Cnuas Calibrated Virtual-time Model, Datasheet

Deterministic cross-component virtual time, physical-target calibration and held-out error reporting

Document DS-CNU-032 • **Revision** B • **Issued** 22 August 2026 • **Status** Published

Cnuas Calibrated Virtual-time Model, Datasheet

Item	Value
Part	`cnuas-timing`
Type	Deterministic timing and performance model
Package	`cnuas-timing`, version 0.1.0
Version	`4b9cc6a-dirty`
Repo	PacketFive/cnuas

1. Overview

`cnuas-timing` composes analytic timing costs for Cnuas processors, PCIe, network interfaces, switches, accelerators, accelerator links, management buses and facility control loops. It uses integer-picosecond event scheduling, models queues and parallel lanes, and can fit component predictions to a named physical target.

The supplied profile is uncalibrated. It produces deterministic virtual-time estimates, not physical performance claims.

Key features

- QEMU `icount` mapping for deterministic guest instruction time.
- Instructions, cycles, bytes, packets and operations in one workload type.
- Fixed latency, resource rates, serial or overlapping costs, and parallel lanes per component.
- Stable event ordering, queue time, service time and end-to-end traces.
- Affine physical-target calibration with retained target identity.
- Held-out MAE, RMSE, MAPE and maximum absolute error.
- Strict JSON profiles and pipelines.
- Machine-readable command output.

2. Module map

Module	Responsibility
`model.py`	Work, component profiles, pipeline stages, event scheduler and result metrics
`calibration.py`	Observation schema, least-squares fitting and held-out errors
`profiles.py`	Strict deterministic JSON profile serialization
`qemu.py`	Instruction-count shift mapping and deterministic TCG arguments
`cli.py`	Prediction, pipeline simulation, fitting, validation and QEMU commands

3. Timing resources

Resource	Profile rate	Service-time term
Guest instructions	MIPS	instructions divided by instruction rate
Component cycles	MHz	cycles divided by clock
Transfer volume	Gbit/s	eight times bytes divided by bandwidth
Packet work	Mpacket/s	packets divided by packet rate
Compute work	TOPS	operations divided by operation rate

Overlapping components use the largest active resource term. Serial components sum them. Fixed latency is added before a positive scale and finite offset from calibration are applied.

4. Scheduling

Property	Specification
Time base	Integer picoseconds
Resource sharing	Per-component lane availability
Lane selection	Earliest available lane, then lowest lane number
Queueing	Start time minus stage arrival time
Pipeline	Ordered component stages per job
Arrivals	Simultaneous or fixed interval
Output	Mean and nearest-rank p95 latency, throughput, optional event trace
Replay	Equal profile, pipeline and arrivals produce an equal trace

5. Calibration

Item	Specification
First model	`physical = max(0, scale * analytic + offset)`
Fit	Ordinary least squares for two or more distinct predictions
Single point	Scale through the origin, preliminary use only
Required identity	Component and named physical target
Rejected fit	Non-positive scale
Held-out metrics	MAE, RMSE, MAPE, maximum absolute error

A physical result must additionally record hardware and firmware revisions, clock and link configuration, software versions, workload shape, queue and flow counts, measurement method, and fitting versus held-out partitions.

6. Reference component coverage

Profile key	Domain
`guest-cpu-icount-shift0`	QEMU guest processor
`pcie-gen5-x16`	Host-to-device transport

Profile key	Domain
`cnuasnic`	RDMA adapter packet and byte work
`cnuasswitch`	Fabric-hop packet and byte work
`cnuasgpu-compute`, `cnuasgpu-memory`	Accelerator arithmetic and memory
`cnuaslink`	Accelerator peer fabric
`bmc-rs485`	ORV3 Modbus RTU segment
`facility-control`	Telemetry and site power control cadence

7. Command reference

Command	Purpose
`predict`	Compute service time for one component and work item
`simulate`	Run repeated jobs through a JSON pipeline
`calibrate`	Fit a target correction from training observations
`validate`	Report errors on held-out observations
`qemu-args`	Print deterministic TCG and `icount` arguments

8. Validation

Suite	Cases	Result
Resource arithmetic, profile validation, scheduling and replay	26	Pass
Calibration and held-out errors	9	Pass
Profile I/O, QEMU mapping and CLI	11	Pass
Total	**46**	**Pass**

These cases validate software behaviour. Physical prediction accuracy remains unmeasured until target datasets are collected.

9. Integration

Item	Value
Source	`timing/src/cnuas_timing/`
Reference profile	`timing/profiles/cnuas-analytic-v0.json`
Example pipeline	`timing/examples/rdma-gpu-pipeline.json`
Python	3.10 or newer
Runtime dependencies	None
Test dependency	pytest 8 or newer
Licence	Apache License 2.0

Detailed architecture and the calibration protocol are in the timing model design.

10. Documentation and publication

Resource	Scope
Documentation overview	Entry point and implementation state
Concepts and evidence	Need, reasoning, evidence classes and design choices
Quick start	Installation and first experiment
Architecture and algorithm	Equations, event scheduler and calibration protocol
Profiles and pipelines	JSON schema, work dimensions, modes and lanes
Calibration guide	Target identity, fitting and held-out validation
CLI and Python reference	Commands, outputs, API and error behaviour
Worked examples	NIC, GPU, RS-485, queueing and experiment bundles
Limits and interpretation	Claim boundary and planned work
Dedicated paper	`publications/cnuas-timing/Cnuas-Timing.pdf` in the academic-research repository

11. Revision history

Revision	Date	Change
B	2026-08-22	Added the dedicated documentation section, full user workflow, command and API reference, worked examples, and expanded paper
A	2026-08-21	Initial deterministic scheduler, analytic profile, QEMU mapping, calibration, held-out errors and CLI

PacketFive, Packet Five Networks Ltd., Dublin, Ireland.