User Space Tracing in Small Footprint Devices

(How Low can You Go?)

Jason Wessel

- Product Architect for WR Linux Core Runtime
- Kernel.org KDB/KGDB Maintainer

August 18th, 2011

Agenda

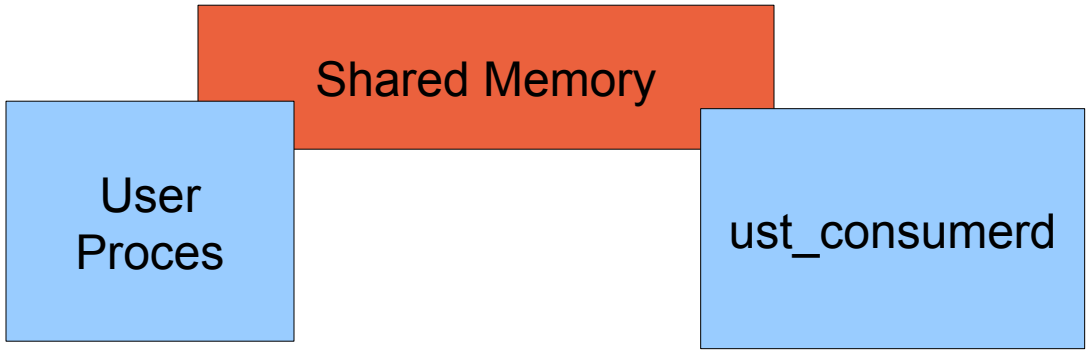
- What is UST?
- How does UST work?
- Can it really replace printf()?
- What is UST going to cost me?
- Future of UST and LTTng

Presentation found at:

http://developer.windriver.com/docs/DOC-1434

What is UST?

- http://lttng.org/ust
- User space shared library and headers
  - Code instrumented with markers or tracepoints
  - A highly efficient lockless tracer populates shared buffers
  - Buffers written to disk by ust_consumerd
  - Managed with ustctl & usttrace commands



What does a marker look like?

```
#include <ust/marker.h>


…
ust_marker(Log_Name_Here, Format_String, Args)
…
```

Now link your app with: -lust

UST Dynamic Logging

- Simple: usttrace -s APP_NAME
- More complex but more dynamic
  - Start ust_consumerd
  - ustctl list-markers PID |grep ust | awk '{print $4}'
    - ust/potential_exec,
    - ust/diaglog1,
    - ust/diaglog2,
    - ust/sleeplen,
  - ustctl enable-marker PID TRACE_NAME ust/diaglog1
  - ustctl create-trace PID TRACE_NAME
- You can log multiple apps in both cases with the same time base

Demonstration 1

- Instrument a custom multi-threaded web server
- Use an external web browser dynamically start / end a trace
- http://www.youtube.com/watch?v=LmOaSW5Il-c

Log everything

- Run the app with usttrace
  - usttrace APP_NAME
- View the log when you are done
  - lttv -m textDump -t /trace_dir/*_*
- View the log in a graphical tool

Demonstration 2

- Show a UST trace in System Viewer from web sever start to web server stop
- Look at time durations and events
- http://www.youtube.com/watch?v=Ufs-mUDwcjE

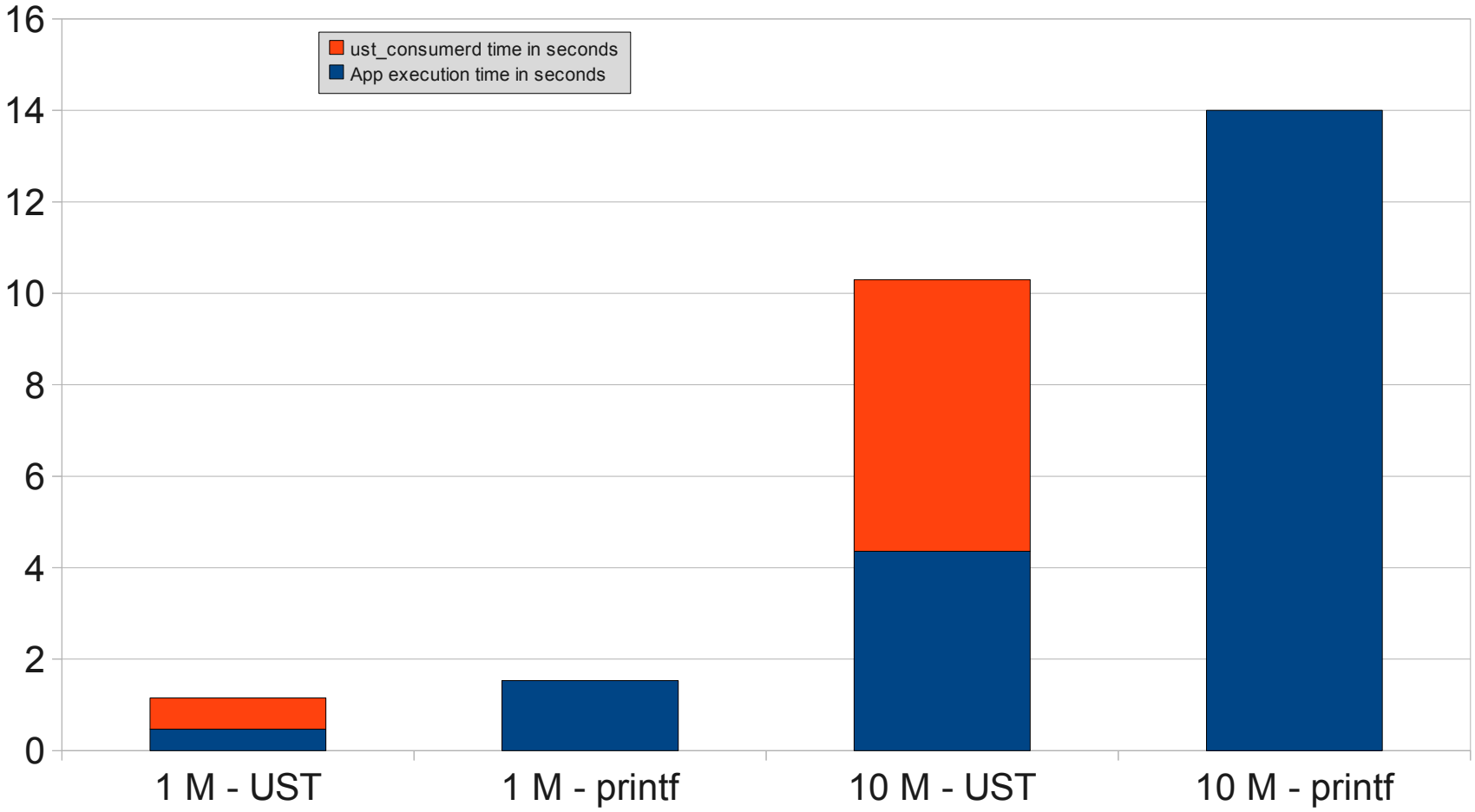
Comparing UST to printf easily?

```
#ifdef USE_UST   /* Compile with: -DUSE_UST -lust */
#include <ust/marker.h>
#define mark_here(x, fmt, args...) \
            ust_marker(x, fmt, ## args)
#else /* use printf() */
#define mark_here(x, fmt, args...) \
            printf(fmt "\n", ## args)
#endif
...
mark_here(sleeplen, "Sleep_Length %i", sleeplen);
```

Speed of UST vs printf

- UST's use of per CPU buffers can cut down bottle necks at chatty times (on a “many” cpu system)
- If the file system is not fast you really _need_ buffering
- Implementing buffering with printf > 0 effort
- Multiple threads with multiple log files?
  - Locks are too high a price with printf!

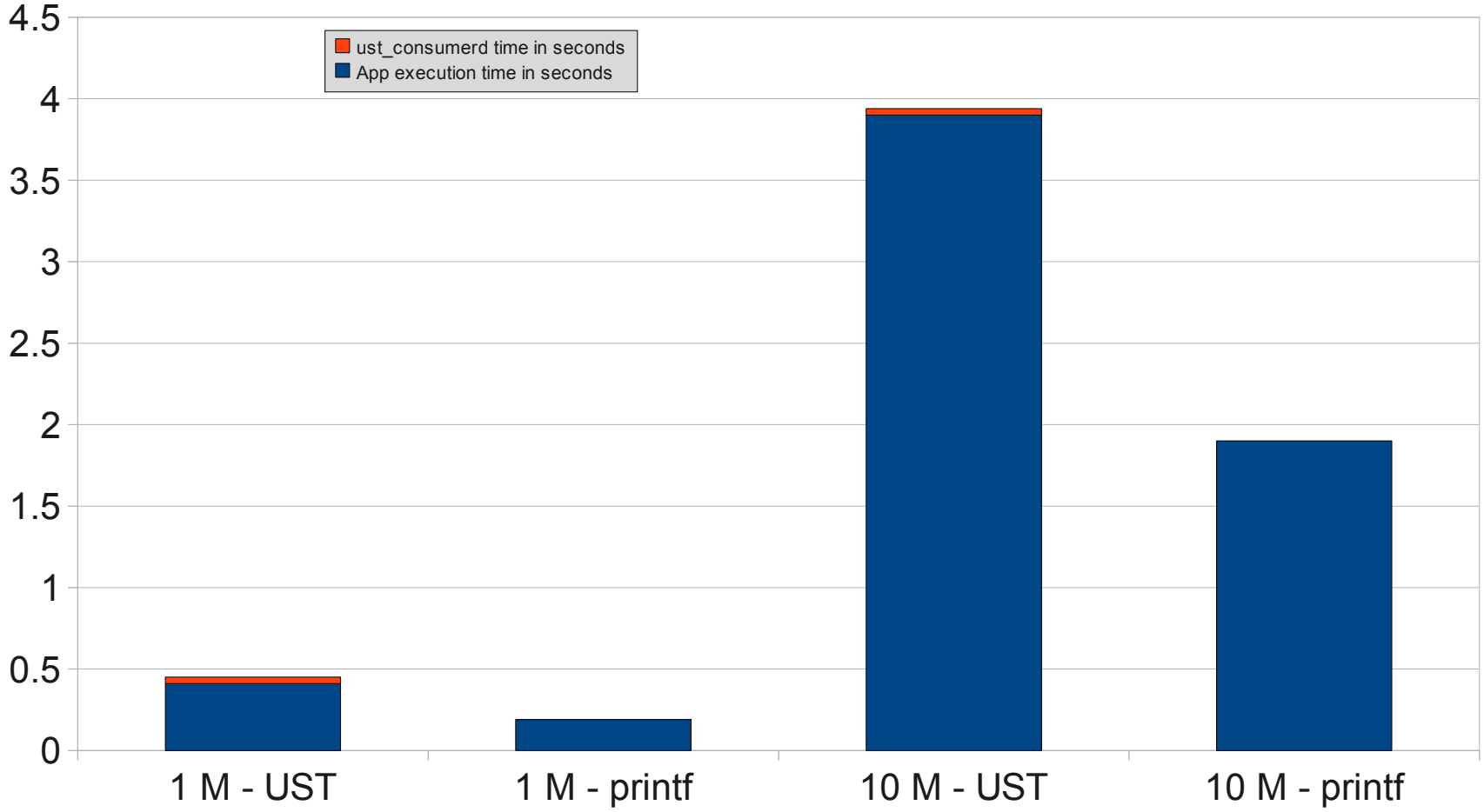
Millions of logs per second



Your log file system matters!

- The previous graph was to logging to an NFS file system
- printf has not lost or won yet...

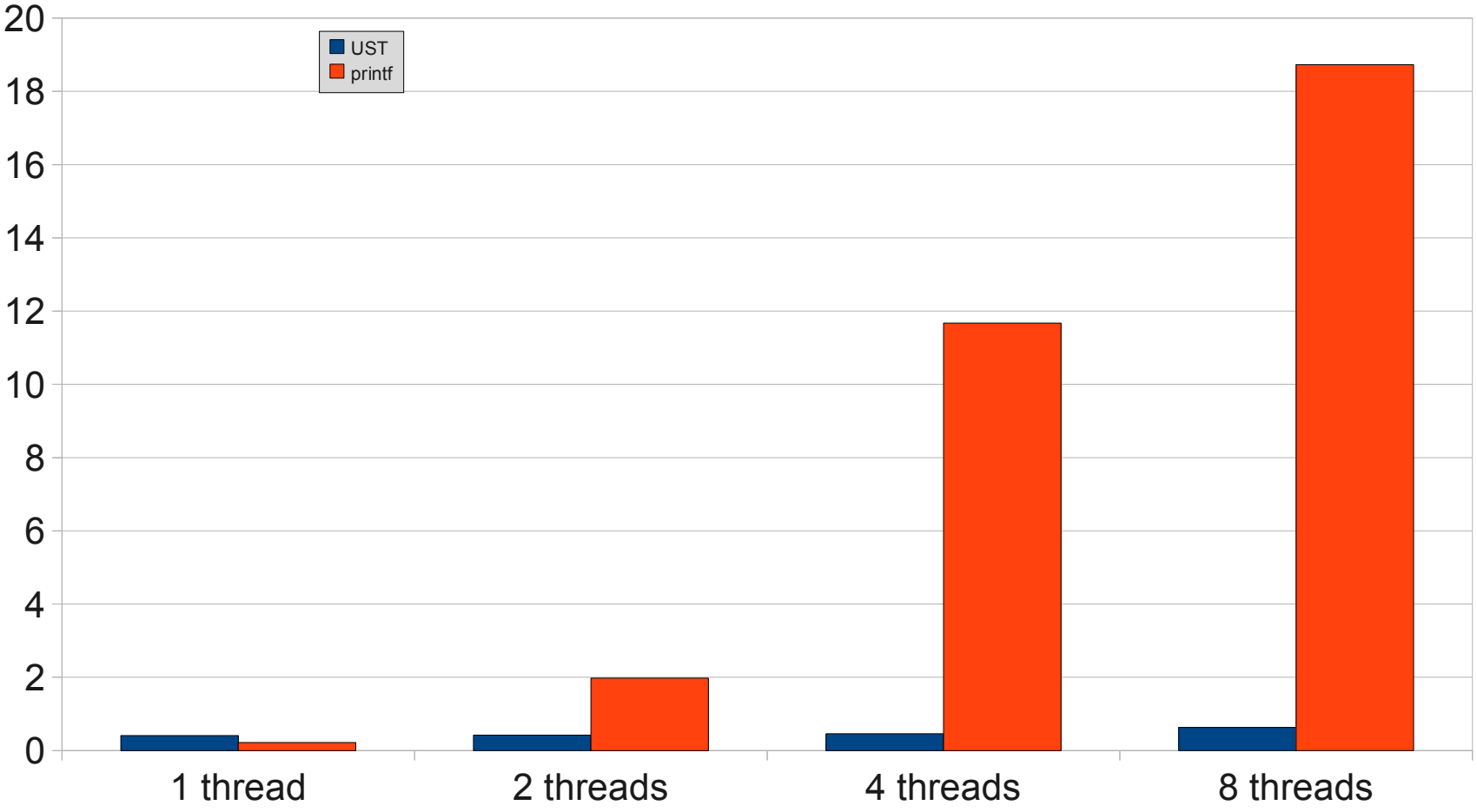
Millions of logs per second → tmpfs



Bring on the threads!

- 8 cpu KVM instance and mutex for printf() serialization vs UST for 1 Million Logs per thread > tmpfs
- UST vs printf – 1 thread
  - 0.41 vs 0.22
- UST vs printf – 2 threads
  - 0.42 vs 1.98
- UST vs printf – 4 threads
  - 0.46 vs 11.67
- UST vs printf – 8 threads
  - 0.63 vs 18.73

1 Million log entries per thread



Cost of UST for foot print?

- 400K on the x86_64 platform for the shared libraries
  - File system was < 5 MB with ssh & apps & busybox
- The difference in the binaries? 2520 bytes
  - 4824 bytes for log_test_printf
  - 7344 bytes for log_test_ust

To UST or Not to UST...

- Good instrumentation is always important in your apps
- Custom printf and log files cost design time
  - BUT... No special viewers needed for a text file!
- Where is the value of UST?
  - Unused trace points are cheap ~ cost of a branch
  - Buffered (and time stamped) per CPU / thread logging
  - Dynamically enable/disable traces without app termination
  - Line up traces with LTT data
  - Commercial linux providing UST out of the box

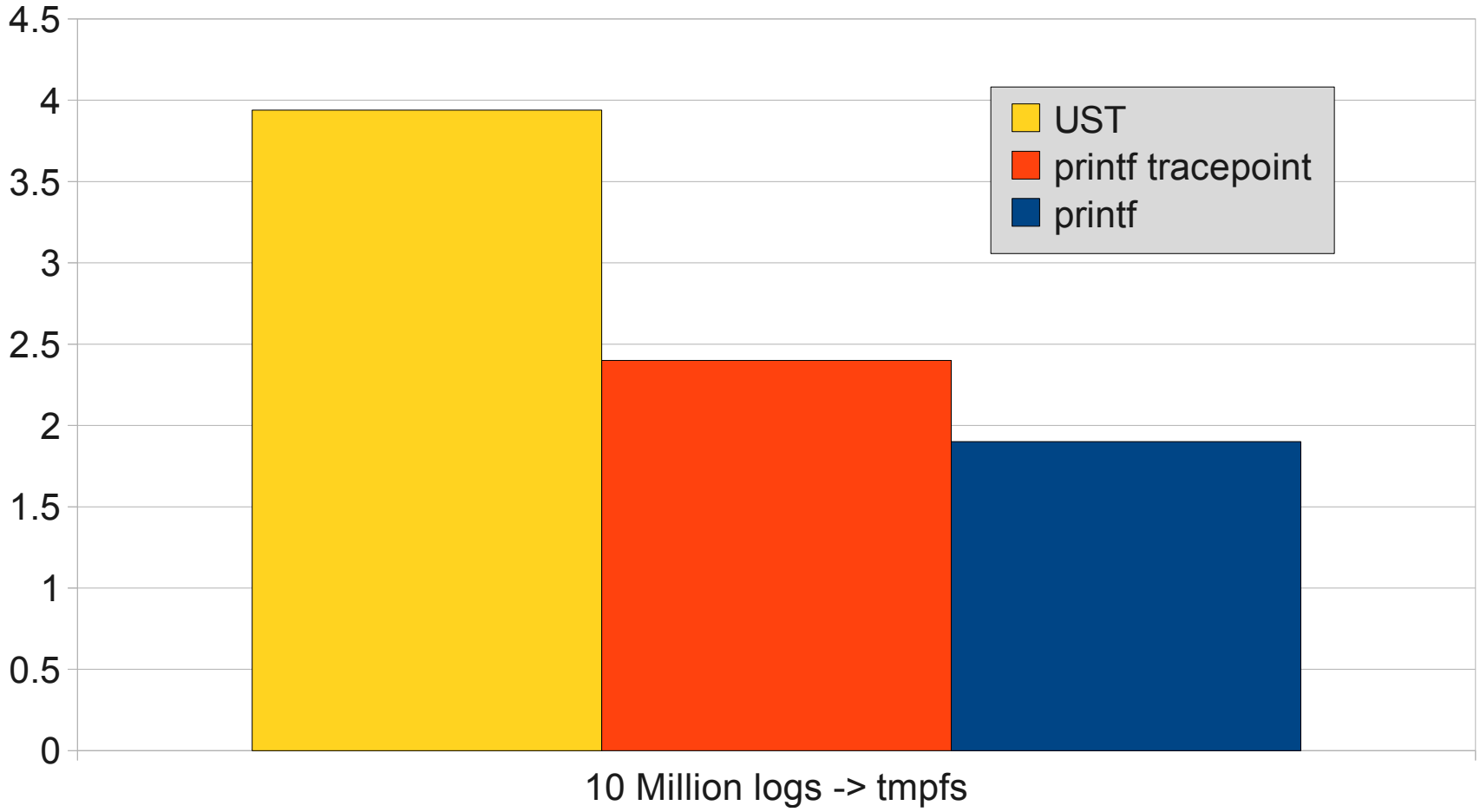
Demonstration 3

- Show LTT data overlayed with UST data
- Look at time delta between UST log and socket operation
- http://www.youtube.com/watch?v=7FvkJuUJmG4

Future of UST

- Dynamic high speed TRACEPOINT_EVENT
  - A forth coming merge of the lttng and UST
  - Source level ust_marker() inteface does not change
- Coordinated kernel and userspace logging
- gdb tracepoints and perf integration future consideration
- Unified lttng / UST graphical tools
- Soon to support MIPS 32/64/BE/LE

ust vs tracepoint printf vs printf



Misc Notes – References

- http://lttng.org/ust
- All demonstrations used WR Linux 4.2 / WB 3.3.1 with:
  - userspace-rcu 0.6.4
  - ust 0.15
- The bench marks were run on:
  - Single CPU Intel(R) Xeon(R) CPU X5520 @ 2.27GHz

WIND RIVER