Scaling FreeSWITCH Performance

ClueCon, August 2015

Moisés Silva <moy@sangoma.com>
Manager, Software Engineering

SANGOMA

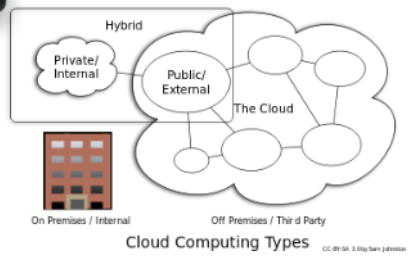
CONNECT WITH SANGOMA

About Sangoma

- Industry pioneer with over 25 years of experience in communications hardware and software
- Publicly traded company since 2000
  - TSXV: STC
- One of the most financially healthy companies in our industry
  - Growing, Profitable, Cash on the Balance Sheet, No Debt
- Mid-market sized firm with just under 100 staff in all global territories
  - Offices in Canada (Toronto), US (CA, NJ), EU (UK & Holland), APAC (India), CALA (Miami)
- World wide customer base
  - Selling direct to carriers and OEMs
  - Selling to the enterprise through a network of distribution partners

Broad Line of Great Products

- Voice Telephony Boards
  - Analog/digital/hybrid, WAN, ADSL
- Session border controllers
- Microsoft Lync
- VoIP Gateways
  - NetBorder SIP to TDM
  - SS7 to SIP
- Software Applications
  - NetBorder Express, Call Progress Analyzer…
- Transcoding (boards/appliances)
- Fiber connectivity (STM1)
- Wireless products (GSM)



We're Hiring

- Linux developers C/C++ or Python
- Anywhere in the world, relocation paid or full time remote opportunities
- Fun and relaxed work environment

Agenda

- Performance Basics
- FreeSWITCH Core Basics
- Performance Tweaks
- Feature Performance Cost
- Final Thoughts

Performance Basics

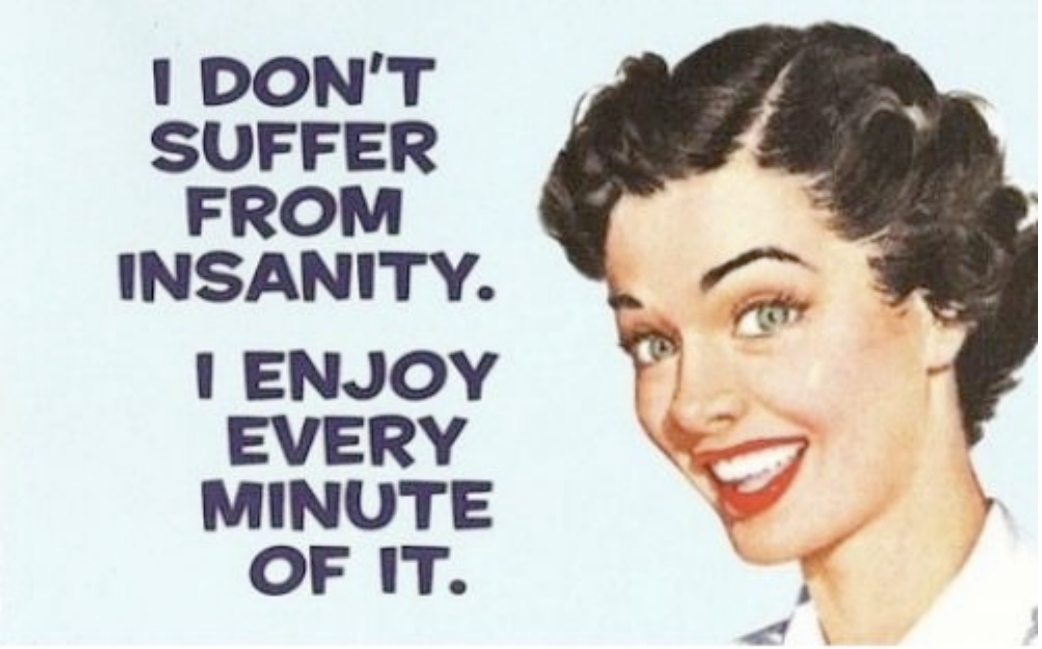
- Performance testing and measurement is hard to do and very prone to errors
- Performance can change widely from seemingly minor hardware/software changes
- This presentation focuses on Linux and SIP call bridging performance

Performance Basics

- CPU-Bound
- I/O Bound
- Threads, Resource/Lock Contention
- You cannot improve what you don't measure

FreeSWITCH Core

- FreeSWITCH is an insanely threaded system
  (the good kind of insane)



FreeSWITCH Core

- Most threads are I/O bound
  - But transcoding, transrating, tone generation introduce CPU-bound elements into the mix
- FreeSWITCH core I/O model is blocking, not async

FreeSWITCH Core

- Every call leg has its own session thread walking through a state machine, roughly, like this:
  - init -> routing -> execute app -> hangup -> reporting -> destroy

FreeSWITCH Core

- Monitoring threads per signaling stack (e.g sofia, freetdm)
  - These threads are long-lived and perform very specific tasks (e.g process SIP signaling out of a call context, initial invite etc)
- Event subsystem launches threads for event dispatch

FreeSWITCH Core

- Conferences duplicate your use of threads per call leg. For each participant you have 2 threads:
  - Session thread (handles call state and media output)
  - Input conference thread (launched when joining the conference, reads media from the session)

FreeSWITCH Core

- Even small features might launch threads
  - e.g. Setting timer=soft when performing a playback() launches an extra thread to consume media from the session

Performance Tweaks

- Logging adds stress to the event subsystem
- Every log statement is queued as an event
- Every log statement is delivered to logger modules (syslog, file, console)
- Set core logging level to warning in switch.conf.xml

Performance Tweaks

- Do not write debug logs to an SSD in a loaded system. You'll kill the SSD soon ☺
- If you want to keep debug level, you can put logs into tmpfs and rotate often

Database

- The native sqlite core database must go to tmpfs to avoid I/O bottlenecks
- On tmpfs however you risk losing SIP registration data on a power outage or any sudden restart (e.g kernel panic)
- Most other data is transient (e.g channels, sip dialogs, etc)

Database

- Eventually you might need to migrate to pgsql, mysql or some other database via odbc
- Allows you to move db workload elsewhere
- Better performance for applications that read the core info (channels, calls, etc)

Database

- Tables such as channels, calls, tasks, sip_dialogs, do not need to persist. You can move those tables to memory (e.g MEMORY engine on MySQL) if you don't need fault tolerance

- Remember to set auto-create-schemas=false and auto-clear-sql=false if you create the db schema on your own (see switch.conf.xml)

Database

- If using MySQL:
  - Use the InnoDB engine for better concurrency in data that requires persistence (e.g SIP registration)
  - innodb_flush_log_at_trx_commit=0
  - sync_binlog=0

SIP Stack

- Sofia launches the following threads per profile:
  - Main profile thread (runs sofia UA stack scheduling)
  - Worker thread (checks expired registrations)
  - Stack listener thread (accepting inbound traffic)
- You can distribute your traffic among more sofia profiles for improved concurrency

Memory Allocation

- FreeSWITCH uses memory pools
- Using modules that depend on libraries or modules not using pools can benefit from using an alternative memory allocator

Memory Allocation

- tcmalloc and jemalloc are good alternatives
- Reports on the mailing list of improvement if using mod_perl
- Sangoma found very significant improvement on its SBC (based on FreeSWITCH)

Memory Allocation

- Easy to try on your own workload:
- LD_PRELOAD=“libtcmalloc.so.x.x.x" ./freeswitch
- Recommended to run mysql with either tcmalloc or jemalloc

Dialplan

- Careful planning of your dialplan goes a long way
- Do not enable functionality you don't need, everything has a cost
- Just loading a module might be consuming precious cpu cycles

Dialplan

- Common performance factors to consider (mind the performance cost of those features):
  - Media relay
  - Tone Detection
  - Recording
  - Transcoding

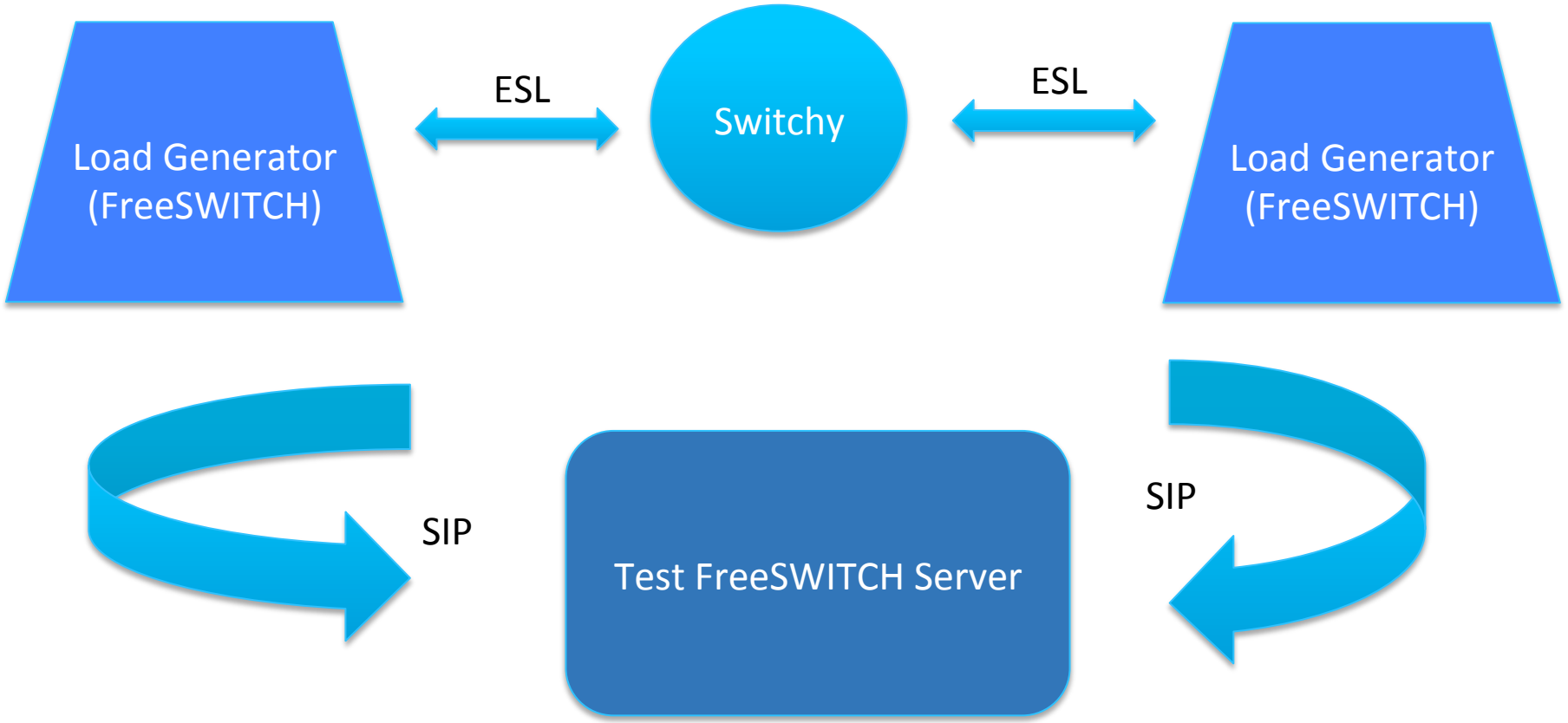
Measurement Tools

- switchy: A distributed load-generator
  - https://github.com/sangoma/switchy
- vmstat plotter
  - https://clusterbuffer.wordpress.com/2014/09/21/vmstat_plotter/

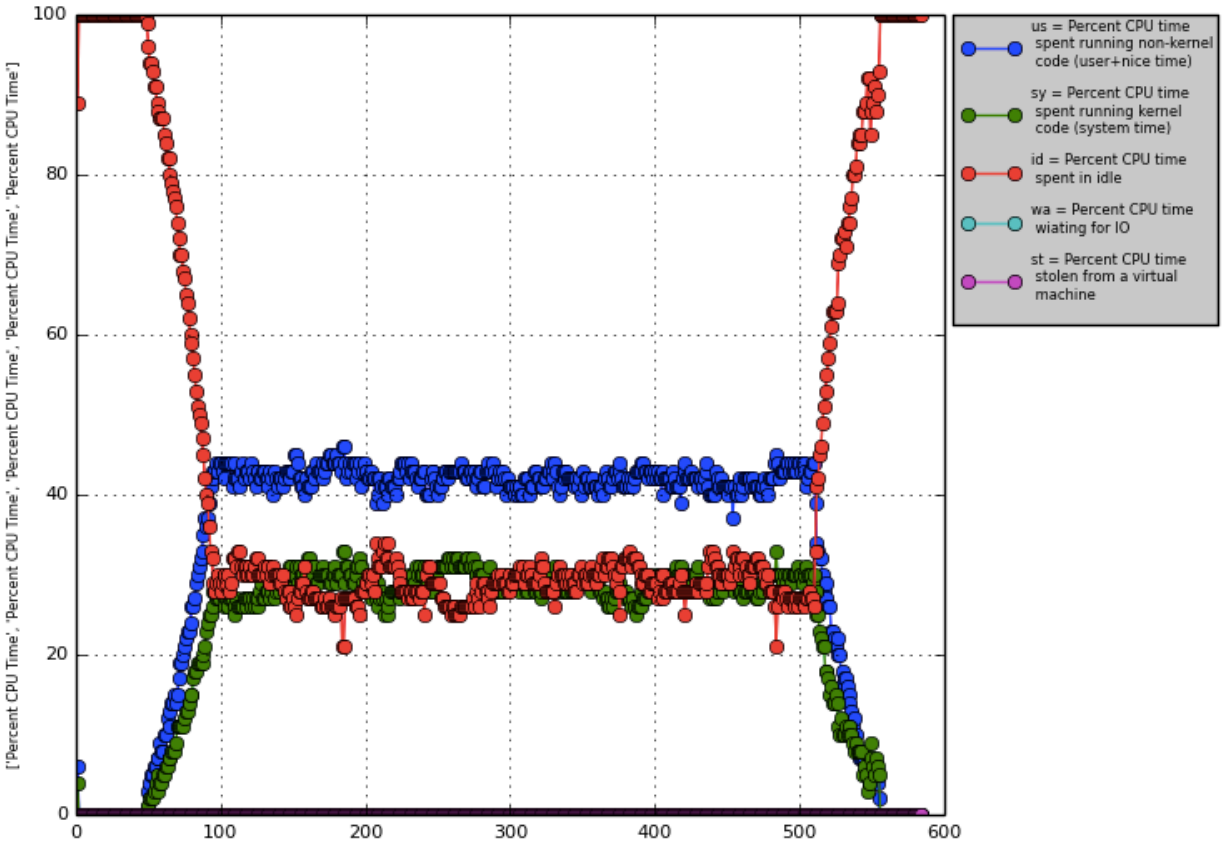
Test Server

- Linux CentOS 6 (kernel 2.6.x)
- FreeSWITCH v1.4 git branch
- Intel Xeon 64bit processor w/ 8 cores
- Intel SSD
- 16GB of RAM

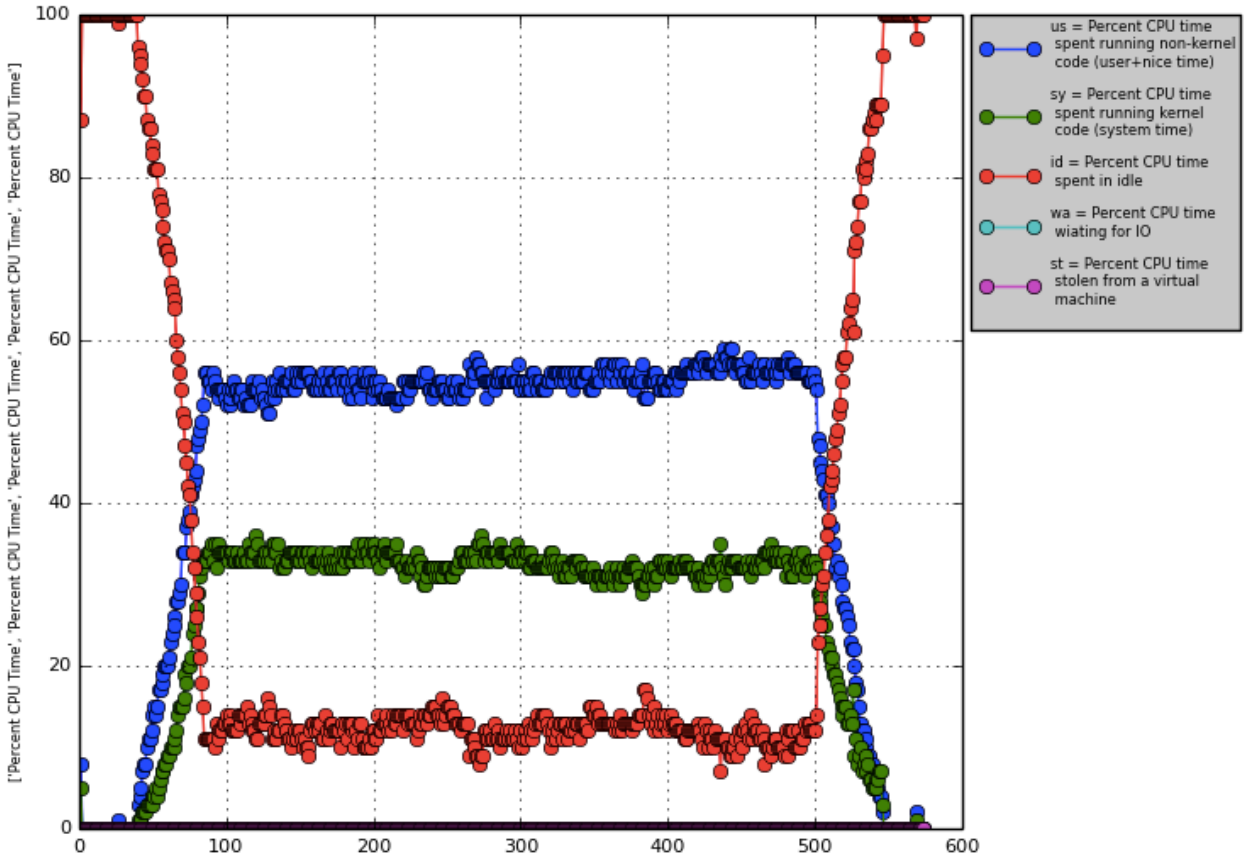
Test Lab


Load Generator
(FreeSWITCH)
ESL
Switchy
ESL
Load Generator
(FreeSWITCH)
SIP
Test FreeSWITCH Server
SIP

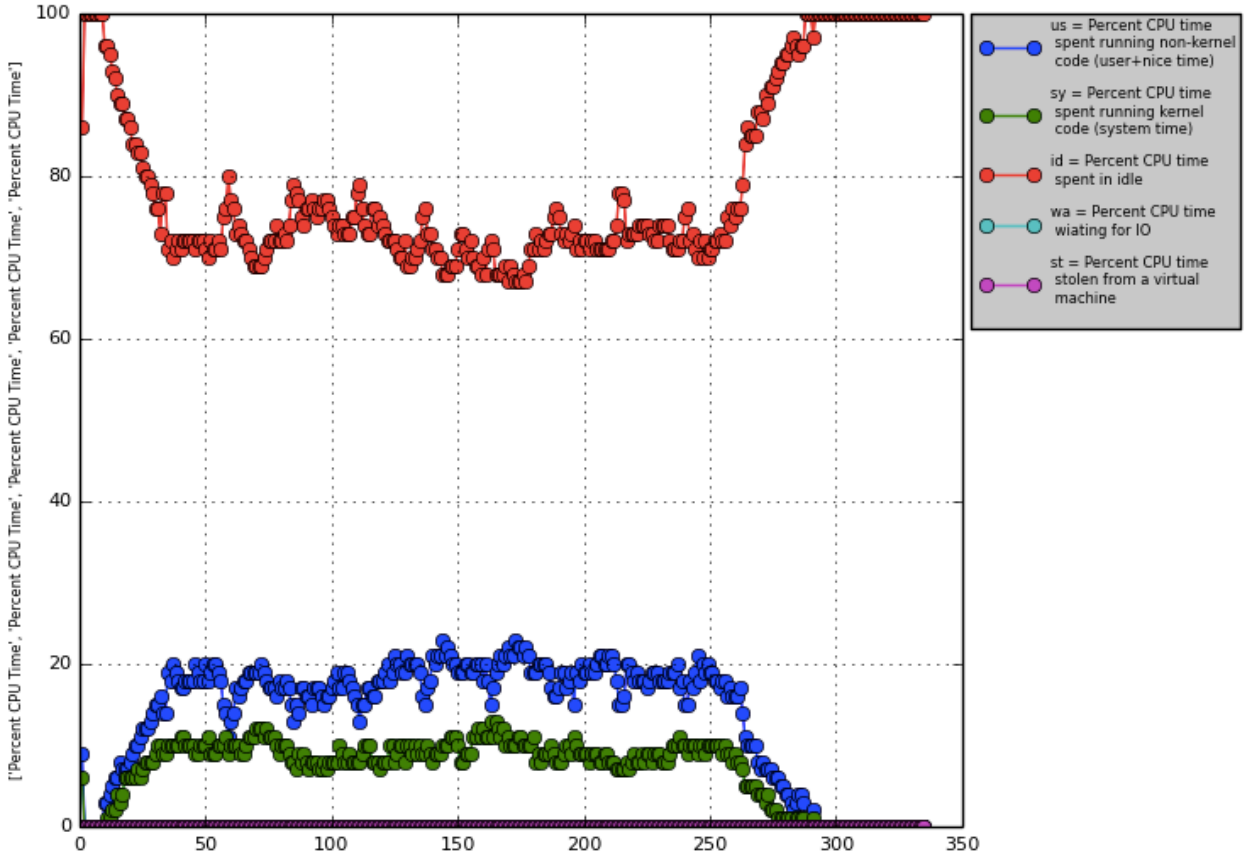
2k@50cps simple audio bridge



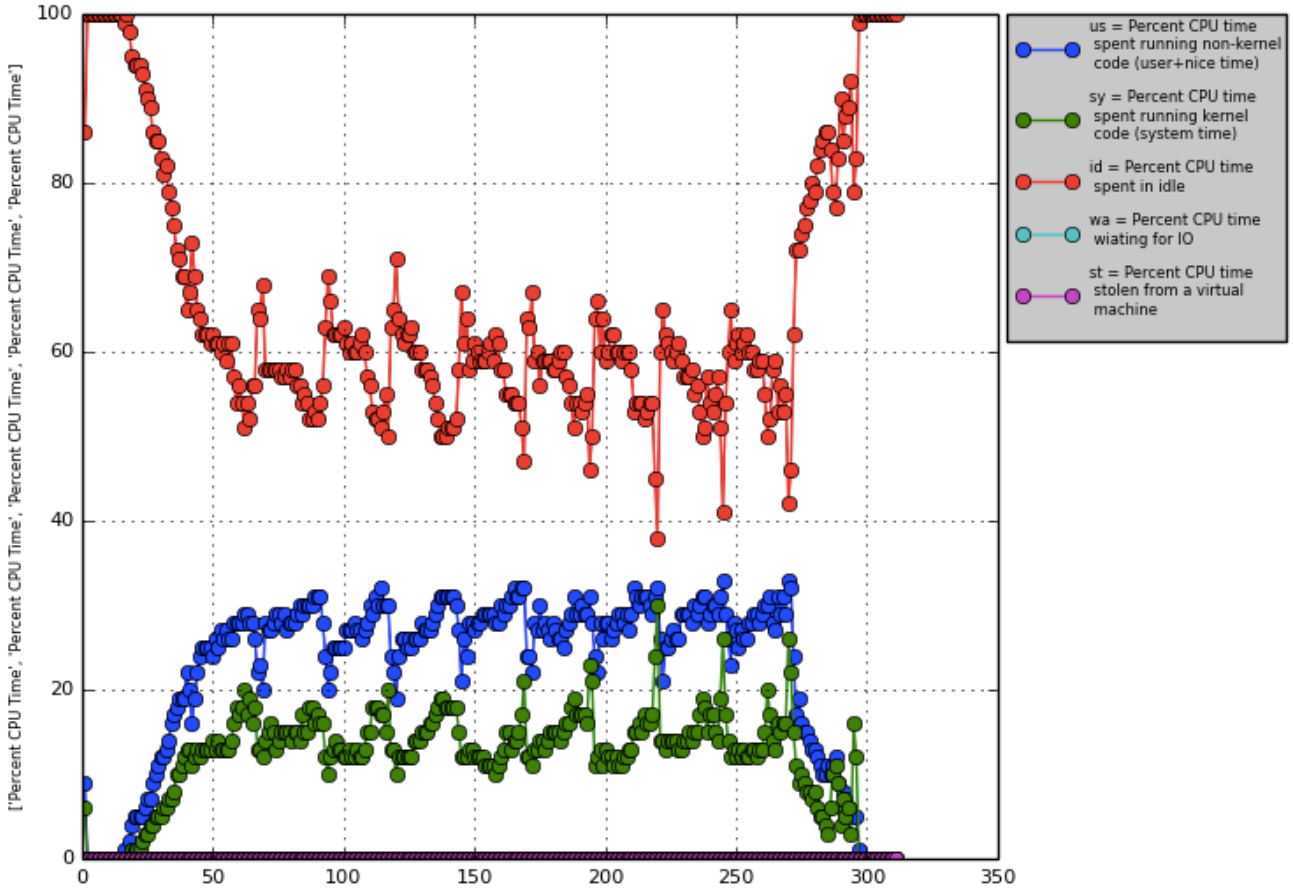
2k@50cps tone detection



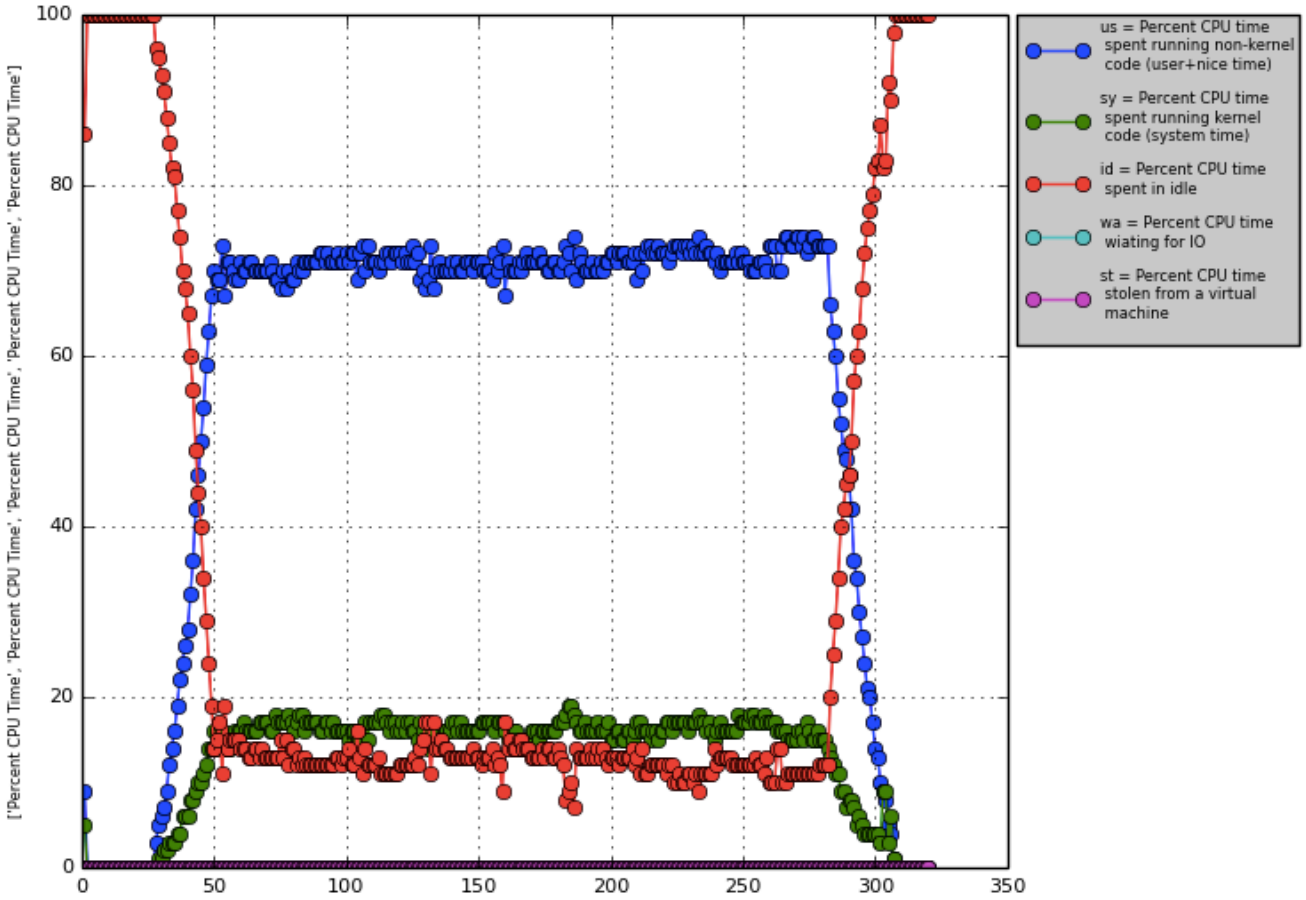
1k@50cps simple audio bridge



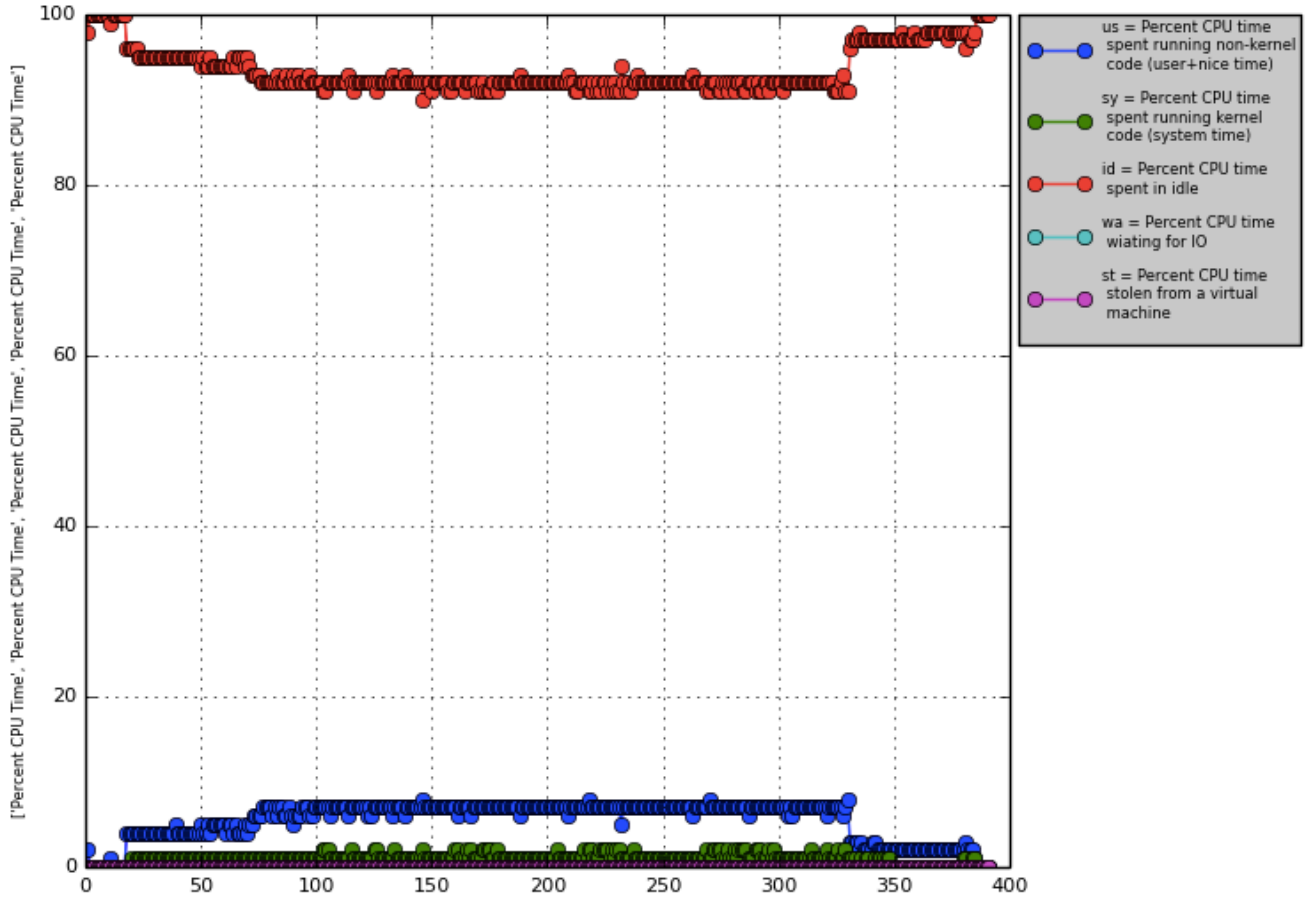
1k@50cps session recording



1k@50cps transcoding PCMU/G722



4k@80cps bypass media



Dialplan

- Use bypass media selectively whenever you can
- Avoid transcoding, use late-negotiation and inherit_codec=true
- If you must do transcoding, you can offload to a hardware transcoder

Final Thoughts

- You have to measure your own work load
- No easy answers with performance, but you have the tools to find what works for you

QUESTIONS

Contact Us

- Sangoma Technologies

  100 Renfrew Drive, Suite 100
  Markham, Ontario L3R 9R6
  Canada

- Website

  http://www.sangoma.com/

- Telephone

  +1 905 474 1990 x2 (for Sales)

- Email

  sales@sangoma.com

/Sangoma

/Sangoma

/SangomaTechnologies

blog.sangoma.com

THANK YOU

SANGOMA

CONNECT WITH SANGOMA