Samsung R&D Institute Russia
Center of System Software

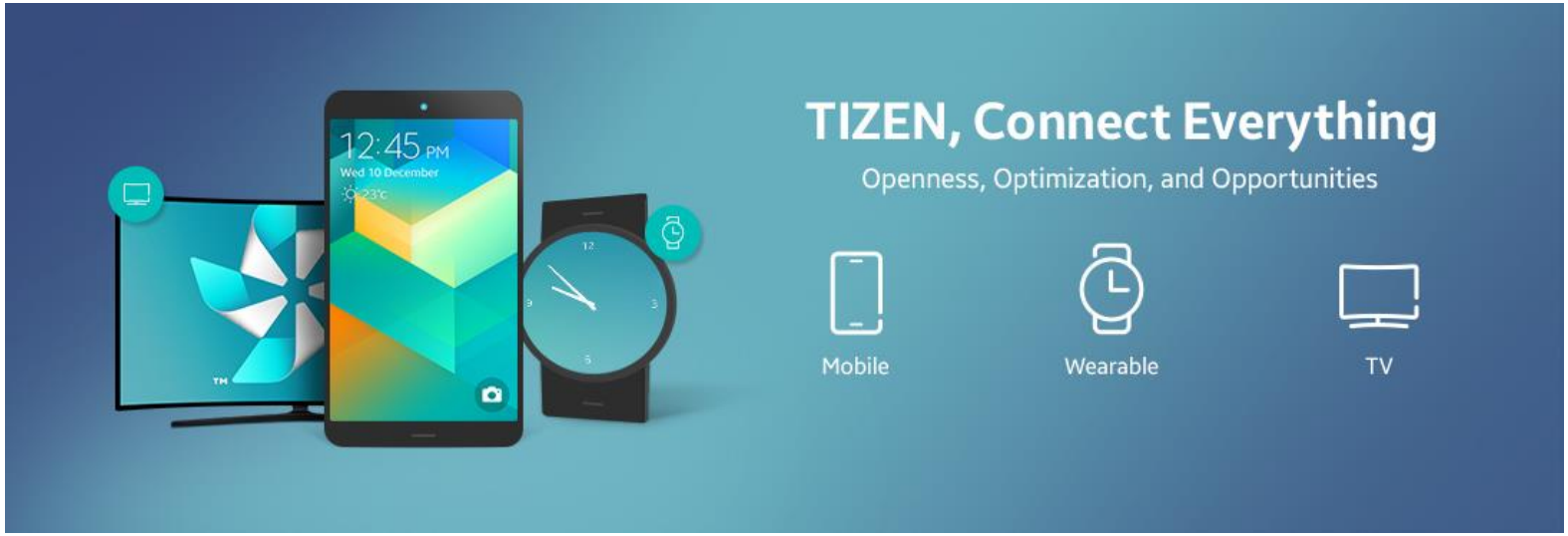
Debugging and Profiling .NET applications in Tizen OS

Dmitri Botcharnikov

Agenda

- What is Tizen
- .NET for Tizen
- Tizen Extension for Visual Studio
- Tizen .NET Debugger internals
- Tizen .NET Profiler internals
- Future plans

Tizen OS



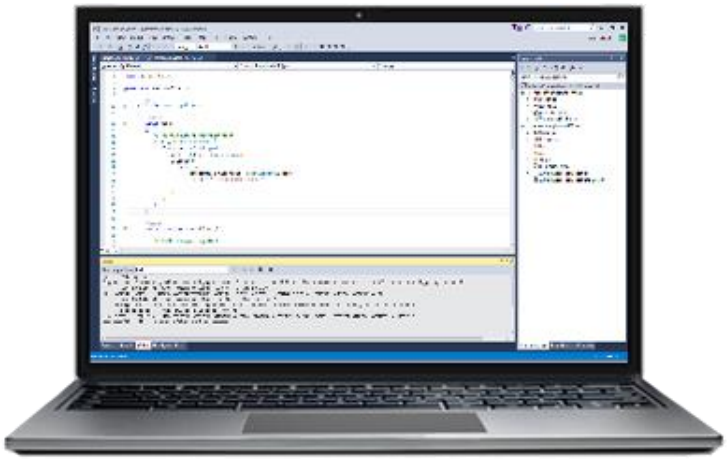
- Open OS based on Linux: kernel + libraries
- Runs on million devices: Smart TV, Smart Watches, Smartphones
- Flexible, configurable

Application Development

HTML5 OR C and EFL ?

- Visual Studio integration
- Tizen Emulator
- Xamarin.Forms
- .NET Core
- Tizen platform-specific API

TIZEN Xamarin Visual Studio .NET



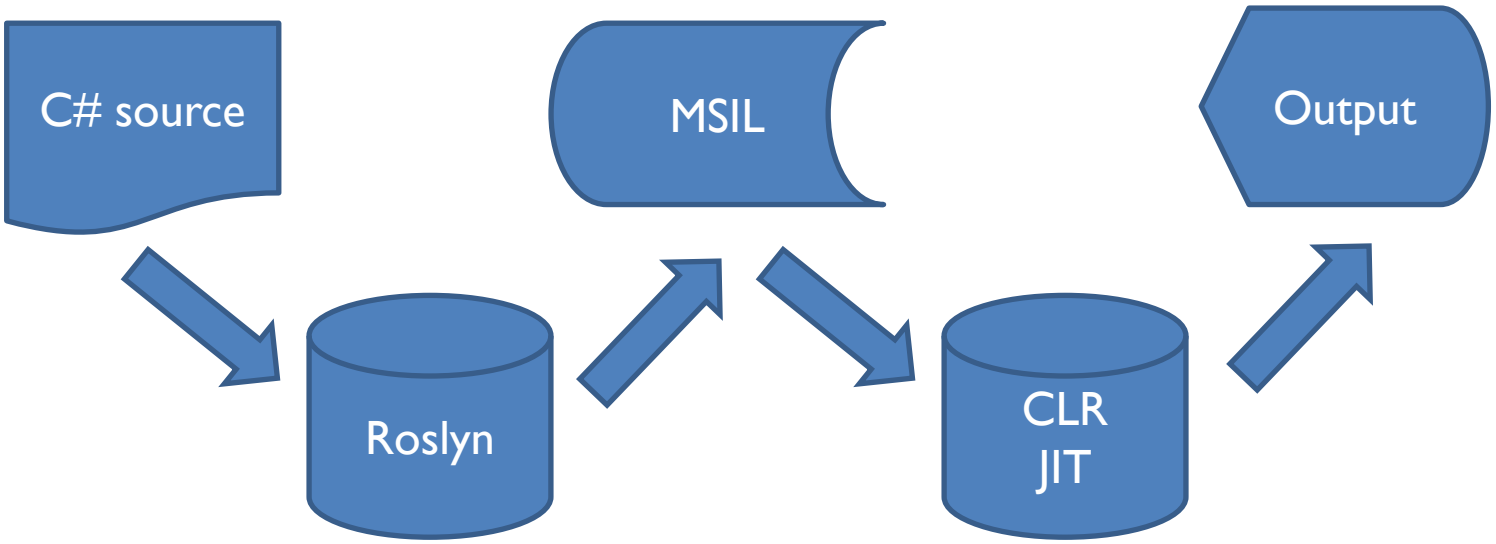
Tizen .NET for Visual Studio

- Application templates
- Emulator Manager
- Certificate Manager
- Smart Debug Bridge
- **.NET Debugger for Tizen**
- **.NET Profiler for Tizen**
- http://developer.tizen.org

.NET Debugger for Tizen

- Challenges in debugging dynamic languages
- Debugger architecture
- Components of .NET Debugger
  - GDB JIT
  - GDB/MI
- Demo

C# Compilation & Execution

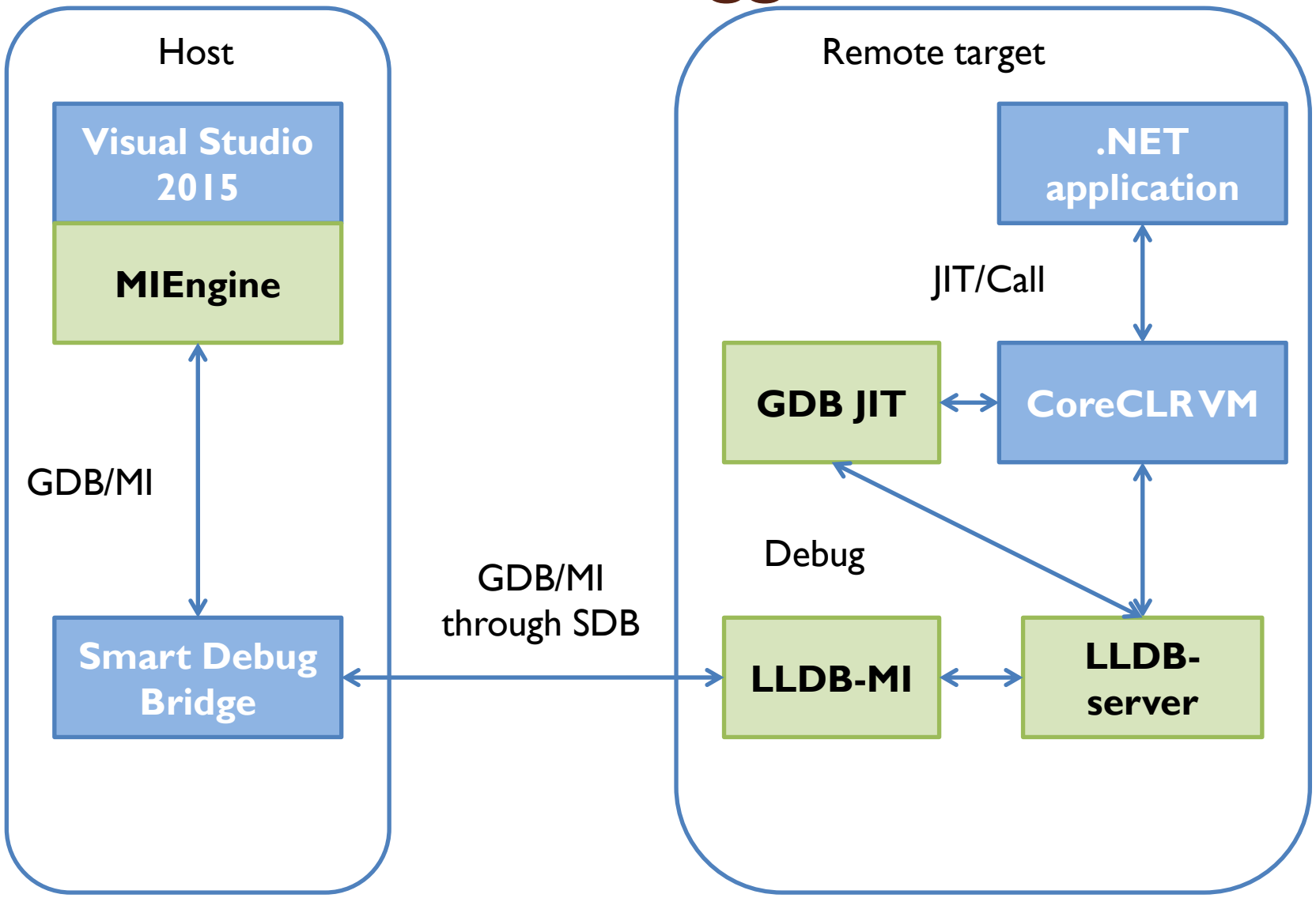


- Language-specific compiler: C# => MSIL
- CLR JIT compiler: MSIL => native code

Debugging Challenges

- Source code to native code mapping
  - C# compiler generates debugging information for source code to MSIL mapping
- Stepping in and over
  - Stepping into not yet compiled code
  - Managed exception handlers
  - Lambdas, closures & iterators
- Local variables & arguments inspection
  - C# compiler generates debugging information for MSIL variables

Tizen .NET Debugger



LLDB

- Subproject of LLVM (http://lldb.llvm.org)
- Native debugger builds on LLVM and Clang libraries
- Supports X86 and ARM architectures

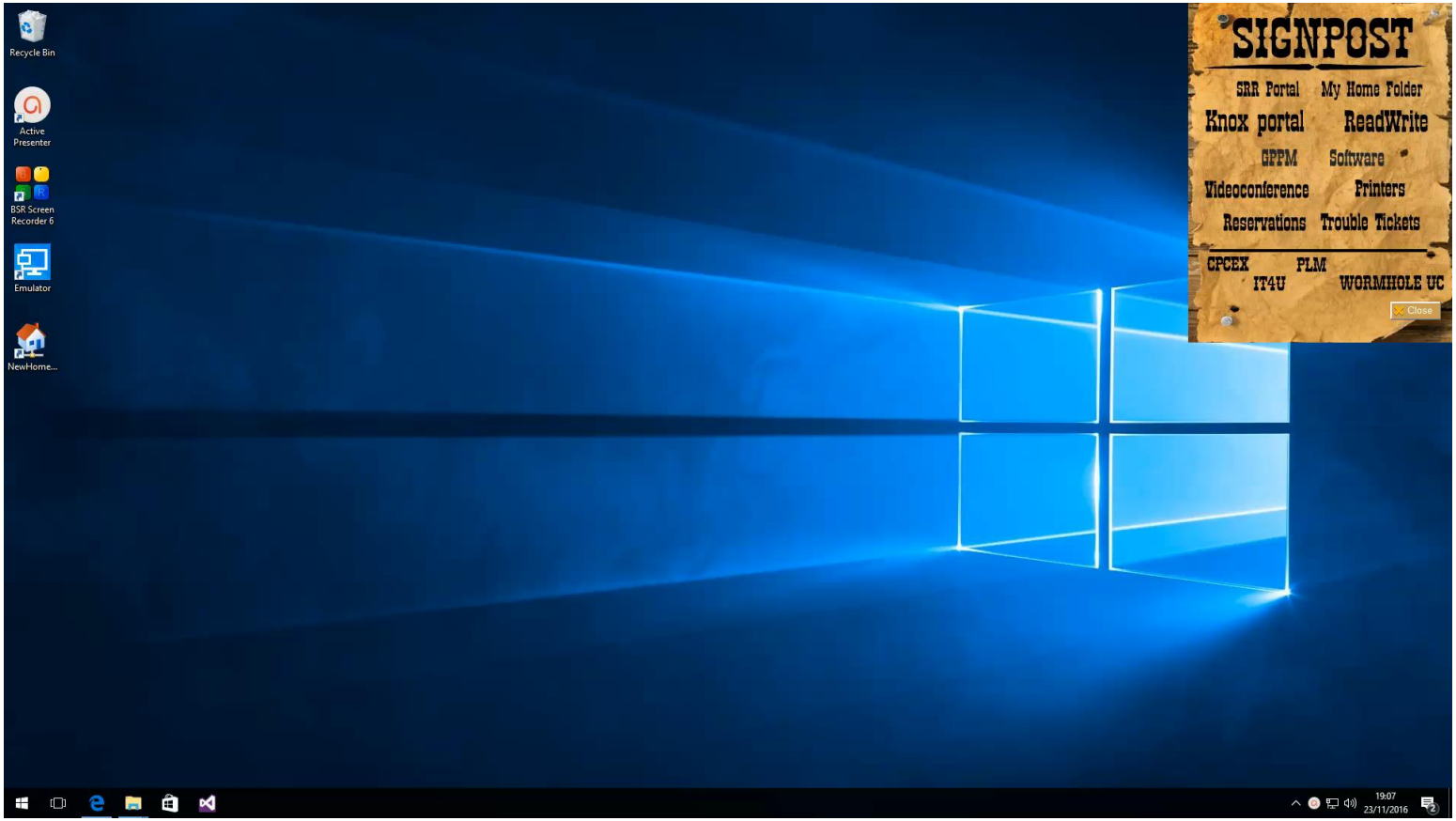
GDB JIT Interface

- Interface for registering JITed code with debuggers
- VM should construct in-memory ELF+DWARF image and call predefined function
  - __jit_debug_register_code
- Debugger puts breakpoint on this function
- On breakpoint hit loads constructed image and resume execution
- GBD JIT drawbacks:

GDB/MI & Microsoft MIEngine

- GDB/MI: machine oriented text interface
- Supported by Eclipse CDT, Emacs & others
- Visual Studio MI Debug Engine is an open source VS extension that provides support for GDB/MI
- Modified to support Tizen Application Framework

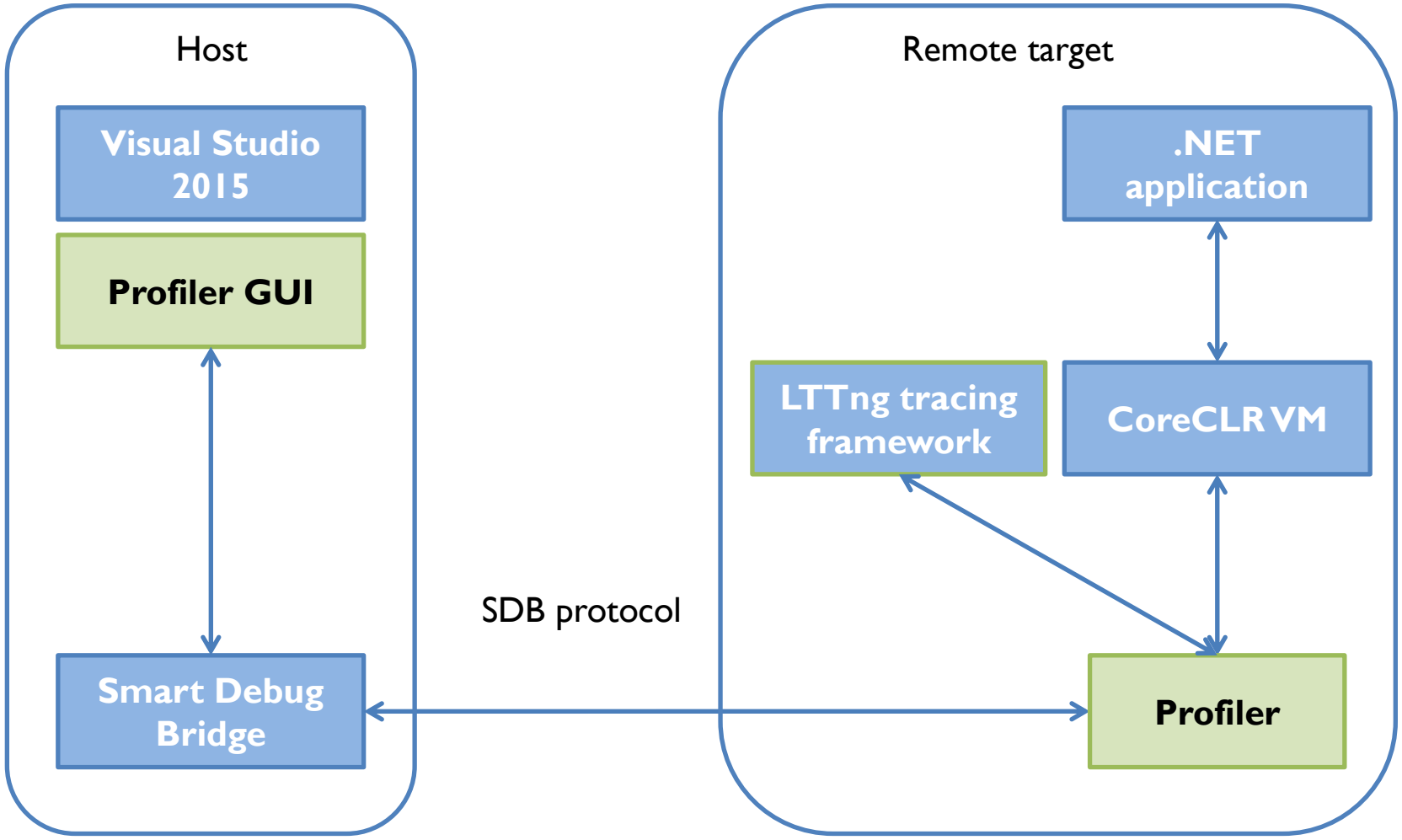
Demo time


Recycle Bin
Active Presenter
BSR Screen Recorder 6
Emulator
NewHome...
SIGNPOST
SRR Portal
My Home Folder
Knox portal
ReadWrite
GPPM
Software
Videoconference
Printers
Reservations
Trouble Tickets
CPCEX
PLM
IT4U
WORMHOLE UC
Close
19:07
23/11/2016

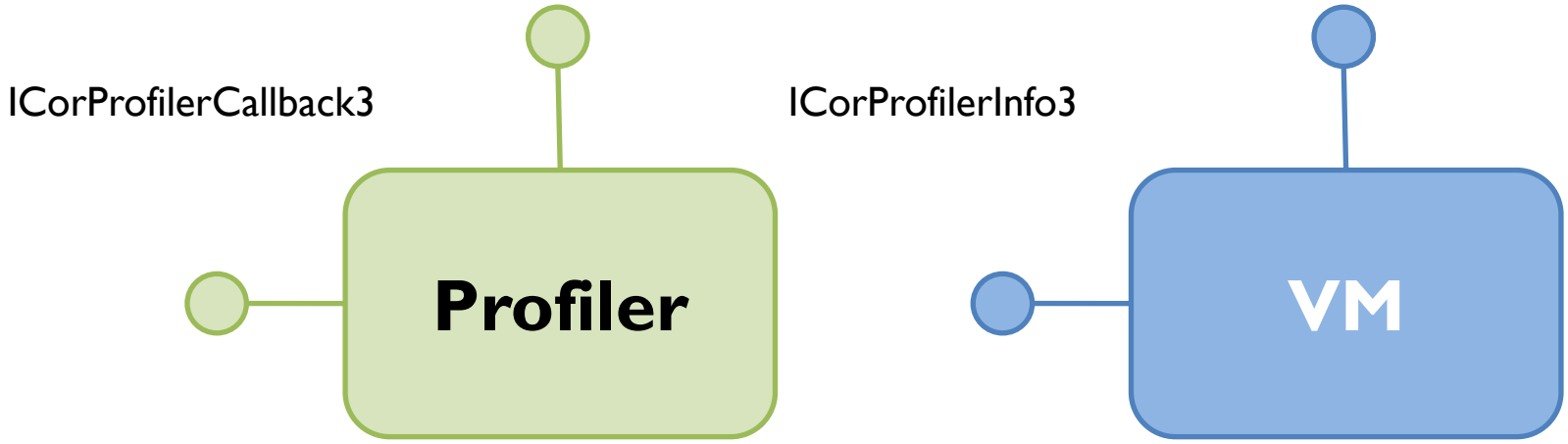
.NET Profiler for Tizen

- Profiler architecture
- .NET Profiler infrastructure
- Linux Trace Toolkit Next Generation
- Demo

Profiler Architecture



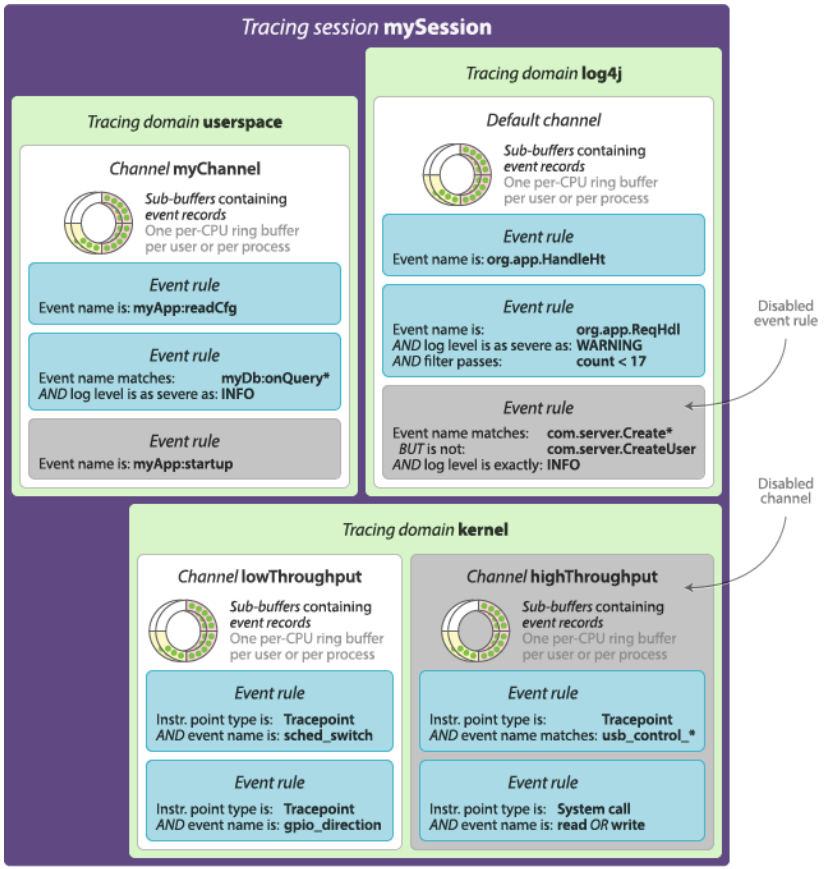
.NET Profiling Infrastructure



- CoreCLR expects profiler to implement ICorProfilerCallback
- VM calls profiler through this interface at appropriate time
- Profiler can use ICorProfilerInfo for more info

Linux Trace Toolkit Next Generation

- LTTng is an open source toolkit for tracing kernel, applications and libraries
- VM generate events collected by session daemon
- http://lttng.org



Demo time

Results & Future Plans

- Tools run on Z300 ARM Tizen phone and on x86/x86_64 Tizen simulators
- Finish development of C# language type plug-in and .NET runtime plug-in for LLDB
  - Get LLDB knows about C# type system
  - Generic instantiation types available during method execution
  - Better support for CoreCLR stubs
- Develop full-fledged Historical debugger
- Refine profiler implementation

Thank you!

Dynamically compiled languages

- Dynamically (Just-In-Time) compiled languages
- VM manages low-level details: memory allocation, exception handling
- But for debuggers…

SOS debugger plug-in

- Plug-in for LLDB (libsosplugin.so, libsos.so)
- Port of SOS.dll (SOS Debugging extension) to Linux platform
- Provides low-level information about internals of CLR environment
- Useful for CoreCLR developers, but not so for application developers

GDB JIT: Pro & Cons

- Pro
  - Supported by both GDB and LLDB
  - Integrated into debugger infrastructure
  - The easiest way to add support for JITed language
- Cons
  - Invasive (only needed for debugging)
  - Memory consuming (~700 b on ARM, ~1kb on x86_64)
  - Inherently static: generated before execution

Stepping over and in

- Stepping in and over
  - Stepping into still not compiled code
  - Managed exception handlers: stack unwinding
  - Lambdas, closures & iterators
- CoreCLR implements calls through stubs dispatch which is dynamically changed
- Solution
  - Generate symbols for stubs in GDB JIT in-memory image
  - Modify LLDB thread plans to follow these symbols

Visual Studio Extension

- Profiler control to start/pause/stop execution of app under profiler
- Collection of profiler info from target
- Profiler GUI for parsing and display collected info

Historical debugging PoC

- Allows you to move backward and forward through the execution of your application and inspect its state
- Implemented in CoreCLR through ICorProfiler interface
- Requires implementation of platform-specific profiler hooks (OS + arch)
- Developed Proof-of-Concept realization for ARM & x86_64 Linux