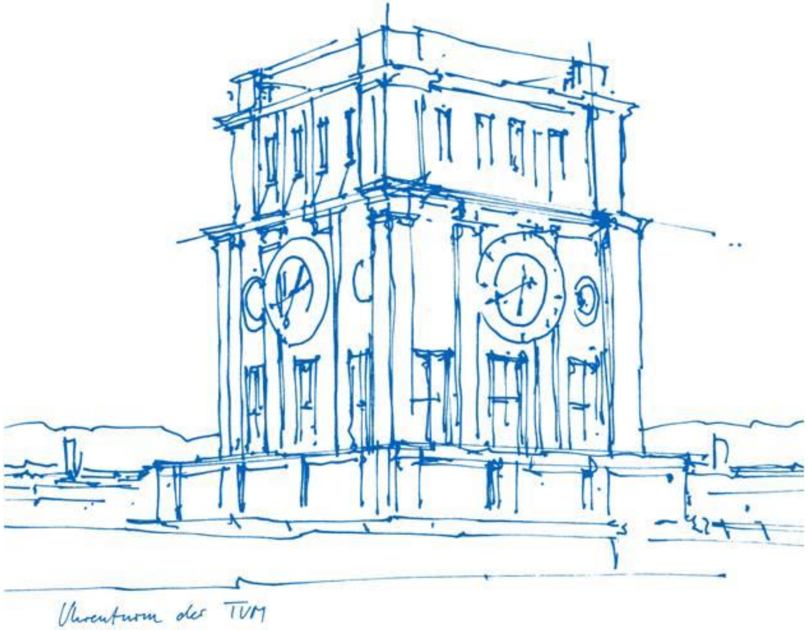
Linux Power Management –
Challenges in Modern Servers and how SmartNICs can help

Marco Liess, Andreas Herkersdorf

Technical University of Munich

Chair of Integrated Systems

Berlin, November 13th 2025



About Me

Marco Liess (marco.liess@tum.de)

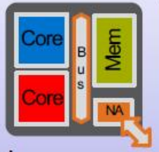
PhD Candidate @ Chair of Integrated Systems

with Prof. Andreas Herkersdorf

- M. Sc. in Electrical and Computer Engineering @ TUM
- Research on energy-efficient and adaptive SmartNICs (“intelligent network interfaces”) for 6G RAN/Edge Nodes
- Teaching assistant for Chip Multicore Processors (EI7271)

Chair of Integrated Systems @ TUM

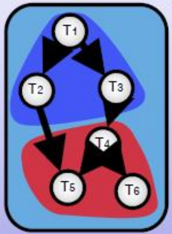
Architectures



- MPSoC platforms
- Hardware accelerators
- 2D/3D NoC interconnect
- Memory hierarchies

Methods

- Cross-layer resilience
- Bio-inspired reinforcement ML
- Power management
- Diagnosis on Chip



Application Areas

- Networking
- Automotive
- Visual Computing
- Near Memory Acceleration

Prototyping Lab

- FPGA-based emulation
- Network testing
- Spectrum Analyzer
- x86-based Manycore

Importance of Energy-efficiency

Energy Production and Compute Consumption

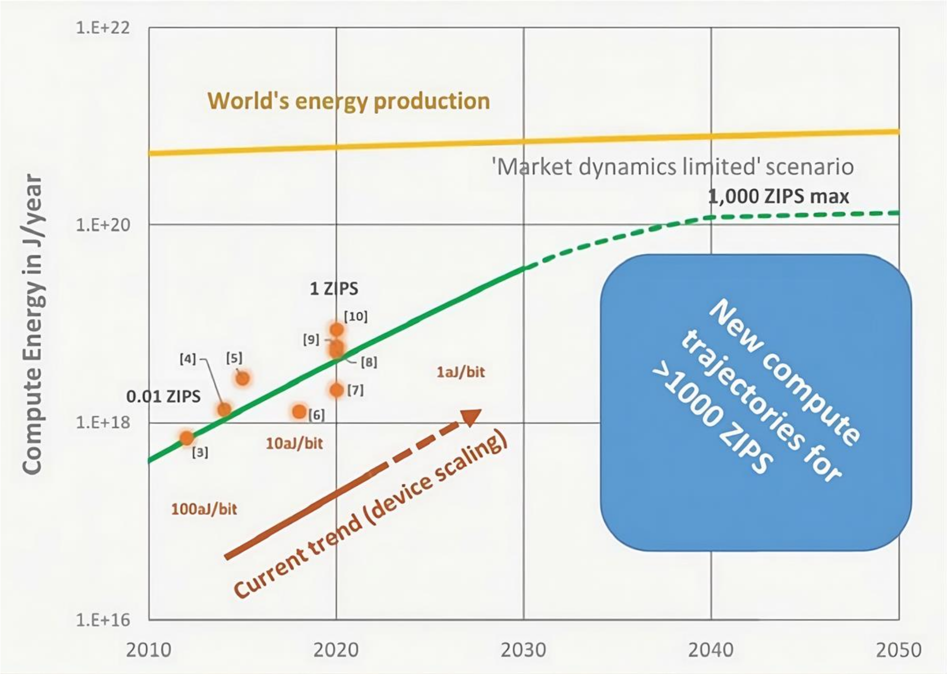
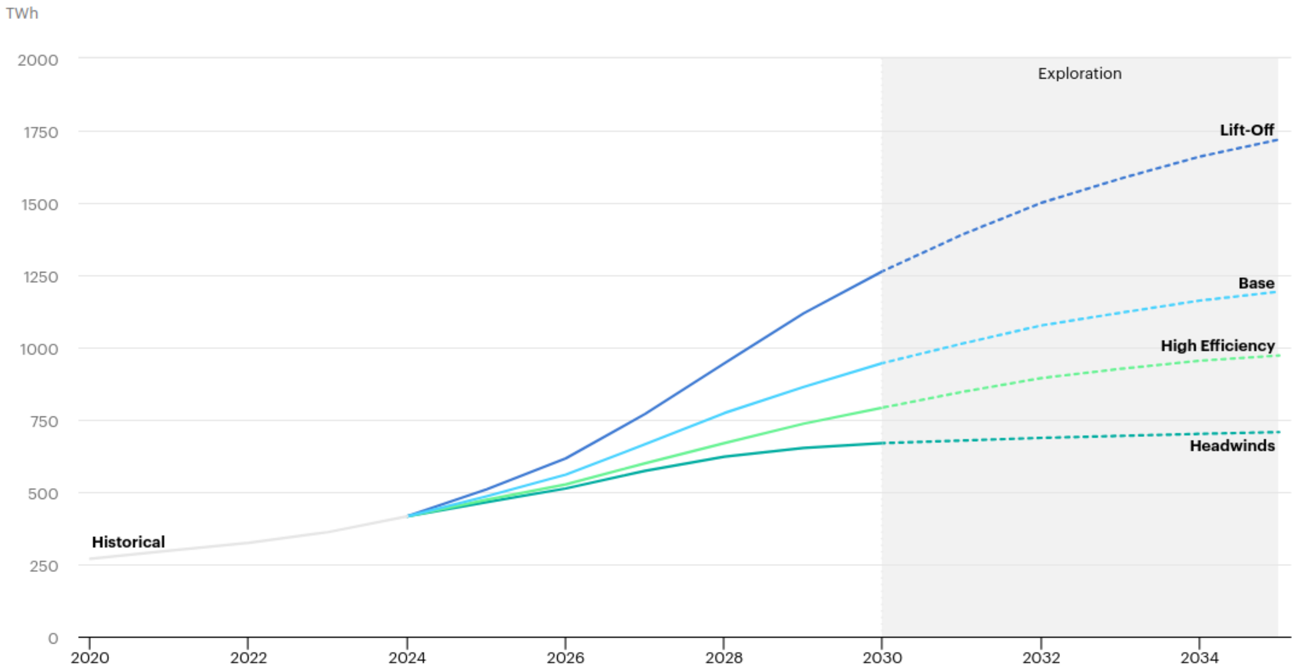


Figure: Worldwide Energy Production and Compute Consumption [1]

[1] Semiconductor Research Corporation. *The Decadal Plan for Semiconductors 2021*. 2021

Importance of Energy-efficiency

Global Data Center Electricity Consumption [1]



[1] IEA (2025), *Energy and AI*, IEA, Paris https://www.iea.org/reports/energy-and-ai, Licence: CC BY 4.0

Importance of Energy-efficiency

How can it be improved (from a Hardware perspective)?

Importance of Energy-efficiency

How can it be improved (from a Hardware perspective)?

Specialized Architectures
(e.g. Tensor Processing Unit) → *High Ops/Watt*

Importance of Energy-efficiency

How can it be improved (from a Hardware perspective)?

Specialized Architectures
(e.g. Tensor Processing Unit) → *High Ops/Watt*

Heterogeneous Architectures
(„*Choosing the right tool for the right job*“)

Importance of Energy-efficiency

How can it be improved (from a Hardware perspective)?

Specialized Architectures
(e.g. Tensor Processing Unit) → *High Ops/Watt*

Heterogeneous Architectures
(„*Choosing the right tool for the right job*")

Intelligent Resource Management
(Power management, scheduling, task distribution, etc…)

Overview

1. CPU Power Management – Linux CPUFreq
2. Analytic Considerations of Asynchronous Traffic-driven DVFS
3. EcoNIC: SmartNIC-assisted Linux Power Management
4. Conclusion & Outlook

CPU Runtime Power Management

Between Running, Walking, Dozing and Sleeping

Runtime Configuration Options:

- Frequency Adjustments
  - Main driver is dynamic power consumption on clock ticks
  - Less clock ticks → less power
  - Further enables voltage reduction → nonlinear savings

- „Sleeping“
  - „Clock gating“ → disable parts of the CPU
  - “Turn off“ supply voltage

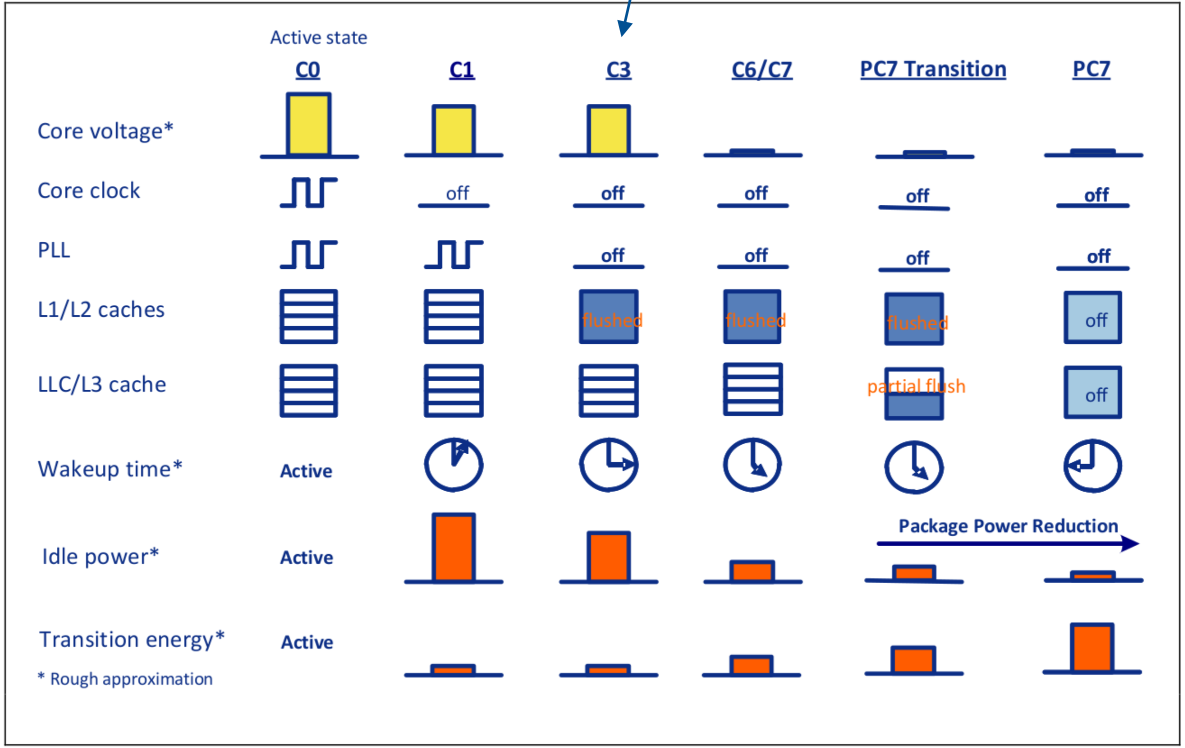
CPU Runtime Power Management

Sleep states – C-States

→ CPU Architecture supports certain C-States

→ C-State activation / selection managed by OS

e.g. 350 µs on Ryzen 5 5600G



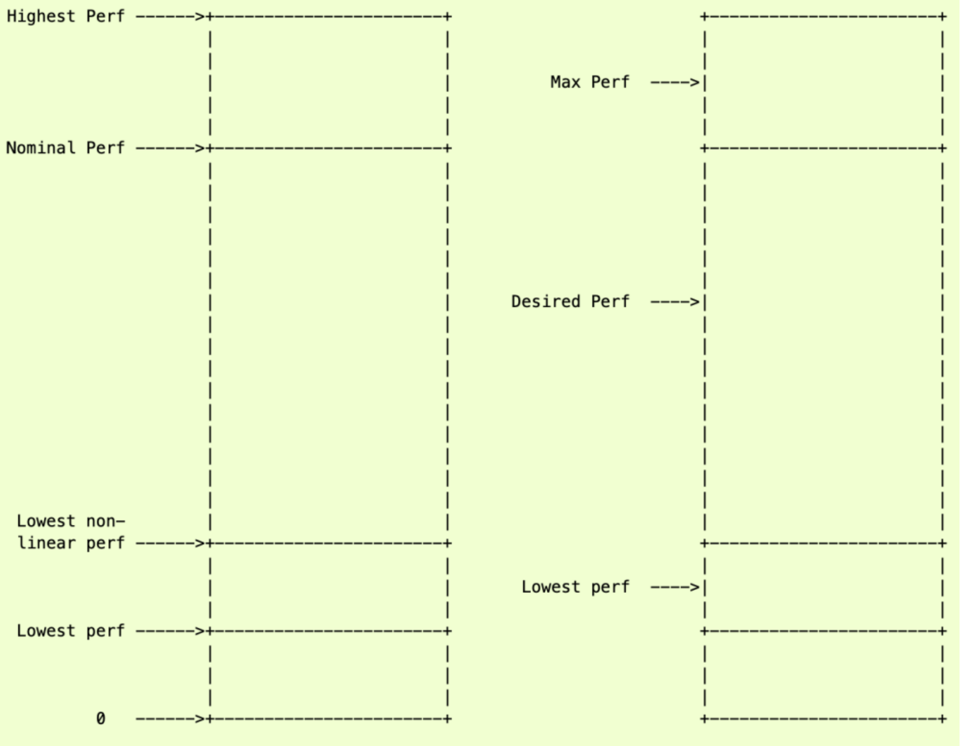
https://www.thomas-krenn.com/en/wiki/Processor_P-states_and_C-states

CPU Runtime Power Management

Dynamic Voltage Frequency Scaling (DVFS) – P-States

→ P-State availability and characteristics (transition latency, frequencies, etc.) depend on CPU architecture

→ P-State management can be performed by OS or CPU itself (e.g. Intel SpeedShift, AMD CPPC)



AMD P-States Performance Scale

https://www.kernel.org/doc/html/v6.0/admin-guide/pm/amd-pstate.html

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

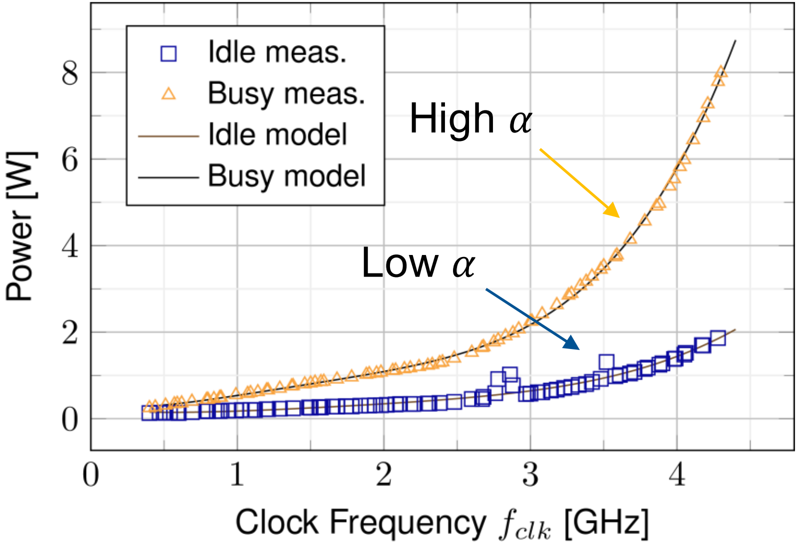


Figure: Power consumption of one core of a Ryzen 5 5600G 6-core CPU

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

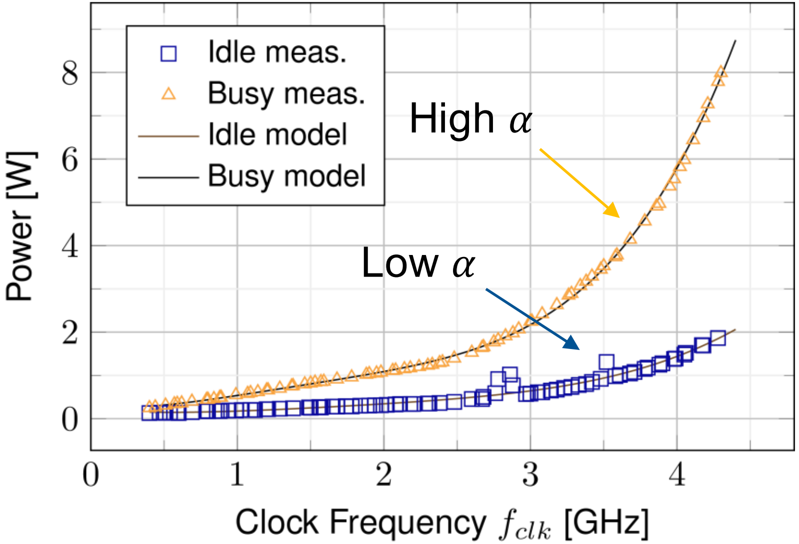


Figure: Power consumption of one core of a Ryzen 5 5600G 6-core CPU

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

3) Required processing capacity

$$MIPS_{req}$$

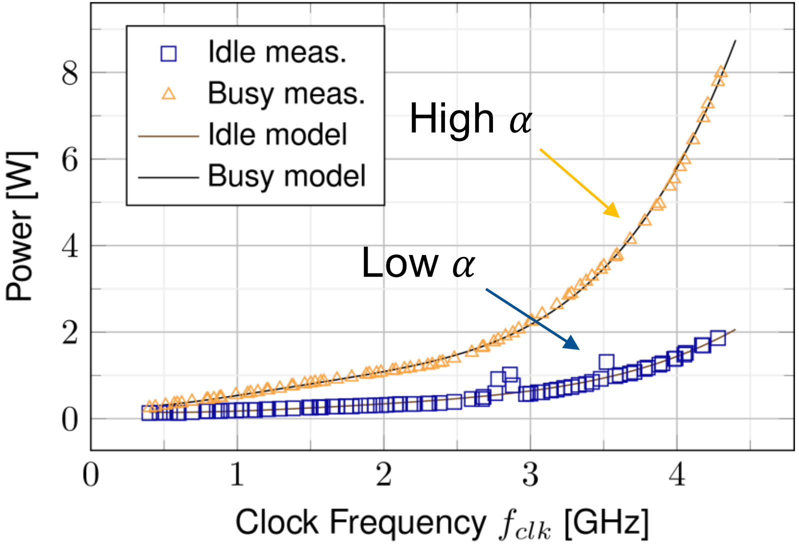


Figure: Power consumption of one core of a Ryzen 5 5600G 6-core CPU

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

3) Required processing capacity

$$MIPS_{req}$$

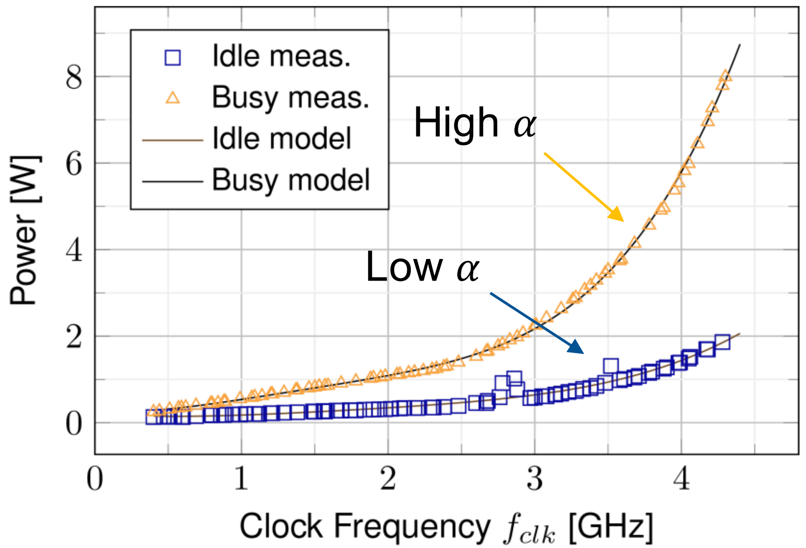


Figure: Power consumption of one core of a Ryzen 5 5600G 6-core CPU

➡ Use f_{clk} to match supply to demand!

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

3) Required processing capacity

$$MIPS_{req} = ?$$

Depends on application and can be hard to predict!

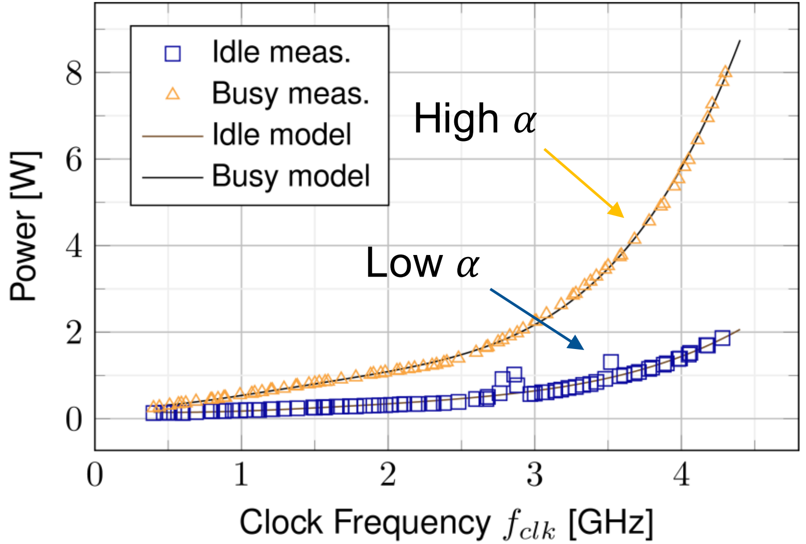


Figure: Power consumption of one core of a Ryzen 5 5600G 6-core CPU

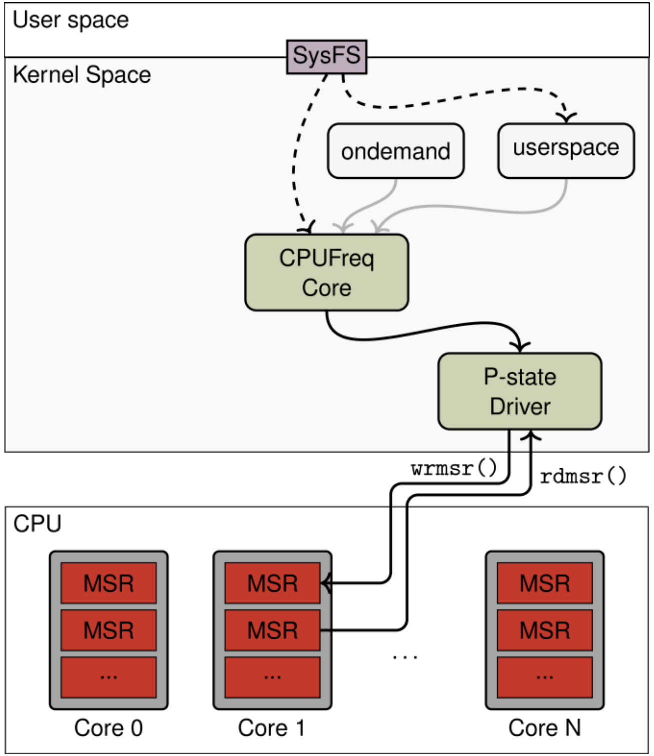
➡ Use f_{clk} to match supply to demand!

DVFS – P-State Management

Linux CPUFreq Subsystem

Basics:

- CPUFreq Core is abstraction layer
- Driver implements CPU Hardware features
- Governor makes decisions



DVFS – P-State Management

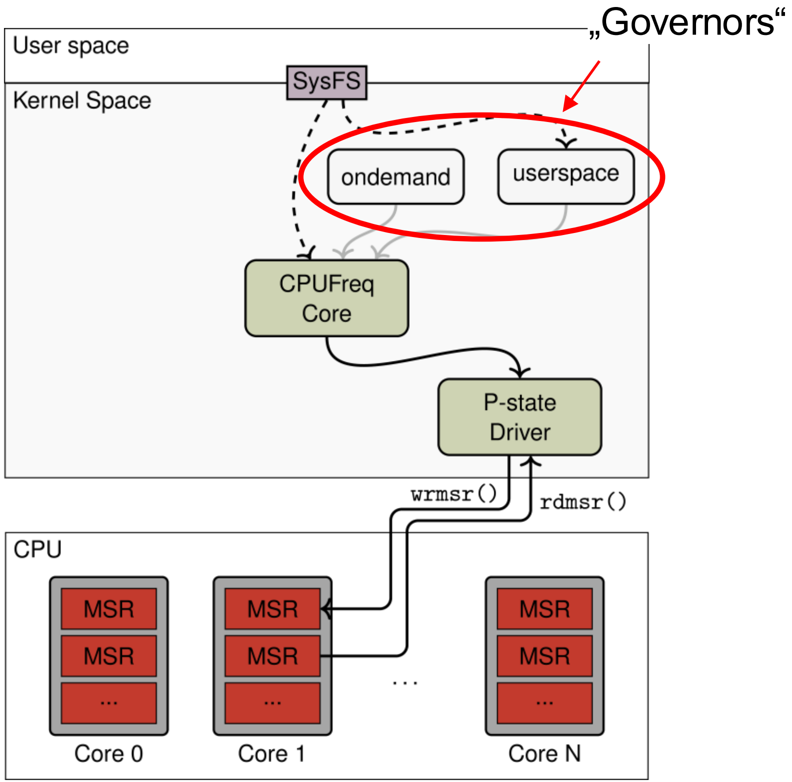
Linux CPUFreq Subsystem

Basics:

- CPUFreq Core is abstraction layer
- Driver implements CPU Hardware features
- Governor makes decisions

Popular Linux Governors:

• performance	always highest frequency
• powersave	always lowest frequency
• ondemand	based on current system load
• schedutil	based on scheduler‘s PELT



DVFS – P-State Management

Linux CPUFreq Subsystem

```
root@ryzen01 / # ll /sys/devices/system/cpu/cpufreq/policy0
total 0
drwxr-xr-x  2 root root    0 Nov 11 17:57 ./
drwxr-xr-x 15 root root    0 Nov 11 17:57 ../
-r--r--r--  1 root root 4096 Nov 11 17:57 affected_cpus
-r--r--r--  1 root root 4096 Nov 11 17:57 amd_pstate_highest_perf
-r--r--r--  1 root root 4096 Nov 11 17:57 amd_pstate_lowest_nonlinear_freq
-r--r--r--  1 root root 4096 Nov 11 17:57 amd_pstate_max_freq
-rw-r--r--  1 root root 4096 Nov 11 17:57 boost
-r--r--r--  1 root root 4096 Nov 11 17:57 cpuinfo_max_freq
-r--r--r--  1 root root 4096 Nov 11 17:57 cpuinfo_min_freq
-r--r--r--  1 root root 4096 Nov 11 17:57 cpuinfo_transition_latency
-r--r--r--  1 root root 4096 Nov 11 17:57 related_cpus
-r--r--r--  1 root root 4096 Nov 11 17:57 scaling_available_governors
-r--r--r--  1 root root 4096 Nov 11 17:57 scaling_cur_freq
-r--r--r--  1 root root 4096 Nov 11 17:57 scaling_driver
-rw-r--r--  1 root root 4096 Nov 11 17:57 scaling_governor
-rw-r--r--  1 root root 4096 Nov 11 17:57 scaling_max_freq
-rw-r--r--  1 root root 4096 Nov 11 17:57 scaling_min_freq
-rw-r--r--  1 root root 4096 Nov 11 17:57 scaling_setspeed
```

DVFS – P-State Management

Linux CPUFreq Subsystem

ondemand

```
(.menv) root@ryzen01 ~ # ll /sys/devices/system/cpu/cpufreq/ondemand/
total 0
drwxr-xr-x  2 root root    0 Nov 12 16:32 ./
drwxr-xr-x 16 root root    0 Nov 11 17:57 ../
-rw-r--r--  1 root root 4096 Nov 12 16:32 ignore_nice_load
-rw-r--r--  1 root root 4096 Nov 12 16:32 io_is_busy
-rw-r--r--  1 root root 4096 Nov 12 16:32 powersave_bias
-rw-r--r--  1 root root 4096 Nov 12 16:32 sampling_down_factor
-rw-r--r--  1 root root 4096 Nov 12 16:32 sampling_rate
-rw-r--r--  1 root root 4096 Nov 12 16:32 up_threshold
(.menv) root@ryzen01 ~ #
```

- Takes current CPU load information from scheduler
- "Short-term" history has benefits and problems
- Typically executed every 10ms

schedutil

ewma_sum(u) := u_0 + u_1*y + u_2*y^2 + ...

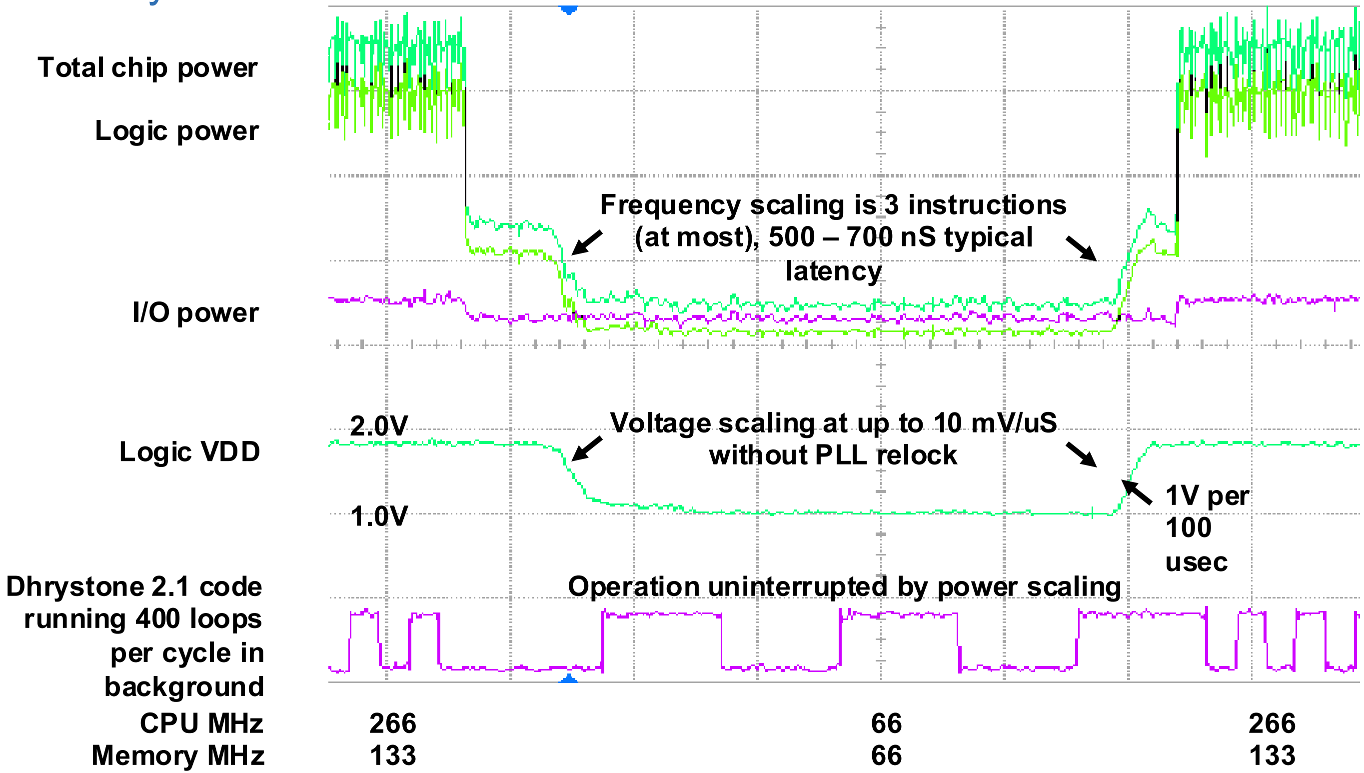
ewma(u) = ewma_sum(u) / ewma_sum(1)

```
           max( running, util_est );  if UTIL_EST
u_cfs := { running;                   otherwise

             clamp( u_cfs + u_rt , u_min, u_max );    if UCLAMP_TASK
u_clamp := { u_cfs + u_rt;                            otherwise

u := u_clamp + u_irq + u_dl;          [approx. see source for more detail]

f_des := min( f_max, 1.25 u * f_max )
```

- Exponential Weighted Moving Average (EWMA)
  > 50% last 32ms, 50% rest of history
- "Long-term" history less flexible
- Executed everytime with scheduler

DVFS – P-State Management

Transition Latency



Source: IBM PowerPC 405 LP NetSeminar

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

3) Required processing capacity

$$MIPS_{req} = ?$$

Depends on application and can be hard to predict!

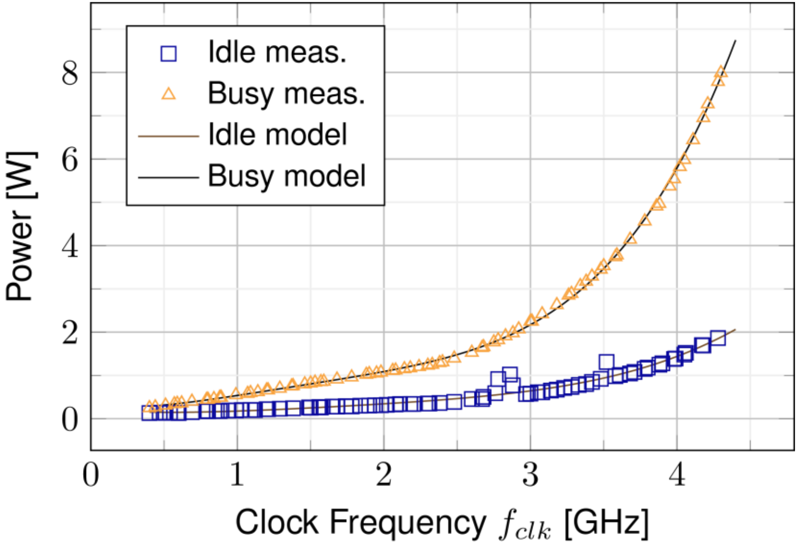
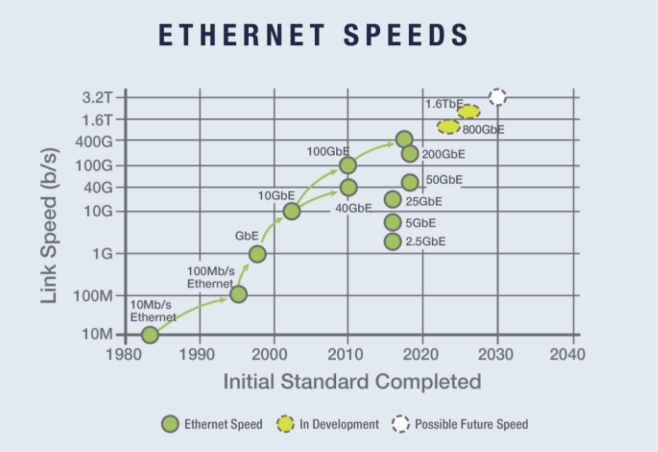


Figure: Power consumption of one core of a Ryzen 5 5600G 6-core CPU

Use f_{clk} to match supply to demand!

Processing Demand in High-Speed Networking

The required processing capacity of NIC depends on:



Source: https://iebmedia.com/wp-content/uploads/2023/05/Ethernet-Speeds.png

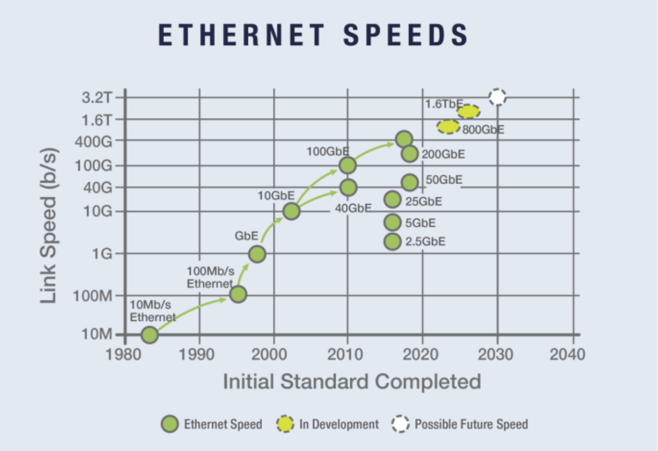
Processing Demand in High-Speed Networking

The required processing capacity of NIC depends on:

Packet rate / Packet interarrival time

→ *packets per second (PPS)* $PPS = \frac{link\_speed}{packet\_size}$

Link speed	40 Gbps		100 Gbps		800 Gbps	
Size [Byte]	64	512	64	512	64	512
PPS [M]	78	9.8	195	24	1,563	195
1 / PPS [ns]	12.8	102	5.1	41	0.64	5.1

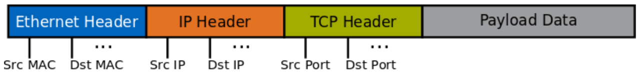


Source: https://iebmedia.com/wp-content/uploads/2023/05/Ethernet-Speeds.png

$Date\_rate = w_{Data} \times f$

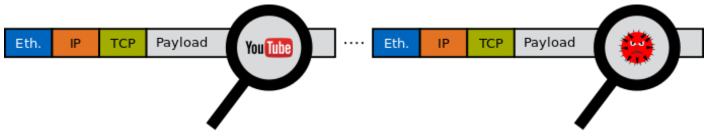
Processing Demand in High-Speed Networking

Header Processing

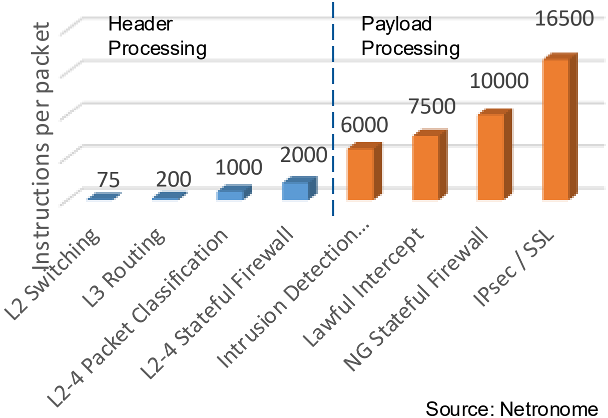


- Fixed header location allows simple parsing
- **Common use-cases:** routing, switching, IP/port-based firewalls, …

Payload Processing



- **Common use-cases:** application/session/user identification for firewalls or bandwidth throttling, cryptology, intrusion detection, virus scanning, …



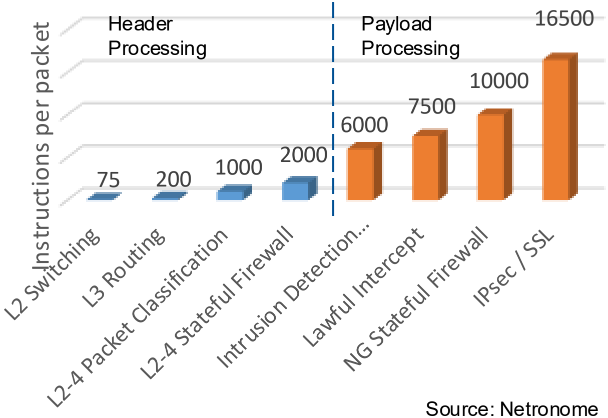
Source: Netronome

Processing Demand in High-Speed Networking

The processing complexity of a networking function:

→ *instructions per packet (IPP)*

Required processing capacity $\mathrm{IPS} = PPS * IPP\ [\frac{instructions}{second}]$



Source: Netronome

@ 100Gbps

L2 Switching (64B): $PPS\ * IPP = 195.313.000 \frac{packets}{second} * 75 \frac{instructions}{packet} = 14.6 * 10^9 \frac{instructions}{second}\ (GIPS)$

Intrusion Detection (512B): $146 * 10^9 \frac{instructions}{second}$ IPSec (512B) : $403\ * 10^9 \frac{instructions}{second}\ (GIPS)$

Processing Demand in High-Speed Networking

AMD Ryzen 5 5600: 6 cores @ 3.9 GHz, 65 W TDP

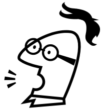
$$IPS = 6\,cores * f_{clk} * IPC_{core} \approx 23{,}4 * 10^9 \frac{instructions}{second}$$

$$IPC_{core} \approx 1$$

Networking Function	100Gbps
L2 Switching	$14.6 * 10^9$
Intrusion Detection	$146 * 10^9$
IPSec	$403 * 10^9$

L2 Switching ✓ Intrusion Detection ✗ IPSec ✗

IPSec @ 100Gbps would require 18 CPUs with a TDP of 1.1 kW!



DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

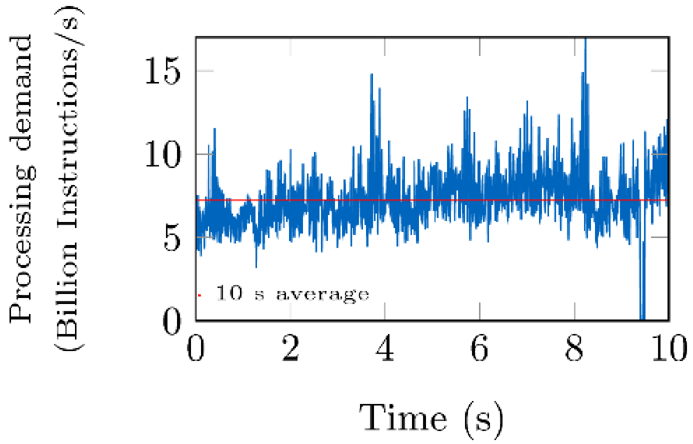
$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

3) Required processing capacity:

$$MIPS_{req} = MPPS \cdot IPP$$

$MPPS$: million packets per second, IPP: instructions per packet



Oeldemann, A., Wild, T., & Herkersdorf, A. (2017, March). Reducing data center resource over-provisioning through dynamic load management for virtualized network functions. In *International Conference on Architecture of Computing Systems* (pp. 234-247). Cham: Springer International Publishing.

→ Use f_{clk} to match supply to demand!

DVFS – P-State Management

Power consumption of digital circuits

1) Dynamic power consumption:

$$P_{dyn} = \alpha C \cdot f_{clk} \cdot V_{dd}^2$$

α: activity, C: capacity,
f_{clk}: clock frequency, V_{dd}: supply voltage

2) Available processing capacity:

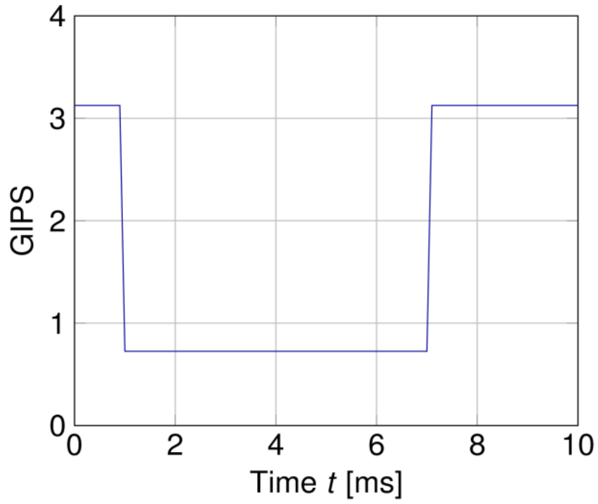
$$MIPS_{prov} = N \cdot f_{clk} \cdot IPC$$

N: # of cores, f_{clk}: clock frequency, IPC: instructions per cycle

3) Required processing capacity:

$$MIPS_{req} = MPPS \cdot IPP$$

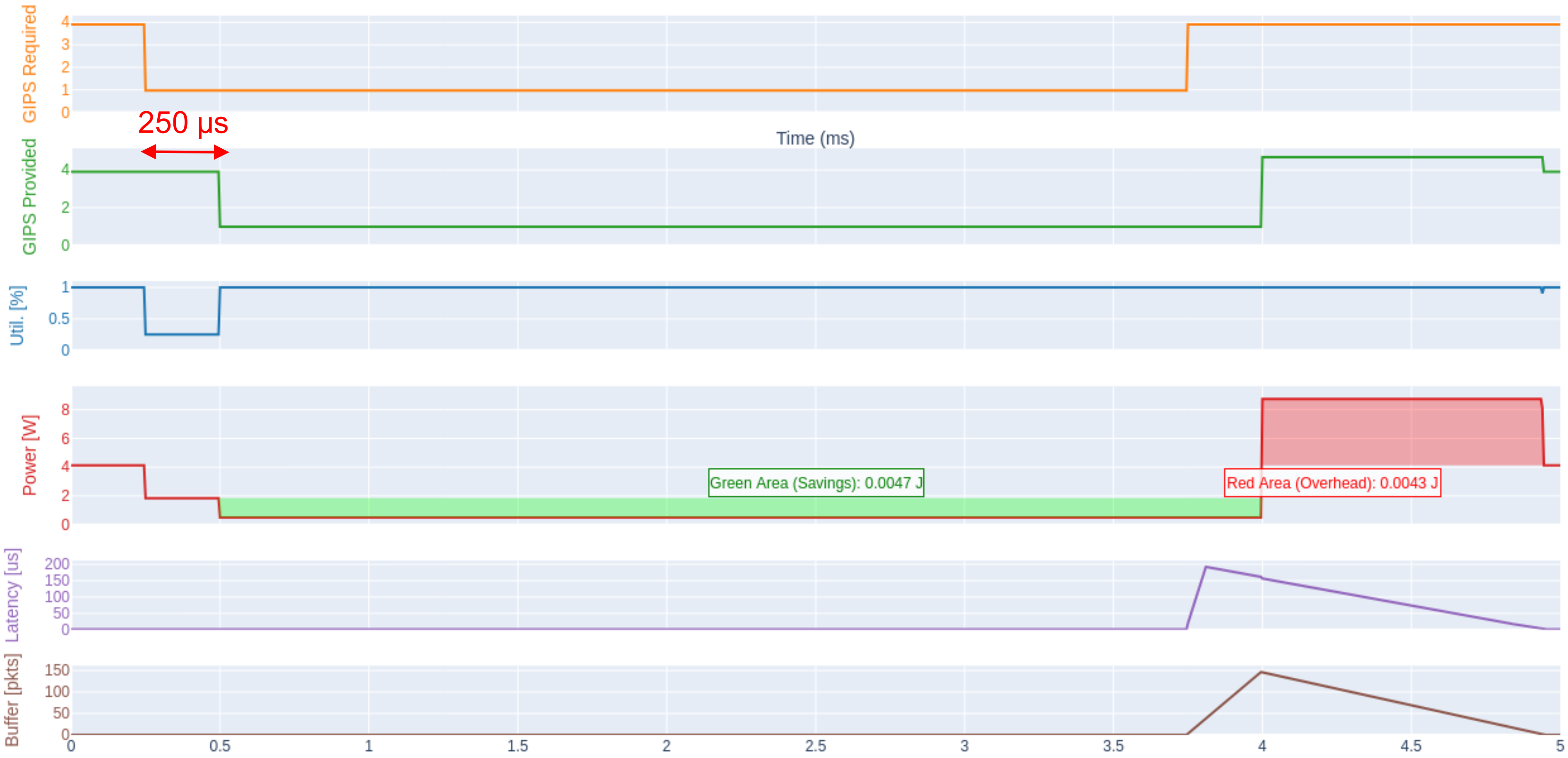
$MPPS$: million packets per second, IPP: instructions per packet



→ Use f_{clk} to match supply to demand!

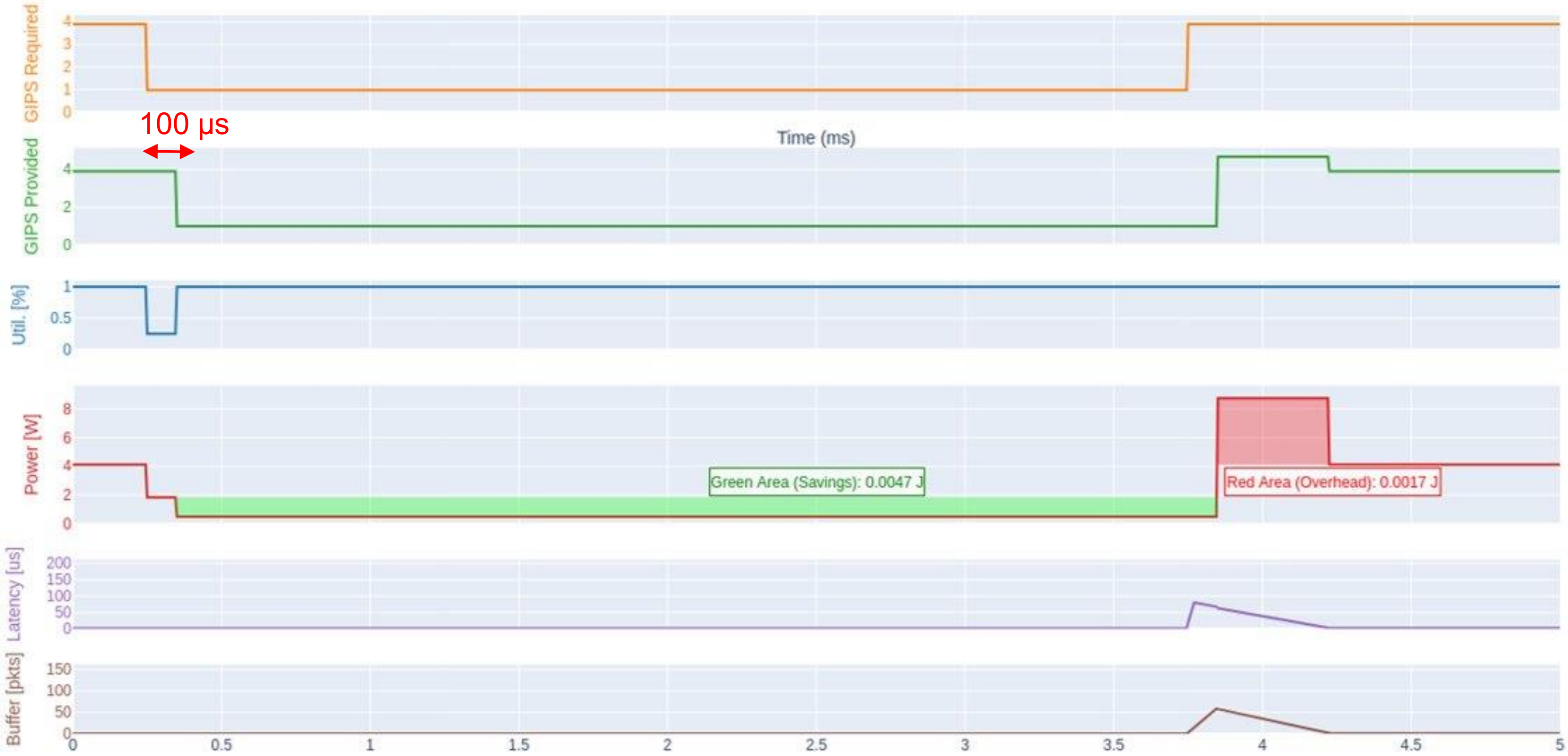
Analytic Model

Timeline visualization of DVFS



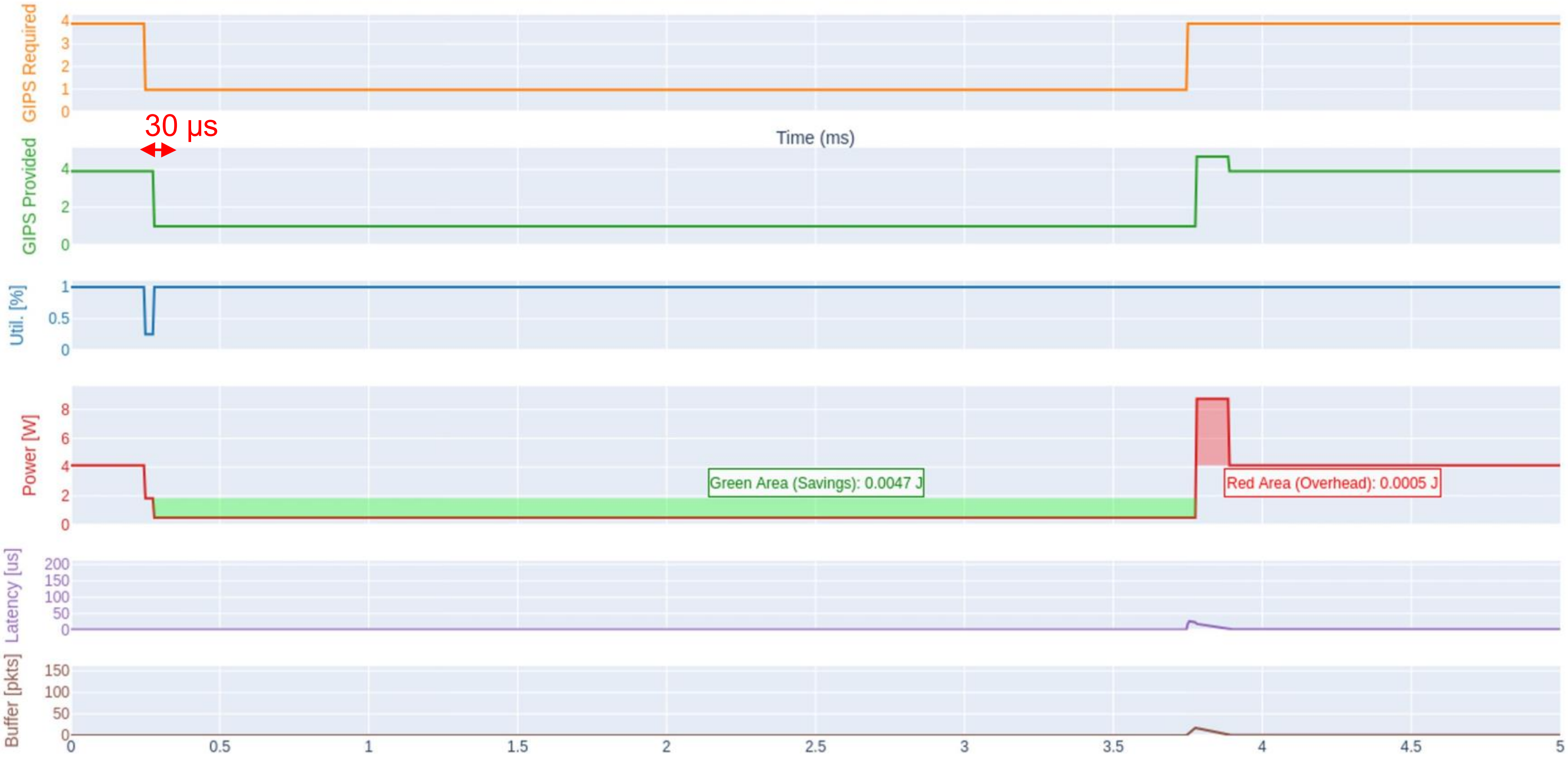
Analytic Model

Timeline visualization of DVFS



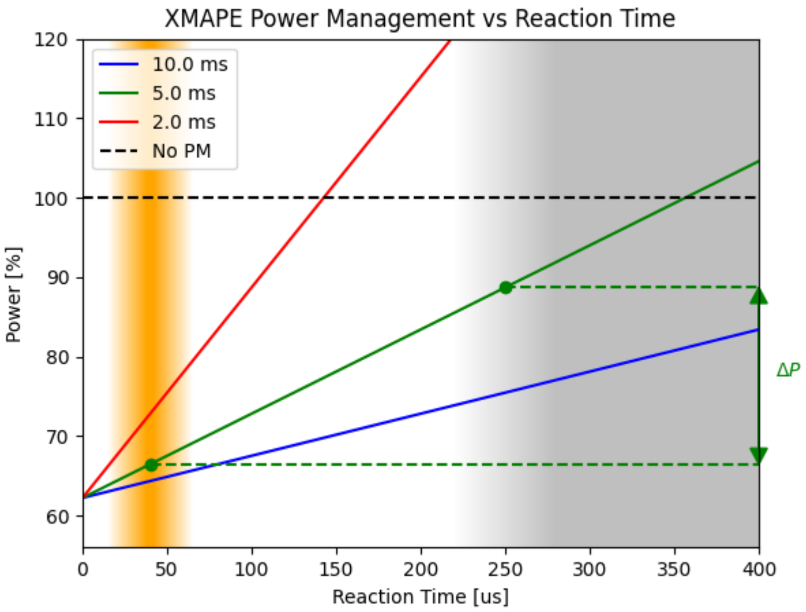
Analytic Model

Timeline visualization of DVFS



Analytic Model

Learnings



- Traffic fluctuations and flow diversity lead to quickly changing processing demand
- Event-driven, short-term compute demand difficult to manage for OS
- Minimal reaction time necessary to
  - → Achieve maximum energy savings
  - → Minimize negative impact on latency

Liess, M., Demicoli, J., Tiedje, T., Lohrmann, M., Nickel, M., Luniak, M., ... & Herkersdorf, A. (2023, November). X-mape: Extending 6g-connected self-adaptive systems with reflexive actions. In *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (pp. 163-167). IEEE.

SmartNICs

Motivation

Performance and Energy Efficiency

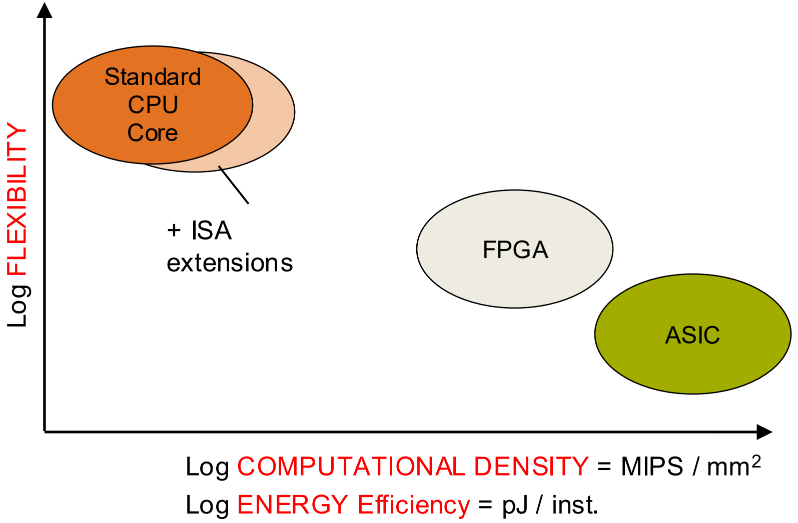
- cope with very high data rates (up to hundreds of Gbps)
- lowest packet delay as possible
- low power consumption

➡ Customized **ASIC**

Flexibility

- adapt to evolving packet processing applications
- efficient resource sharing among network applications

➡ Programmable **CPU / ASIP**



Source: T. Noll / H. Blume et al., „Model-based exploration of the Design Space for Heterogeneous System on Chip“, 2002

SmartNICs

Motivation

Performance and Energy Efficiency

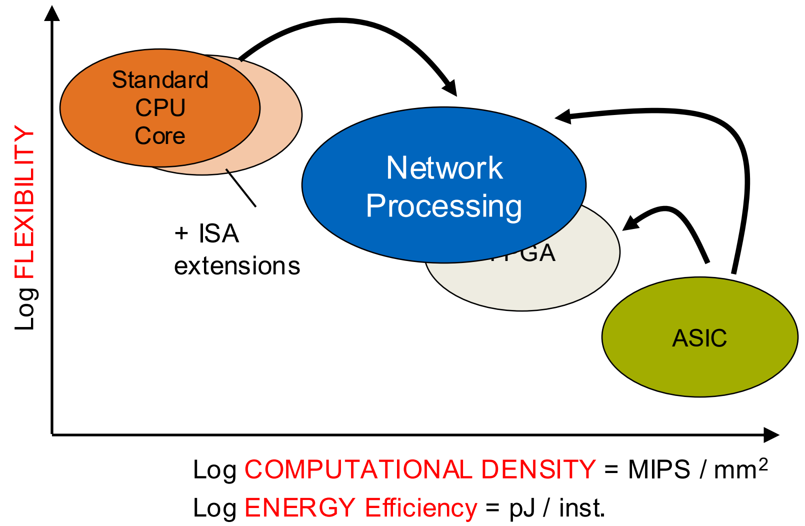
- cope with very high data rates (up to hundreds of Gbps)
- lowest packet delay as possible
- low power consumption

→ Customized **ASIC**

Flexibility

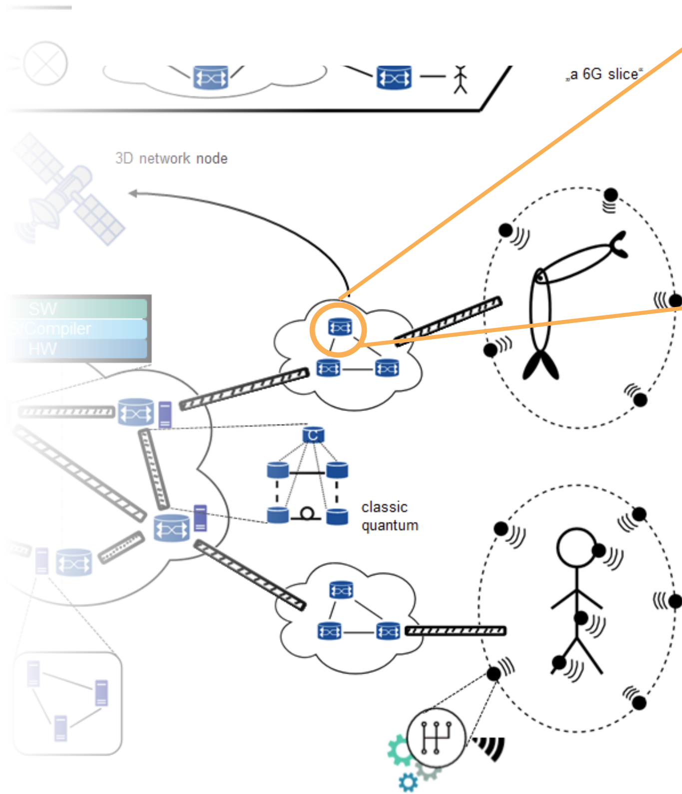
- adapt to evolving packet processing applications
- efficient resource sharing among network applications

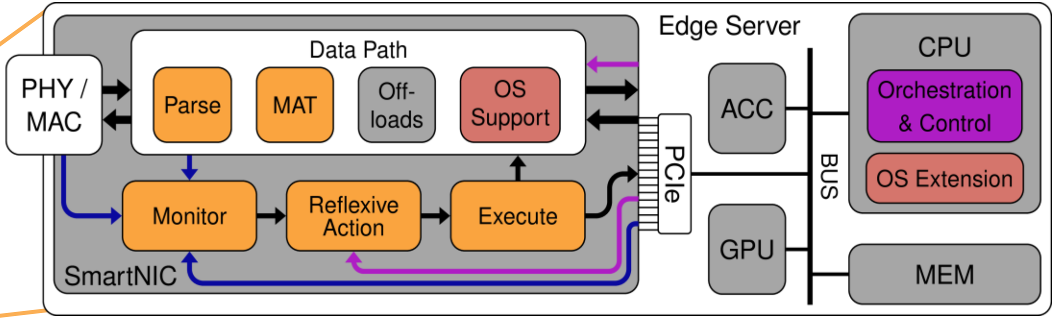
→ Programmable **CPU / ASIP**



Source: T. Noll / H. Blume et al., „Model-based exploration of the Design Space for Heterogeneous System on Chip“, 2002

SmartNIC Extensions @ LIS





Actions:

- Power management (e.g. DVFS of CPU cores)
- Offloading / Forwarding to other nodes in case of overload

Enabled by:

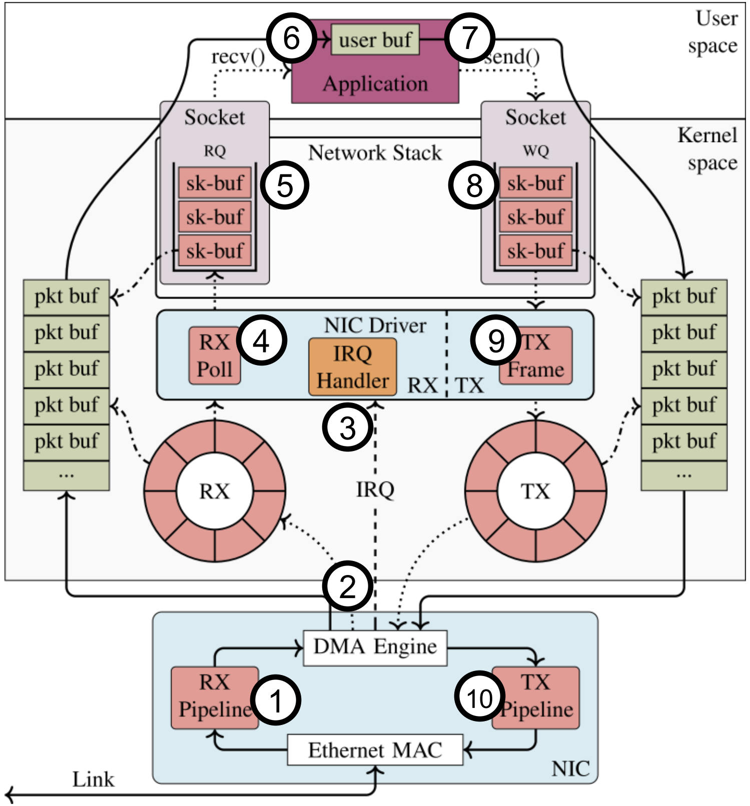
- ➢ Hardware monitoring of data path
- ➢ Configurable LUTs/MATs for system-level decisions
- ➢ Threshold-based triggering of actions

Linux Network Stack

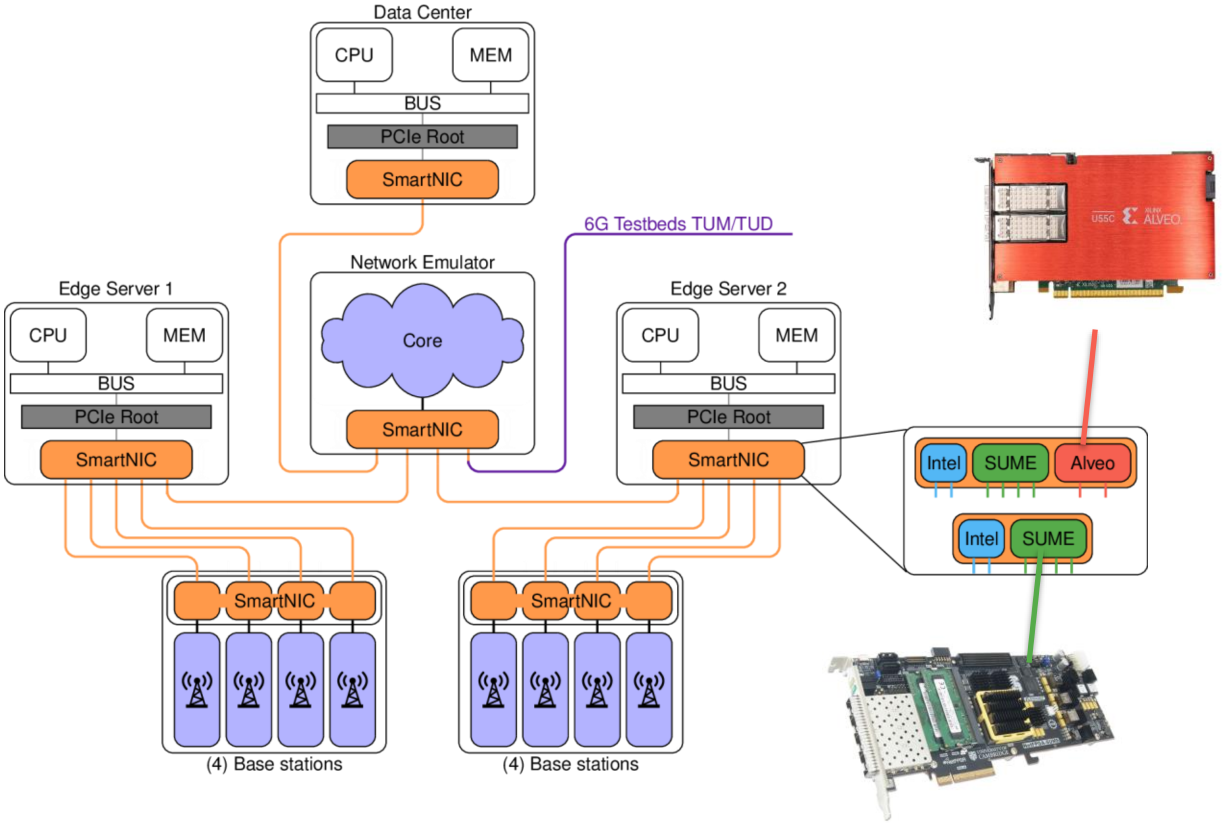
OS Support?

Challenges:

- Driver, Network Stack and App not „synchronized“
- Buffers disaggregate between functions
  → Limited info about state of other components
- Processing demand highly dependent on App
  → Difficult to predict without feedback



NETTB - NETworking TestBed @ LIS



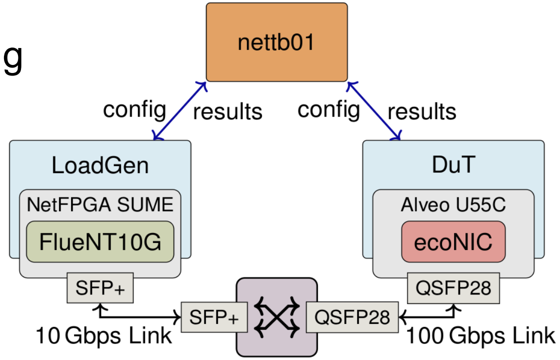
NETTB - NETworking TestBed @ LIS

1. Configuration

benchmark_settings.yaml > gen_trace

```yaml
dut_pc: ryzen01
dut_benchmark: l2fwd
dut_threads: [6,12]
dut_pstate: [ondemand,performance]
gen_pc: ryzen02
gen_trace: caida
```

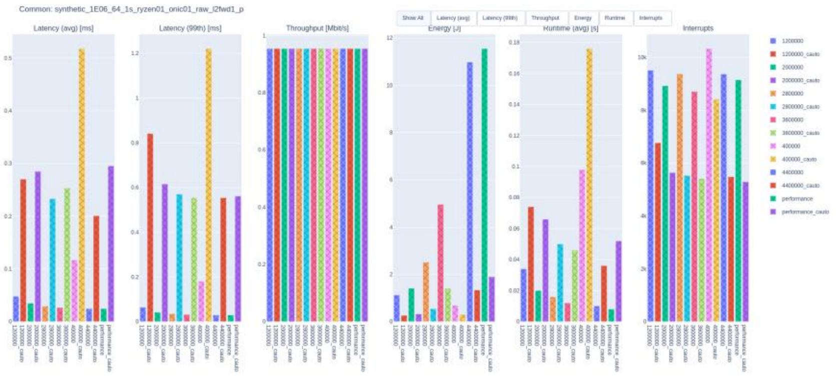
2. Running



3. Results

- run_5
- run_4
- run_3
- run_2
- run_1
- usage_summary.csv
- stats_summary.csv
- power_summary.csv
- latency_summary.csv
- interrupts_summary.csv
- gen.log
- dut.log
- benchmark_settings.yaml

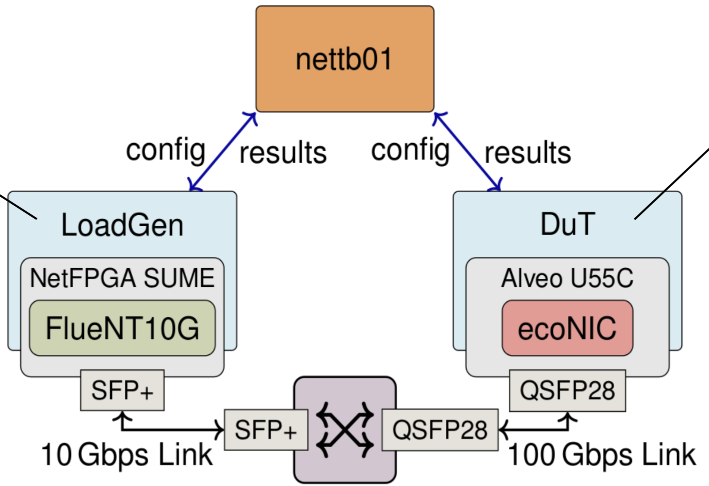
4. Postprocessing



Experimental Evaluation

Experimental Setup

CAIDA Network Trace 2019 (10Gbps)



AMD Ryzen 5 5600G

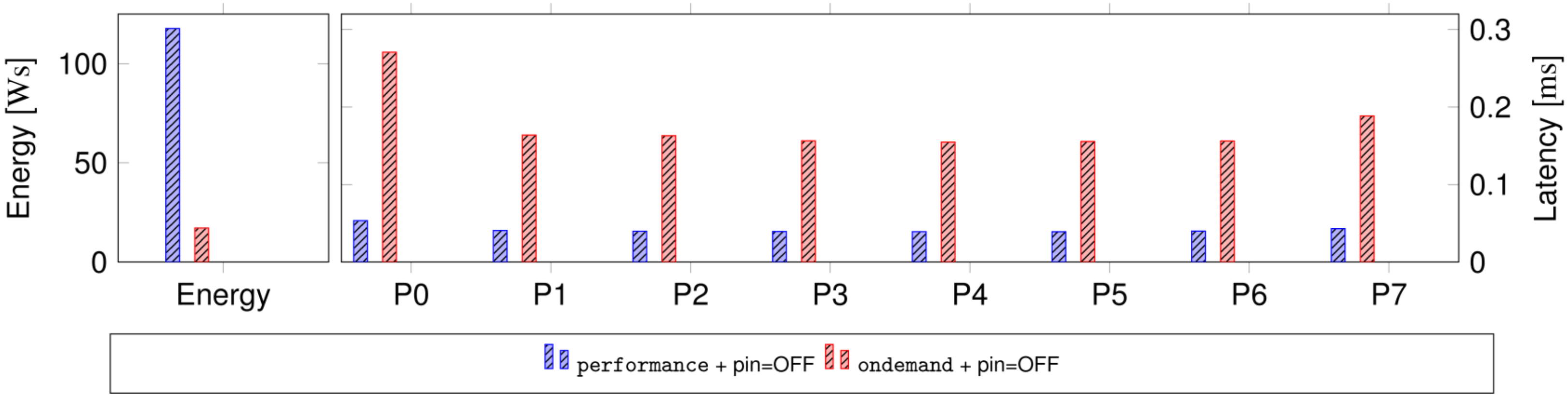
Linux 6.8 (Mainline)

AMD “OpenNIC“ Driver

Benchmark: L2 Forwarding

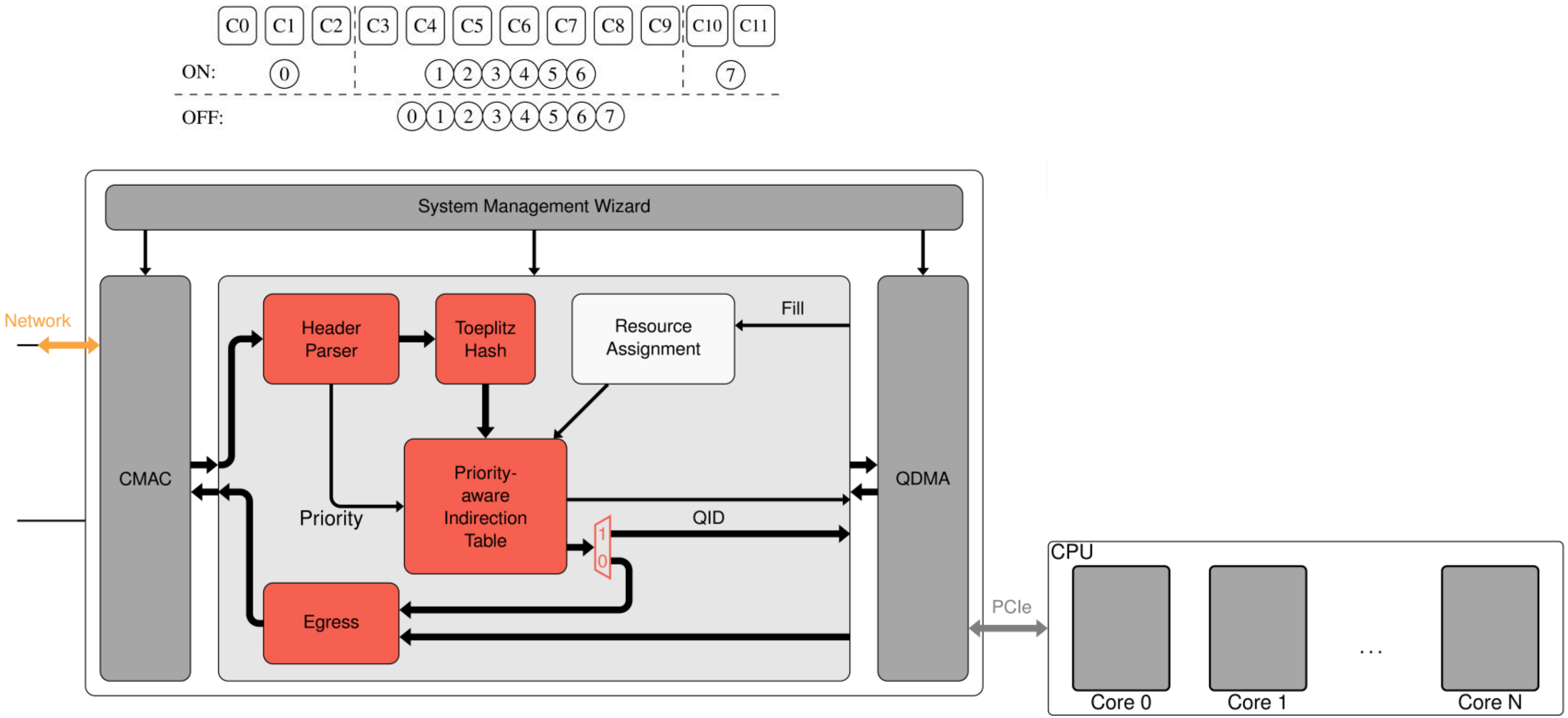
Experimental Evaluation

Linux Governors



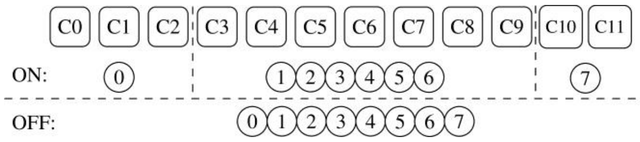
Experimental Evaluation

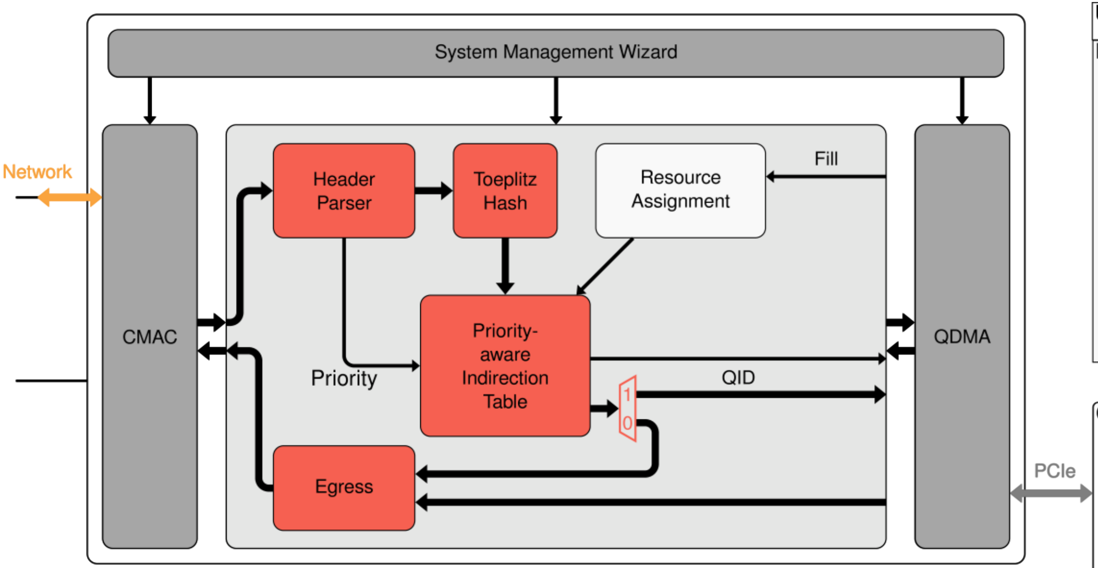
Hardware Priority Pinning

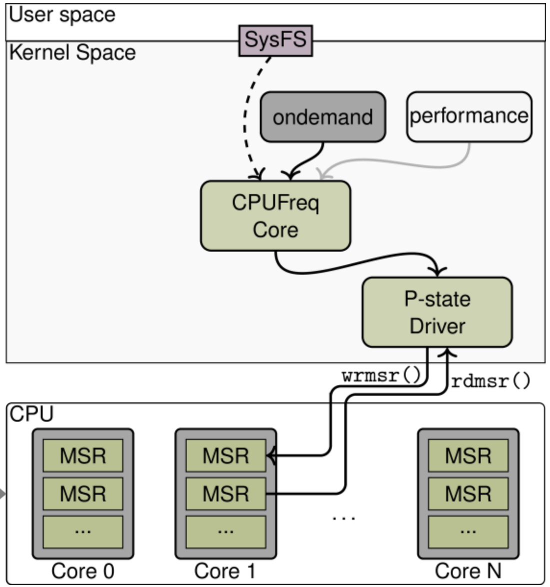


Experimental Evaluation

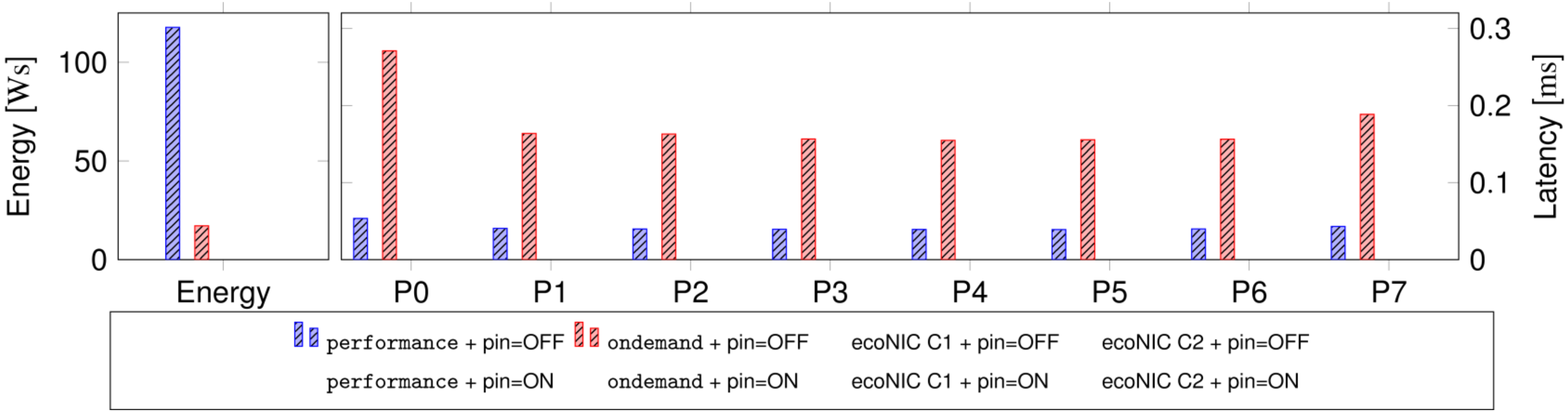
Hardware Priority Pinning





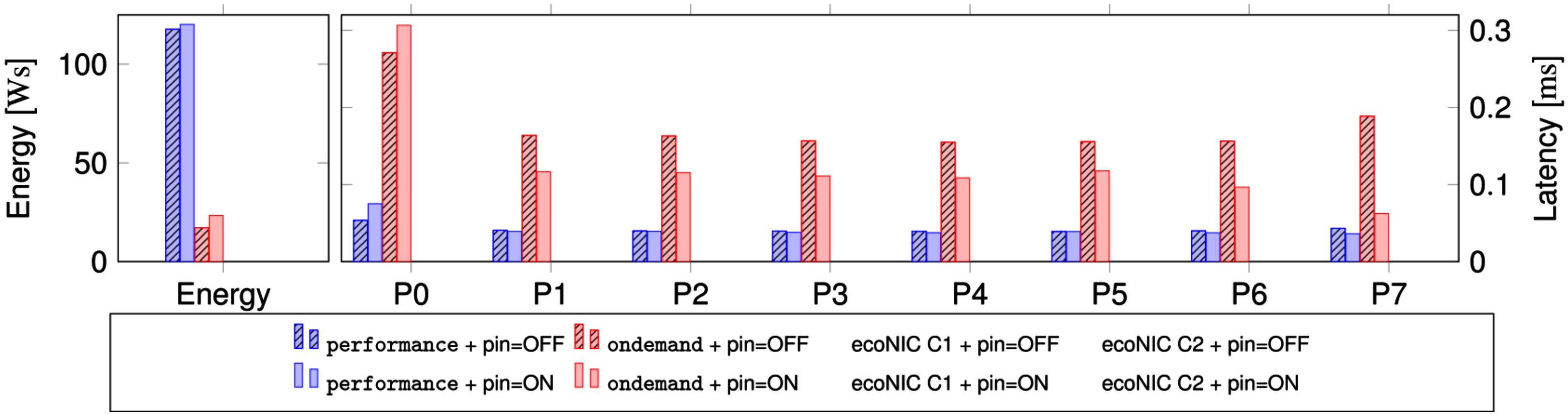


Experimental Evaluation
Linux Governors



Experimental Evaluation

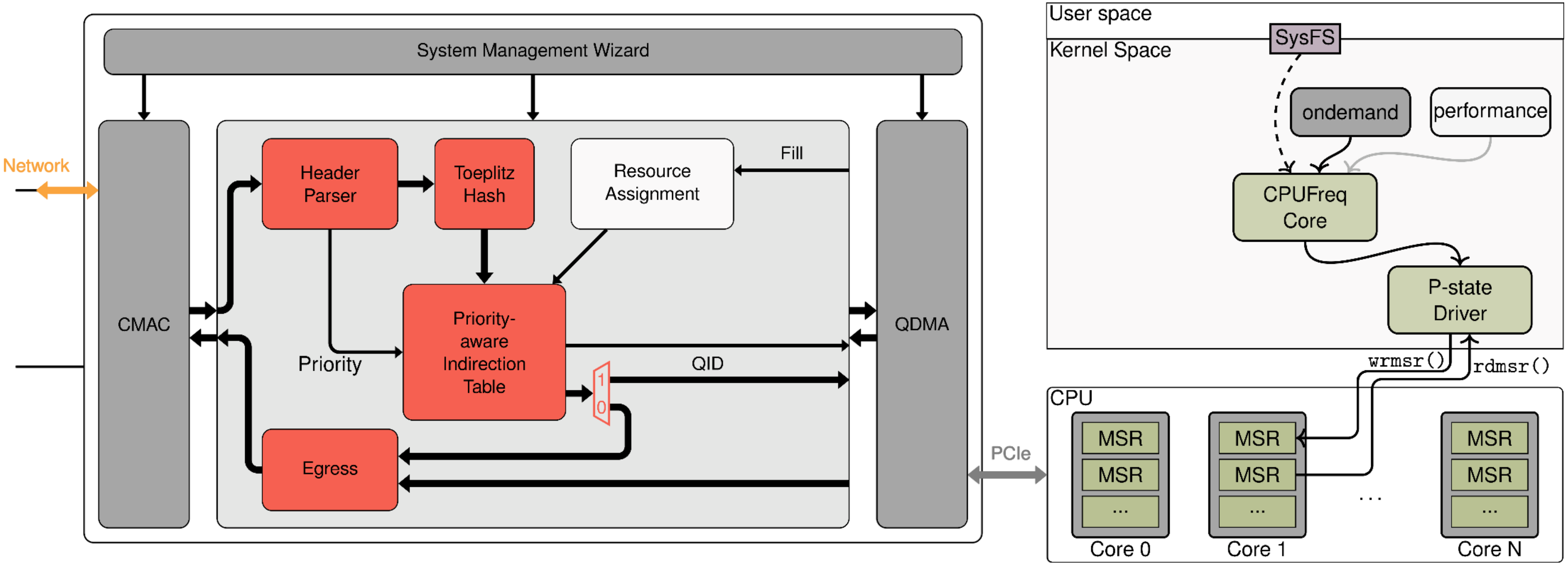
Linux Governors



Experimental Evaluation

ecoNIC: SmartNIC-assisted Power Management [1]

Combines traffic-dependent power management in OS / Linux
with priority-aware traffic steering / pinning in SmartNIC HW

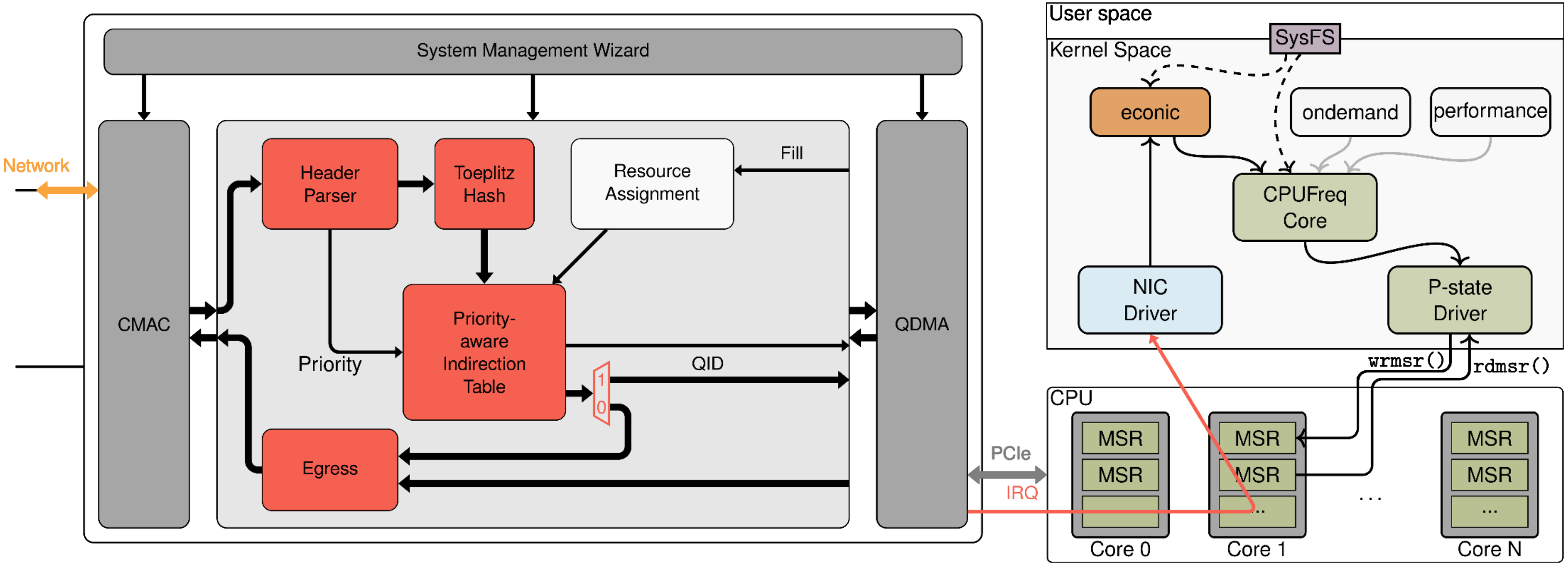


Liess, M., Biersack, F., Nolte, L., Wild, T., & Herkersdorf, A. (2025). ecoNIC: SmartNIC-assisted power management for networking workloads in Linux servers. Microprocessors and Microsystems, 105209.

Experimental Evaluation

ecoNIC: SmartNIC-assisted Power Management [1]

Combines traffic-dependent power management in OS / Linux
with priority-aware traffic steering / pinning in SmartNIC HW

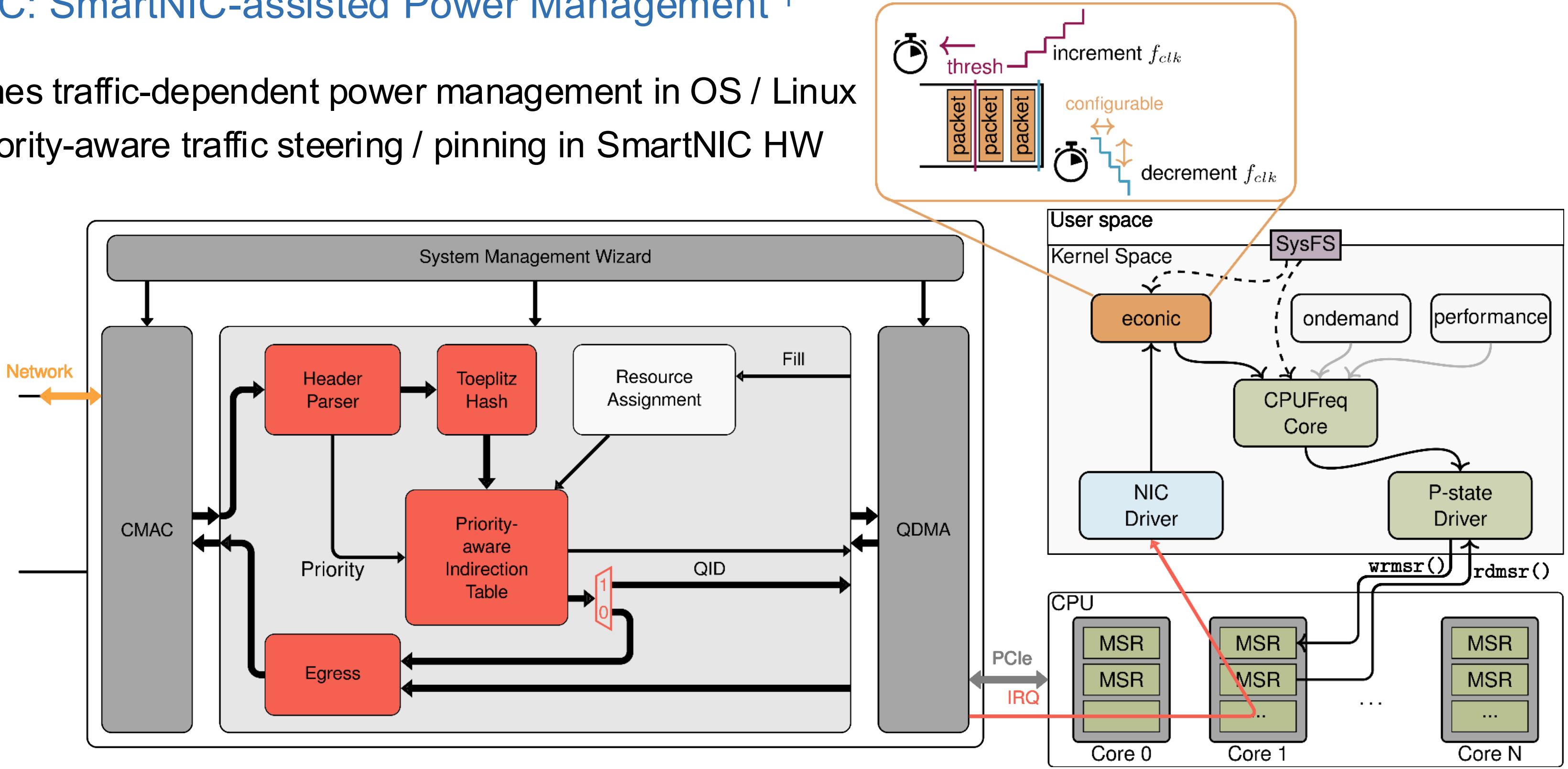


Liess, M., Biersack, F., Nolte, L., Wild, T., & Herkersdorf, A. (2025). ecoNIC: SmartNIC-assisted power management for networking workloads in Linux servers. Microprocessors and Microsystems, 105209.

Experimental Evaluation

ecoNIC: SmartNIC-assisted Power Management [1]

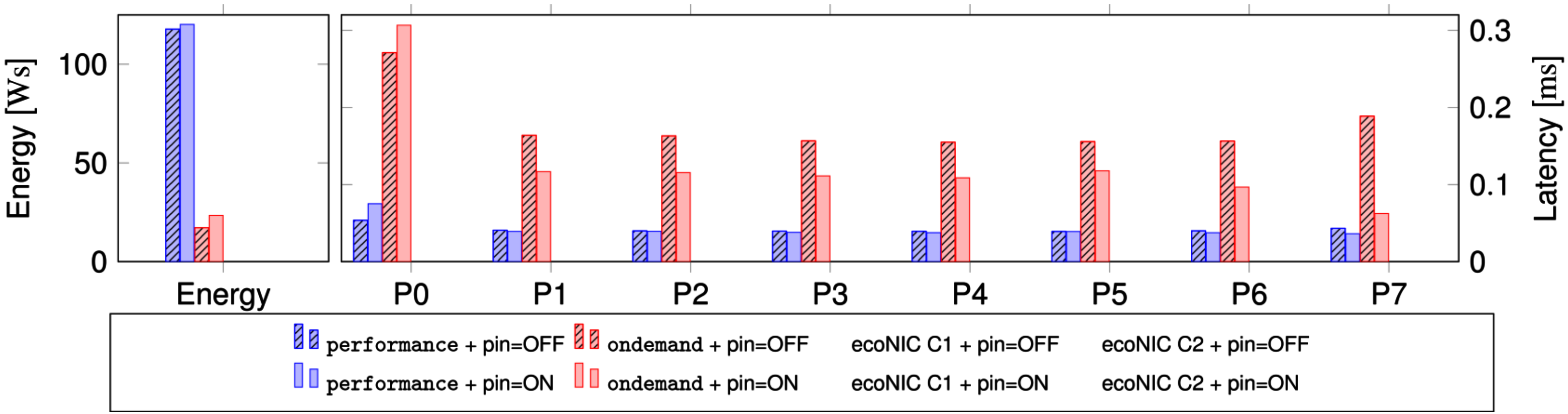
Combines traffic-dependent power management in OS / Linux with priority-aware traffic steering / pinning in SmartNIC HW



Liess, M., Biersack, F., Nolte, L., Wild, T., & Herkersdorf, A. (2025). ecoNIC: SmartNIC-assisted power management for networking workloads in Linux servers. Microprocessors and Microsystems, 105209.

Experimental Evaluation

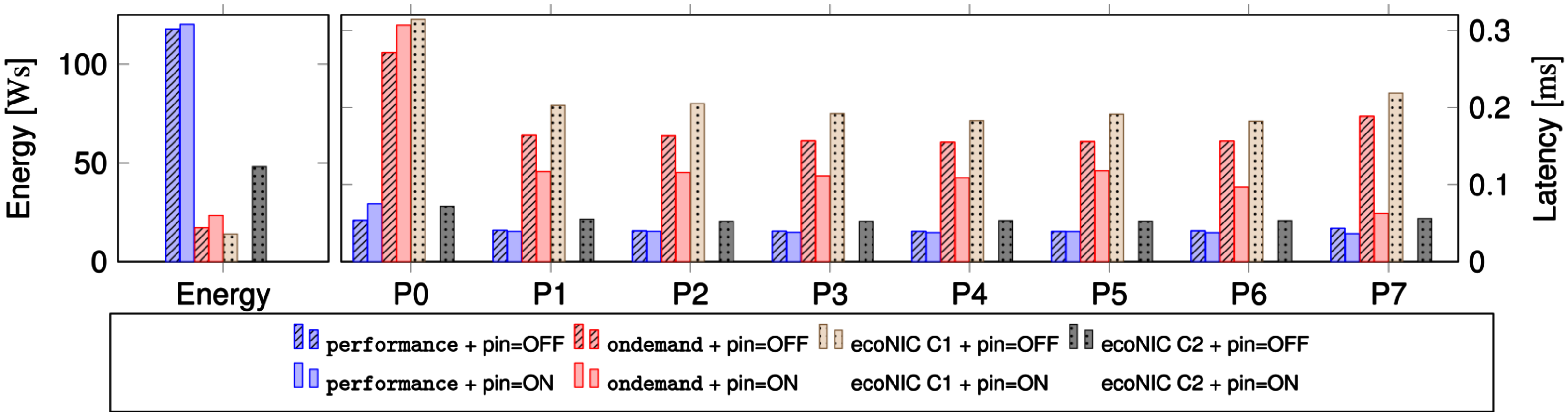
ecoNIC: SmartNIC-assisted Power Management [1]



Liess, M., Biersack, F., Nolte, L., Wild, T., & Herkersdorf, A. (2025). ecoNIC: SmartNIC-assisted power management for networking workloads in Linux servers. Microprocessors and Microsystems, 105209.

Experimental Evaluation

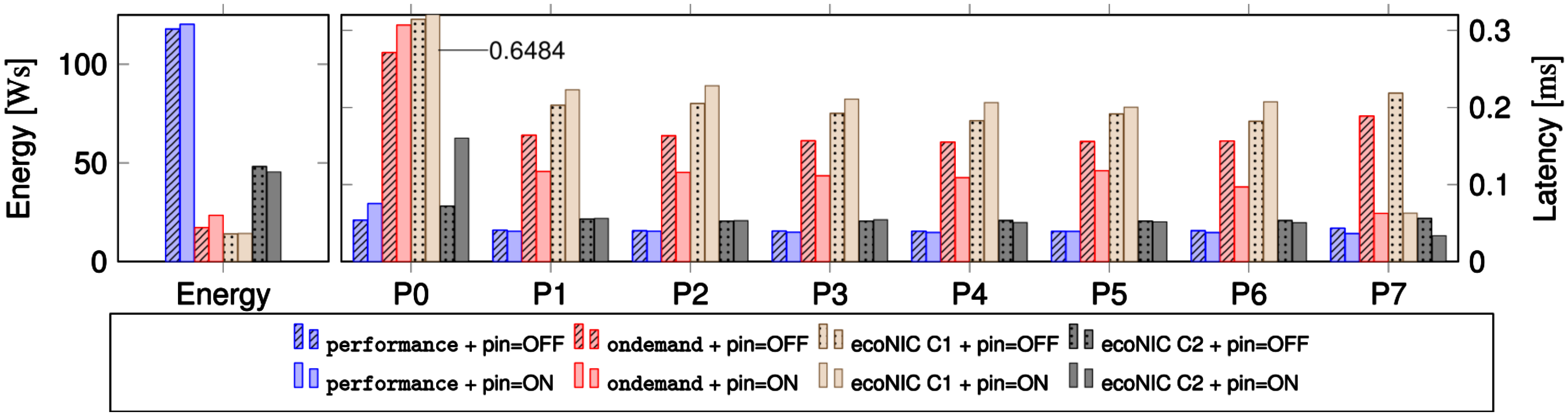
ecoNIC: SmartNIC-assisted Power Management [1]



Liess, M., Biersack, F., Nolte, L., Wild, T., & Herkersdorf, A. (2025). ecoNIC: SmartNIC-assisted power management for networking workloads in Linux servers. Microprocessors and Microsystems, 105209.

Experimental Evaluation

ecoNIC: SmartNIC-assisted Power Management [1]



Liess, M., Biersack, F., Nolte, L., Wild, T., & Herkersdorf, A. (2025). ecoNIC: SmartNIC-assisted power management for networking workloads in Linux servers. Microprocessors and Microsystems, 105209.

Conclusions & Outlook

- Power Management essential for energy-efficient server operation
- Fluctuating, unpredictable network processing demand poses huge challenge
- CPU / OS power management not optimized for external demand

→ SmartNICs can monitor traffic demand right at the source
→ offers improvement in adaptability and energy-efficiency

Outlook:

- Extend SmartNIC with workload prediction, application information, etc.

Let‘s discuss!

Acknowledgement
We are greatful for the financial support by the *Bavarian Ministry of Economic Affairs, Regional Development and Energy* in the *6G Future Lab Bavaria* project and the *Federal Ministry of Education and Research* of Germany in the *6G-life* project with project identification number: 16KISK002

6G BAVARIA

6G-life

Bayerisches Staatsministerium für Wirtschaft, Landesentwicklung und Energie

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung