

Seminar: Leistungsanalyse unter Linux

Performance analysis using **GPerfTools and LTTng**

Heye Vöcking

March 27, 2012

Contents

- Great Performance Tools
 - Overview and History
 - Tools offered for profiling and analysis
- Linux Trace Toolkit next generation
 - Overview and History
 - Tracing using Tracepoints
 - Analysis
- Live demonstration

Profiling and Tracing

- What?
 - Collecting information during execution
- Why?
 - Understanding software for debugging / tuning (esp. on multicore systems)
- How?
 - “Great Performance Tools” and “Linux Trace Toolkit next generation”

Great Performance Tools

- Distributed under BSD license
- Community run
- Libraries are available for Windows, Linux, Solaris, and Mac
- Supports C/C++ and all languages that can call C code

GPerfTools – History

- Originally called “Google Performance Tools”
- Started in March 2005
- Version 1.0 early 2009
- Version 2.0 since February 2012



GPerfTools – History

- Main contributor “csilvers” stepped down since v. 2.0
- Google withdrew ownership → now completely community run
- Renaming to “gperftools”, where “g” stands for “great”
- Main developer from now on is David Chappelle

GPerfTools – Overview

- CPU Profiler – performance of functions
- TCMalloc – fast, thread aware malloc
更快的，线程感知的malloc
- Heap Leak Checker – **memory leak detector**
这部分代码和头文件是C++的，C语言好像用不了
- Heap Profiler – record program stack
记录程序栈

GPerfTools – CPU Profiler

注意：CPU相关的分析器

- To use the CPU Profiler you have to
 - Link the library with `-lprofiler` 这个动态库，
 - Set `$CPUPROFILE` to the path where to save the profile
 - Surround the code to be profiled with
 - 分析器开始
 - `ProfilerStart("profile name")`
 - `ProfilerStop()` 分析器停止
- Output can be analyzed with `pprof` (we'll get back to `pprof` later)

使用pprof工具进行对输出文件的分析

GPerfTools – TCMalloc

- To use TCMalloc simply link the library with `-ltcmalloc`
- Faster than `clib` malloc
- Low overhead on small objects
- Reduces thread lock contention in multithreaded environments
- The `Heap Leak Checker` and `Heap Profiler` work with TCMalloc

内存泄漏检测也是C++的

GPerfTools – Heap Leak Checker

- To use the Heap Leak Checker you have to
 - Link the library with `-ltcmalloc`
 - Set `$HEAPCHECK` to the desired mode
 - minimal, normal, strict, draconian, as-is, local
- Output can be analyzed with `pprof` (we'll get back to `pprof` later)

GPerfTools – Heap Profiler

堆分析器

- To use the **Heap Profiler** you have to
 - Link the library with `-ltcmalloc`
 - Set `$HEAPPROFILE` to the path where to save the profile
 - Surround the code to be profiled with
 - `HeapProfilerStart("prefix name")`
 - `HeapProfilerStop()`
- Output can be analyzed with `pprof` (we'll get back to `pprof` later)

GPerfTools – Profiler Insights

获取栈帧

此接口找不到

- Function of GetStackFrames () :

```
main() { foo(); }
```

```
foo() { bar(); }
```

```
bar() {
```

```
    void* result[10];
```

```
    int sizes[10];
```

```
    int depth = GetStackFrames(result,sizes,10,1); }
```

```
[root@localhost api]# gcc GetStackFrames.c -lprofiler -  
ltcmalloc_and_profiler  
/tmp/cc1CwkCH.o: 在函数‘main’中:  
GetStackFrames.c:(.text+0x3e): 对‘GetStackFrames’未定义的引用  
collect2: 错误: ld 返回 1
```

- Result:

- depth = 2

- result[0] = foo, sizes[0] = 16

- result[1] = bar, sizes[1] = 16

GPerfTools – pprof

- Analyzes profiles generated by

CPU分析器

- CPU Profiler – Weighted call graph with timing information

堆泄漏分析器

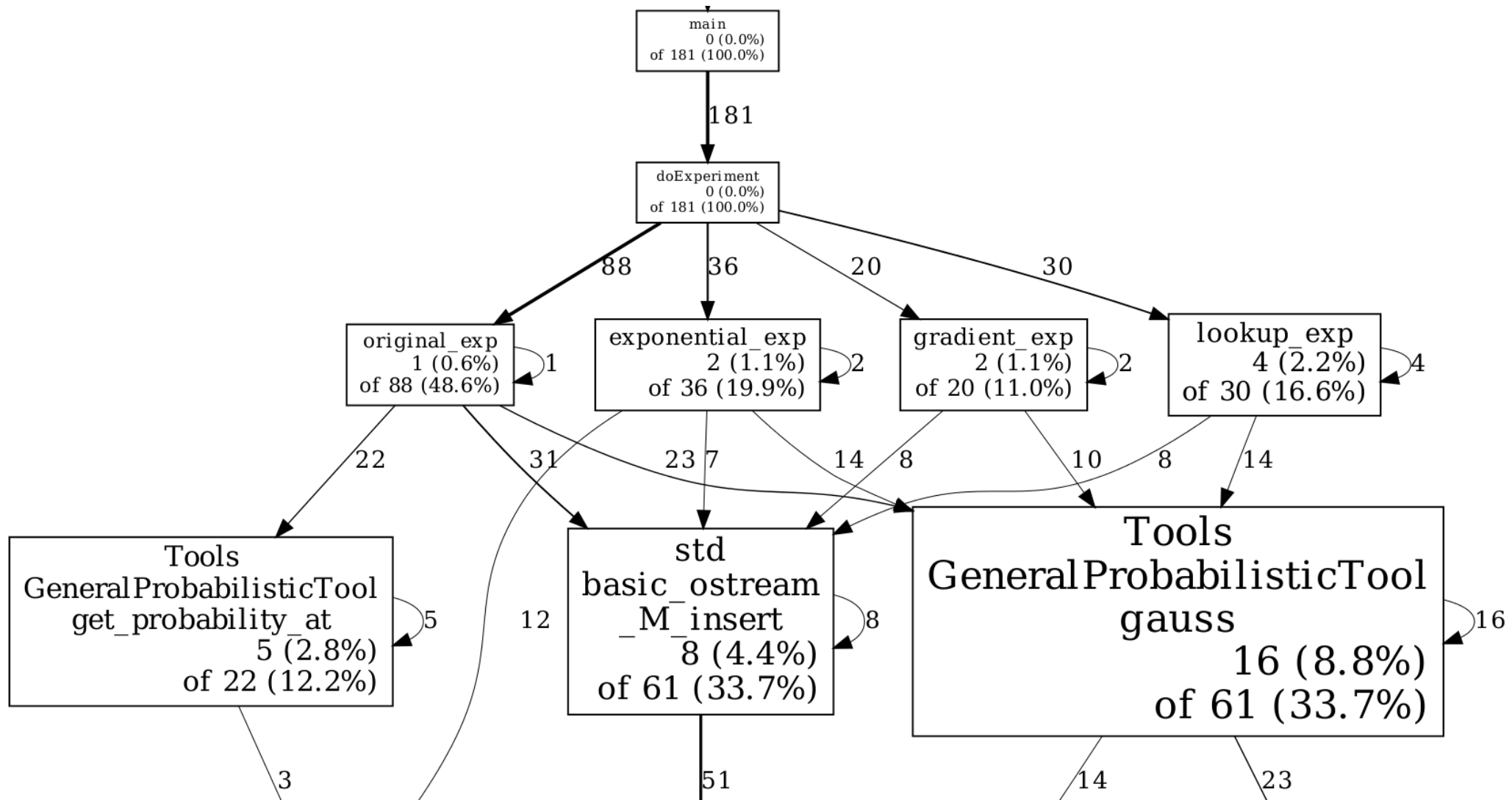
- Heap Leak Checker – call graph of reported leaks

堆分析器

- Heap Profiler – weighed directed graph of memory allocations

- Text, postscript (gv), dot, pdf, gif, source-code listings, & disassembly

GPerfTools – pprof



Linux Trace Toolkit next generation

- Distributed under GPLv2
- Community run
- Offers Kernel and User Space Tracing
- Only available for Linux (32 & 64 bit)
- Supports C/C++ and all languages that can call C code

LTTng – History

- Successor of Linux Trace Toolkit
- Launched in 2005
- Version 2.0 since March 2012
- Written and maintained by Mathieu Desnoyers

LTTng – Overview



- Consists of 3 parts:
 - Kernel part → Kernel tracing
 - User space command-line application (lttcl)
 - User space daemon (ltttd)
- Modular: 5 modules
 - ltt-core, generates events, controls:
 - ltt-heartbeat, ltt-facilities, ltt-statedump
 - ltt-base, built in kernel object, keeps symbols & data structures

LTTng – Tracing

- Observes operating system kernel events such as:
 - system calls, interrupt requests, scheduling & network activities
- Multiple traces can be recorded simultaneously (on mult. CPUs)
- Events are recorded by so called “Tracepoints”

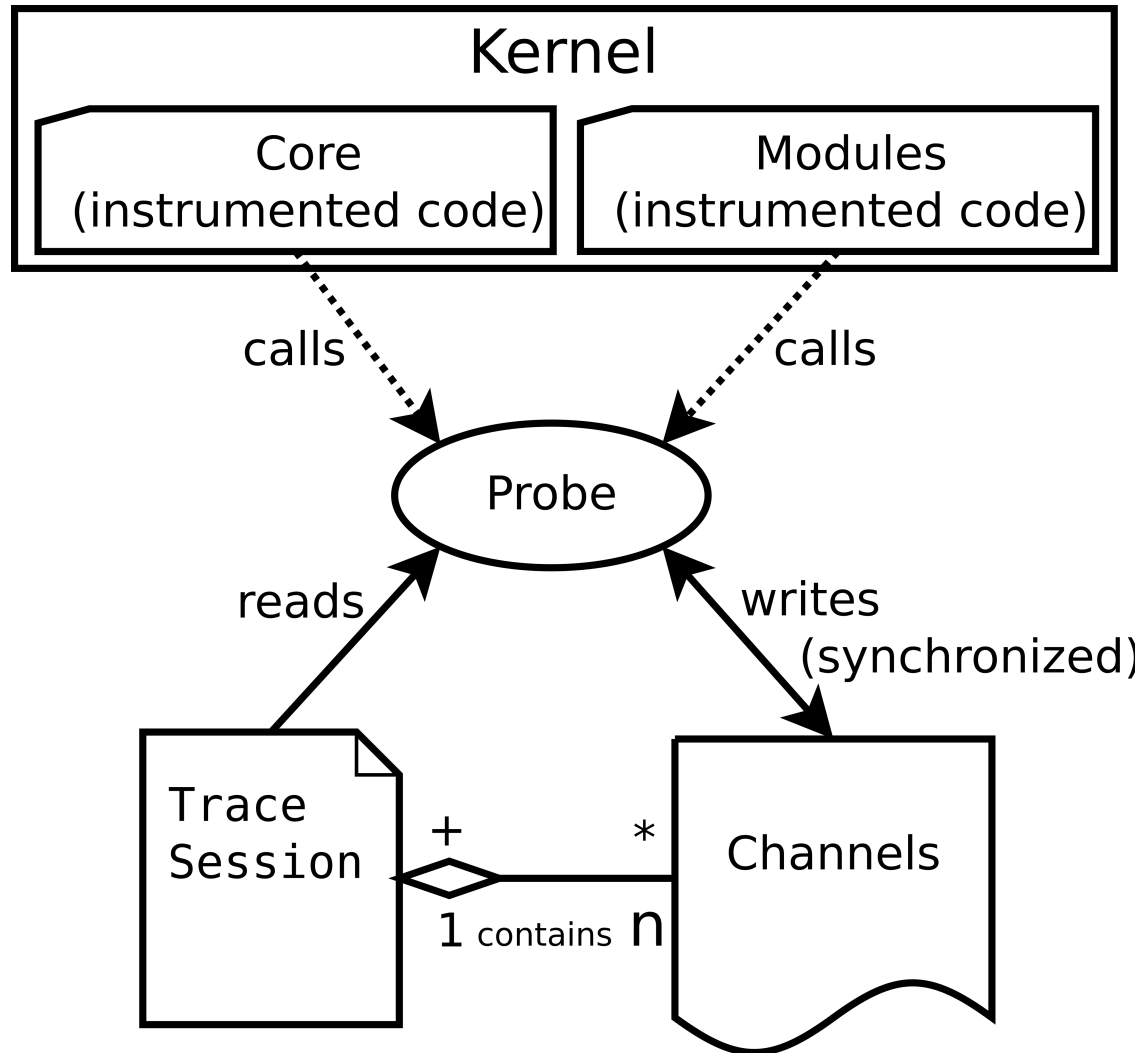
LTTng – Tracepoints

- Tracepoints are small pieces of code, like this:

```
if (tracepoint_1_active)
    (*tracepoint_1_probe)(arg1, arg2);
```

- Almost no overhead when inactive
- Overhead when active comparable to a C function call

LTTng – Tracepoints



* one trace session contains multiple channels

LTTng – Storage & Tracemodes

- Data can be written to disk or sent to a network stream
- Available tracemodes are:
 - Write-to-disk, flight recorder
 - Still under development: stream to remote disk & live monitoring

LTTng – Overhead

Load size	Test Series	CPU time (%)			Data rate (MiB/s)	Events/s
		load	probes	lttd		
Small	mozilla (browsing)	1.15	0.053	0.27	0.19	5,476
Medium	find	15.38	1.150	0.39	2.28	120,282
High	find + gcc	63.79	1.720	0.56	3.24	179,255
Very high	find + gcc + ping flood	98.60	8.500	0.96	16.17	884,545

Source: Desnoyers: *A low impact performance and behavior monitor for GNU/Linux.*

LTTng – Post-Processing

- Babeltrace
 - Text based, LTTng 2.0 traces in CTF (Common Trace Format)
- LTT Viewer
 - Visual analyzer, written in C, extendable with plug-ins
- Eclipse
 - TMF (Trace Monitoring Framework) plug-in, displays:
 - Control Flow, Resources, and Statistics

Areas of application

- GPerfTools
 - Google, ...
- LTTng
 - IBM, Siemens, Autodesk, Ericsson, ...
 - Included in packages of Montavista, Wind River, STLinux, & Suse

Summary

- GPerfTools:
 - CPU Profiler, TCMalloc, Heap Leak Checker, Heap Profiler
 - Pprof for analysis
- LTTng:
 - Kernel & user space tracer → tracepoint → probe → channel → I/O
 - Babeltrace, LTT Viewer, and eclipse for analysis

Sources

- **GperfTools:** <http://gperftools.googlecode.com>
- **Desnoyers:**
 - *Low-Impact Operating System Tracing*
 - *A low impact performance and behavior monitor for GNU/Linux*
- **LTTng:** <http://www.lttng.org>
- **Chakraborty, Anjoy:** *Efficiency of LTTng as a Kernel and Userspace Tracer on Multicore Environment*