Visualizing gem5 via ARM DS-5 Streamline

Dam Sunwoo (dam.sunwoo@arm.com)

ARM R&D

December 2012

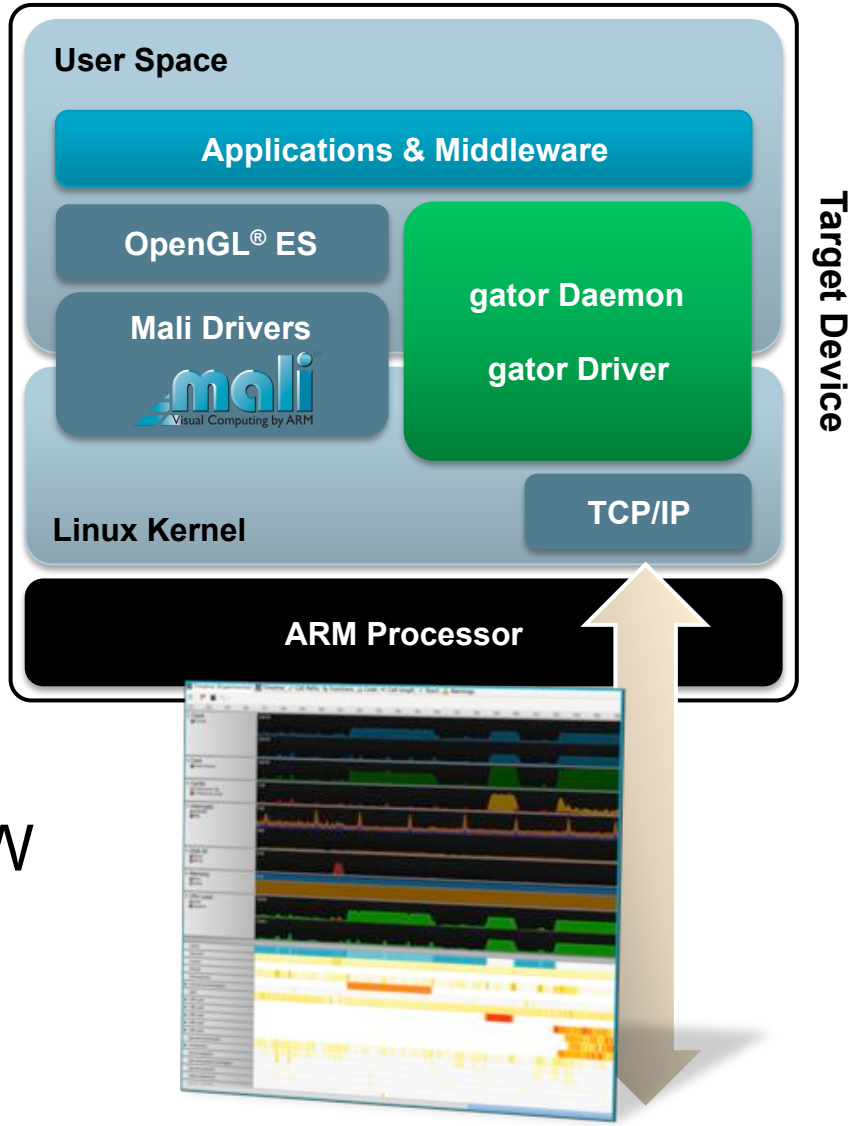
The Challenge

- System-level research and performance analysis becoming ever so ***complicated***
  - More cores and IPs in system
  - More threads in workloads
- Many interesting aspects of system remain in ***thread-level*** and ***temporal*** behavior
- Many architectural simulators (including gem5) only provide text-based statistics
  - Hard to get insight into complex system-level behavior

Good *visualization* is key!

ARM DS-5 Streamline:
System Performance Analyzer for Linux and Android

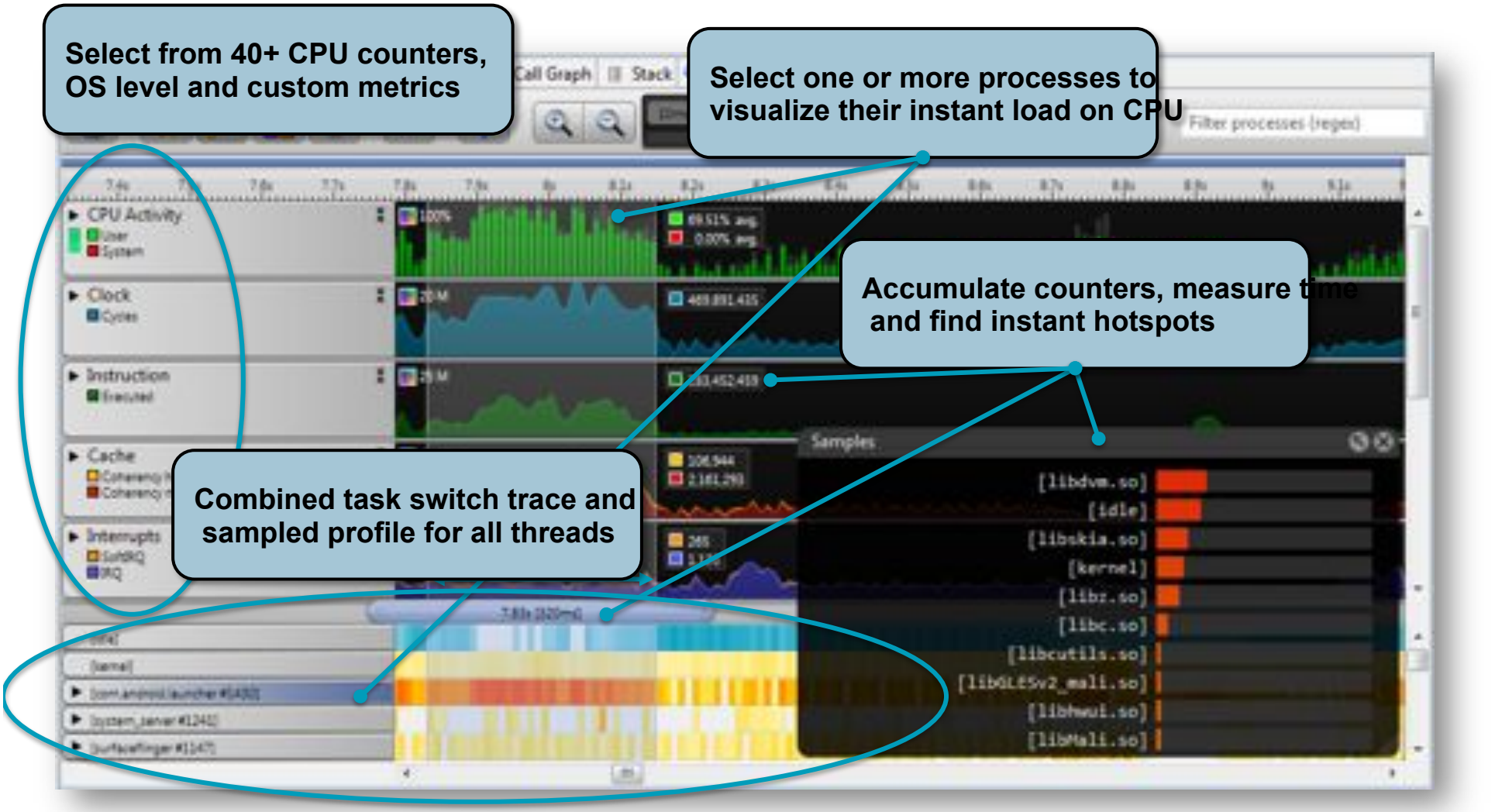
- Software based solution
  - Support for Linux kernel 2.6.32+ on target
  - Eclipse plug-in or command line
- Lightweight sample profiling
  - Time- or event*-based sampling
  - Process to C/C++ source code profiler
  - Low probe effect; <5% typically
- Multiple data sources
  - CPU and GPU H/W and S/W counters
  - Tracepoints
  - Code instrumentation
- Originally developed for real H/W platforms



* Event-based sampling is available on kernels 3.0 or later

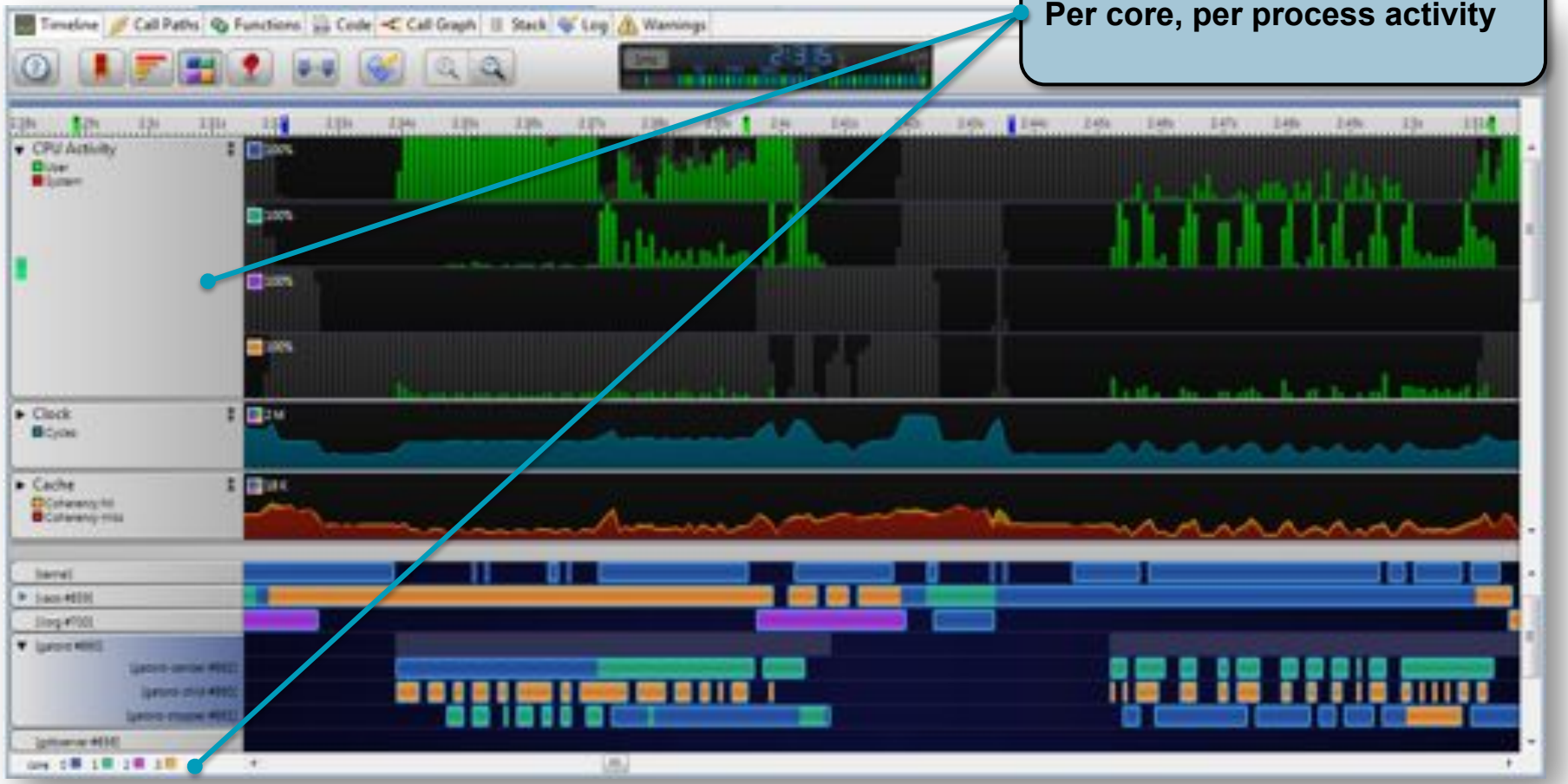
Timeline: The Big Picture

- Find hotspots, system glitches, critical conditions at a glance



SMP Analysis

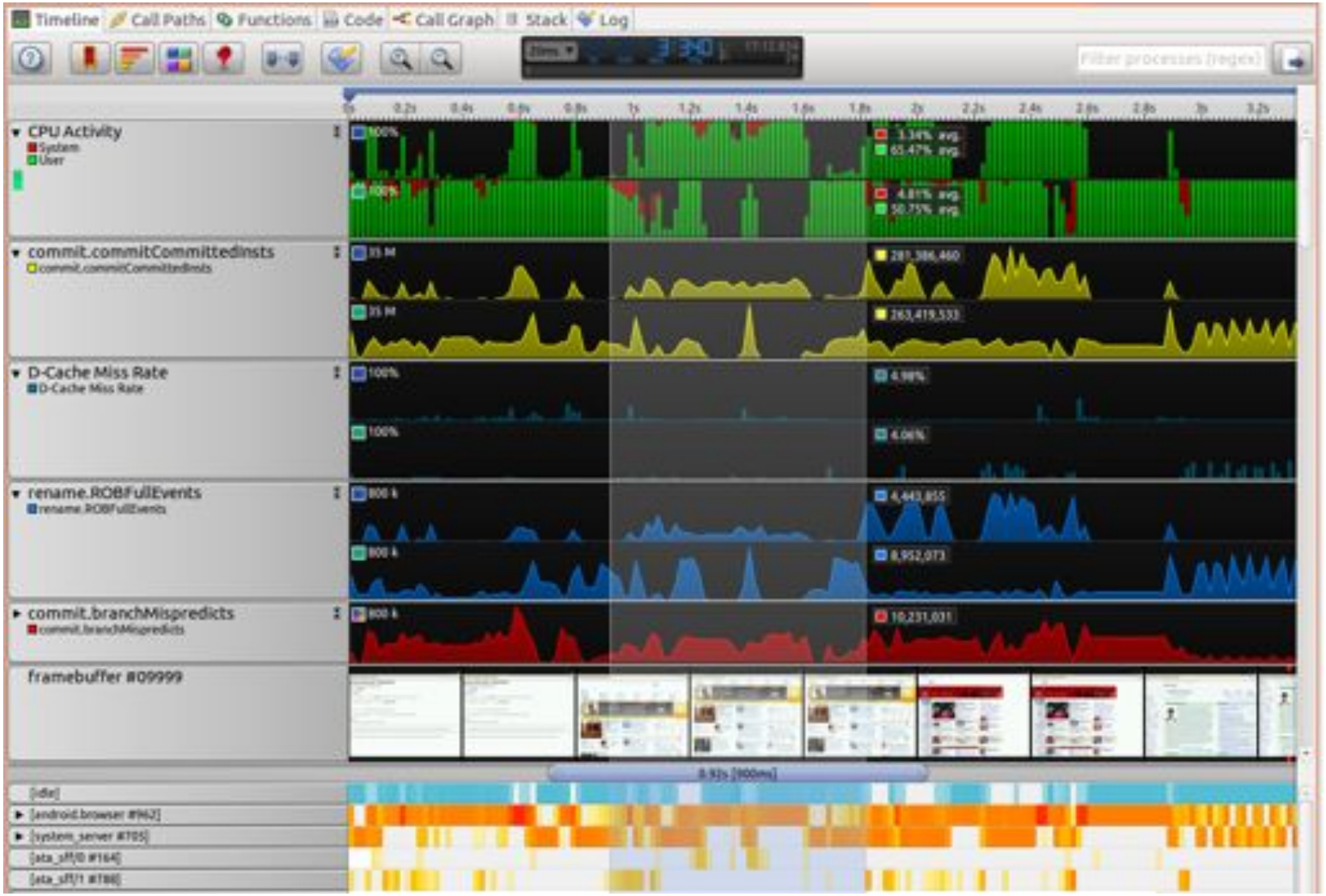
- Take advantage of multicore SMP platforms
  - Visually trace core migration and per-core statistics
  - Spot non-optimal thread synchronization and improve parall



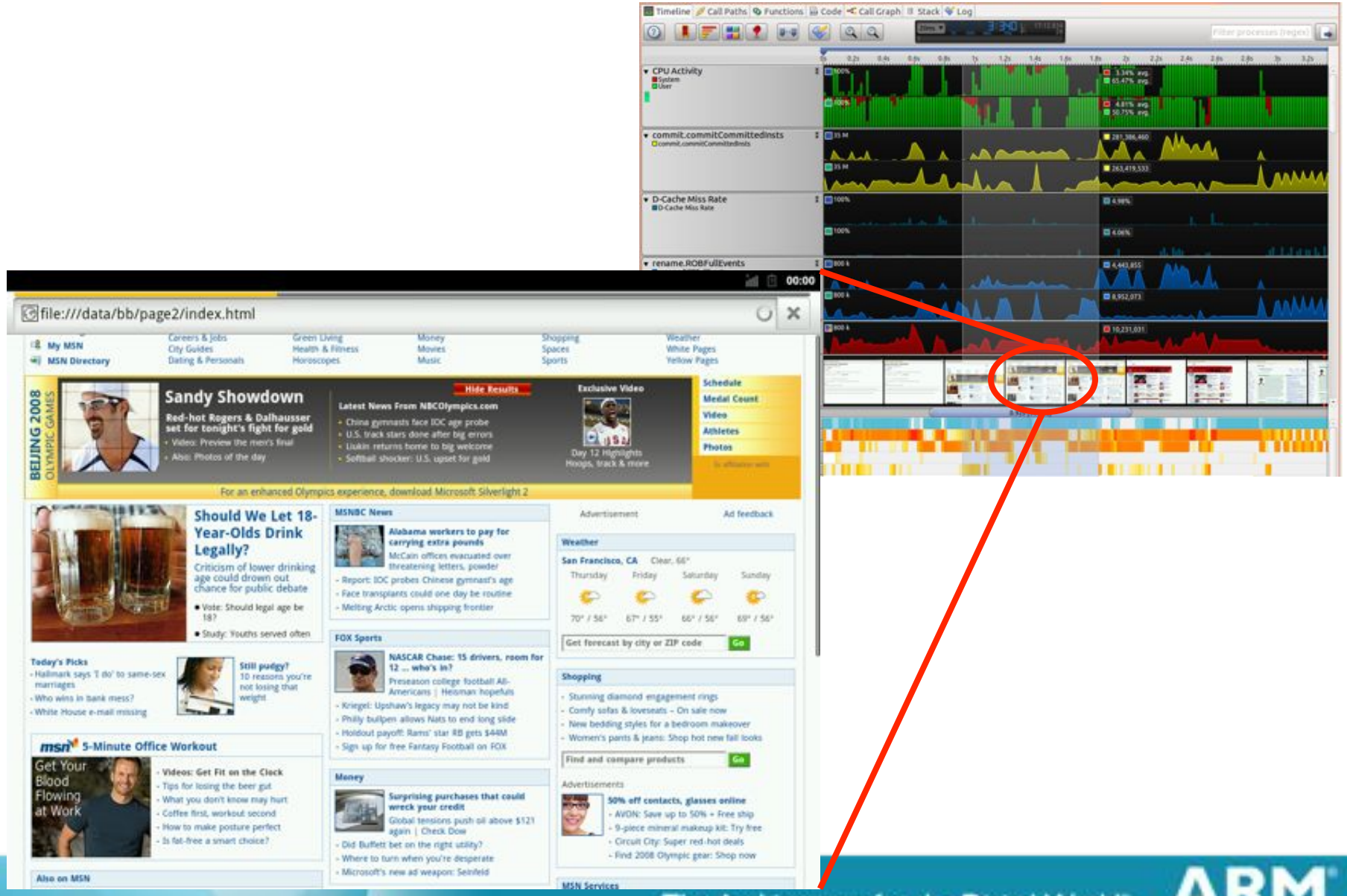
Streamline + gem5

Demo

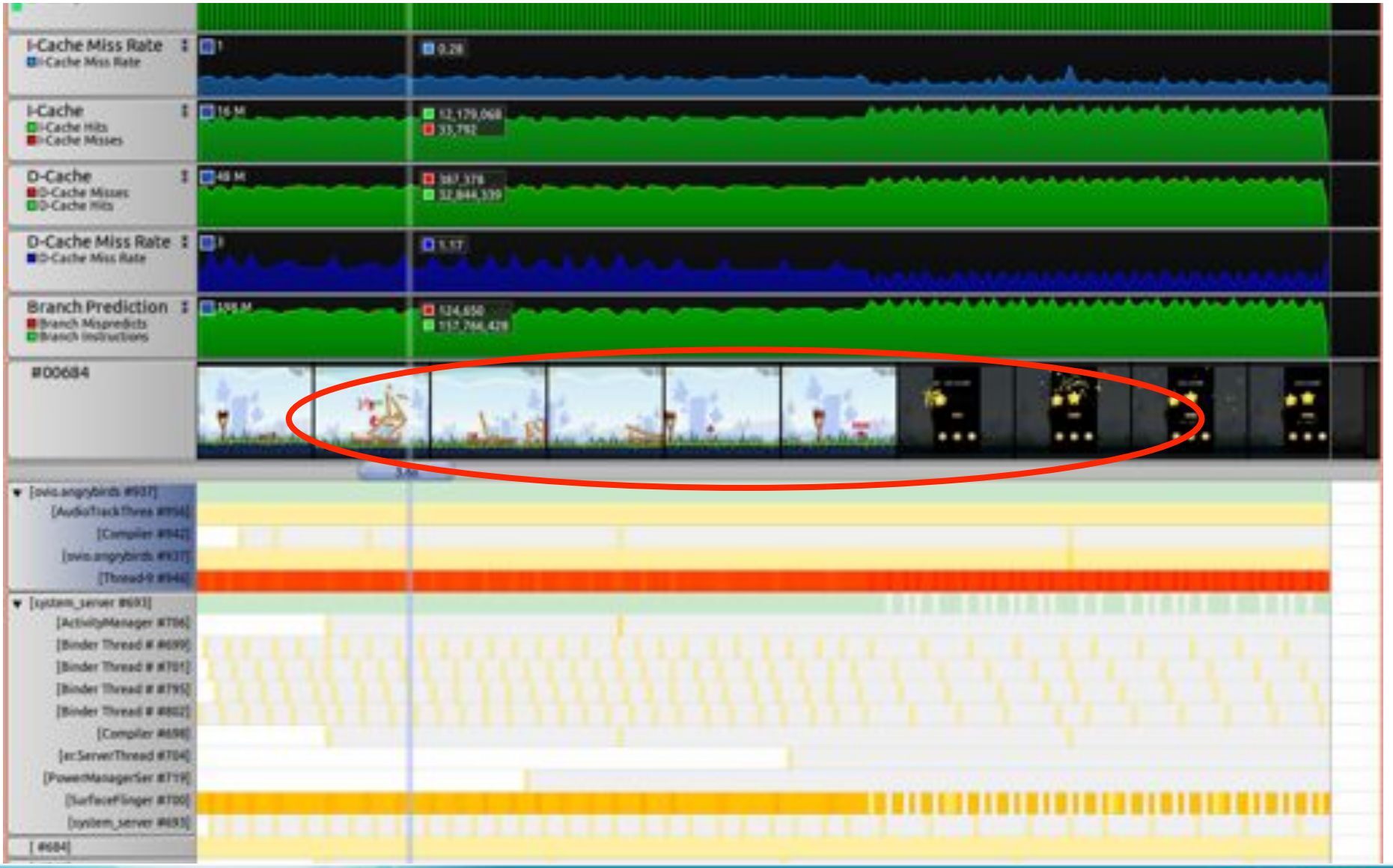
Sample Screenshot running BBench



Visual Annotation of LCD Frame Buffers



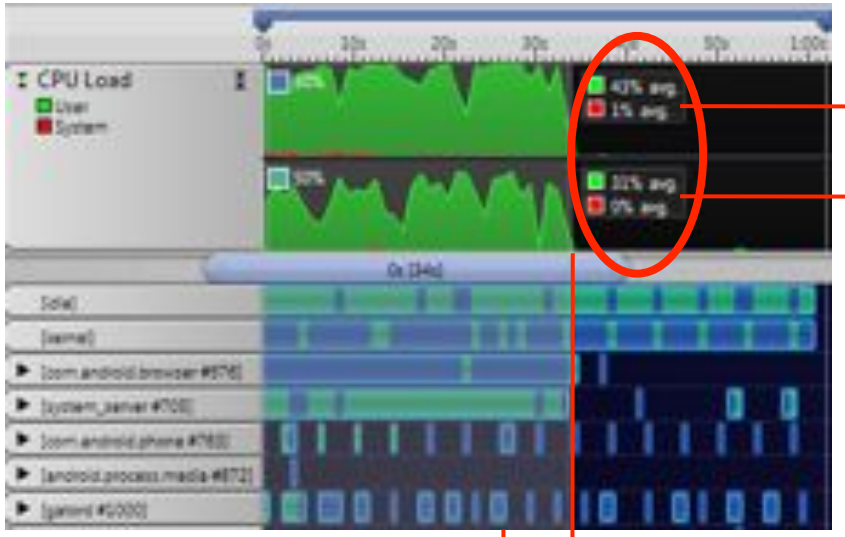
Sample Screenshot running Angry Birds



CPU Load Comparison on “MP-little-big” Config

- The two BBench runs with different schedulers resumed from exact same checkpoint
- aMP-aware scheduler correctly puts more load on big core
- BBench finishes 23% sooner with aMP-aware scheduler in this experiment

▪ **Default Scheduler**



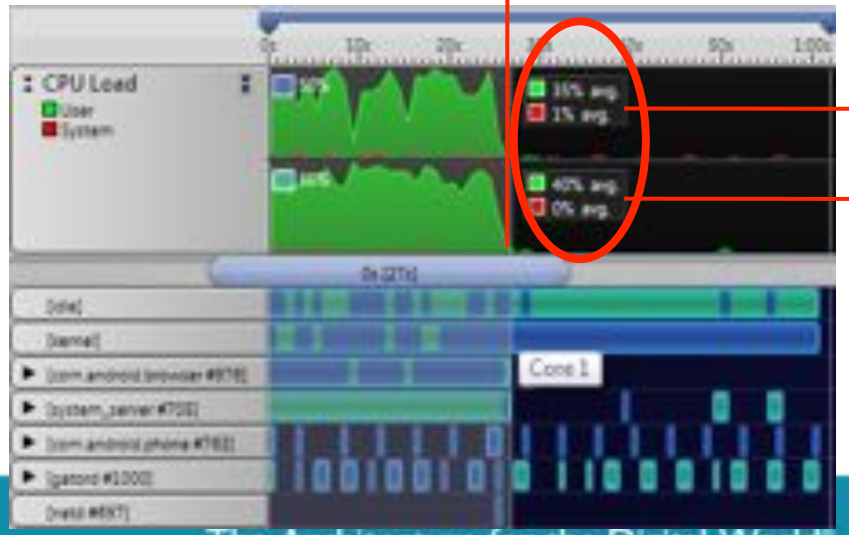
CPU loads out of 50% per core

Little Core Load

Big Core Load

23% improvement

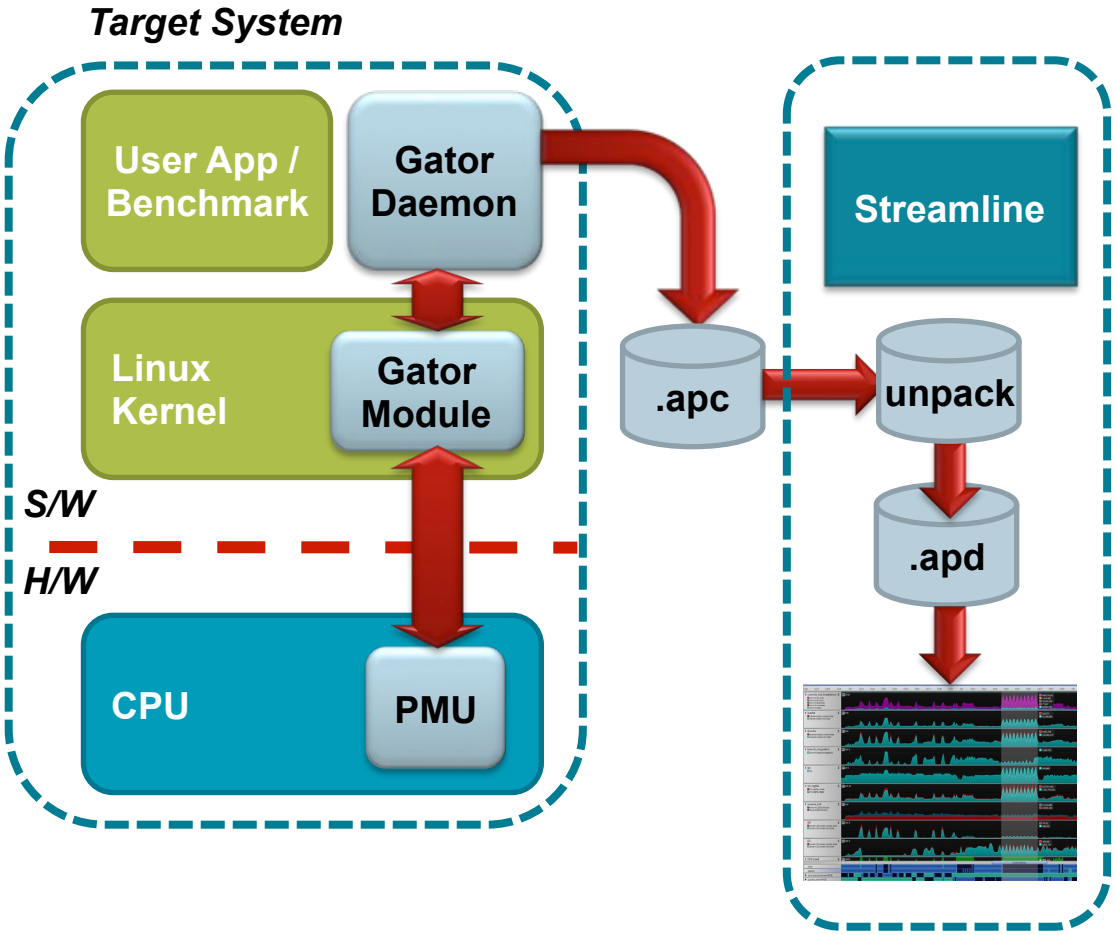
▪ **aMP-aware Scheduler**



Little Core Load

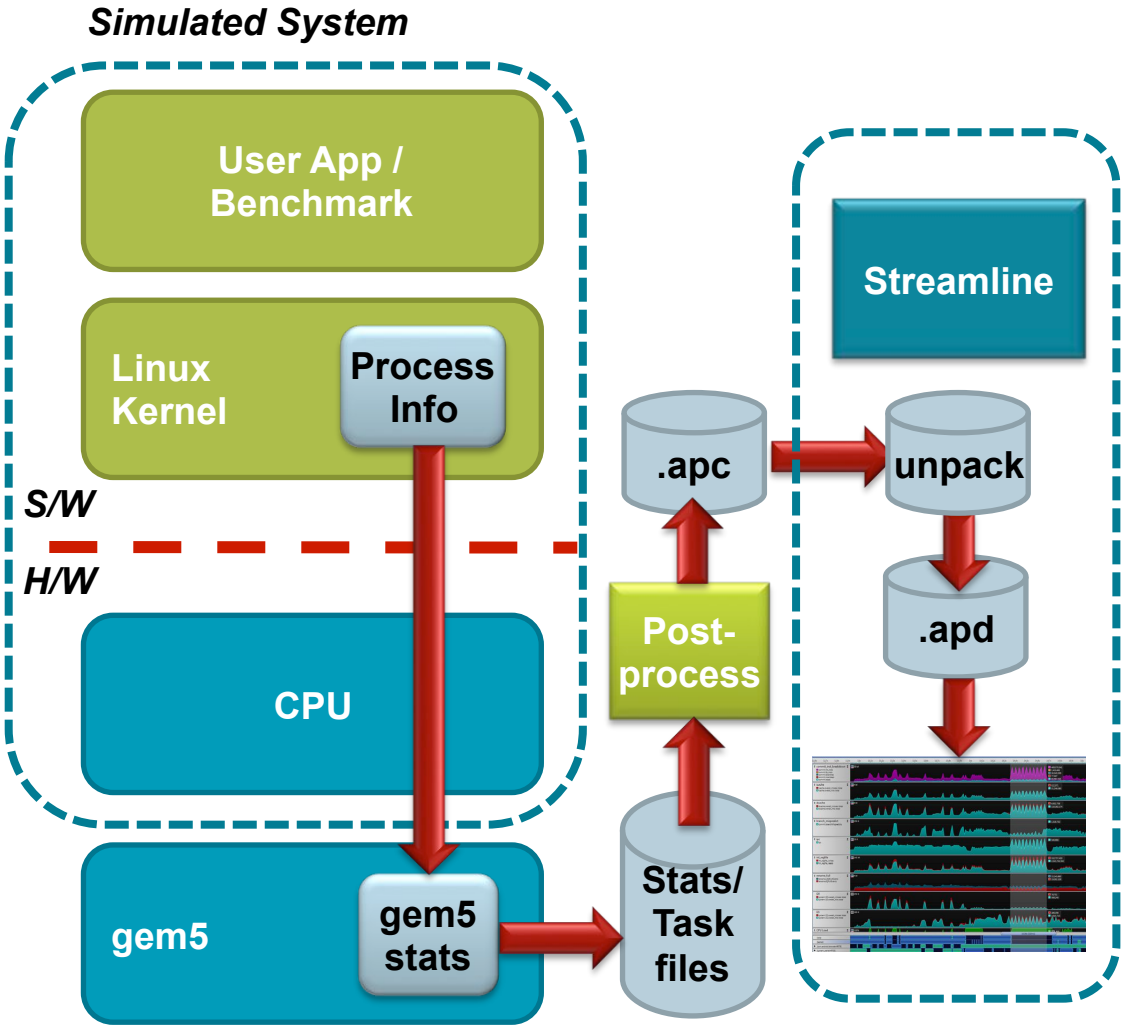
Big Core Load

Original Streamline Capture Flow



- Relies on “gator” kernel module and daemon
- Reads out counters and process information and dumps to file

Streamline+gem5 Flow



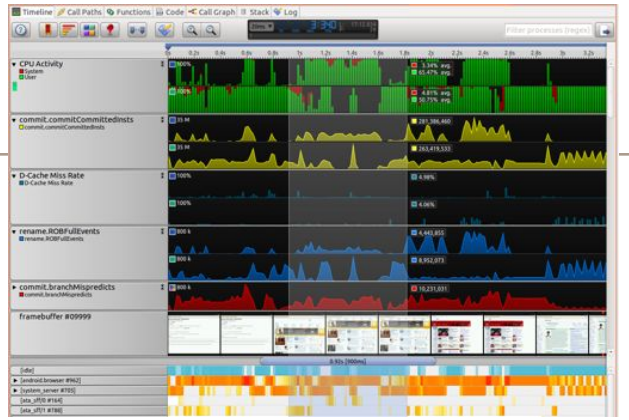
- ***Gator-free*!**
  - Special kernel module/ daemon not required anymore
  - Zero probe effects
  - Can capture data for bare-metal runs as well
- Linux process/thread info and gem5 stats dumped at every context switch
- Simple single-pass flow

How do I get started?

- Streamline 5.12 Community Edition available now for free!
  - Details on http://www.arm.com/products/tools/streamline-for-gem5.php
- Slightly modified Linux/Android kernel
  - Add “m5struct” to let gem5 know of offsets of certain kernel struct fields (pid, tgid, comm (task name), mm (mem map), etc.)
- Enable enableContextSwitchStatsDump flag in LinuxArmSystem
  - Dumps stats at context switches (callback for __switch_to())
  - Dumps process info (pid, tgid, task name, cpu id) at context switches
- Enable frame_capture (optional)
  - Dump frame-by-frame output in gzipped bmp format for visual annotation
- Post-process script
  - Uses gem5 stats / process info / frames to generate Streamline .apc project file from scratch (without gator)

Streamline available for download now!
gem5 changes and scripts to be available very shortly. Stay tuned!

Summary



- **Streamline+gem5** enables great ***visualization*** of complex system behavior in an effortless manner
  - Process / Thread information
    - Crucial in understanding OS scheduling behavior in complex multi-threaded benchmarks
  - Temporal behavior of benchmarks
    - Easier to digest than Giga-bytes of text in stats file
  - Better visualization
    - Various features and views to help better understand results
    - Pretty screenshots for papers and presentations ☺
- Any questions or feedback are welcome (dam.sunwoo@arm.com)