PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

Selecting the Right Tools for Performance Analysis

William Cohen
Performance Tools Engineer, Red Hat
06.27.12

To measure is to know.
-Lord Kelvin

If you can not measure it,
you can not improve it.
-Lord Kelvin

The Importance of Correct Metrics

- Have standard econbox car
- Want it “faster”

The Ugly Result of Using the Wrong Metric

http://commons.wikimedia.org/wiki/File:Poser.JPG

What are your important metrics?

- Response time for a query
- Maximum number of simultaneous requests
- Memory required per query
- Time spent in key sections of code

Acceptable measurement cost?

- Zero cost
- Low cost
- Cost is no object

When is it “Good Enough”?

- Fine as is
- Nice if a bit better
- Need to meet some specific performance metric
- Never “good enough”

Want data that guide actions

- Some metrics show there is a problem
- Want metrics to show how or where to fix the problem

Map your metrics to tool metrics

- Your metrics probably not the same as the tool's metrics
- Need to translate into available tool measurements
- Typical tool metrics:
  - Software events (for example page faults)
  - Latency
  - Code hotspots
  - Processor microarchitecture performance

What region metrics measured?

- Single process
- User-space
- Kernel-space
- System-wide

Many Linux Performance Tools

- Tools developed to address specific issues
- Provide different:
  - Measurements
  - Performance impact
  - Filtering
  - Data analysis

Instrumentation Techniques

Statistical Sampling

- Every Nth (+/-) sample record details
- Number of samples should be proportional to events at that location
- Reduces overhead
- Often used with hardware performance counters

Probing

- Predefined points in software:
  - kernel tracepoints
  - user-space markers
- Use breakpoints instructions:
  - Kprobes
  - Uprobes
- Only encounter cost when hit
- Can be used on already running code

Binary Rewrite

- Examine the code as it is loaded
- Take machine code apart
- Add instrumentation and generate new code
- Allow instrumentation of code without recompilation of source
- Cannot attach to already running apps
- Can significantly slow the startup and execution of code

Shared Library Wrapper

- Inserts a wrapper function between caller and callee in shared library
- Lower overhead than binary rewrite
- Limit to function call boundaries
- Cannot attach to already running apps
- Implemented with LD_PRELOAD=libwrapper.so

Mapping Data Back to Source Code

- Most performance tools collect instruction addresses
- Want to map instruction address to source code
- Debuginfo (-g) provides mapping
- Debuginfo does not affect runtime performance
- Can strip out debuginfo to make executables smaller

Mapping Data Back for Interpreters

- Instruction address maps to runtime
- Just-In-Time translation no debug info for address
- Some interpreters provide profilers:
  - Python (python -m cProfile.py ..)
  - Perl (perl -d:DProf ...)
  - Ruby (ruby-prof ...)
- Some interpreters have hooks for other tools:
  - Java jvmti and jvmpi interfaces

Where are the processors spending time?

Why?

OProfile

- Useful for finding code hotspots
- Not useful for I/O or latency problems
- Uses performance monitoring hardware
- Performs sampling
- Collects system-wide data
- Requires root privileges
- Can record data for Java applications
- Supports a wide range of processors
- Available on most distributions

Perf

- Provides access to:
  - performance monitoring hardware
  - Kernel software events
- Data metrics:
  - Count events (stats)
  - Trace of samples (record)
- Available to normal user
- Requires newer Linux kernels (>= 2.6.32)

What about watching events across processes?

What about watching events in both user- and kernel-space?

SystemTap

- Can instrument already running apps
- Probes system-wide (both user- and kernel-space)
- Very flexible filtering
- Requires special privileges

Why is so much memory being used?

Valgrind

- Originally a tool to look at memory issues:
  - Allocation, initialization, and freeing
- Actually a very flexible binary rewrite framework:
  - Memory diagnostics, memory allocation, cache simulator, threading
- Instruments shared libraries, but not kernel code
- Slows application (4x-100x slower)
- Run as normal user
- Cannot attach to already running code

Valgrind (cachegrind) Example

valgrind --tool=cachegrind stap -p4 disktop.stp

...

==10672==

==10672== I   refs: 4,556,135,689

==10672== I1 misses: 2,192,463

==10672== LLi misses: 30,208

==10672== I1 miss rate: 0.04%

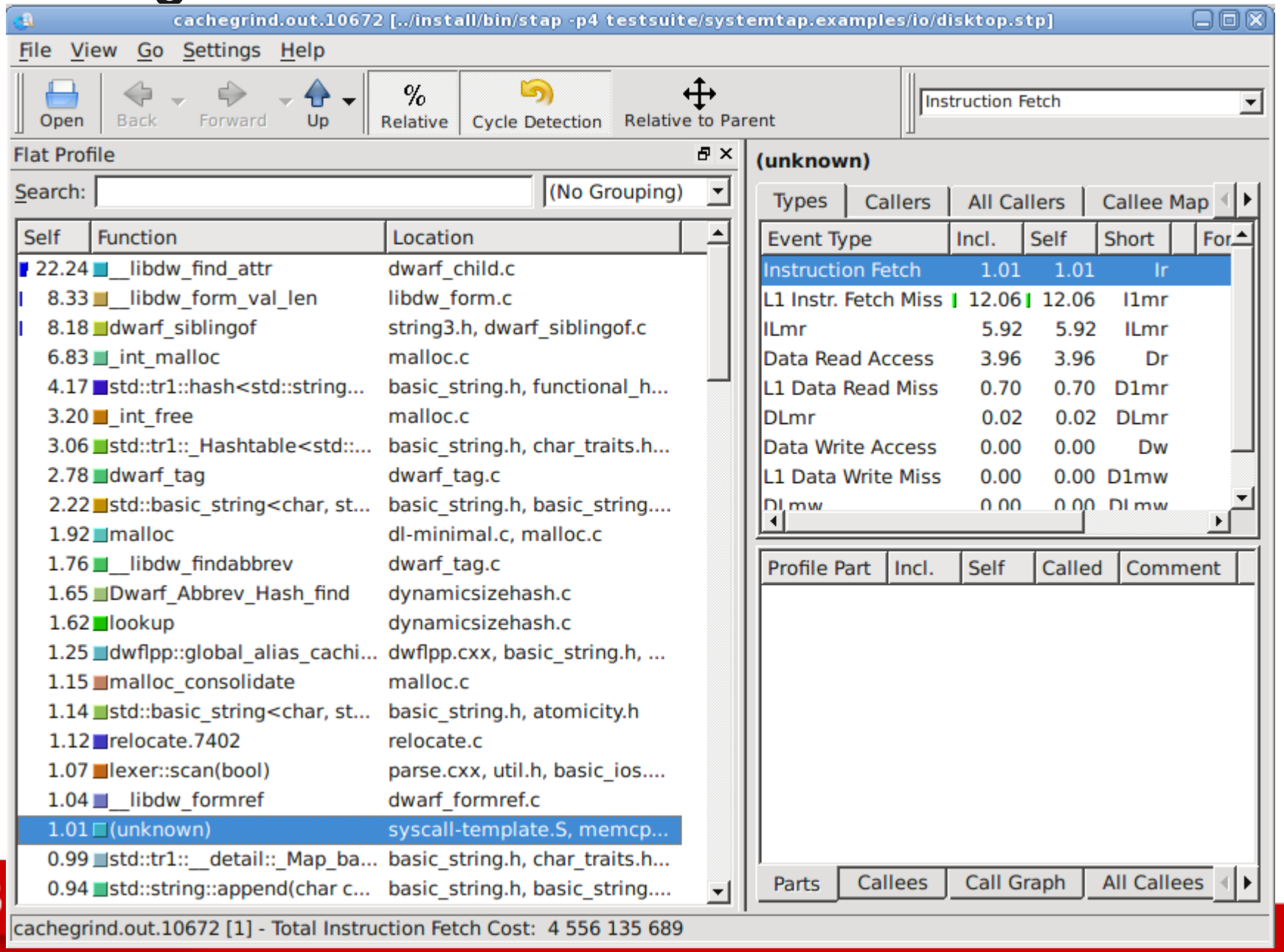
==10672== LLi miss rate: 0.00%

...

Valgrind (cachegrind) Example (cont.)

```
==10672==
==10672== D   refs:      1,846,734,454  (1,161,695,711 rd   + 685,038,743 wr)
==10672== D1  misses:       15,404,912  (   13,300,378 rd   +   2,104,534 wr)
==10672== LLd misses:        8,070,771  (    6,160,076 rd   +   1,910,695 wr)
==10672== D1  miss rate:           0.8% (          1.1%     +         0.3%  )
==10672== LLd miss rate:           0.4% (          0.5%     +         0.2%  )
==10672==
==10672== LL refs:          17,597,375  (   15,492,841 rd   +   2,104,534 wr)
==10672== LL misses:         8,100,979  (    6,190,284 rd   +   1,910,695 wr)
==10672== LL miss rate:            0.1% (          0.1%     +         0.2%  )
```


Kcachegrind GUI

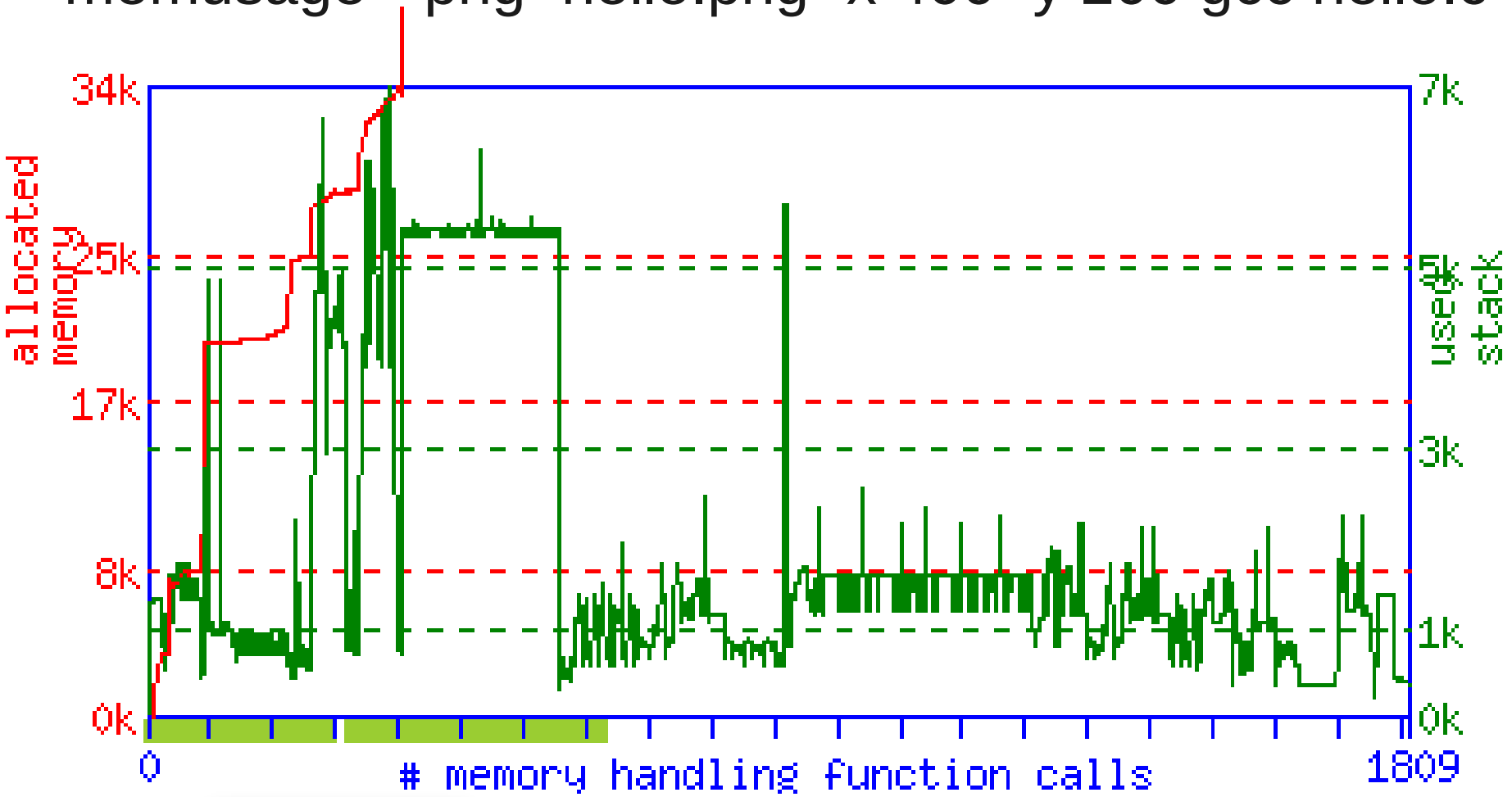


Glibc-utils

- Analysis more limited than Valgrind:
  - Only analyzes memory allocation/free operations
  - No checks for use of uninitialized memory
- Faster than valgrind:
  - Intercepting calls for memory allocation/free
  - Not rewriting and instrumenting the binary code
- Tools in the glibc-utils rpm

Glibc memusage demonstration

memusage --png=hello.png -x 400 -y 200 gcc hello.c



Graphical User Interface?

- Eclipse IDE
- Plug-ins:
  - Systemtap
  - Oprofile
  - Valgrind
- Navigate data by point and click

Further Information

- Developer Guide on http://docs.redhat.com/docs/en-US/index.html
- Oprofile - http://oprofile.sourceforge.net/news/
- Perf – perf help
- SystemTap - http://sourceware.org/systemtap/
- Valgrind - http://valgrind.org/
- glibc utils – memusage –help

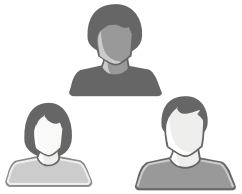
Stay connected through the Red Hat Customer Portal

Instrumenting the Linux Kernel with SystemTap

Review Tech brief

Performance Issues in Red Hat Enterprise Linux (Part 1)

Review Tech brief



Join Group

Join Red Hat Enterprise Linux

access.redhat.com

LIKE US ON FACEBOOK

www.facebook.com/redhatinc

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#redhat

READ THE BLOG

summitblog.redhat.com

GIVE US FEEDBACK

www.redhat.com/summit/survey