**NAME:** Maria Krstevski

**DATE:** JUL 17, 2024

**Part 1: Research Paper Exploration**

1. **Summary:**

* Read the research paper thoroughly.
* Write a concise summary (300-400 words) explaining the key objectives, methodologies, and findings of the study.

1. **Critical Analysis:**

* Identify and discuss three major advantages of computing applications as presented in the paper.
* Identify and discuss two potential limitations highlighted by the authors.

1. **Application:**

* Explain how the methodologies and findings from this paper could be applied to your course activities

1. The research paper focused on advancements and techniques in computer architecture simulation by emphasizing the evaluation and comparison of different simulation methods and tools. Some key objectives of the paper include providing an updated survey of computer architecture simulation techniques, categorizing and analyzing various simulators, and comparing their performance and accuracy to real hardware. The paper provided insight for researchers interested in working with simulations, to assist in choosing appropriate simulators and validating simulation results effectively. Different types of computer architecture simulators were analyzed by organizing them into groups and examining them in detail. This involved looking at things like features, capabilities, and performance metrics to understand the strengths and weaknesses of the simulators, and to compare them. By doing this, the paper helps in identifying which simulators might be best suited for their specific needs and applications. Both functional and timing simulators were the focus of the research. The methodologies involve comparing the accuracy of simulation results against real hardware, using methods like instructions per cycle (IPC) and memory performance. Different simulation techniques, like cycle-level and event-driven simulations are examined to discuss the validation methods and challenges in ensuring simulator accuracy.

**Survey of simulators:** An up-to-date review of different simulator, evaluating and organizing different computer architecture simulators based on how detailed their simulations are, and what they are used for. This is helpful, to understand which simulators are detailed or general, and the specifics of their application like processor design or system analysis. It highlights the strengths and weaknesses of functional versus timing simulators, offering a detailed comparison of 6 modern x86 simulators.

**Performance and Accuracy Comparison:** Identifies key differences in performance and accuracy among various simulators. For example, functional simulators are faster but less detailed, while timing simulators offer in depth performance metrics, but are slower and use more memory.

**Challenges and Solutions:** Common challenges in simulation are discussed and looked at, such as issues with validation and accuracy. Some solutions that were suggested for improving simulator accuracy and performance include using dynamic binary instrumentation tools and refining simulation methodologies.

**Findings:**

**Simulation Speed**: ZSim is the fastest simulator, making it great for quick testing. The paper compares average simulation times and shows ZSim’s efficiency.

**Validation and Calibration**: Validation and calibration are crucial for accuracy. Sniper and ZSim are validated for Intel Nehalem and Westmere cores, ensuring their results closely match real hardware.

**Accuracy and Error Analysis:** The paper identifies sources of inaccuracies in simulators. For example, PTLsim has issues with floating-point benchmarks, and gem5 and Multi2Sim struggle with branch prediction and cache misses.

**Full-System Simulation Support:** Simulators like gem5 and MARSSx86 support full-system simulations, which is important for studying OS and hardware interactions.

**Relative Performance and Sensitivity Tests:** Sensitivity tests show how changes in microarchitectural parameters affect performance. Sniper and gem5 are more sensitive to these changes.

**Configuration and Flexibility:** Sniper and ZSim are accurate and fast but less flexible for modeling new features compared to gem5 and MARSSx86.

1. **Advantages:**

* Timing simulators provide detailed performance data, such as instructions per cycle (IPC) and memory usage, which are important for optimizing processor designs and understanding how the system behaves in depth. Because of this enhanced performance analysis, architects can make more informed decisions about system improvements using time simulators. For example, by analyzing IPC, architects can identify bottlenecks in the processor pipeline and make informed decisions to enhance system efficiency.
* Various simulation tools and methods are reviewed, focusing on current and up-to-date options. These up-to-date comparisons help researchers select the most suitable simulators for their needs. For example, if a researcher is working on a project that requires detailed timing analysis, they might choose the gem5 timing simulator over a functional simulator. This is particularly useful in saving time and resources by ensuring the right tool is used for the specific task.
* Advanced methods for validating simulators are talked about, ensuring that the simulation results closely match real world performance. To have an effective design and analysis, accurate validation is crucial because it ensures that the results from the simulation are not misleading, which can lead to poor design choices. The simulated performance should reflect actual behaviour, and having important validation techniques ensures reliability in designing and analyzing systems effectively.

**Limitations:**

* Cycle-level simulators give detailed information about how processors work, but because of how much memory and resources it uses, it is slower compared to a functional simulator. This is because they simulate every single processor cycle in detail, which increases accuracy, but also requires significant computing power, and time. As a result, they can be less practical for large or urgent simulations. For example, while they provide the precise data needed for in depth analysis, their slow speed can be a problem when quick results are needed.
* Sampled simulation techniques can speed up simulations, but there are some limitations. One big issue is getting an accurate starting point for the simulation, which can consume a lot of time, especially if points are far part. Alo, these techniques may not work well with the multi threaded applications because tools like Simpoint and SMARTS don’t support them. This can make it difficult to accurately simulate modern multi-core processors.

1. **Application:** When it comes to our final project, deciding which simulations to use to understand our topic can be challenging. This paper was insightful, as it helped me understand the difference between functional and timing simulators. With this knowledge, I can use the paper as a reference to guide my choice of appropriate tools for my final project research. Now, I can identify whether a simulator is functional or timing and determine which simulators are best suited for specific tasks, especially when selecting a topic for the final project.

**Part 2: Evaluating emulsiV**

**Overview**

EmulsiV is a visual simulator designed to help beginners at ESEO learn about computer architecture and digital design. It uses a simple 32-bit RISC processor called Virgule, which only accepts the basic instructions that a C compiler would generate from a standalone C program. This makes it easier for students to understand how processors work. The main goal of EmulsiV is to provide a hands-on learning experience. Students can explore how software and hardware interact and practice low-level programming using assembly and C languages. It’s especially useful in courses focused on computer architecture and digital circuit design. By simulating the Virgule processor, EmulsiV allows students to see how each instruction affects the processor, making complex concepts easier to grasp in an interactive environment. Virgule was chosen because it meets several key requirements: it has an open specification, a simple and regular instruction set, and is supported by a free, open-source toolchain. These features make it an ideal teaching tool that reflects current industry standards. In a computer architecture course, students can learn how a processor works and what kinds of languages and tools can be used to create low-level programs.

**Advantages and Disadvantages**

Advantages:

* *Educational Focus:* Simplifies complex concepts, great for beginners.
* *Open-Source:* Free to use and customize for different needs.
* *Comprehensive Environment:* Simulates the whole system, including peripherals and interrupts.

Disadvantages:

* *Limited Instruction Set:* Only supports a small part of RISC-V, which might limit advanced research.
* *Single Privilege Level:* Only supports machine level, missing user and supervisor modes for complex OS and security studies.

**Comparison with Traditional Tools**

Compared to gem5, a more advanced simulation tool, emulsiV is easier to use and aimed at education. gem5 supports multiple architectures and detailed performance metrics, suitable for deep research but needs more setup and resources. emulsiV is quicker and simpler to set up, making it better for students and beginners.

**Practical Application**

Project Idea: Developing a Simple Educational Operating System

Goals:

* Teach students basic OS concepts like scheduling, memory management, and interrupts.
* Give hands-on experience with system programming.

How emulsiV Helps:

* EmulsiV simulates a whole system and supports interrupts, so students can see how their code interacts with hardware.
* Since it’s open-source, students can change and expand the simulator to test and build OS components, helping them learn by doing.