



COTSon: Infrastructure for Full-System Simulation

Ayose Falcón

HP Labs — Exascale Computing Lab



“Oh no, not Yet Another Simulator!”

- We don't want to re-invent the wheel
 - Not a “new simulator”, but a “new framework”
- **Solid Foundations**
 - CPU simulation uses established, validated, supported tools with support for commodity OSs, complex multi-tier applications, new platforms
- **Leverage**
 - A “pluggable” architecture to quickly leverage other existing simulators for individual sub-components (such as disks or networks)
- **Statistical Approach**
 - Abandon the idea of a ‘always-on’ cycle-based simulation in favor of statistical sampling across the board

Why COTSon?



The challenge

Fully and faithfully simulate
a large scale system of
thousands of computing nodes,
with hundreds of cores each,
in an affordable amount of time

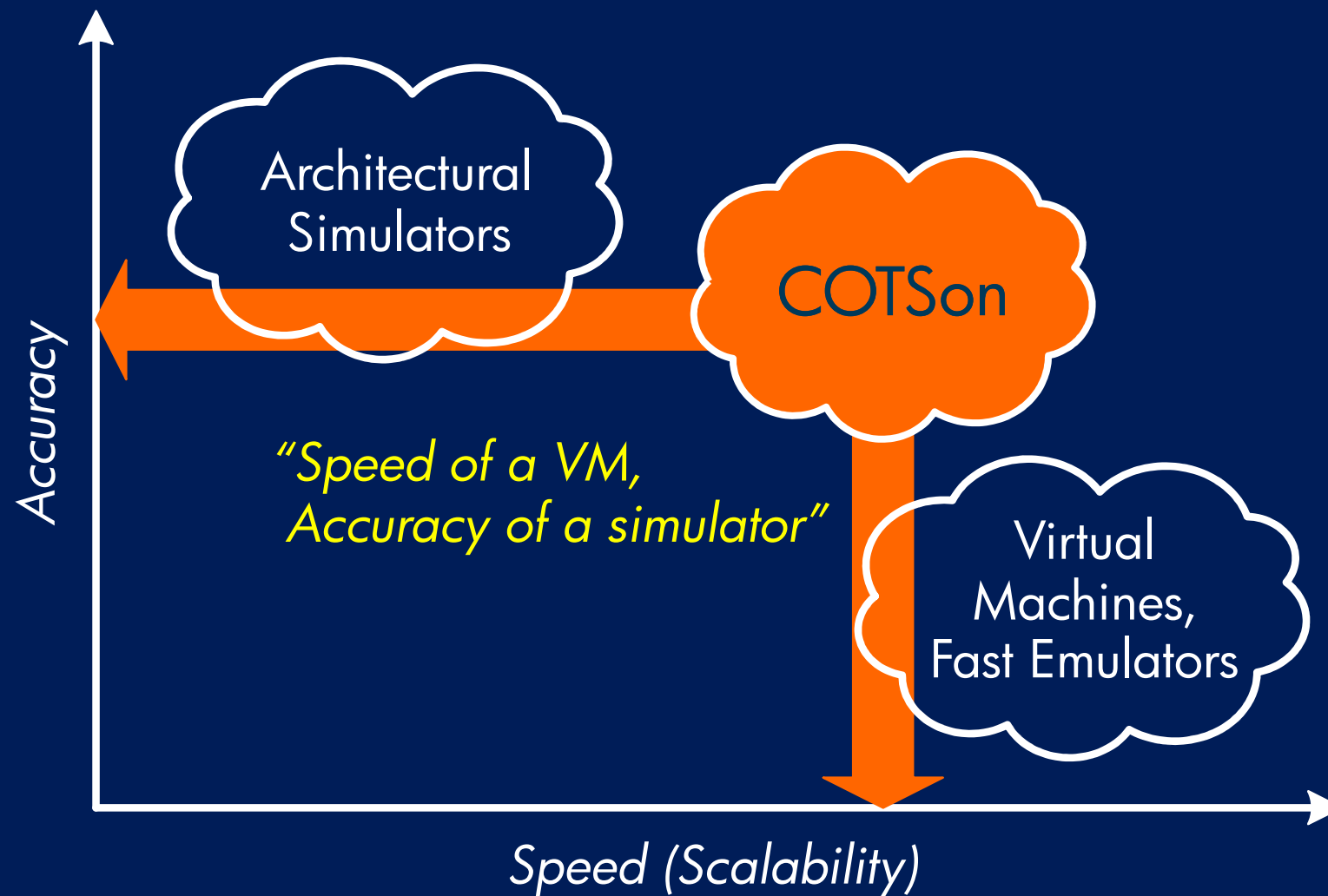
Why is this important? [research]

- Simulation is the basis of architecture research
 - Too complex/expensive/long to build prototypes
 - Analytical models miss important behavior
 - Adding instrumentation to real systems may not be feasible
- Traditional simulation does not address **new needs**
 - Multiprocessor and multithreaded simulations
 - Storage subsystem and network simulation
 - Full system, OS and application support
 - Supporting micro- and macro-benchmarking
 - Trading off accuracy and speed
 - Robust and statistically validated

Why is this important? [business]

- A large and growing fraction of HP's server business is in **large "high-value" computing deals**
 - Traditional High-Performance Computing
+ new "Cloud Computing" markets
- Customers want the **optimal performance/\$/Watt**
 - This turns the bid proposal process into a multi-dimensional optimization problem to select the right components
 - New complex emerging workloads break the traditional modeling schemes based on offline characterization and linear extrapolations
- **Better modeling translates to customer value**, increased margins, higher customer loyalty and attach
 - For example, a better CPU choice could swing ~\$100, or about 2–3 points of profitability

We need a new breed of simulators



COTSon: Simulating the future

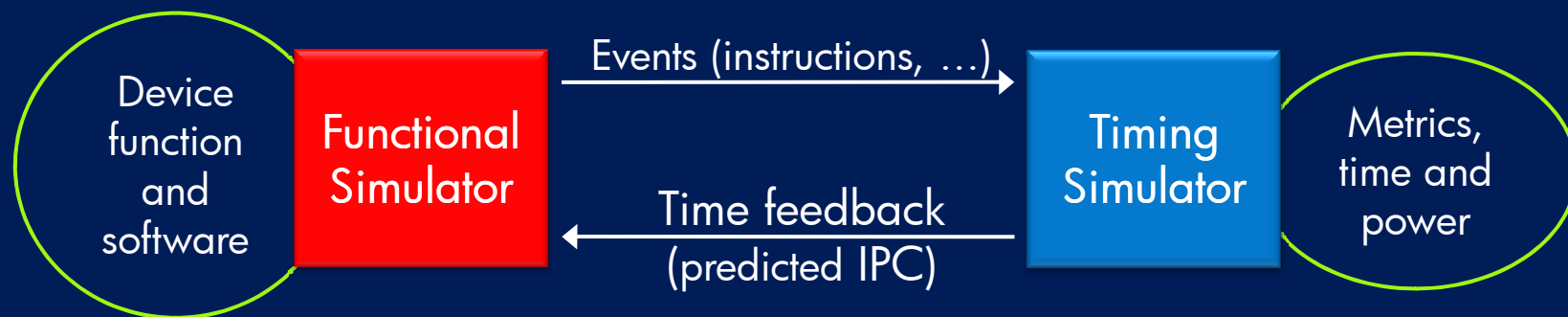
- COTSon is a framework for system-level simulation
 - Fast enough to run the full OS and applications
 - Models clusters of multicore CPUs, networking and storage
- We think of it as a **Virtual Machine with Timing**
 - Full execution-based performance target
 - Fast simulation with accurate timing models and smart sampling
 - Enables speed/accuracy tradeoffs at multiple levels
 - Supports scale-up and scale-out
 - Tested up to ~100 nodes and ~1000 cores
 - Speed/Accuracy: ~15% accuracy error at 15x slower than native
- Key components
 - AMD SimNow™ platform simulator
 - HP Labs timing, sampling and clustering

COTSon architecture

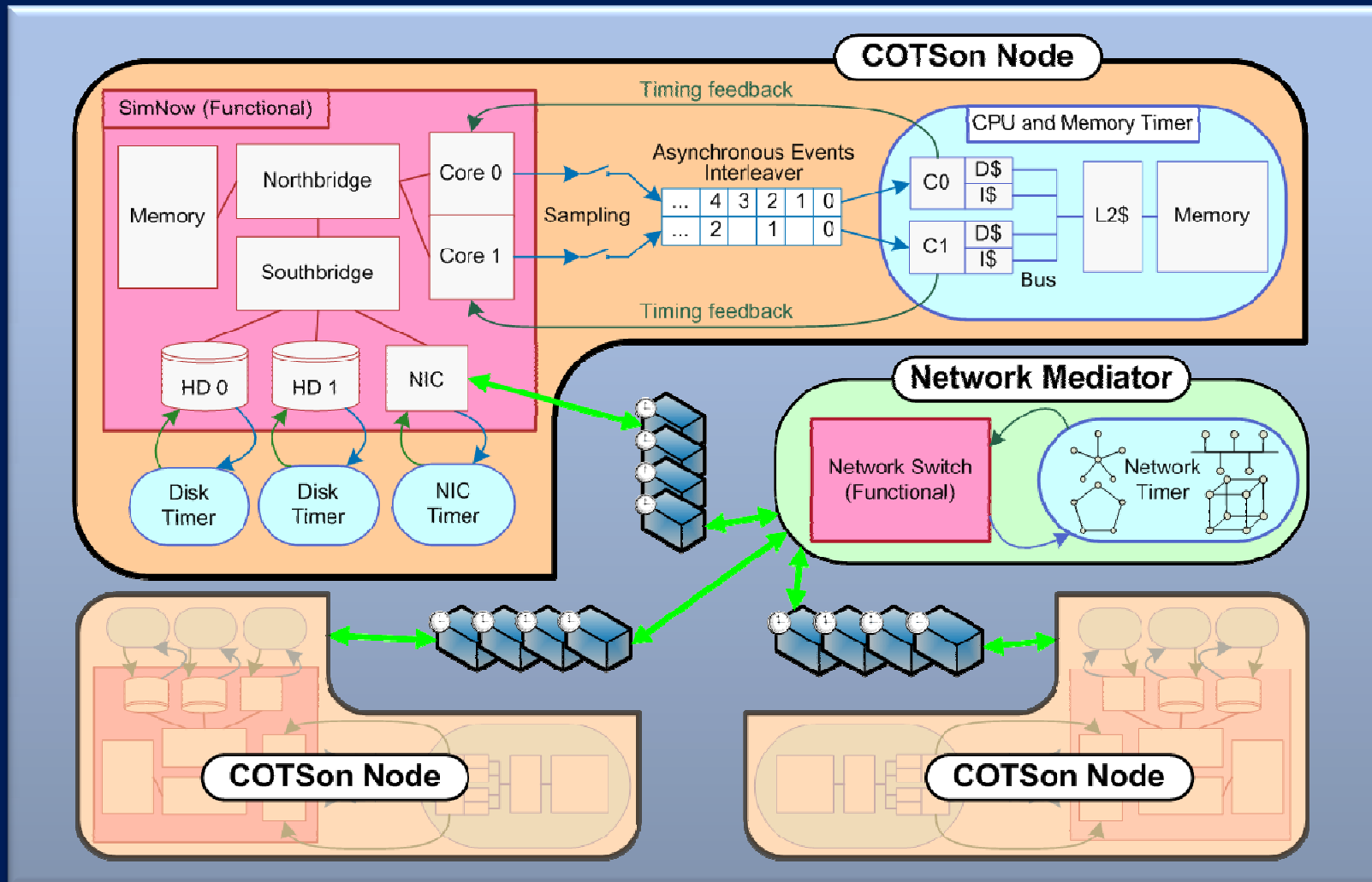


Decoupling simulation

- **Functional Simulation (fast)**
 - Emulates the behavior of all the components of our system
 - Disks, video, network cards, etc.
 - Necessary to verify correctness, run software
- **Timing Simulation (slow)**
 - Models the timing of all the components
 - Used to measure performance (or power)
- **COTSon approach:**
“Functional Directed with sampling and time feedback”

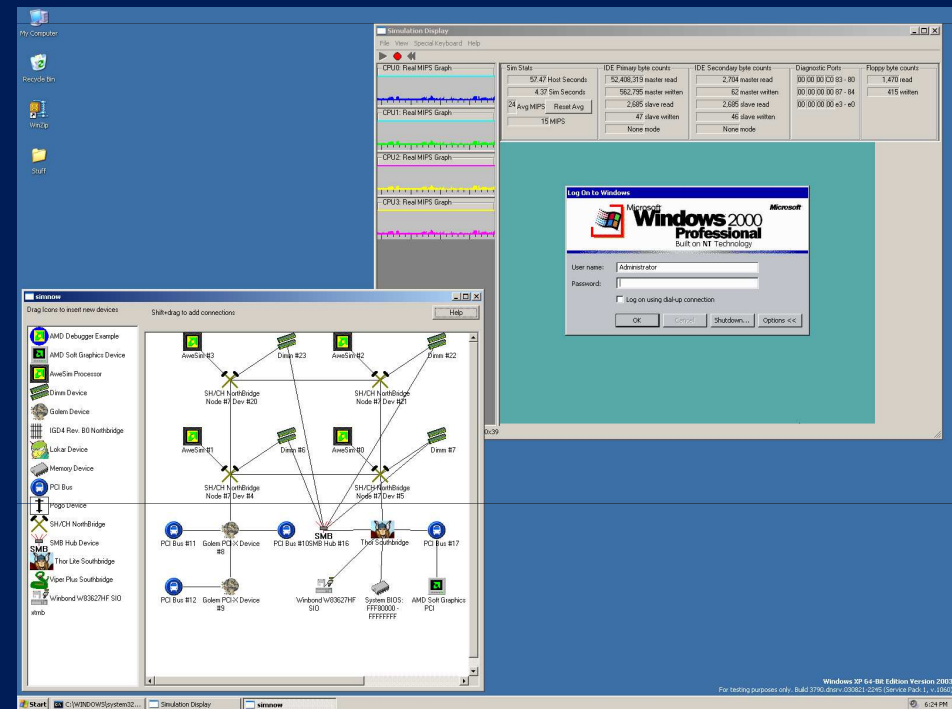


COTSon components



AMD SimNow™

- Fast and configurable x86-64 full-system emulator, using dynamic compilation and caching techniques
 - Supports unmodified BIOS's and OS's and executing real applications over it
 - Models the entire computer (CPU + devices)
 - Faster than other x86 simulators (~10x slowdown)
 - Used internally at AMD for
 - Performance analysis of new CPUs/GPUs/Chipsets
 - OS and BIOS testing
 - Application and driver testing



<http://developer.amd.com/simnow.jsp>

Samplers

- Decide when and how much to simulate and when to move from one simulation state to another
 - **Functional** → fast forward to the next state as quickly as possible
 - **Warming** (simple/detailed) → get data in stateful structures (e.g., caches), but do not account for time
 - **Simulation** → account for time
- Pluggable architecture
- Many implementations
 - Smarts, SimPoint, Dynamic Sampling, Random, Interval-based, ...
- Samplers are what provide the **major acceleration component**
 - Even for very accurate (hence slow) timing models, a good sampler only needs to invoke the timing model $< 1\%$ of the time.

Timers (a.k.a. CPU/device models)

- Pluggable architecture
- **CPU timers**
 - Accept instructions, process them and update metrics
 - All timers share the memory hierarchy
 - Some “must have” metrics → cycles and instructions
 - Current CPU models
 - Timer0 → simple “linear” model + cache hierarchy
 - Timer1 → Timer0 + in-order pipeline
 - Bandwidth → Only limited by memory bandwidth
 - PTLsim (open source) → full x86, OoO superscalar (linked to COTSon)
 - Not only CPU models, but also:
 - Profiling
 - Disassembly/trace generation
 - “SimPoint”-like analysis
- **Disk/network timers**
 - DiskSim simulator
 - Network topologies

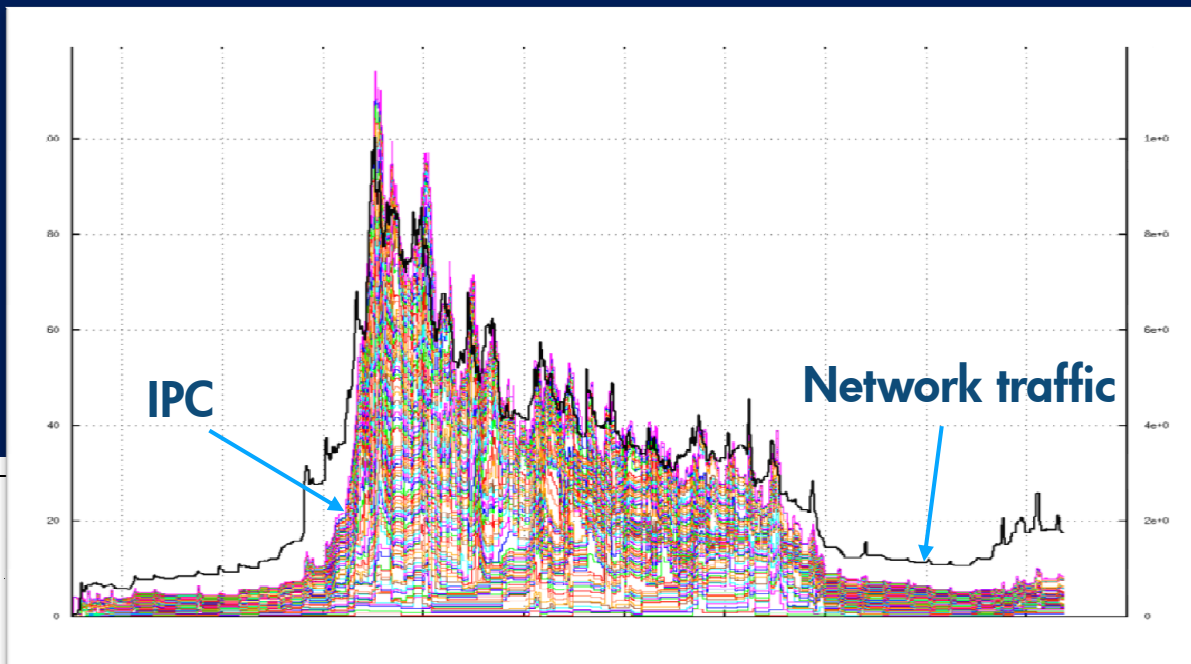
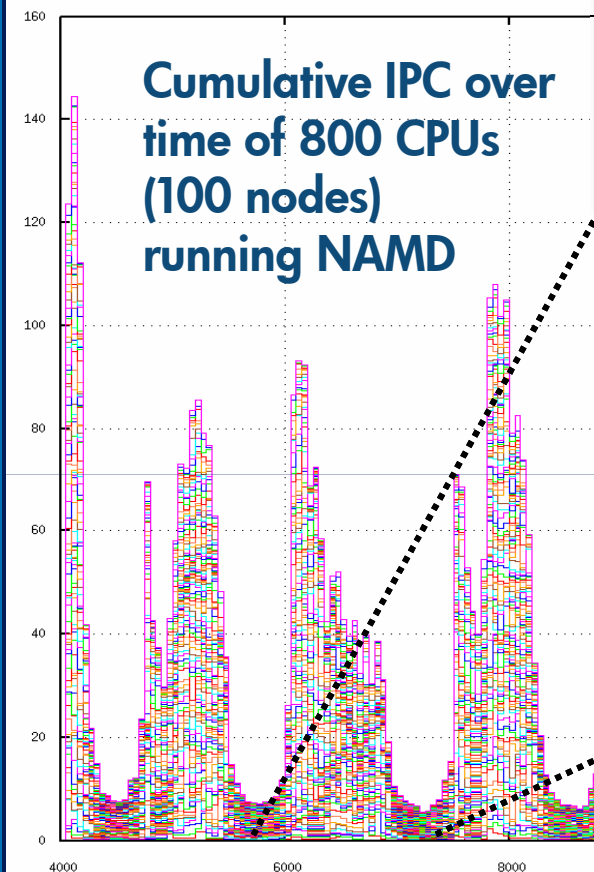
Current status and future work



Speed vs. Accuracy Tradeoffs

- We can play the speed vs. accuracy game at several control point
 - Within a node: dynamic sampling sensitivity
 - At cluster level: adaptive synchronization quantum range
- By choosing the appropriate values we can reach
 - Single node accuracy in the order of 10–15% error
 - Networking accuracy (microbenchmark) up to 15 Gb/s
 - All of the above with self-relative slowdown (vs. native) of ~15–30x
- **Improvement Areas**
 - Better CPU models (if needed), especially in the SMP coherency area
 - SMP and cluster validation on larger applications
 - Distributed simulation sometimes “unstable” for large clusters (> 50 nodes)
 - Make it easier the accuracy-speed tradeoff selection for non-expert users

Putting it all together



COTSon Success Stories

- **Fault isolation / availability for commodity architectures**
 - “Configurable isolation: building high-availability systems with commodity multi-core processors” (ISCA’07)
 - “Isolation in Commodity Multicore Processors” (IEEE MICRO’07)
 - “Implementing High Availability Memory with a Duplication Cache” (MICRO’08)
- **Nanophotonics architecture investigation**
 - “Corona: System implications of emerging nanophotonic technology” (ISCA’08)
- **Last level cache technologies study (CACTI)**
 - “A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies” (ISCA’08)
- **Web 2.0 workload analysis**
 - “Microblades and megaservers: system architectures for emerging Web 2.0 / internet workloads” (ISCA’08)
- **...and some other internal projects at HP Labs**

What's next

- We have invested many engineering years on this
 - SimNow fully validated and supported by AMD
 - We are a research lab, not a development team
 - HP is not in the simulation market
- We are now open for academic collaborations
 - Initially not as open-source model, but through SDKs
 - COTSon plug-ins for PTLSim, DiskSim, CACTI, ... could be opened
- We see value in COTSon becoming a “standard” tool
 - Fostering research on technology HP cares about
 - Widely used and supported by the community
 - Integration of **other CPU front-ends** (e.g., QEMU)
 - Integration of **other CPU back-ends** (SimpleScalar, M5, PTLSim, ...)
 - Integration of **other device simulators** (network, power models, ...)

Thanks

