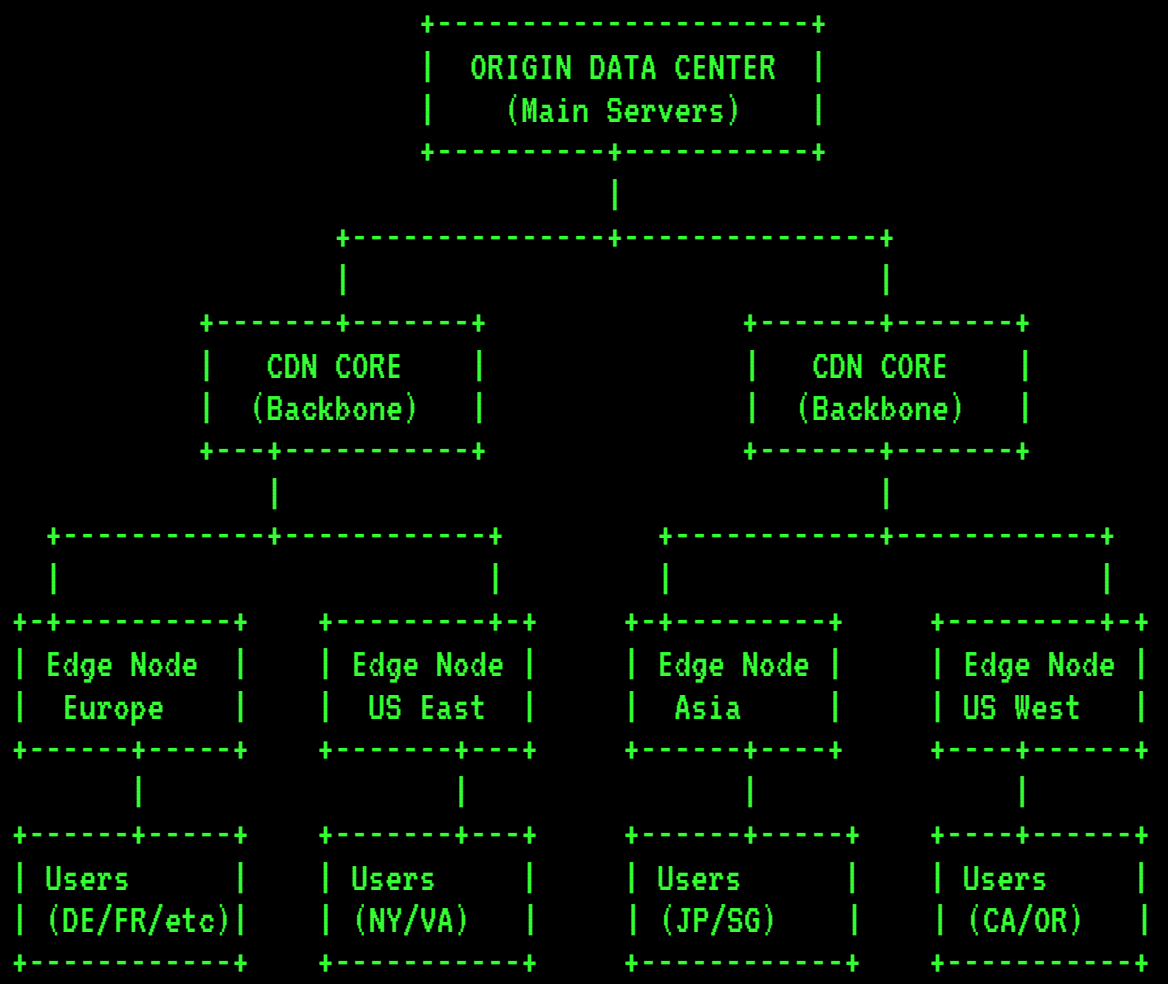
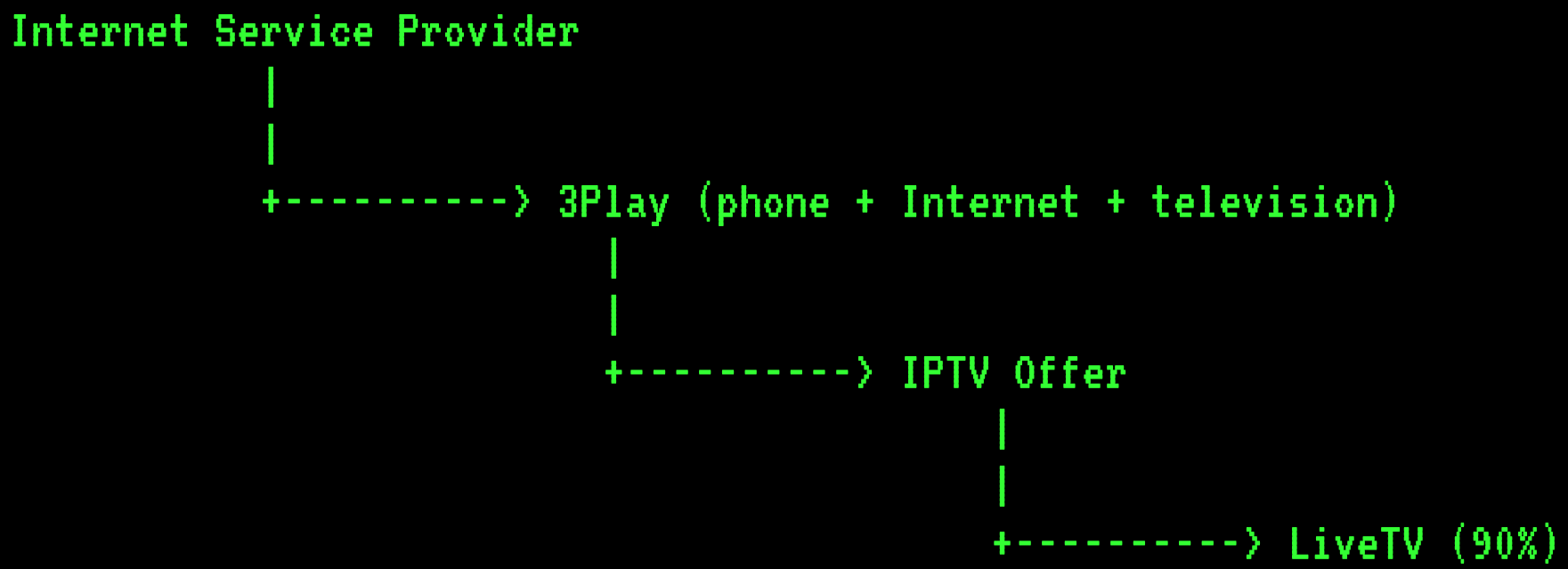
Carbon-Aware Data Centers Orchestration
CDN as Case Study

Tamás Jursonovics

Jiyan Salim Mahmud



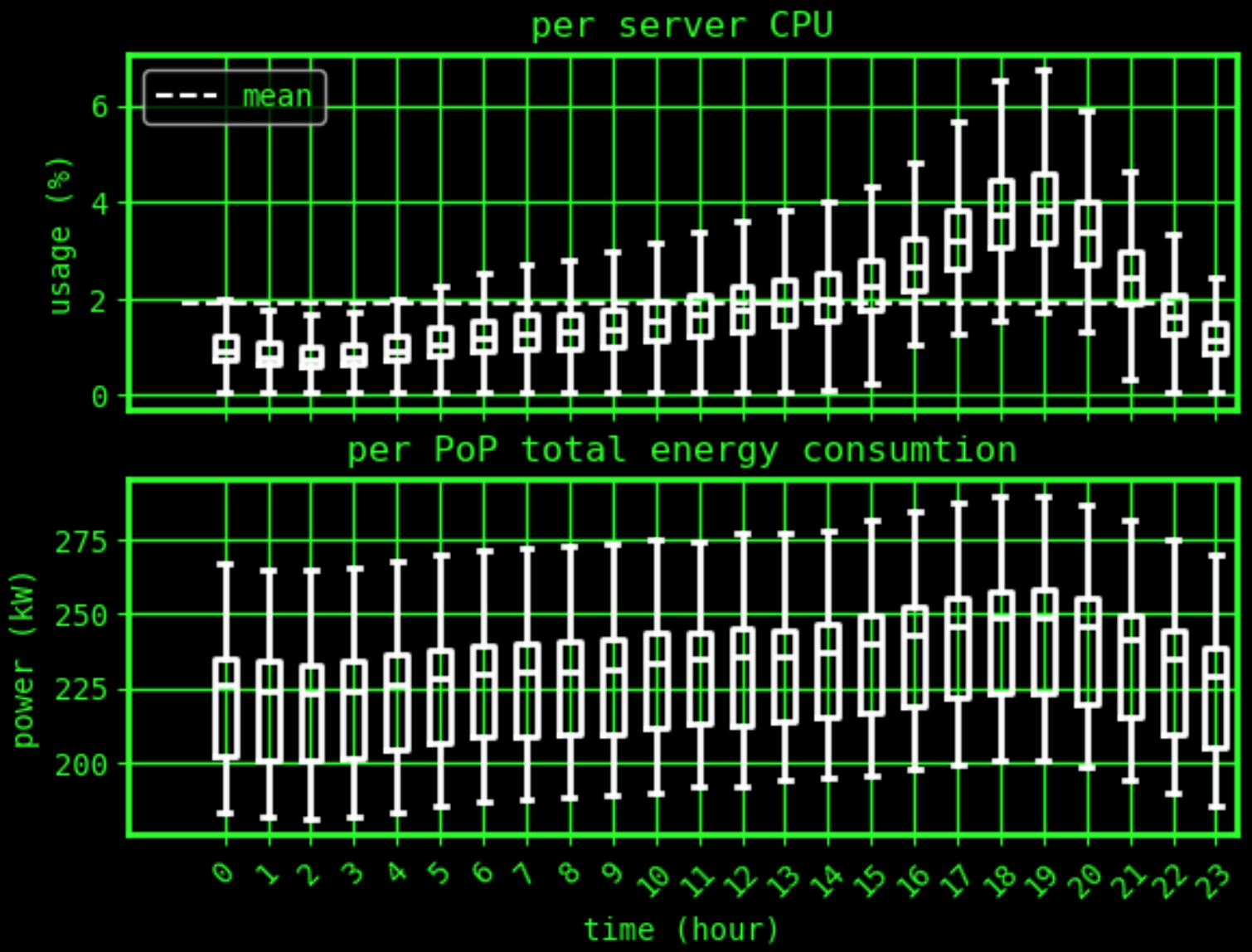
Need for a CDN



Building a CDN

Abandon multicast ----------+
                            |
Network monetization -------+
                            |
Cost efficiency ------------+
 - cheap* COTS hardware |
 - maturity of open-source SW |
 - know-how |
                            |
                            |
   Large, inefficient CDN <---+
   ====================

* before the RAM crisis

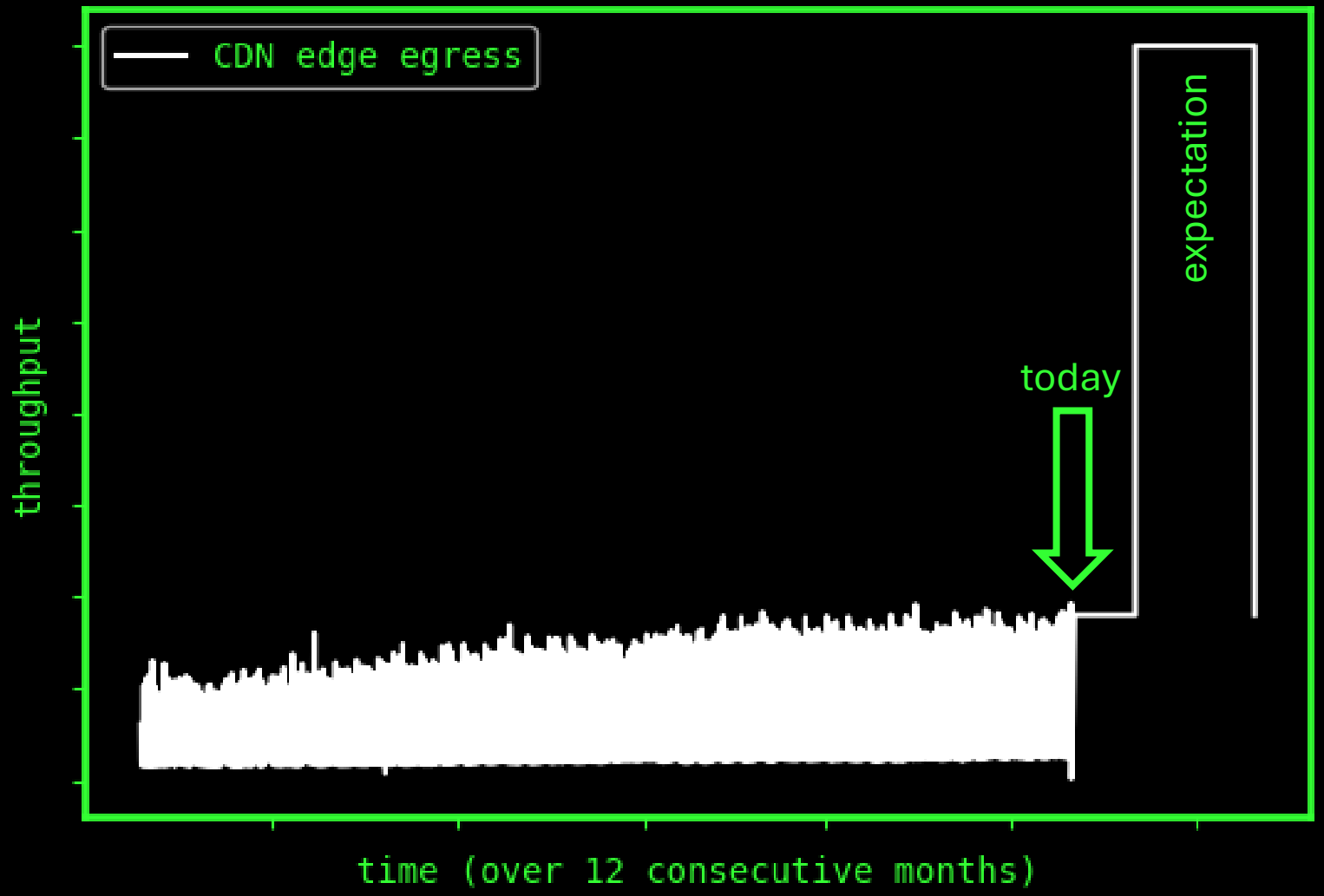


Quality First!

Events
- soccer

Benchmark
- relevant for customers

No shared platform
- bandwidth not available
- resource commitment



Options

- Balanced power advisor --> QoE issues (running OS control, perf. govenor)
- SW stack optimization --> already performance tuned (XDP, io_uring)
- CPU HW optimization --> 5G core, but limited gain in CDN, ARM: no perf.
- Chasse optimization --> constant ~400W, opposing interest of HW vendors
- Immersive cooling --> lack of data-center infrastructure

```
+-------------------------------------------------+
| On-demand orchestration: load consolidation and shutdown |
|                    - low effort                 |
|                    - ambition: save 80-85%      |
+-------------------------------------------------+
```

Maintain Quality of Experience!

Mgmt. API + Automation --> no-brainer (chatGPT) --> GitLab --> savings ensured



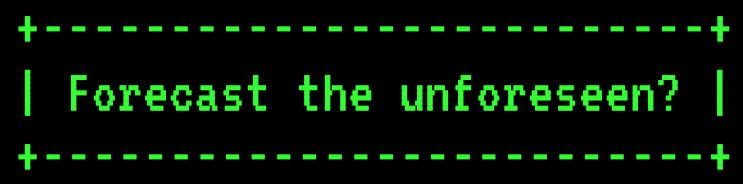
Two main contributions

- determination of the number of active servers
- seamless scaling

Resource Model and Forecasting

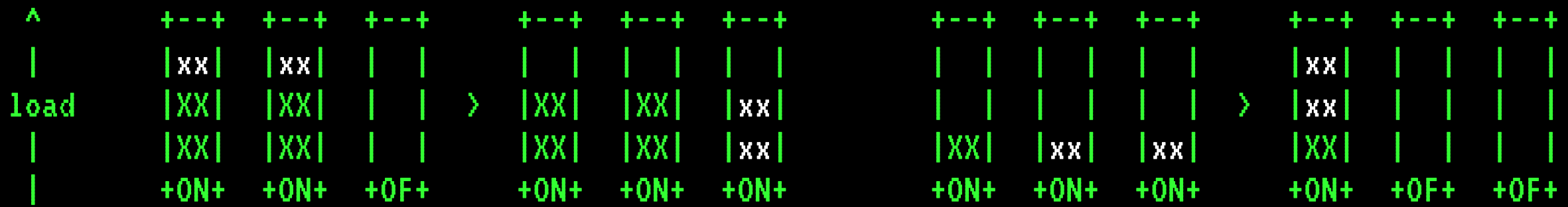
CPU the first bottleneck
- capacity changes over time (CPU frequency scaling)
- utilization --> number of CPU instructions executed
- non-linearity (CPU thermal limitation)

Short term (next 10 min) forecast
- 3.5 min server power-up (HPE Fast Boot)
- daily load pattern is easy



linear regression from the last 15 min to the next 5 min

Active Redistribution and Drain



scale out (redistribution)

scale in (drain)

- probabilistic approach: HTTP 200 + Connection Close header / go-away frame
- adaptive server weights

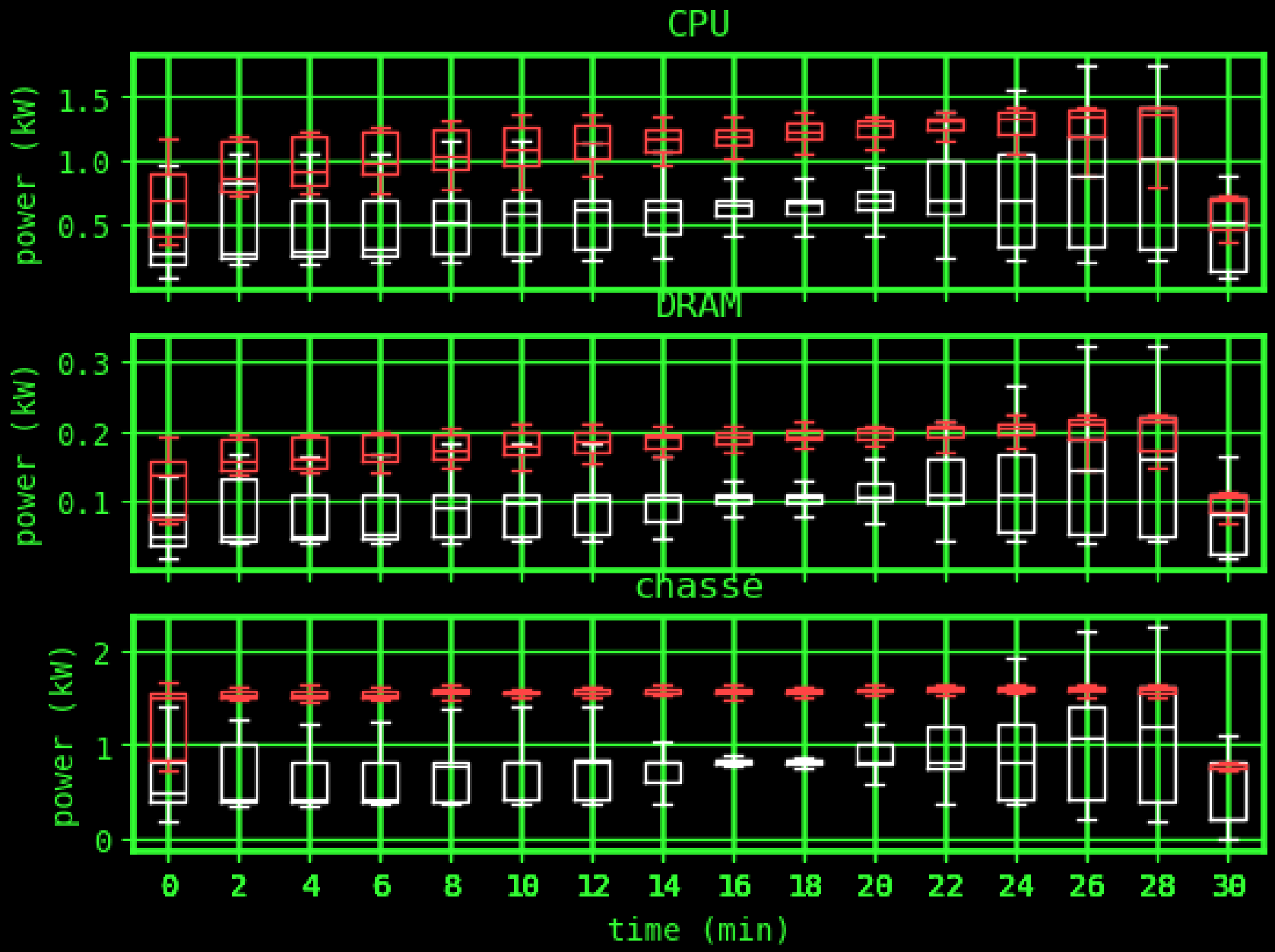
(L4 loadbalancing, DNS based location assignment)

Results

Environment
- 7 physical server
- locust.io as loadtester
- CPU capped at 4 cores
- iLO, RAPL power metrics

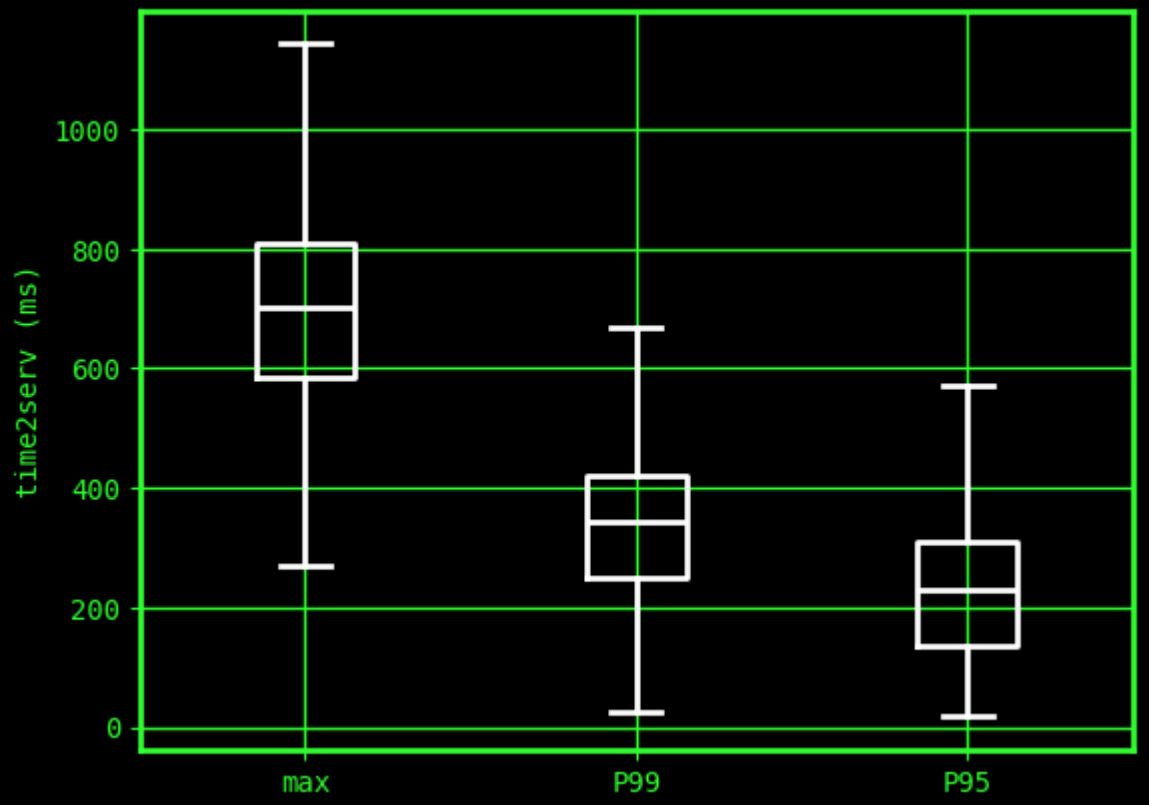
Load pattern
- gaussian random walk
- 100 orchest. run
- 24 baseline run

Quality KPI
- time2serv (<2s: OK)



Results - Quality

Orchestration



Baseline

