

# Application Testing under Realtime Linux

Luis Claudio R. Gonçalves  
Red Hat - Realtime Team  
Software Engineer

# Agenda

- \* Realtime Basics
- \* Linux and the PREEMPT\_RT patch
- \* About the Tests
- \* Looking for bad programming practices
- \* Bad Priority Assignment
- \* Resource Allocation
- \* Changing Application Behavior
- \* Comparing Apples to Apples

Conclusion

# Realtime Basics

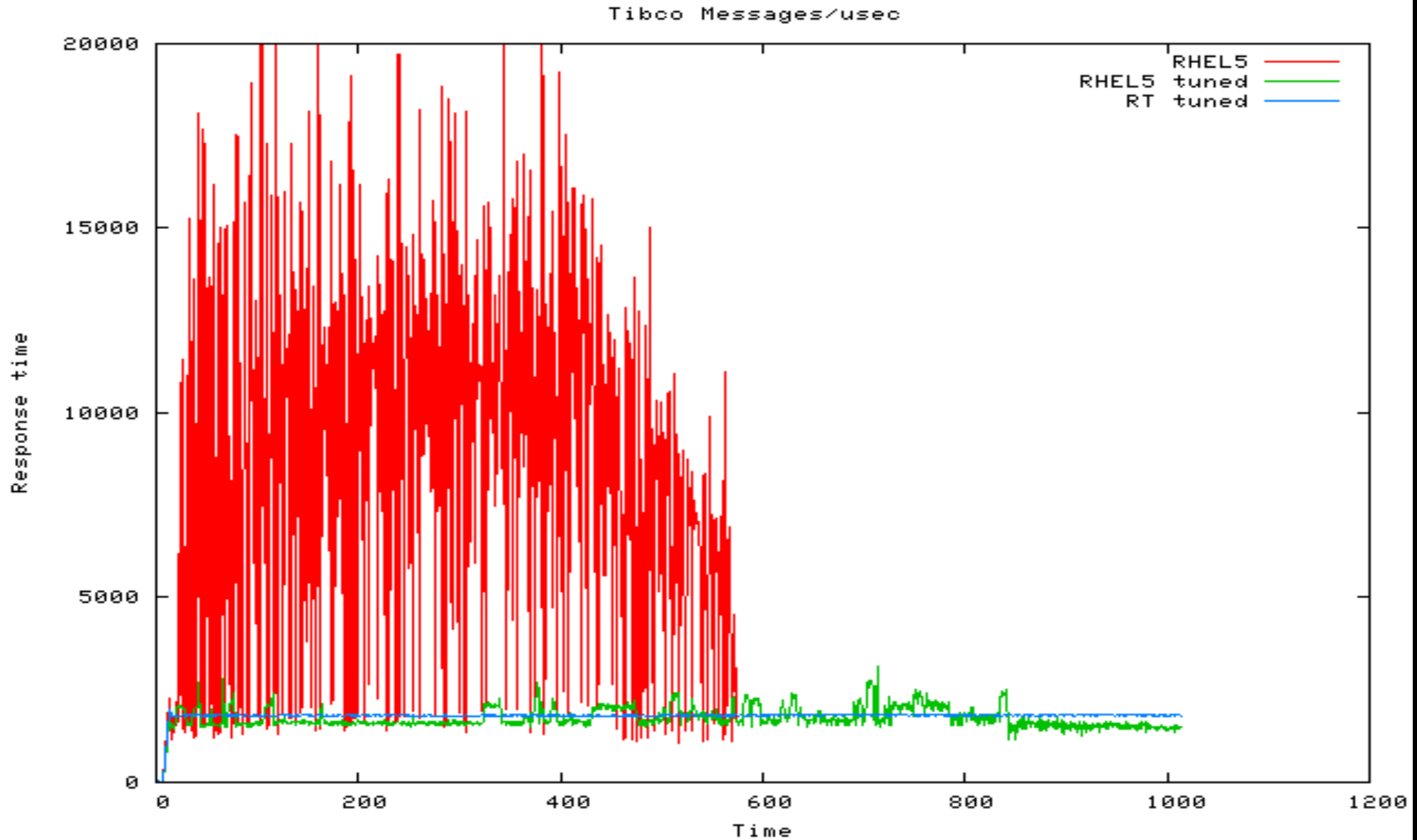
# Realtime Basics

- \* Determinism – ability to schedule the highest priority tasks in a consistent and predictable way.
- \* Latency – time between a given event and the desired effect.

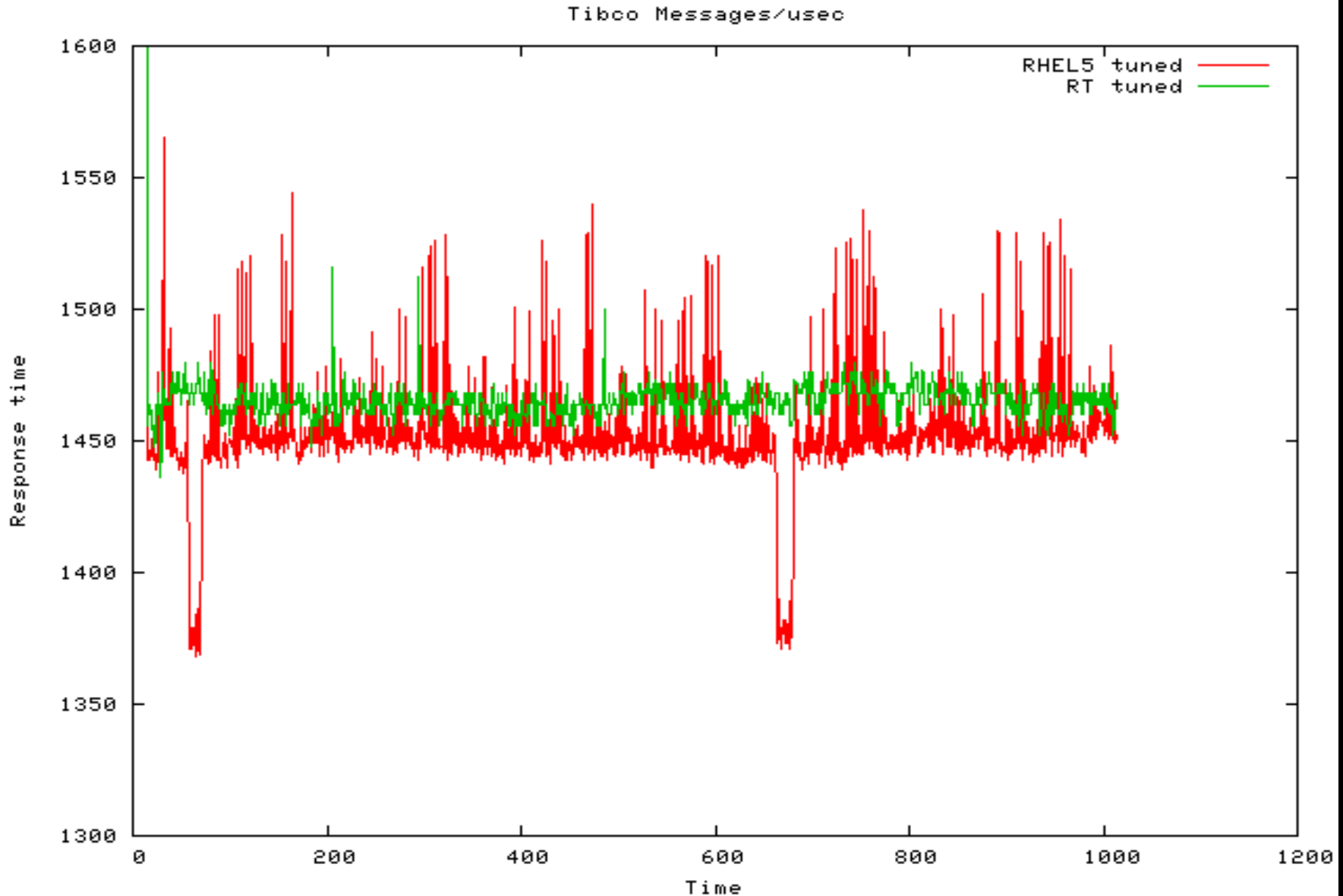
# 'Almost' is not enough...

- \* When determinism matters...
- \* Navy, Financial Services, Federal sector, Telco/Network
- \* Financial Services:
  - first to answer has better chances on the bids
  - legal requirements for consistency in operations
- \* Telco / Network
  - network packet management
  - QoS

# Latency...



# Latency in detail...



# Sources of latency

## \* Application Priority

- one application blocks (or preempts) another
- holds a contended resource (lock)

## \* The Kernel

- when the kernel runs apps stop
- the longer the kernel runs the longer applications wait
- determinism gets limited by the longest codepath

High Priority  
Application

Interrupt

Application continues

Kernel: interrupt handler +  
scheduler

```
graph TD; A[High Priority Application] -- Interrupt --> B[Kernel: interrupt handler + scheduler]; B --> C[Application continues];
```

# Linux + PREEMPT\_RT

# PREEMPT\_RT patch

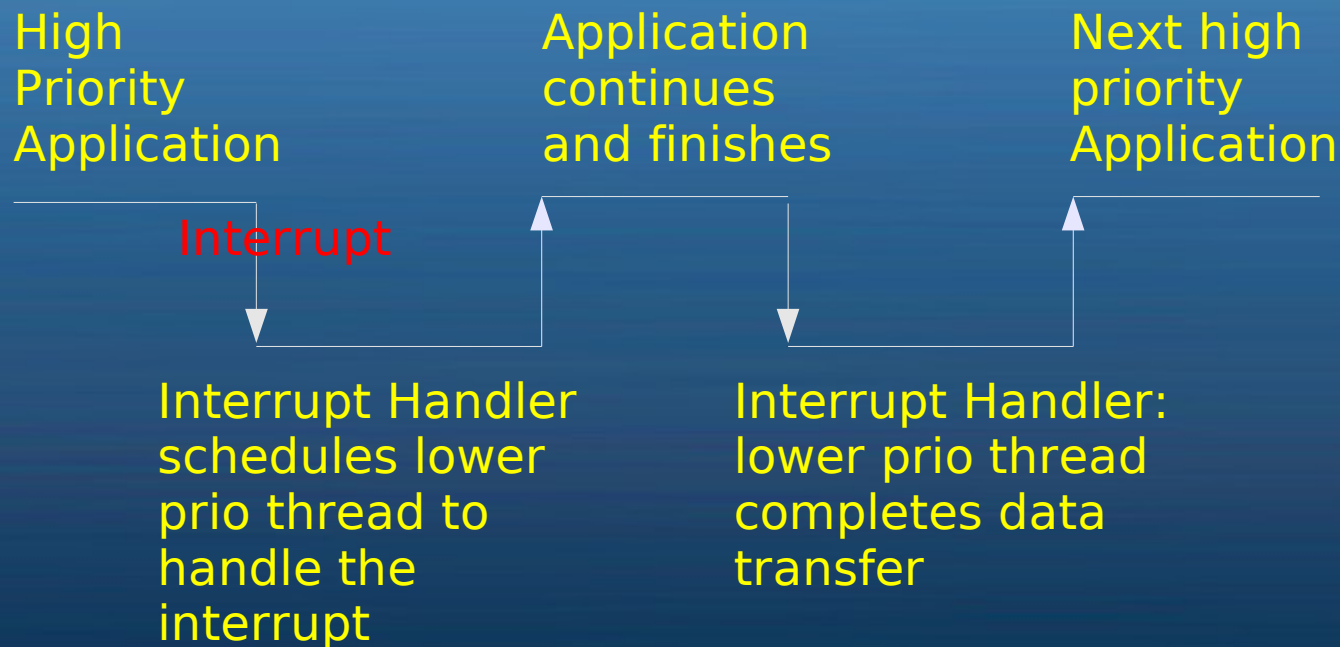
- \* Methodical implementation, focusing upstream adoption
- \* Started by the building blocks that would enable the implementation
- \* Complete RT, not just a VM/scheduler hack
- \* Mature features, that benefits both RT and upstream, have been merged
- \* Community + sandbox (linux-rt-users)

# Building Blocks

- \* Threaded IRQs
- \* Threaded Softirqs
- \* Replaced semaphores and spinlocks by (rt)mutexes
- \* Sleeping spinlocks
- \* Priority Inheritance

# Strategies

- \* Measure and Identify the longest code path in the kernel
- \* Enhance algorithms to enable more concurrency
- \* Shorten the time the kernel runs as non-preemptible
- \* Defer processing to lower priority kernel threads



# About the Tests

# About the tests

- \* Validation of the Kernel and Customer Apps
- \* “It runs slower on RT!”
- \* “Sorry, I can't show you or change the code...”
- \* Where the problem lies? (kernel, app, comparison)

# Bad Practices

# Bad Coding Practices

- \* Bad coding practices (focus on Realtime)
  - some practices are bad only in the context of PREEMPT\_RT
  - apps ported from other/older OS
  - tunings that no longer make sense
- \* How to spot bad programming practices?
  - system calls that should be avoided
  - bad parameter assignment
- \* Reading the source code
- \* Systemtap scripts
- \* Strace
- \* Example: avoid sched\_yield()

# Example

\* How to spot a given syscall usage

```
# cat test_yield.stp
```

```
probe syscall.sched_yield
{
    printf(" %s(%d:%d) Called sched_yield()\n", execname(), pid(),tid())
}
```

```
# stap -v test_yield.stp
```

```
rsyslogd(1702:1705) Called sched_yield()
rsyslogd(1702:26304) Called sched_yield()
rsyslogd(1702:26304) Called sched_yield()
```

# Priority Assignment

# The Short Version...

- \* Prioritize what is important
  - chrt is your friend
  - sched\_setscheduler()
  - always check 'man 3p <function>'
  - avoid SCHED\_FIFO priority 99!!!
- \* That Spiderman principle...
  - avoid SCHED\_FIFO priority 99!!!
- \* Have I said...
  - avoid SCHED\_FIFO priority 99!!!

# Priorities

\* Is the App suffering non-voluntary preemption?

- Tuna
- check `/proc/<pid>/status` - (nonvoluntary\_ctxt\_switches)
- use the function `getrusage()`
- create a `systemtap` script to verify this (and other) information
- check the priority of the application and its threads

```
[root@void ~]# ps -emo pid,tid,policy,pri,rtprio,cmd
```

```
...  
3826  - -  -  - /usr/lib64/.../firefox-bin  
- 3826 TS 19  - -  
- 3848 TS 19  - -  
- 3849 TS 19  - -  
- 3855 TS 19  - -  
- 3856 TS 19  - -  
- 3857 TS 19  - -  
- 3869 TS 19  - -  
- 8854 TS 19  - -
```

```
...
```

# Priorities

## \*Example systemtap script

```
# if you have problems hooking on exit_mm,  
# use profile_task_exit instead.  
  
probe kernel.function("exit_mm")  
{  
    printf("%s(%d:%d) stats:\n", execname(), pid(), tid())  
  
    printf("\tPeakRSS: %dKB \tPeakVM: %dKB\n",  
        $task->mm->hiwater_rss*4, $task->mm->hiwater_vm*4)  
  
    printf("\tSystem Time: %dms \tUser time: %dms\n",  
        $task->stime, $task->utime)  
  
    printf("\tVoluntary Context Switches: %d \tInvoluntary: %d\n",  
        $task->nvcs, $task->nivcs)  
  
    printf("\tMinor Page Faults: %d \tMajor Page Faults: %d\n",  
        $task->minflt, $task->majflt)  
}
```

# Priorities

## \* Results:

sshd(3098:3098) stats:

PeakRSS: 1584KB	PeakVM: 47764KB
System Time: 1ms	User time: 3ms
Voluntary Context Switches: 32	Involuntary: 0
Minor Page Faults: 340	Major Page Faults: 0

id(3102:3102) stats:

PeakRSS: 756KB	PeakVM: 76164KB
System Time: 2ms	User time: 0ms
Voluntary Context Switches: 3	Involuntary: 0
Minor Page Faults: 262	Major Page Faults: 0

# Priorities

- \* Notes about the systemtap script

- gathers information from task\_struct
- exit() -> do\_exit()
- exit\_mm()

- \* More options are coming

# Priorities

\* Prioritize kernel threads (IRQs, Softirqs, ...)  
I.e.: Applications depending on network and not using disk I/O

```
4 [posix_cpu_timer]
5 [softirq-high/0]
6 [softirq-timer/0]
7 [softirq-net-tx/]
8 [softirq-net-rx/]
9 [softirq-block/0]
10 [softirq-tasklet]
11 [softirq-sched/0]
12 [softirq-hrtimer]
88 [kacpid]
90 [IRQ-9]
181 [ksuspend_usbd]
246 [aio/0]
350 [IRQ-8]
384 [IRQ-14]
406 [IRQ-16]
407 [pccardd]
```

# Resource Allocation

# Good Habits

## \* Allocate Resources in advance

- minor / major page faults
- memory (mlock)
- create thread pool in advance

## \* Bind processes to CPUs and Isolate CPUs

- taskset
- isolcpus in the kernel command line
- Tuna

# Resource Allocation

## \* Memory Allocation

- minor page faults
- major page faults
- mlock() and friends
- don't forget touching memory before mlock'ing
- can be checked via /proc, getrusage() or systemtap scripts

## \* Thread creation

- can take more than 100us
- can suffer page faults

## \* Threads

- some system calls can generate page faults (i.e. fopen)
- select() has a 1ms resolution

# CPU Affinity

## \* Allocate processes to CPU

- i.e. avoid running sender and receiver on the same CPU
- sometimes, try running sender and receiver on the same CPU
- taskset, exidus
- some kernel threads can't be moved
- try Tuna

## \* CPU Isolation

- boot parameter, isolcpus
- removes all (possible) processes from the CPU

# Application Behavior

# Changing App Behavior

## \* No access to source code

- When system tuning is not enough...
- What if this or that configuration was in use?
- hacking the kernel
- Using sytemtap
- Library preload tricks
- Kernel hacks

# Changing App Behavior

## \* Libautocork

- Performance is king
- Nagle Algorithm
- TCP\_NODELAY
- TCP\_CORK

```
# LD_PRELOAD=libautocork.so ./customer_app
```

## \* Liblocalize

# Comparing Results

# Apples to Apples

## \* AIT

- Have you attended Arnaldo's presentation?
- Which knobs were set?
- Did the tests run on the same environment?
- What was different?
- How to reproduce a given test on a different system?
- How to compare all the tests that have a given set of features?

# Conclusion

# Conclusion

- \* Some features of PREEMPT\_RT makes it unique with regards to system tuning
- \* Several test cases solved with minimal system and application tuning
- \* There are cases where app behavior kills the benefits of tunings
- \* Clues for fingerprinting

# Questions...?

\* The answer is 42

# References

LIBAUTOCORK

[http://git.kernel.org/p=linux/kernel/git/acme/libautocork.git;a=blob\\_plain;f=tcp\\_nodelay.txt](http://git.kernel.org/p=linux/kernel/git/acme/libautocork.git;a=blob_plain;f=tcp_nodelay.txt)

Real-Time Wiki Page

<http://rt.wiki.kernel.org>

Techniques that can have its behavior changed when the kernel is replaced

<http://oops.ghostprotocols.net:81/acme/unbehaved.txt>

Nettaps

<http://oops.ghostprotocols.net:81/acme/nettaps.tar.bz2>