Erlang-DTrace

Garry Bulmer

Team DTrace: Tim Becker

What I'm going to talk about

- Introduction to DTrace & DTrace Architecture
- Demo of DTrace with ‘one liners’
- Erlang-Dtrace Vision & ‘Fit’
- Erlang VM Architecture
- Current Erlang DTrace Scope
- Erlang-DTrace Demo
- Future

What is DTrace?

*"DTrace is a comprehensive dynamic tracing facility ...
that can be used by administrators and developers on **live production systems** to examine the behavior of both **user programs** and of the **operating system** itself.*

*DTrace enables you to **explore** your system to understand how it works, track down performance problems **across many layers of software**, or locate the cause of aberrant behavior.*

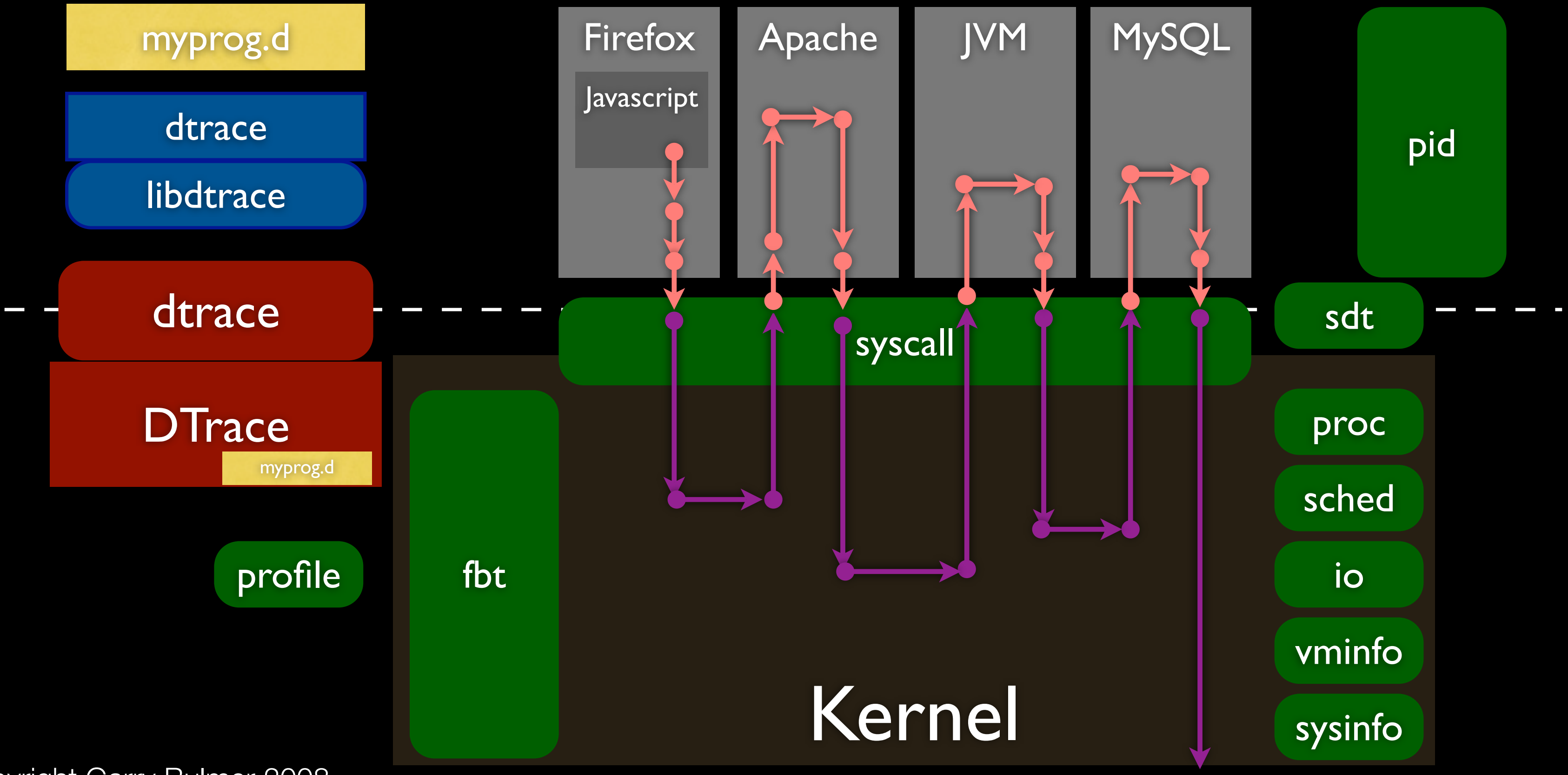
*DTrace lets you create your own **custom programs** to **dynamically instrument** the system and provide immediate, **concise answers to arbitrary questions**"*

Source: Sun Microsystems "Solaris Dynamic Tracing Guide"

How does DTrace work?

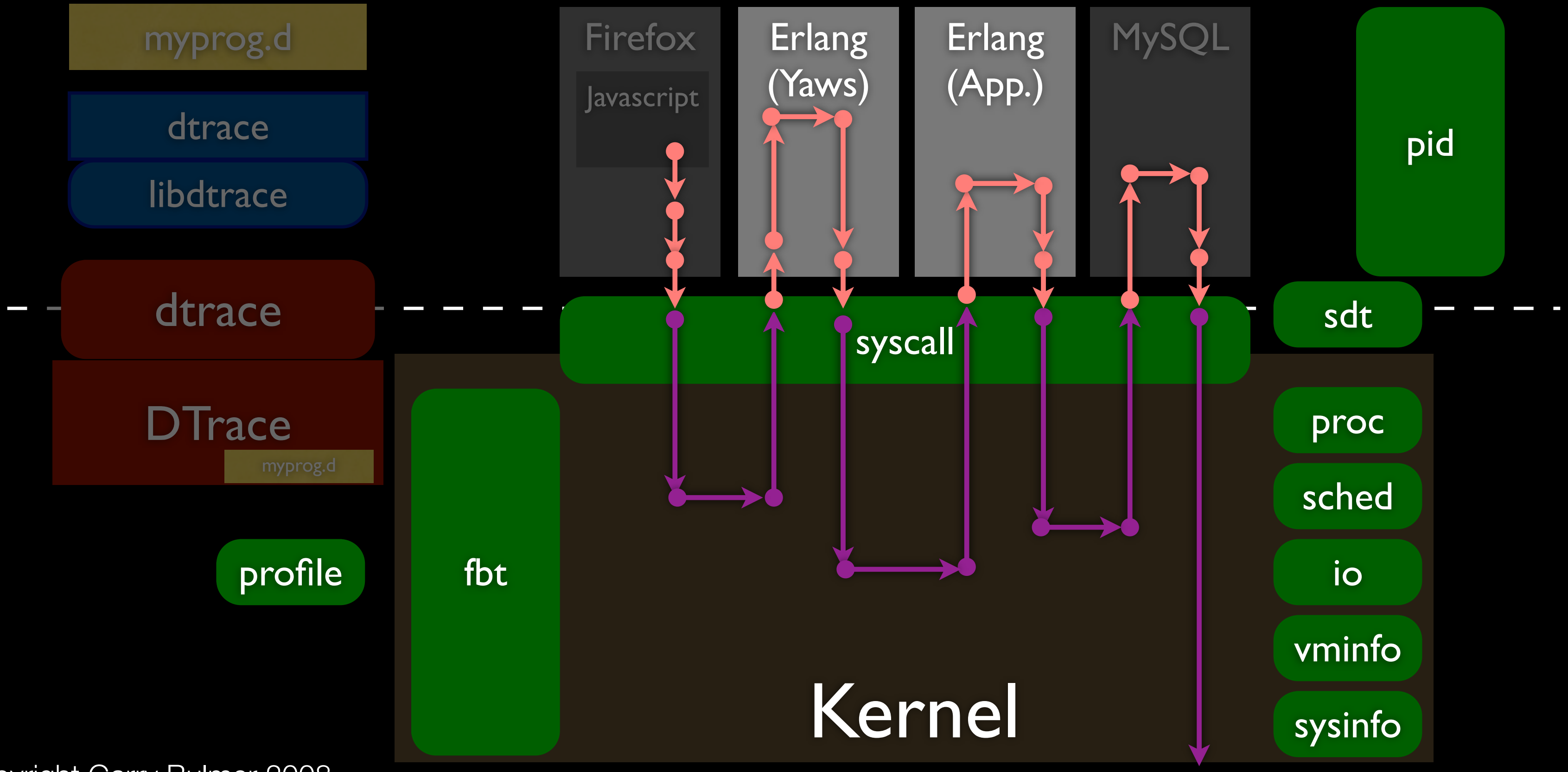
- KEY: Dynamically enabled - even in Production
- Probes within OS kernel - ‘Zero cost’ when disabled
- ‘Providers’ - subsystem managing a group of Probes
  - Probes observe events, and capture data
  - Providers forward events and data to ‘D programs’
- User applications - observed by ‘PID’ Provider
  - Probes observe function entry, exit & parameters

DTrace End-to-End

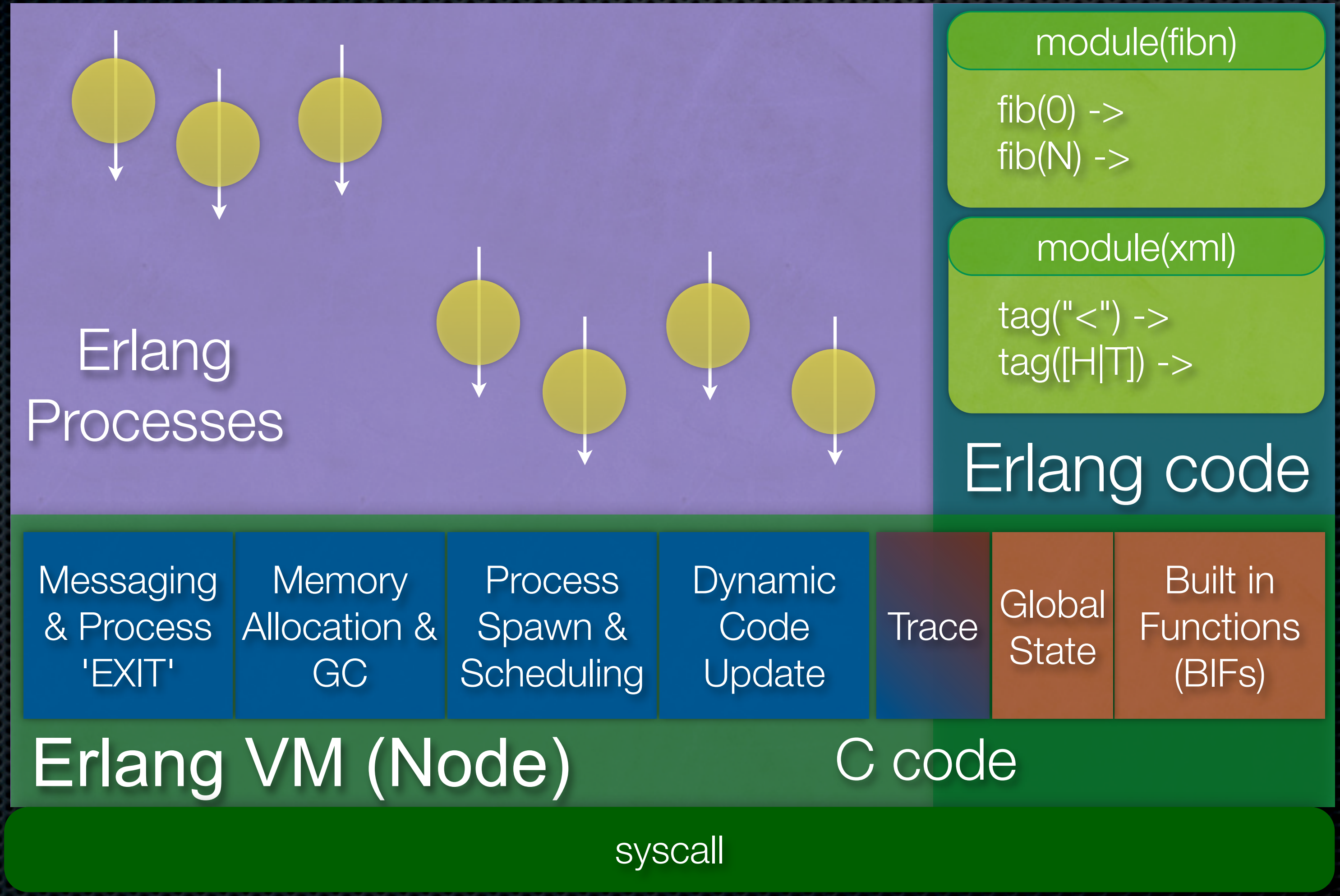


DTrace Demo - ‘one liners’

Erlang-DTrace End-to-End



Erlang VM Architecture



Erlang’s DTrace ‘Fit’

- DTrace ‘PID’ Provider can observe C programs
  - Good: Erlang VM is C
  - Bad: user needs to understand Erlang VM internals !
- Erlang VM-managed, Fine-Grain ‘Process’
  - Erlang processes are invisible to DTrace
- Erlang data is dynamically typed
  - DTrace uses static ‘C-style’ data types
- Erlang scripts are opaque data to DTrace

Erlang DTrace Implementation

- DTrace Statically Defined Tracing (SDT) Probes
  - Insert SDT probes (C) into Erlang VM
- Probes in key parts of Erlang VM
  - Process management, GC, Messaging, Code Load ...
  - ‘Decode’ Erlang scripts (?)
- Add new DTrace functions for Erlang Developers

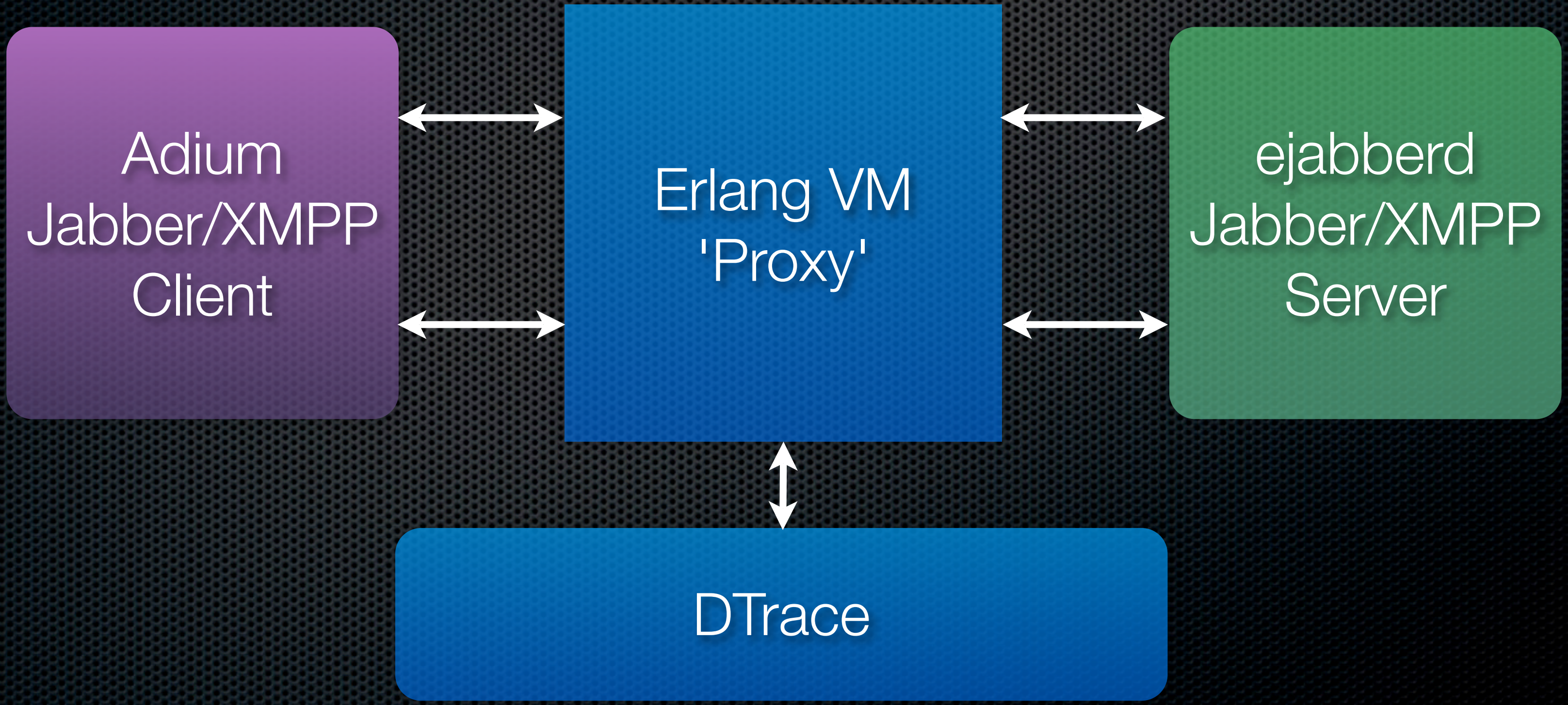
Erlang has Dynamic Tracing !

- Aim to complement, not replace
- Longer term integrate Erlang tracing and DTrace
  - Provide Erlang DTrace interface functions
  - Exploit Erlang’s Dynamic Code Update
    - Can load Erlang code in production

V002 Erlang-DTrace Scope

- New DTrace BIFs (explicitly use DTrace probes in Erlang)
- Statically Defined Tracing Probes inserted into Erlang VM
  - Processes, Memory (GC),
  - Global State (Registry)
- Use Erlang VM Trace facilities from Erlang DTrace BIF’s

Erlang-Dtrace Demo



'Proxy' Code

Future Directions

- Better use of existing Erlang Trace facilities
  - Dynamic DTrace Probes
- Correlate Messages across Erlang Processes
- Like to handle Erlang Data Types (e.g. Lists) in DTrace ...
  - ... and not flatten to strings in probe code
  - Dynamic DTrace Type extensions
- Distributed/Clustered DTrace (one day ...)

More Information

- Erlang-DTrace google group
- Source hosted at opensolaris.org
- Co-developer is Tim Becker
- Thanks to Bryan Cantrill, Sun Microsystems for encouragement and support

Questions