

presented by



Using Performance Measurement Tool to Optimize UEFI Drivers and Systems

UEFI PlugFest – May 13-15, 2014

Seattle, WA

Presented by Jeff Bobzin (Insyde Software)

Agenda



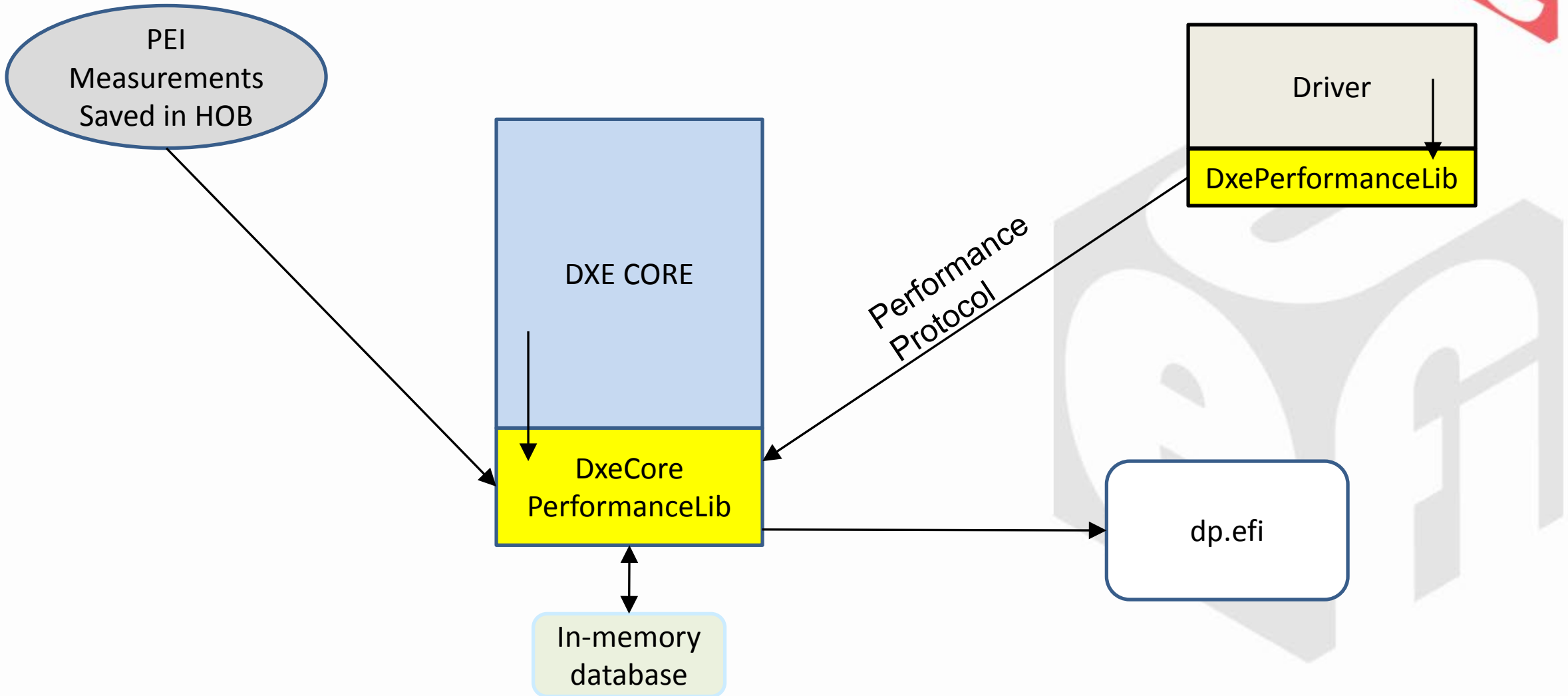
- What is PerformancePkg?
- How Does It Work?
- Real World Example
- What if my IHV test platform does not have a PERF ROM available?
- Late-Loading PERF capability

Performance Measuring Package



- PerformancePkg is available Open Source -
 - Part of EDK2 project on sourceforge.net (*but not in UEFI spec*)
- What can it do?
 - Insert low-overhead measurement start/stop hooks in driver
 - (*low, but not zero overhead. Keep this in mind when nesting.*)
 - Duration measurements based on high accuracy CPU timer
 - Ongoing measurements are stored in central buffer
 - Later able to display results in UEFI Shell

PerformanceProtocol in Operation



PerformanceLib.h Defines PERF Macros



PERF_START_EX(Handle, Token, Module, TimeStamp, Identifier)

<i>Handle</i>	Useful Identifier from context
<i>Token</i>	ASCII String used to id component
<i>Module</i>	ASCII String used to id module
<i>TimeStamp</i>	[OPTIONAL] Caller-supplied start time (use current if 0)
<i>Identifier</i>	Additional 32-bit constant used for class of similar measurement

ALTERNATE:

PERF_START(Handle, Token, Module, TimeStamp)

Equal to PERF_START_EX with *Identifier* as 0

End Measurements



PERF_END_EX(Handle, Token, Module, TimeStamp, Identifier)

PERF_END(Handle, Token, Module, TimeStamp)

PERF_END_EX, is matched to PERF_START_EX by Handle, Token, Module, Identifier

PERF Macros In Code



DxeMain.c

```
//  
// Initialize the DXE Dispatcher  
//  
PERF_START (NULL,"CoreInitializeDispatcher", "DxeMain", 0) ;  
CoreInitializeDispatcher ();  
PERF_END (NULL,"CoreInitializeDispatcher", "DxeMain", 0) ;
```

Dp.efi
output

```
==[ General ]=====
```

Index	Name	Description	Time(us)
3:		PreMem	2174540
55:		PostMem	376006
56:		DisMem	17215
66:	DxeMain	CoreInitializeDispatcher	3337

MACRO On/Off



This PCD turns On/Off the PERF MACROs, so they can remain in production source

`gEfiMdePkgTokenSpaceGuid.PcdPerformanceLibraryPropertyMask | 1`



dp.efi output snippets



DP Build Version: 2.3
System Performance Timer Frequency: 2,296,380 (KHz)

==[Major Phases]=====

SEC Phase Duration: 28701 (us)
PEI Phase Duration: 2556 (ms)
DXE Phase Duration: 15432 (ms)
BDS Phase Duration: 1215 (ms)
Total Duration: 19231 (ms)

==[Drivers by Handle]=====

Index:	Handle	Driver Name	Description	Time(us)
712:	[9E]	HeciDxe	StartImage:	14656754
757:	[A0]	VariableRuntimeDxe	StartImage:	2394
868:	[AB]	SmbiosDxe	StartImage:	1820
1024:	[BD]	PiSmmIpl	StartImage:	2355
1097:	[D7]	BootScriptExecutorDxe	StartImage:	1699



Special Measurement Categories



==[General]=====

Index	Name	Description	Time(us)
3:		PreMem	2174540
55:		PostMem	376006
56:		DisMem	17215
66:	DxeMain	CoreInitializeDispatcher	3337
67:	DxeMain	CoreDispatcher	15041351
786:		StartImage:	2558
4165:	BDS	PlatformBds	1213860
54745:		PostBDS	284242

==[Cumulative]=====

(Times in microsec.)					
Name	Count	Cumulative Duration	Average Duration	Shortest Duration	Longest Duration
LoadImage:	206	36707	178	12	8653
StartImage:	203	14780092	72808	2	14656754
DB:Start:	63	1063861	16886	0	519179
DB:Support:	54298	82441	1	0	57195

Adding A Cumulative Category



In dp.c

```
/// Items for which to gather cumulative statistics.
PERF_CUM_DATA CumData[] = {
    PERF_INIT_CUM_DATA (LOAD_IMAGE_TOK),
    PERF_INIT_CUM_DATA (START_IMAGE_TOK),
    PERF_INIT_CUM_DATA (DRIVERBINDING_START_TOK),
    PERF_INIT_CUM_DATA (DRIVERBINDING_SUPPORT_TOK)
    //add your custom cumulative here, and rebuild dp.efi
};
```



Is PERF Useful for IHVs?



- For System PERF, PerformanceProtocol must be enabled when DxeMain is built
- So what if no PERF build available for my test platform?
- Can IHV use PerformancePkg to measure IHV driver?
- Usage Goal:
 1. Boot To Shell
 2. Run Driver to Start Performance Protocol
 3. Test IHV Driver in Shell

Insyde Reveals – Enable PERF ‘on-the-fly’



PerfDxe.c

```
#include <Uefi.h>
#include <Library/PerformanceLib.h>
EFI_STATUS
EFIAPI
PerfDxeInit(
    IN EFI_HANDLE          ImageHandle,
    IN EFI_SYSTEM_TABLE    *SystemTable
)
{
    return EFI_SUCCESS;
}
```

The Driver C Source is simple. How does this enable PERF?

PerfDxe.inf



```
[Defines]
  INF_VERSION           = 0x00010006
  BASE_NAME             = PerfDxe
  FILE_GUID             = 8C270BB5-CEBB-4A39-BAFB-5BCA79303C77
  MODULE_TYPE          = UEFI_DRIVER
  VERSION_STRING        = 1.0
  ENTRY_POINT          = PerfDxeInit
#
# The following information is for reference only and not required by the build tools.
#
# VALID_ARCHITECTURES   = IA32 X64 IPF EBC
#
[Sources]
  PerfDxe.c

[Packages]
  MdePkg/MdePkg.dec
  MdeModulePkg/MdeModulePkg.dec
  PerformancePkg/PerformancePkg.dec

[LibraryClasses]
  UefiDriverEntryPoint
  PerformanceLib
```

Nothing unusual in the inf either!

PerformancePkg.dsc Overrides Are the Key!



```
[Components]
```

```
PerformancePkg/Dp_App/Dp.inf
```

```
//add lines below ...
```

```
PerformancePkg/PerfDxe/PerfDxe.inf {
```

```
<LibraryClasses>
```

```
UefiDriverEntryPoint|MdePkg/Library/UefiDriverEntryPoint/UefiDriverEntryPoint.inf
```

```
TimerLib|PerformancePkg/Library/TscTimerLib/DxeTscTimerLib.inf
```

```
PerformanceLib|MdeModulePkg/Library/DxeCorePerformanceLib/DxeCorePerformanceLib.inf
```

```
<PcdsFixedAtBuild>
```

```
gEfiMdePkgTokenSpaceGuid.PcdPerformanceLibraryPropertyMask|1
```

```
gEfiMdeModulePkgTokenSpaceGuid.PcdMaxPeiPerformanceLogEntries|64 }
```

```
build -p PerformancePkg\PerformancePkg.dsc -a X64 -t VS2010x86 -b DEBUG
```

PerfDxe.c – final version



```
#include <Uefi.h>
#include <Library/UefiBootServicesTableLib.h>
#include <Library/PerformanceLib.h>
EFI_STATUS
EFIAPI
PerfDxeInit(
    IN EFI_HANDLE          ImageHandle,
    IN EFI_SYSTEM_TABLE    *SystemTable
)
{
    EFI_STATUS      Status;
    EFI_EVENT       Timeout;
    UINTN           Index;
    Status = gBS->CreateEvent (EVT_TIMER,0,NULL,NULL,&Timeout);
    PERF_START_EX(ImageHandle,"CALIBRATE","PerfDxe",0,0);
    Status = gBS->SetTimer (Timeout, TimerRelative, 10000000);
    gBS->WaitForEvent(1,&Timeout,&Index);
    PERF_END_EX(ImageHandle,"CALIBRATE","PerfDxe",0,0);
    return EFI_SUCCESS;
}
```

Final Version Includes Calibration!

dp.efi output after load of PerfDxe.efi



```
==[ Drivers by Handle ]=====
```

Index:	Handle	Driver Name	Description	Time(us)
1:	[1CC]	PerfDxe	CALIBRATE	991985
2:	[1CF]	MyDriver	START	39567

Summary



- PerformancePkg is a EDK-II tool
- Measurement start/stop macros are added to driver source
- Helps to Identify slow sections
- If test platform is not PERF enabled, there is a Shell-Loading PERF capability

For more information on the
Unified EFI Forum and UEFI
Specifications, visit
<http://www.uefi.org>



presented by

