

# DTrace for Java

**The most advanced operating system on the planet**

이창재 (eggboy@gmail.com)

OpenSolaris kr lead, KSUN

# Overview

- ✓ Dtrace Introduction & Architecture
- ✓ Provider & Probe
- ✓ D-language
- ✓ Why DTracing Java?
- ✓ JVMPI & JVMTI 소개
- ✓ Hotspot provider 소개
- ✓ Chime 및 jdtrace 프로젝트 소개

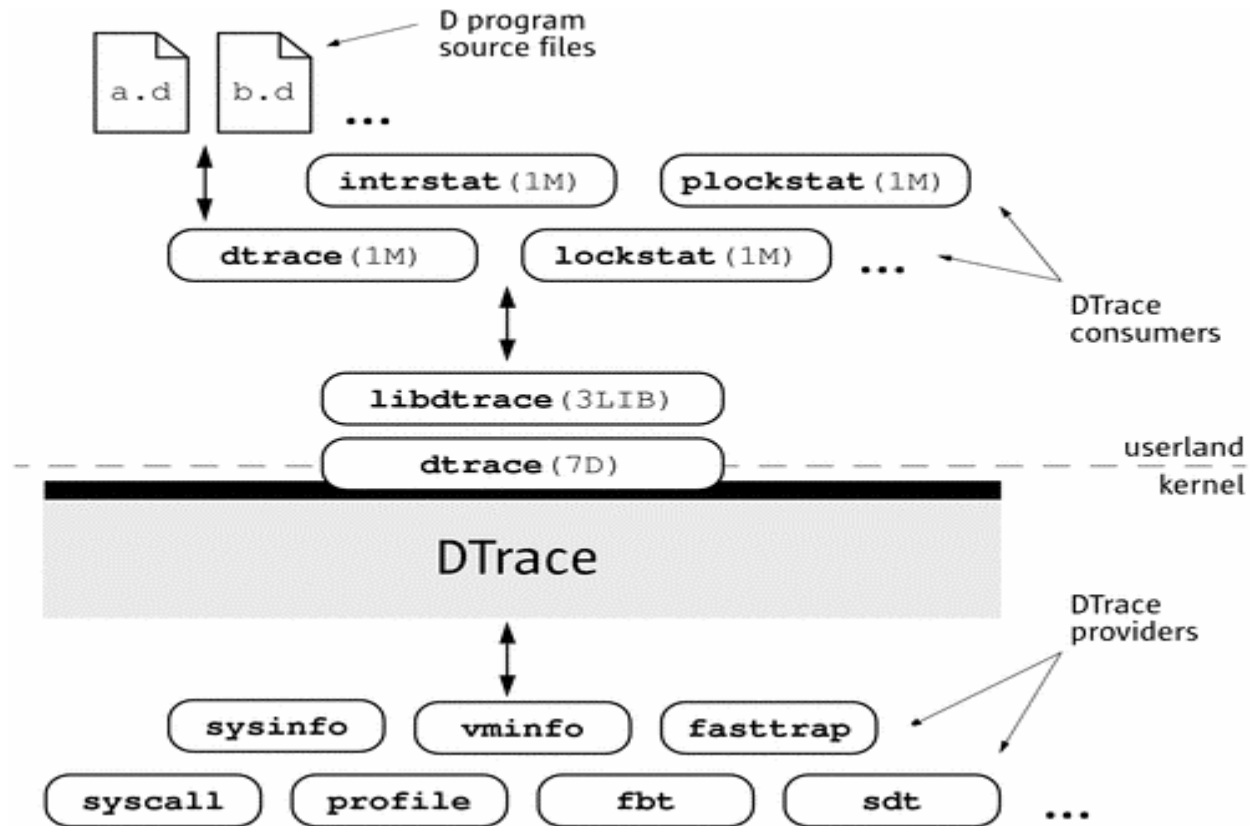
# DTrace Introduction

- ✓ Solaris 10 에 처음 소개
- ✓ 동적 추적 시스템
  - ✓ 추적하려는 코드에 어떠한 변환도 불필요
- ✓ 실시간 동작
  - ✓ 아무런 중단 없이 probe 의 삽입
- ✓ 시스템의 전체 layer 모니터링 가능
  - ✓ 유저 어플리케이션 부터 커널까지

# DTrace Introduction(Cont'd)

- ✓ Production 시스템을 위해 디자인됨
- ✓ 사용하지 않을 때에는 어떠한 퍼포먼스 저하도 없음
- ✓ 완벽한 안정성 – **panic** 이나 **crash** 유발이 구조적으로 불가능함

# DTrace Architecture



# Probe 와 Provider

- ✓ DTrace 의 정보는 Probe 를 통해 수집됨
- ✓ 이러한 Probe 는 Provider 에 의해 생성됨
- ✓ Probe 는 시스템 전체를 모니터링 할 수 있는 프로그래밍 가능한 센서
- ✓ Provider 는 기본적으로 Probe 를 생성하는 커널 모듈임

# Providers

- ✓ 각각 독립적으로 동작함
- ✓ 다수의 Provider 가 존재
  - ✓ System call (app/kernel interface)
  - ✓ Function boundary call(entry, exit)
  - ✓ I/O Events (started, done)
  - ✓ Profiling

# Providers (계속)

- ✓ Solaris 는 기본적으로 16 개 의 provider 를 제공함
  - ✓ function boundary tracing(fbt) : 커널 내 함수의 entry 및 return
  - ✓ system call tracing (syscall): 시스템 호출 테이블
  - ✓ lockstat : 커널내 동기화 변수들(유저 레벨은 plockstat)
  - ✓ profile : 시간 기반으로 지속적으로 이벤트를 발생
  - ✓ 이외에 pid, profile, sdt, sched, io, proc, vminfo, sysinfo 등등이 존재

# Probe

**provider:module:function:name**

- ✓ Provider
  - ✓ 해당 Probe 를 만든 Provider (예: fbt)
- ✓ Module
  - ✓ probe 가 속한 모듈. 커널일때에는 모듈의 이름 (예: ufs), 어플리케이션일때에는 library 의 이름 (예: lib.c)

# Probe (계속)

```
provider:module:function:name
```

- ✓ function
  - ✓ Probe 와 연관된 함수 이름. 커널의 경우 `ufs_write()`, `clock()`. 유저 어플리케이션의 경우 `libc` 의 `printf()`
- ✓ name
  - ✓ probe 의 이름. 예를 들어 커널 함수의 경우 `entry`, `return I/O probe` 의 경우 `start 스케줄링 probe` 의 경우 `on-cpu`

# D-language

- ✓ Dtrace 만을 위한 C 와 비슷한 언어로 awk 와 비슷한 사용 방법
- ✓ execname, timestamp 같은 내장 변수 제공
- ✓ 조건식을 이용한 수집 데이터의 선별 가능
- ✓ 수집된 데이터에 대한 액션 및 스택의 백 트래이스 그리고 특정 부분에서 어플리케이션의 중지 가능함

# DEMO

간단한 DTrace 스크립트 실행

# Why DTracing Java?

- ✓ 시스템의 재 시작이나 소스의 수정이 필요 없음
- ✓ 소프트웨어의 전 스택(자바 코드, JVM, native 코드, 커널 등) 을 전부 하나의 툴로 추적할 수 있는 강력한 기능
- ✓ D 언어를 이용하여 프로그래밍 적으로 모니터링이 가능

# DTrace for Java 의 역사

- ✓ Solaris10 출시 당시에는 `jstack()` 만 지원
- ✓ Pre-Mustang(Java 6.0 이전) 버전을 위한 VM Agent 프로젝트가 `java.net` 에서 진행
- ✓ Mustang 버전 부터는 Solaris 용 JVM 에서 probe 를 native 하게 지원(hotspot provider)

# JVMPI, JVMTI 소개

- ✓ DTrace Probe 를 VM Agent 의 도움을 받아서 JVM 에 load 하는 방식
- ✓ JNI 를 이용한 방식
- ✓ J2SE 1.4.2 는 JVMPI(JVM Profiler Interface)
- ✓ J2SE 5.0 은 JVMTI(JVM Tool Interface)

# JVMPI, JVMTI 소개(계속)

- ✓ dvmpci agent 는 5.0 혹은 이전 버전에서 사용 가능
- ✓ dvmpci agent 는 experimental version
- ✓ dvmti agent 는 5.0 혹은 그 이상 버전에서 사용 가능

# DEMO

JVMTI agent 사용 예

# Hotspot Providers

- ✓ J2SE 6 부터 JVM 에 native 하게 embed 되서 제공(솔라리스)
- ✓ hotspot, hotspot\_jni 두개의 provider 제공
- ✓ 500 개 정도의 probe 가 제공됨

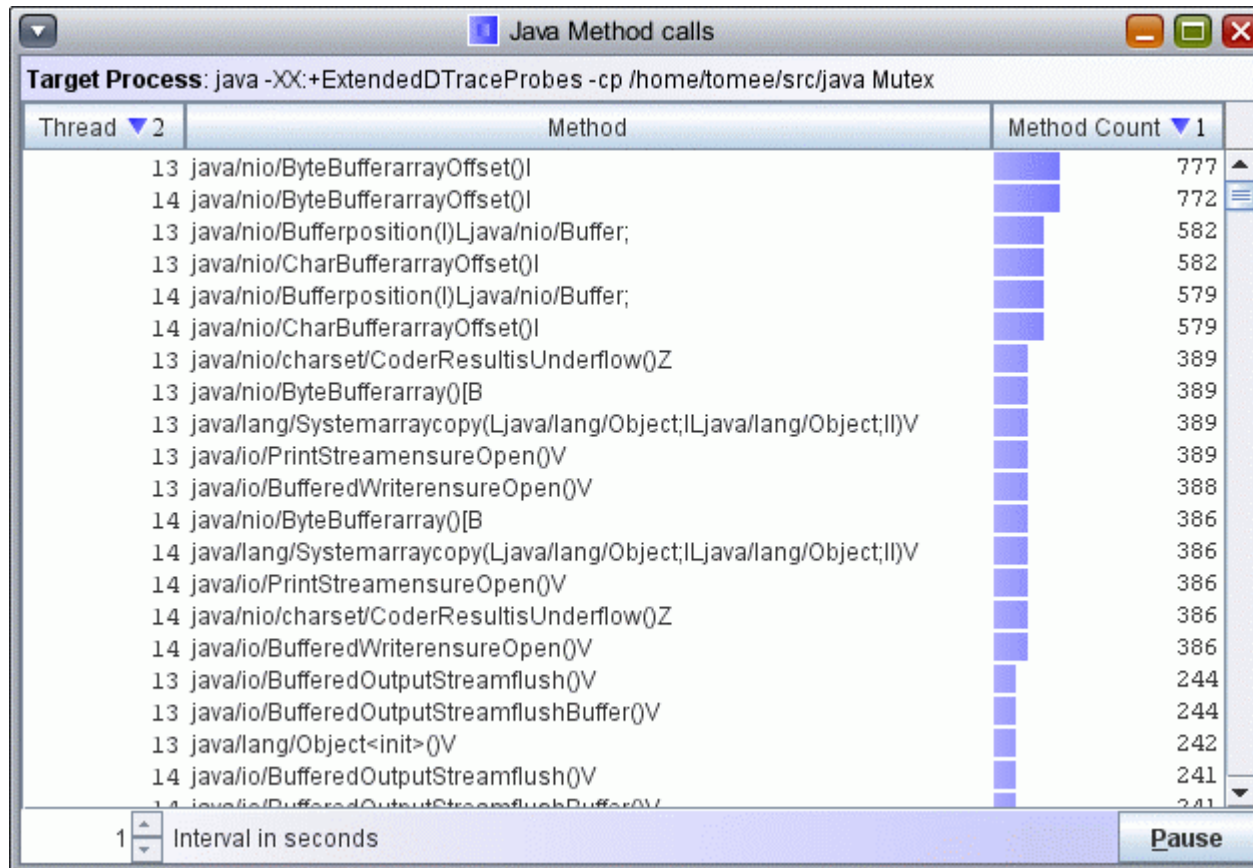
# 주요 Hotspot Probes

<i>probe 이름</i>	<i>설명</i>
class-loaded	A class loaded
class-unloaded	A class unloaded
method-entry	A method begins
method-return	A method completed
object-alloc	An object was allocated
gc-begin	System wide GC begins
gc-end	System wide GC ended
thread-start	A thread has started
thread-stop	A thread completed

# DEMO

hotspot provider 사용 예

# Chime : OpenSolaris Project



# Q & A

# References

- ✓ OpenSolaris.org
- ✓ SDN & SKDN
- ✓ Bryan Cantrill, Adam Leventhal Blog

# 감사합니다

**The most advanced operating system on the planet**

이창재 (eggboy@gmail.com)

OpenSolaris kr lead, KSUN