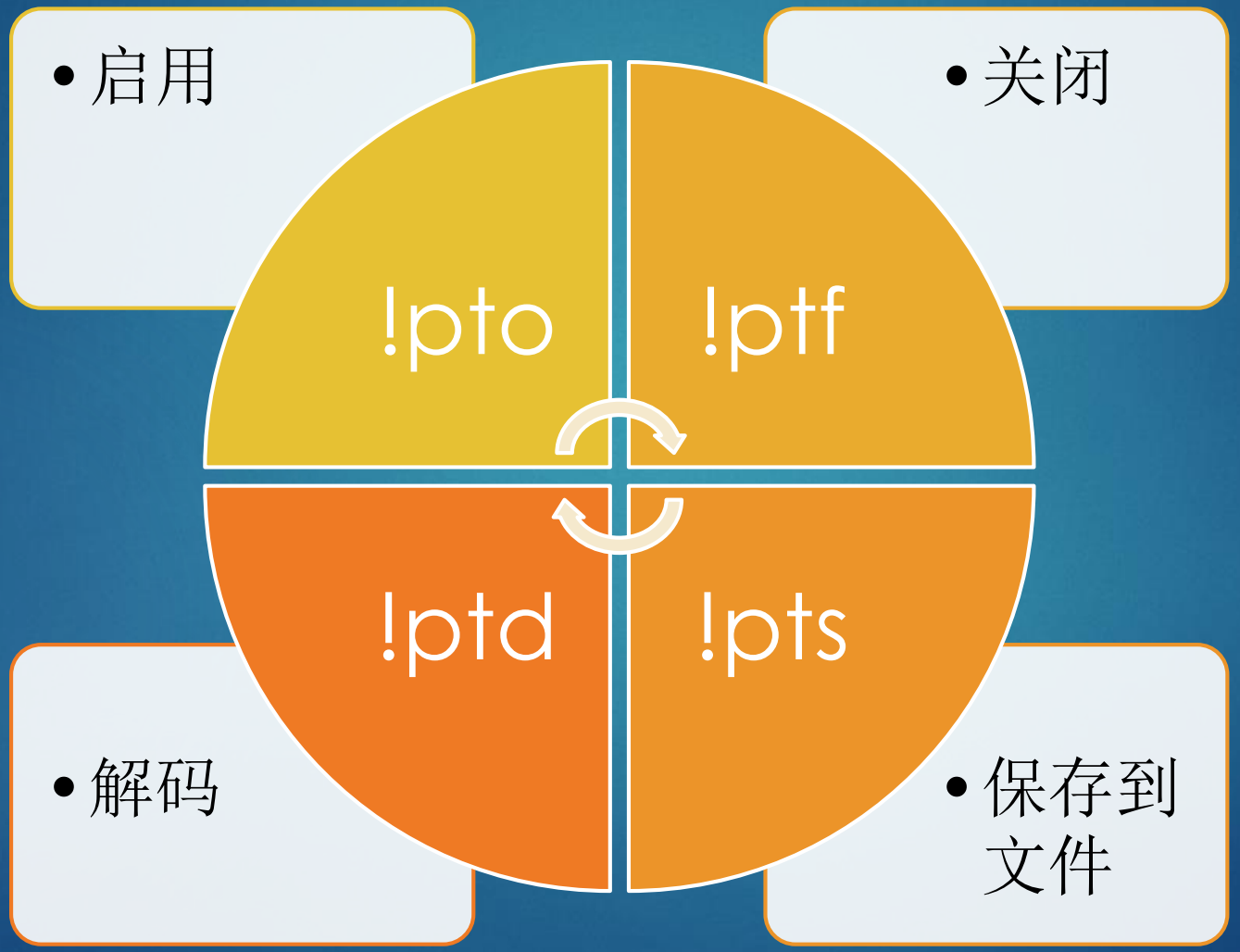
单机	Perf
	Simple-pt
	Windows
双机	Vtune
	Ptext64.dll
	Nano Debugger

Ptext64.dll

- Intel 开发的WinDBG扩展
- 可以用在Nano中
- 实现了启用、过滤、解码等基本功能

```
!cleartrace           - Clear the trace buffer
!decode               - Decode Intel(R) Processor Trace and display
                        reconstructed execution history
!delcr3filter         - Disable Intel(R) Processor Trace CR3 filtering
!delipfilter          - Delete Intel(R) Processor Trace IP filter region based
                        on an address range
!delipfilterall       - Delete all Intel(R) Processor Trace IP filter regions
!delipfilterfunction - Delete Intel(R) Processor Trace IP filter region based
                        on a function symbol
!delipfilterid        - Delete Intel(R) Processor Trace IP filter region based
                        on its ID
!delipfiltermodule    - Delete Intel(R) Processor Trace IP filter region based
                        on a module name
!disable              - Disable tracing
!disablering3trace    - Disable Intel(R) Processor Trace for User-Space code
!disabletimestamps    - Disable timestamps
!enable               - Enable tracing
!enablering3trace     - Enable Intel(R) Processor Trace for User-Space code
!enabletimestamps     - Enable timestamps (only JTAG connection)
!help                 - Displays information on available extension commands
!init                 - Init of Intel(R) Processor Trace extension
!ptdump               - Dump assembled PT buffers to file
!resettrace           - Set the Intel(R) Processor Trace hardware state to
                        reset values
!setcr3filter         - Configure Intel(R) Processor Trace CR3 filtering
!setipfilter          - Configure Intel(R) Processor Trace IP filtering
!setipfilterfunction - Configure Intel(R) Processor Trace IP filtering using
                        a function symbol
!setipfiltermodule    - Configure Intel(R) Processor Trace IP filtering using
                        a module name
!setoutputconfig      - Set a simple output configuration
!showconfig           - Display Intel(R) Processor Trace configuration
!showcr3filter        - Show Intel(R) Processor Trace CR3 filter configuration
!showenable           - Display Intel(R) Processor Trace enable state
!showipfilters        - Show Intel(R) Processor Trace IP filters
!showoutputconfig     - Show Intel(R) Processor Trace output configuration
!version              - Display Intel(R) Processor Trace extension version
                        information
```

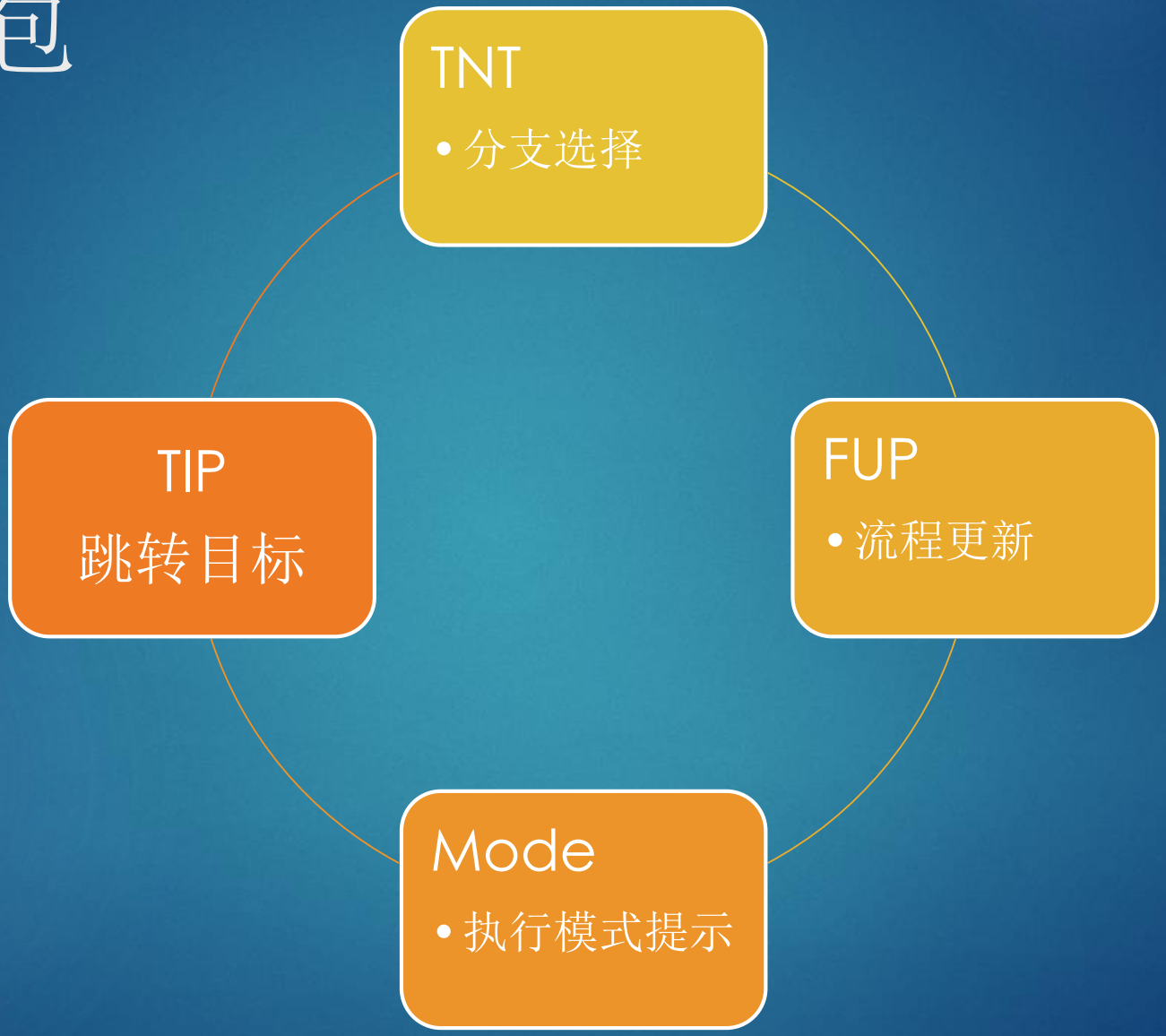
Nano Debugger的扩展命令




RTIT技术背景
通过DCI会话启用RTIT
追踪包
演示
应用

基础信息包

程序流程包



其它包

- PTWRITE
- 电源事件
- CPU群组事件

Packet Stream Boundary (**PSB**)

- 每4K一次
- 有心跳作用

PIP

```
D:\work\nano\nd\libipt\build\bin\Debug>ptdump --no-pad c:\temp\cm.dmp3 | more
0000000000000df0  psb
0000000000000e00  mode.tsx
0000000000000e06  pip        12bee0000                cr3  000000012bee0000
0000000000000e10  cbr        11
0000000000000e14  psbend
0000000000000e16  pip        12bee1800                cr3  000000012bee1800
0000000000000e26  pip        12bee0000                cr3  000000012bee0000
0000000000000e36  pip        12bee1800                cr3  000000012bee1800
0000000000000e46  pip        12bee0000                cr3  000000012bee0000
0000000000000e56  pip        12bee1800                cr3  000000012bee1800
0000000000000e66  pip        12bee0000                cr3  000000012bee0000
0000000000000e76  pip        12bee1800                cr3  000000012bee1800
0000000000000e86  pip        12bee0000                cr3  000000012bee0000
0000000000000e96  pip        12bee1800                cr3  000000012bee1800
0000000000000ea6  pip        12bee0000                cr3  000000012bee0000
0000000000000eb6  pip        50e0e000                 cr3  0000000050e0e000
0000000000000ec0  cbr        11
0000000000000ed0  cbr        11
```

- PIP – Paging Information Packet

不是page fault

通常代表CR3变化

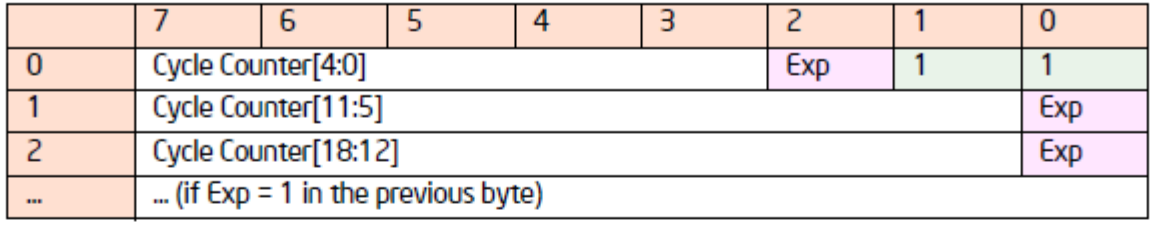
切换进程

Dependencies	TriggerEn && ContextEn && IA32_RTIT_CTL.OS	Generation Scenario	MOV CR3, Task switch, INIT, SIPI, PSB+, VM exit, VM entry
Description	The CR3 payload shown includes only the address portion of the CR3 value. For PAE paging, CR3[11:5] are thus included. For other paging modes (32-bit and 4-level paging[1]), these bits are 0. This packet holds the CR3 address value. It will be generated on operations that modify CR3: ▪ MOV CR3 operation ▪ Task Switch ▪ INIT and SIPI ▪ VM exit, if "conceal VMX from PT" VM-exit control is 0 (see Section 35.5.1) ▪ VM entry, if "conceal VMX from PT" VM-entry control is 0 PIPs are not generated, despite changes to CR3, on SMI and RSM. This is due to the special behavior on these operations, see Section 35.2.8.3 for details. Note that, for some cases of task switch where CR3 is not modified, no PIP will be produced. The purpose of the PIP is to indicate to the decoder which application is running, so that it can apply the proper binaries to the linear addresses that are being traced. The PIP packet contains the new CR3 value when CR3 is written. PIPs generated by VM entries set the NR bit. PIPs generated in VMX non-root operation set the NR bit if the "conceal VMX from PT" VM-execution control is 0 (see Section 35.5.1). All other PIPs clear the NR bit.		

<table>
<tr><td>Name</td><td colspan="4">Core:Bus Ratio (CBR) Packet</td></tr>
<tr><td>Packet Format</td><td colspan="4">
<table>
<tr><th></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td colspan="8">Core:Bus Ratio</td></tr>
<tr><td>3</td><td colspan="8">Reserved</td></tr>
</table>
</td></tr>
<tr><td>Dependencies</td><td>TriggerEn</td><td>Generation Scenario</td><td colspan="2">After any frequency change, on C-state wake up, PSB+, and after enabling trace packet generation.</td></tr>
<tr><td>Description</td><td colspan="4">Indicates the core:bus ratio of the processor core. Useful for correlating wall-clock time and cycle time.</td></tr>
<tr><td>Application</td><td colspan="4">The CBR packet indicates the point in the trace when a frequency transition has occurred. On some implementations, software execution will continue during transitions to a new frequency, while on others software execution ceases during frequency transitions. There is not a precise IP provided, to which to bind the CBR packet.</td></tr>
</table>

CYC包

- 变长负载

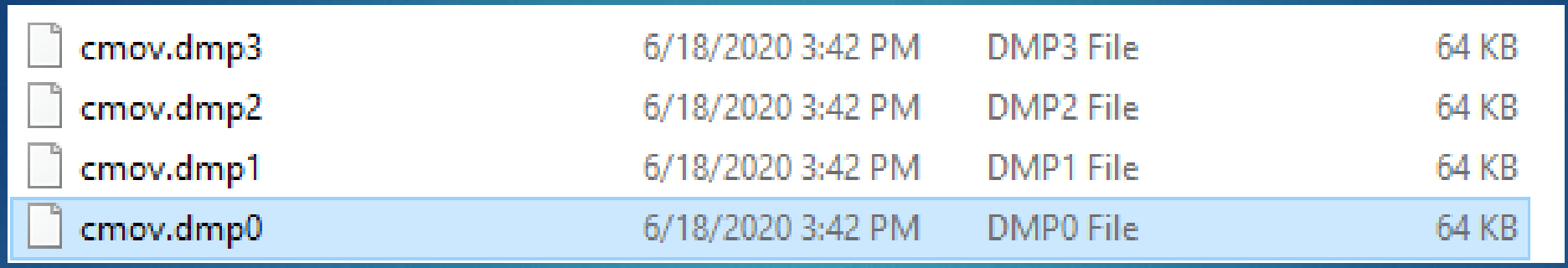
Name	Cycle Count (CYC) Packet		
Packet Format			
Dependencies	IA32_RTIT_CTL.CYCEn && TriggerEn	Generation Scenario	Can be sent at any time, though a maximum of one CYC packet is sent per core clock cycle. See Section 35.3.6 for CYC-eligible packets.
Description	The Cycle Counter field increments at the same rate as the processor core clock ticks, but with a variable length format (using a trailing EXP bit field) and a range-capped byte length. If the CYC value is less than 32, a 1-byte CYC will be generated, with Exp=0. If the CYC value is between 32 and 4095 inclusive, a 2-byte CYC will be generated, with byte 0 Exp=1 and byte 1 Exp=0. And so on. CYC provides the number of core clocks that have passed since the last CYC packet. CYC can be configured to be sent in every cycle in which an eligible packet is generated, or software can opt to use a threshold to limit the number of CYC packets, at the expense of some precision. These settings are configured using the IA32_RTIT_CTL.CycThresh field (see Section 35.2.7.2). For details on Cycle-Accurate Mode, IPC calculation, etc, see Section 35.3.6. When CycThresh=0, and hence no threshold is in use, then a CYC packet will be generated in any cycle in which any CYC-eligible packet is generated. The CYC packet will precede the other packets generated in the cycle, and provides the precise cycle time of the packets that follow. In addition to these CYC packets generated with other packets, CYC packets can be sent stand-alone. These packets serve simply to update the decoder with the number of cycles passed, and are used to ensure that a wrap of the processor's internal cycle counter doesn't cause cycle information to be lost. These stand-alone CYC packets do not indicate the cycle time of any other packet or operation, and will be followed by another CYC packet before any other CYC-eligible packet is seen. When CycThresh>0, CYC packets are generated only after a minimum number of cycles have passed since the last CYC packet. Once this threshold has passed, the behavior above resumes, where CYC will either be sent in the next cycle that produces other CYC-eligible packets, or could be sent stand-alone. When using CYC thresholds, only the cycle time of the operation (instruction or event) that generates the CYC packet is truly known. Other operations simply have their execution time bounded: they completed at or after the last CYC time, and before the next CYC time.		
Application	CYC provides the offset cycle time (since the last CYC packet) for the CYC-eligible packet that follows. If another CYC is encountered before the next CYC-eligible packet, the cycle values should be accumulated and applied to the next CYC-eligible packet. If a CYC packet is generated by a TNT, note that the cycle time provided by the CYC packet applies to the first branch in the TNT packet.		

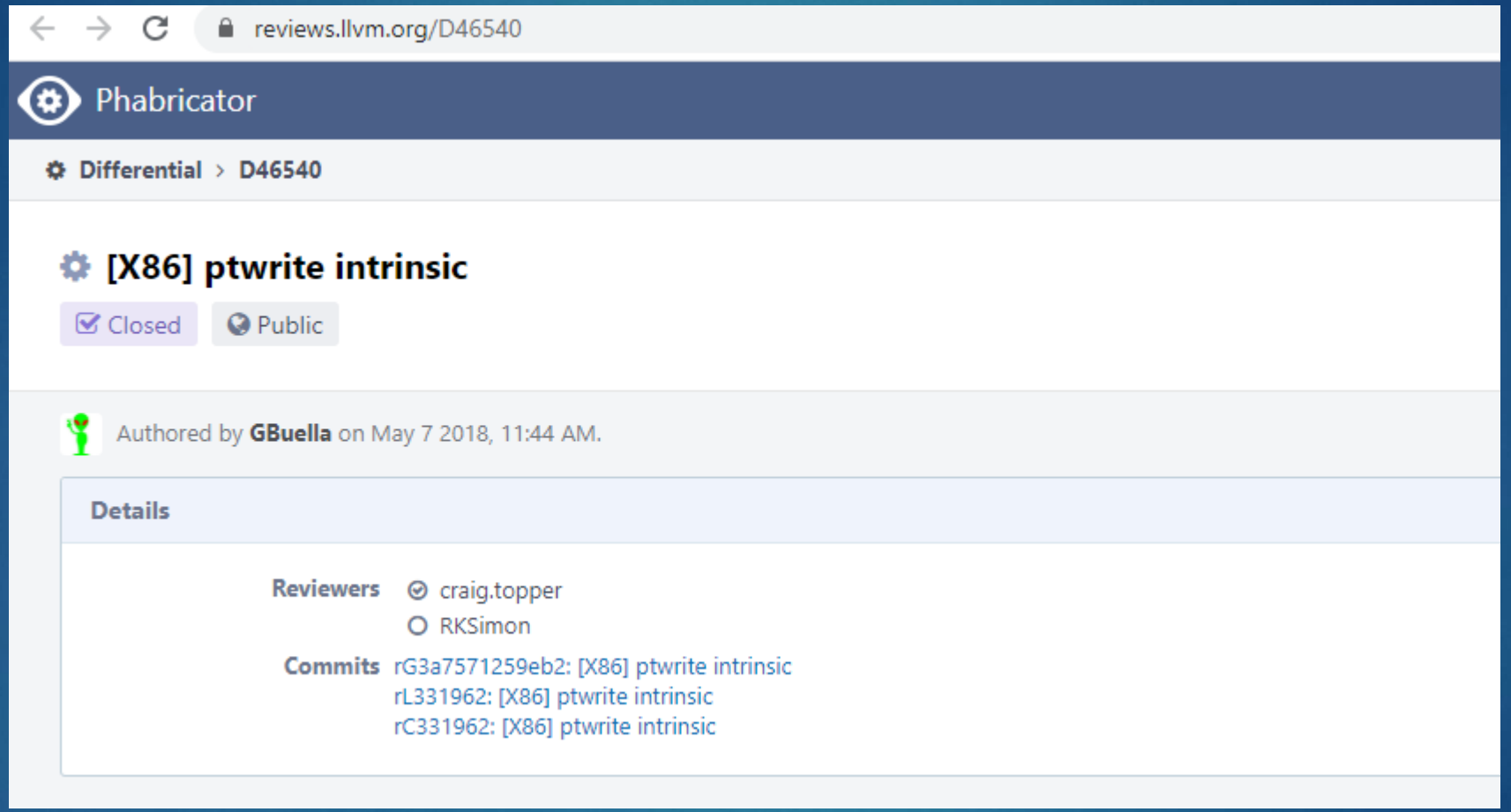
	7	6	5	4	3	2	1	0
0	Cycle Counter[4:0]					Exp	1	1
1	Cycle Counter[11:5]							Exp
2	Cycle Counter[18:12]							Exp
...	... (if Exp = 1 in the previous byte)							

TSC

```
000000000000145c  cyc        dd9
000000000000145e  tsc        125c56c37cf2
0000000000001468  cbr        11
000000000000146c  cyc        fff
000000000000146e  cyc        fff
0000000000001470  cyc        fff
0000000000001472  cyc        fff
0000000000001474  cyc        fff
0000000000001476  cyc        fff
0000000000001478  cyc        fff
000000000000147a  cyc        fff
000000000000147c  cyc        fff
000000000000147e  cyc        fff
0000000000001480  cyc        fff
0000000000001482  cyc        fff
0000000000001484  cyc        fff
0000000000001486  cyc        fff
0000000000001488  cyc        fff
000000000000148a  cyc        fff
000000000000148c  cyc        fff
000000000000148e  cyc        fff
0000000000001490  cyc        fff
0000000000001492  cyc        fff
0000000000001494  cyc        fff
0000000000001496  cyc        fff
0000000000001498  cyc        fff
000000000000149a  cyc        fff
00000000000014a0  cyc        fff
00000000000014a2  cyc        fff
00000000000014a4  cyc        fff
00000000000014a6  cyc        fff
00000000000014a8  cyc        fff
00000000000014aa  cyc        fff
00000000000014ac  cyc        fff
00000000000014ae  cyc        fff
00000000000014b0  cyc        fff
00000000000014b2  cyc        fff
00000000000014b4  cyc        fff
00000000000014bc  cyc        c3
00000000000014be  tsc        125c57445d80
00000000000014c8  cbr        11
```

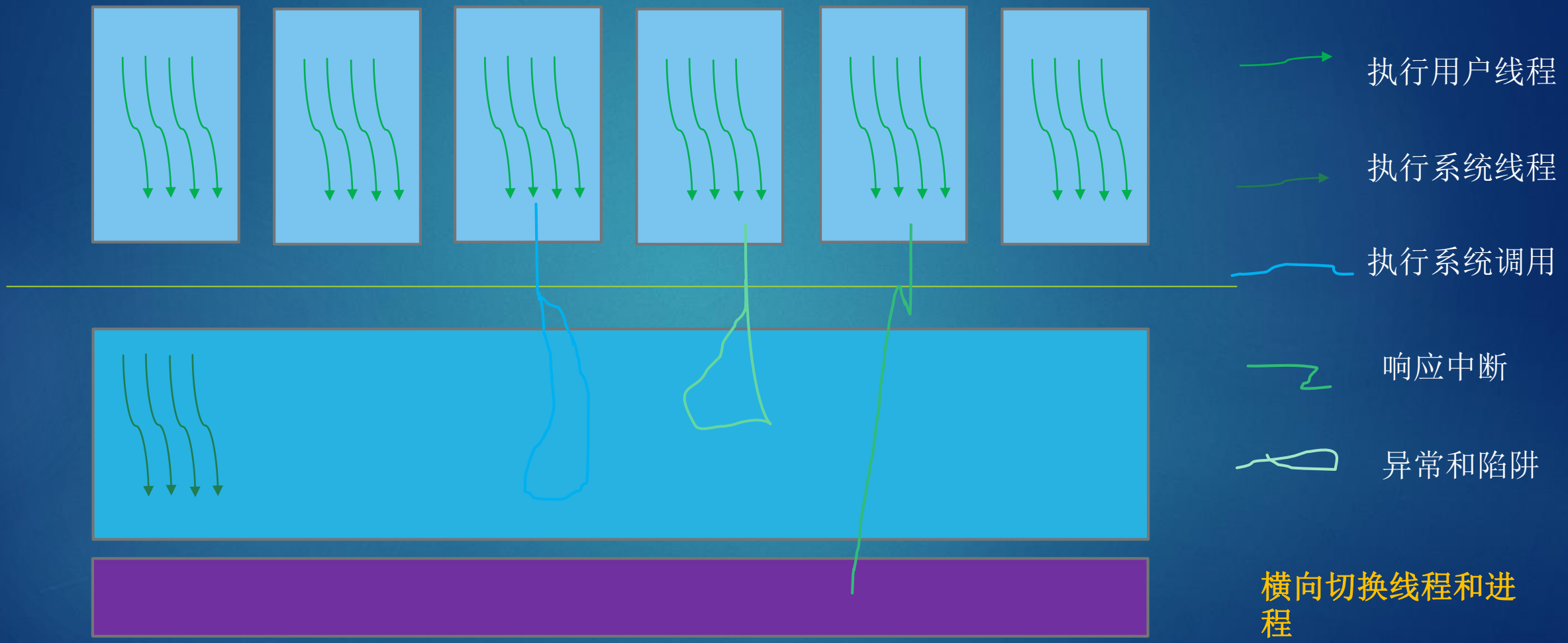
使用ptext64保存到文件




reviews.llvm.org/D46540
Phabricator
Differential › D46540
[X86] ptwrite intrinsic
Closed
Public
Authored by GBuella on May 7 2018, 11:44 AM.
Details
Reviewers
craig.topper
RKSimon
Commits
rG3a7571259eb2: [X86] ptwrite intrinsic
rL331962: [X86] ptwrite intrinsic
rC331962: [X86] ptwrite intrinsic


RTIT技术背景
通过DCI会话启用RTIT
追踪包
演示
应用

CPU的旅行



横向切换线程和进程
纵向切换执行模式

感受纳秒

- 1纳秒里，光在真空中可以行进大约30厘米

光速大约30万公里每秒 3*10 * 10^4 * 10^3m/s

1 ns = 10^ -9 s

d = 3*10^8*10^-9 =3*10^-1 m = 0.3m

绑定任务

- # taskset <掩码> ./geipt
- # taskset –c <CPU序号> ./geipt

CR3过滤

- CR3代表进程
- #define MSR_IA32_CR3_MATCH 0x00000572

!ptext64.showcr3filter

<0> 000000003a26c000 enabled
<1> 000000003a26c000 enabled
<2> 000000003a26c000 enabled
<3> 000000003a26c000 enabled

258D

.formats 258d
Evaluate expression:
  Hex: 00000000`0000258d
  Decimal: 9613
  Octal: 00000000000000022615
  Binary: 00000000 00000000 00000000 00000000 00000000 00000000 00100101 10001101

  Bit 0: TraceEn
  Bit 1: CycEn
  Bit 2: OS
  Bit 3: User
  Bit 7: Enable CR3 Filtering   // 8
  Bit 8: ToPA   // 5
  Bit 10: TSCEn
  Bit 13: BranchEn   // 2

使用ptext64解码文件

- !ptext64.decode /family 6 /model 0n78 /stepping 3 /file c:\temp\gd5.dmp

设置IP过滤

- #objdump –d geipt

```
0000000000001b7a <test_cmov>:
    1b7a:       55                      push   %rbp
    1b7b:       48 89 e5                mov    %rsp,%rbp
    1b7e:       89 7d ec                mov    %edi,-0x14(%rbp)
    1b81:       89 75 e8                mov    %esi,-0x18(%rbp)
    1b84:       89 55 e4                mov    %edx,-0x1c(%rbp)
    1b87:       89 4d e0                mov    %ecx,-0x20(%rbp)
    1b8a:       c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
    1b91:       83 7d ec 00             cmpl   $0x0,-0x14(%rbp)
    1b95:       7e 08                   jle    1b9f <test_cmov+0x25>
    1b97:       8b 45 e8                mov    -0x18(%rbp),%eax
    1b9a:       89 45 f8                mov    %eax,-0x8(%rbp)
    1b9d:       eb 06                   jmp    1ba5 <test_cmov+0x2b>
    1b9f:       8b 45 ec                mov    -0x14(%rbp),%eax
    1ba2:       89 45 f8                mov    %eax,-0x8(%rbp)
    1ba5:       8b 55 ec                mov    -0x14(%rbp),%edx
    1ba8:       8b 45 e8                mov    -0x18(%rbp),%eax
    1bab:       01 c2                   add    %eax,%edx
    1bad:       8b 45 e4                mov    -0x1c(%rbp),%eax
    1bb0:       01 c2                   add    %eax,%edx
    1bb2:       8b 45 e0                mov    -0x20(%rbp),%eax
    1bb5:       01 d0                   add    %edx,%eax
    1bb7:       99                      cltd
    1bb8:       f7 7d f8                idivl  -0x8(%rbp)
    1bbb:       89 45 fc                mov    %eax,-0x4(%rbp)
    1bbe:       83 7d fc 01             cmpl   $0x1,-0x4(%rbp)
    1bc2:       7e 05                   jle    1bc9 <test_cmov+0x4f>
    1bc4:       8b 45 e4                mov    -0x1c(%rbp),%eax
    1bc7:       eb 03                   jmp    1bcc <test_cmov+0x52>
    1bc9:       8b 45 e0                mov    -0x20(%rbp),%eax
    1bcc:       89 45 f8                mov    %eax,-0x8(%rbp)
    1bcf:       8b 45 f8                mov    -0x8(%rbp),%eax
    1bd2:       5d                      pop    %rbp
    1bd3:       c3                      retq
```

使用!ptext64设置IP过滤

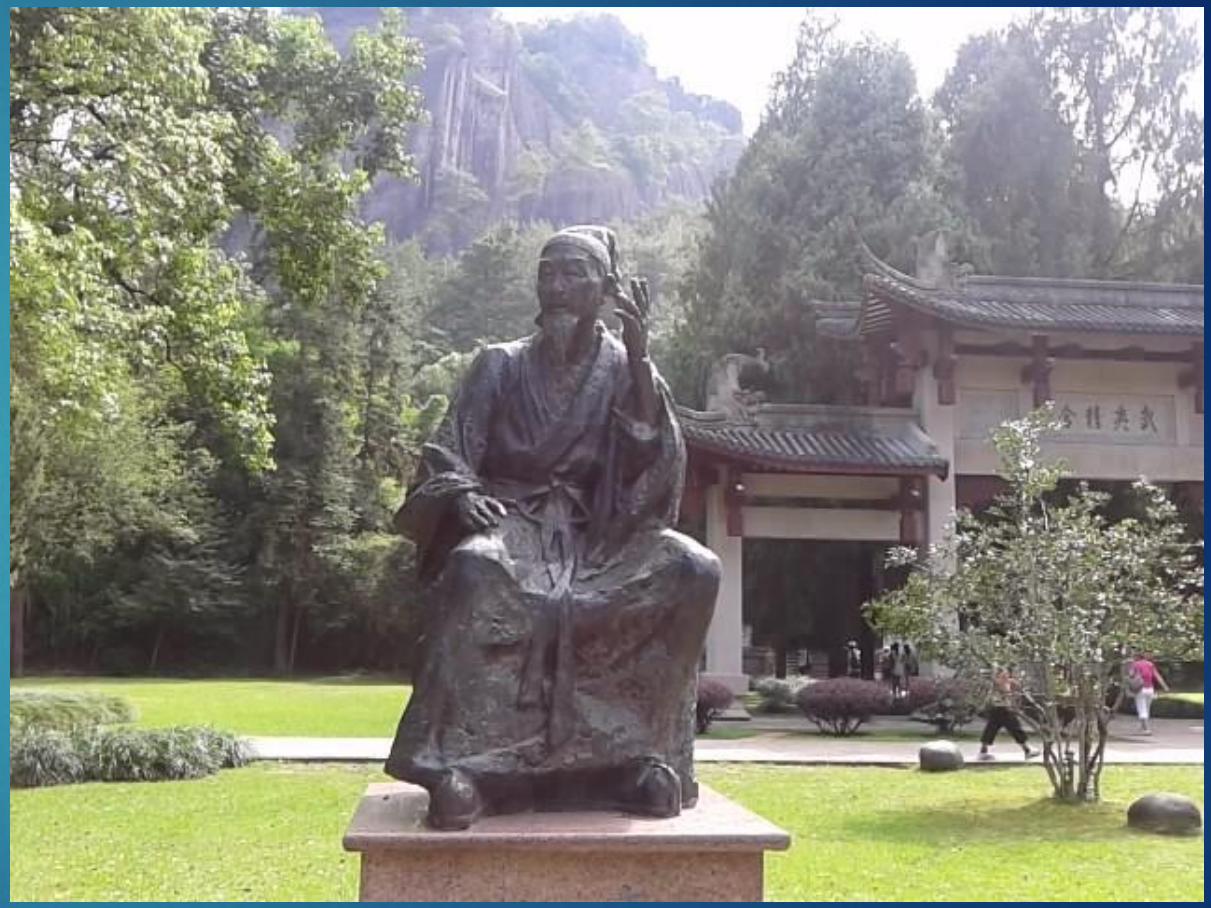
!ptext64.setipfilter 0x555555554840 0x555555554862

```
<0>   0: 000055555554840 - 000055555554862   enable
<1>   0: 000055555554840 - 000055555554862   enable
<2>   0: 000055555554840 - 000055555554862   enable
<3>   0: 000055555554840 - 000055555554862   enable
```

用途很多

千头万绪，终归一理

宋·朱熹 (1130-1200)



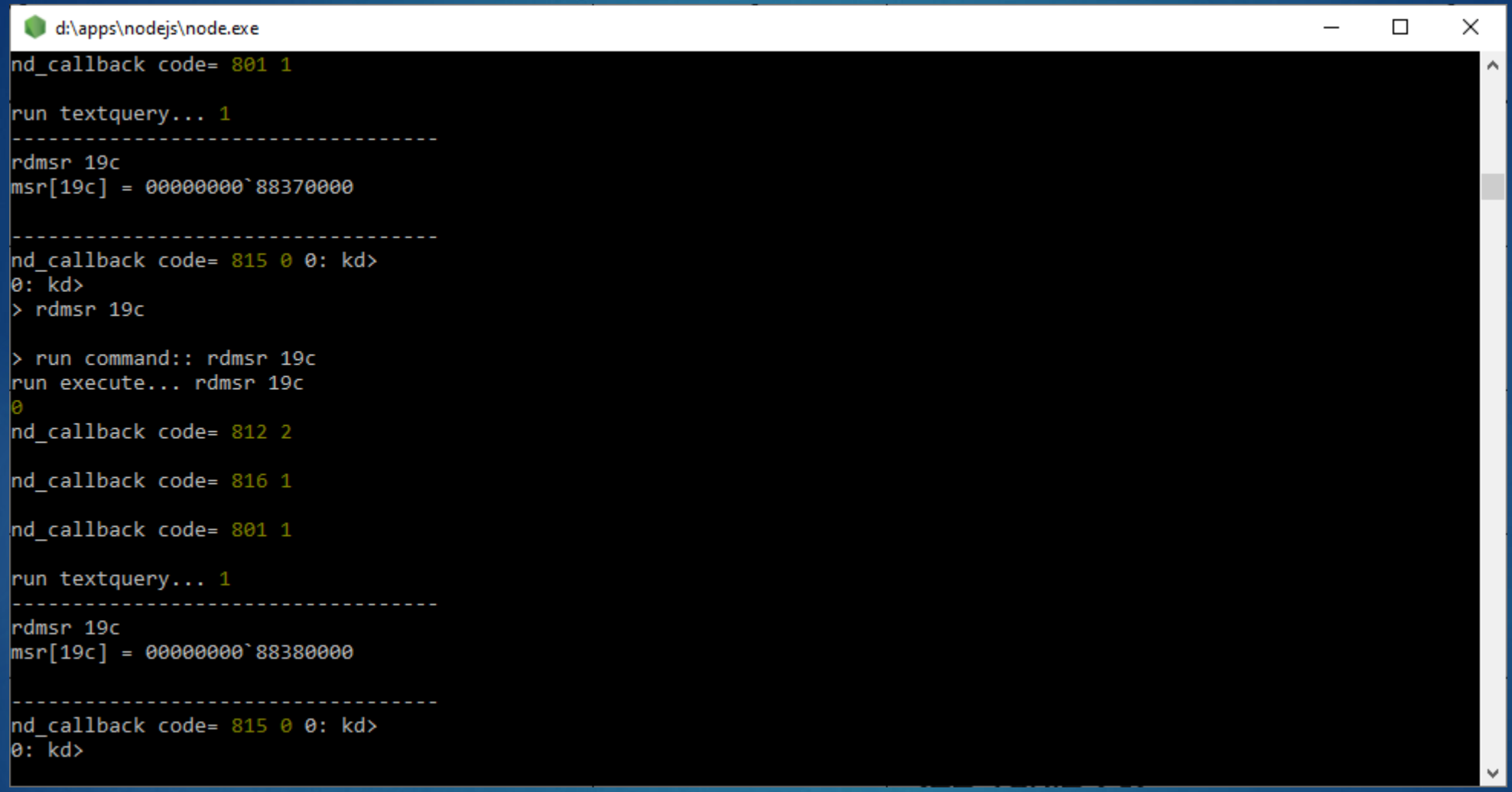
所谓致知在格物者，言欲致吾之知，在**即物**而穷其理也。盖人心之灵莫不有知，而天下之物莫不有理，惟于理有未穷，故其知有不尽也。是以《大学》始教，必使学者即凡天下之物，莫不因其已知之理而益穷之，以求至乎其极。至于用力之久，而一旦豁然**贯通**焉，则众物之表里精粗无不到，而吾心之全体大用无不明矣。此谓物格，此谓知之至也。

问答

http://www.nanocode.cn

http://advdbg.org/gdk

```
unsigned long pid_to_cr3(int pid)
{
    struct task_struct *task;
    struct mm_struct *mm;
    void *cr3_virt;
    unsigned long cr3_phys;

    task = pid_task(find_vpid(pid), PIDTYPE_PID);

    if (task == NULL)
        return 0; // pid has no task_struct

    mm = task->mm;

    // mm can be NULL in some rare cases (e.g. kthreads)
    // when this happens, we should check active_mm
    if (mm == NULL) {
        mm = task->active_mm;
    }

    if (mm == NULL)
        return 0; // this shouldn't happen, but just in case

    cr3_virt = (void *) mm->pgd;
    cr3_phys = virt_to_phys(cr3_virt);

    return cr3_phys;
}
```


d:\apps\nodejs\node.exe
nd_callback code= 801 1
run textquery... 1
rdmsr 19c
msr[19c] = 00000000`88370000
nd_callback code= 815 0 0: kd>
0: kd>
> rdmsr 19c
> run command:: rdmsr 19c
run execute... rdmsr 19c
0
nd_callback code= 812 2
nd_callback code= 816 1
nd_callback code= 801 1
run textquery... 1
rdmsr 19c
msr[19c] = 00000000`88380000
nd_callback code= 815 0 0: kd>
0: kd>

PGD: 0xffff988f7a6ac000

CR3: 3a6ac000

dq ffffffffb1217010
ffffffff`b1217010  00000000`1fc00000

```
unsigned long __phys_addr(unsigned long x)
{
    if (x >= __START_KERNEL_map) {
        x -= __START_KERNEL_map;
        VIRTUAL_BUG_ON(x >= KERNEL_IMAGE_SIZE);
        x += phys_base;
    } else {
        VIRTUAL_BUG_ON(x < PAGE_OFFSET);
        x -= PAGE_OFFSET;
        VIRTUAL_BUG_ON(!phys_addr_valid(x));
    }
    return x;
}
```

? 0xffff988f7a6ac000-ffff800000000000
Evaluate expression: 27004513206272 = 0000188f`7a6ac000

```
static inline unsigned long __phys_addr_nodebug(unsigned long x)
{
	unsigned long y = x - __START_KERNEL_map;

	/* use the carry flag to determine if x was < __START_KERNEL_map */
	x = y + ((x > y) ? phys_base : (__START_KERNEL_map - PAGE_OFFSET));

	return x;
}
```