Introducing MINCER: Performance Measurement & Reproducibility on Chameleon Cloud

By: Jhonny Gonzalez
University of Texas at El Paso

Chameleon Webinar – March 10, 2026

The Reproducibility Challenge in Systems Research

- Reproducing performance results is difficult in modern computing systems
- Hardware, software versions, and configurations vary widely
- GPUs, accelerators, and heterogeneous systems increase complexity
- Slight changes can lead to large performance differences
- This makes results harder to trust, compare, and build upon

The Real Cost of Irreproducibility

- Researchers spend days or weeks re-configuring environments instead of doing science
- Published performance results often cannot be independently verified
- Cross-platform comparisons become unreliable without a common measurement baseline
- HPC and ML workloads are especially vulnerable due to hardware and software diversity
- The community loses trust in results — and progress slows

What is the Central Challenge

- Researchers often struggle to rerun experiments and get comparable results
- Performance measurements depend heavily on:
- Hardware architecture
- Software stack
- Runtime environment
- Without consistent measurement tools, results lack transparency
- This slows progress and makes collaboration harder

If We Standardize Measurement, We Improve Trust

- A consistent performance and monitoring environment can:
  - Improve reproducibility
  - Reduce setup overhead
  - Increase confidence in results
- Researchers should focus on experiments, not environment debugging
- Performance data should be portable and comparable

What Is MINCER?

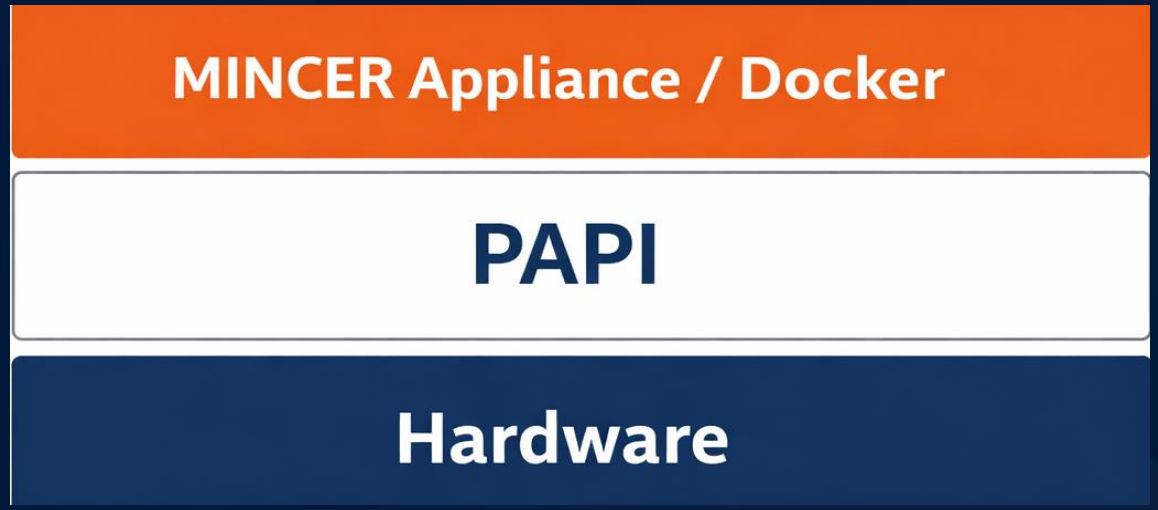
- MINCER is a performance measurement and reproducibility appliance
- Built specifically for Chameleon Cloud
- Designed to:
  - Automate experiment setup
  - Collect consistent performance metrics
  - Support CPUs, AMD GPUs, and NVIDIA GPUs
- Enables reproducible performance studies across architectures

MINCER Architecture Overview

Layer 1 – Hardware Layer
- Intel/AMD CPUs
- NVIDIA GPUs (CUDA)
- AMD GPUs (ROCm/HIP)

Layer 2 – PAPI Measurement Layer
- perf_event → Linux CPU counters
- CUDA component → NVIDIA GPU metrics
- ROCm / ROCm-SMI → AMD GPU metrics
- RAPL → Intel power and energy

Layer 3 – MINCER Appliance Layer
- Architecture-specific Docker containers
- Python experiment runner (run_experiment.py)
- Automatic logging with timestamps and system metadata



How MINCER Works

- Uses Docker containers to standardize environments
- Integrates PAPI for performance and energy measurement
- Automatically records:
  - System metadata (hardware, OS, GPU type)
  - Performance counters
  - Power and energy metrics
- Same scripts run across different architectures

The MINCER Workflow

- **Provision** a Chameleon bare-metal node — CPU, NVIDIA GPU, or AMD GPU
- **Download** the MINCER appliance from GitHub
- **Run** the architecture-specific setup script
- setup_cpu.sh / setup_cuda.sh / setup_amd.sh
- **Build** the appropriate Docker image for your target architecture
- **Execute** your experiment using run_experiment.py
- **Retrieve** timestamped logs containing system metadata and PAPI metrics

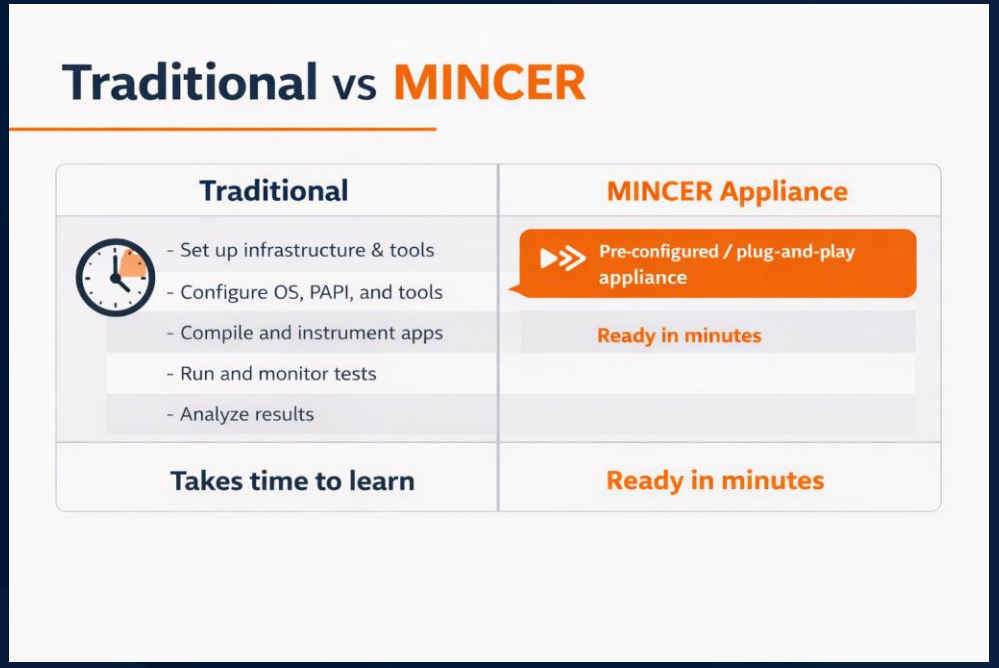
What is the Performance Application Programming Interface (PAPI)

- Cross-platform interface for accessing hardware performance counters.
- Provides real time visibility into hardware behavior:
  - Cache Misses
  - Floating point operations (FLOPS)
- Component Architecture
  - Modular Design that gathers metrics from specific hardware or software layers
  - Examples for MINCER
    - perf_event (Linux CPUs)
    - cuda (NVIDIA GPUs)
    - rocm (AMD GPUs)
    - rapl (Intel power/Energy)

Traditional vs MINCER

- PAPI Traditional Setup:
  - Provision a bare-metal node
  - Install OS and dependencies
  - Download and build PAPI from source
    - Match compiler versions, linking CUDA/ROCm libraries
  - Configure system environment variables and kernel permissions to allow hardware counter access

- MINCER:
  - Pull the MINCER container (Docker)
  - PAPI environment is pre-configured and ready
  - Directly compile and run the experiment



Why PAPI?

- PAPI (Performance Application Programming Interface) provides:
  - Low-level access to hardware performance counters
  - CPU and GPU performance metrics
  - Power and energy measurements
- MINCER uses PAPI to ensure:
  - Consistent metric collection
  - Comparable data across runs and platforms

Real Workloads, Real Insights

MINCER includes example proxy applications:

- **XSBench** – memory and cache behavior
- **LULESH** – computational physics workload
- **Gunrock** – GPU graph analytics
- **3D-UNET** – machine learning workload

These demonstrate reproducibility on:

- CPUs
- AMD GPUs
- NVIDIA GPUs

What MINCER Measures Across Workloads

CPU Metrics

- Total cycles and instructions
- Instructions per cycle (IPC)
- L1 and L2 data cache misses
- Cache miss rates (derived)
- Package and DRAM energy (RAPL)
- Average power consumption (W)

NVIDIA GPU Metrics

- FP64 instructions executed (CUDA)
- DRAM bytes transferred
- L1 texture cache bytes
- L2 cache bytes transferred
- GPU utilization % (NVML)
- Power consumption and temperature

AMD GPU Metrics

- GPU busy percentage (ROCm)
- VALU and SALU instruction counts
- L2 cache hit rate
- Memory fetch and write size
- Power average (ROCm-SMI)
- GPU temperature

Why we Built this on Chameleon

Chameleon provides features critical for systems research:

- Bare-metal access
- Full hardware control
- Support for diverse architectures

Enables accurate, low-noise performance measurements

Ideal platform for reproducibility-focused research

Building and Running MINCER on Chameleon

- Use Chameleon bare-metal instances
- Command-line workflow:
  - Download MINCER appliance
  - Build Docker images per architecture
  - Run experiments using Python scripts
- Results are automatically logged with:
  - Timestamps
  - System metadata
  - PAPI metrics

Understanding MINCER Output

- Each run produces a timestamped folder underexperiment_logs/
- Two files per run:
  - **output.txt** — raw experiment output including PAPI metrics
  - **execution_log_[timestamp].txt** — system metadata + full experimentoutput
- System metadata captured automatically:
  - OS, CPU architecture, RAM
  - GPU type, memory, and driver version
- Energy measurements reported in nanojoules and convertedto joules
- Performance counters include derived metrics: IPC, cachemiss rate, IPS
- **Example Output Highlights (from XSBench CPU run):**
- Runtime: 1.664 seconds
- Total Energy: 968.882 J
- Average Total Power: 582.222 W
- IPC: 0.161 instructions/cycle
- IPS: 510.630 MIPS

Live Demonstration: MINCER on Chameleon

Understanding MINCER Output

- Each run produces a timestamped folder under experiment_logs/
- Two files per run:
  - **output.txt** — raw experiment output including PAPI metrics
  - **execution_log_[timestamp].txt** — system metadata + full experiment output
- System metadata captured automatically:
  - OS, CPU architecture, RAM
  - GPU type, memory, and driver version
- Energy measurements reported in nanojoules and converted to joules
- Performance counters include derived metrics: IPC, cache miss rate, IPS
- **Example Output Highlights (from XSBench CPU run):**
- Runtime: 1.664 seconds
- Total Energy: 968.882 J
- Average Total Power: 582.222 W
- IPC: 0.161 instructions/cycle
- IPS: 510.630 MIPS

```
61  ================================================================
62                                SIMULATION
63  ================================================================
64  Energy + Performance monitoring initialized successfully
65  Energy + Performance measurement started
66  Beginning history based simulation...
67
68  Simulation complete.
69  Energy + Performance measurement stopped
70
71  ================================================================
72                        ENERGY & PERFORMANCE RESULTS
73  ================================================================
74  Runtime: 1.664 seconds
75  ----------------------------------------------------------------
76  ENERGY MEASUREMENTS:
77  Package Energy: 911689646080 nJ (911.690 J)
78  DRAM Energy:    57192677376 nJ (57.193 J)
79  Total Energy:   968.882 J
80  Average Package Power: 547.853 W
81  Average DRAM Power:    34.368 W
82  Average Total Power:   582.222 W
83  ----------------------------------------------------------------
84  PERFORMANCE COUNTERS:
85  Total Cycles:       5263371033
86  Total Instructions: 849745476
87  Frequency:          3.163 GHz
88  IPS:                510.630 MIPS
89  IPC:                0.161 instructions/cycle
90  ================================================================
```

Cross-Architecture Comparison: XSBench

Metric	CPU (Xeon Gold 6240R)	NVIDIA (Quadro RTX 6000)	AMD (MI100)
Runtime	1.664 s	0.327 s	0.18 s
IPC / Throughput	0.161 IPC	51.9M lookups/s	20.7M lookups/s
Cache / Memory	L1 miss rate tracked	12.97 GB DRAM xfer	L2 hit rate: 28%
Power	582 W avg total	144.5 W GPU	46 W GPU
Energy	968.9 J total	-	-

Key Takeaway: Same experiment, same scripts, different hardware → comparable structured output

What we Use Most

- Hardware Catalog:
  - Identify AMD vs NVIDIA GPU nodes
- Bare-metal reconfiguration:
  - Consistent hardware across runs
- OS-level access:
  - Custom PAPI, CUDA, and ROCm versions
- Documentation and host availability tools

What Would Be Hard Without Chameleon

- Performance studies require:
  - OS control
  - Stable hardware
  - Precise measurement
- Without Chameleon:
  - Experiments would be harder to reproduce
  - Less control over software stacks
  - Fewer architecture options
- Chameleon enables experiments that would otherwise be impractical

Sharing and Reproducibility

- MINCER code and scripts are hosted on GitHub
- Repository includes:
  - Setup scripts
  - Measurement examples
  - PAPI data collection tools
- Artifacts are being refined to make reuse easier for other users
- Goal: lower the barrier for reproducible performance research

Who Benefits from MINCER?

- Researchers:
  - Easier experiment setup
  - More trustworthy results
- Students:
  - Learn performance analysis hands-on
- HPC community:
  - More standardized and comparable studies
- Chameleon users:
  - Faster onboarding for systems experiments

What We Want You to Remember

- Reproducibility is a major challenge in systems research
- MINCER provides a standardized, portable measurement solution
- Chameleon Cloud enables this work through bare-metal access
- Performance visibility leads to better science

Questions?
THANK YOU

jgonzalez183@miners.utep.edu