

Capping with Confidence: A Test & Validation Framework for Power Savings and a Lower Data-Center Carbon Footprint

Smitha Kashyap C (smic@microsoft.com) Chockalingam A
(choa@microsoft.com) Vinod Vikraman
(vvikraman@microsoft.com) Srikanth Ambati
(sriambati@microsoft.com) Mihir Kikani
(mikikani@microsoft.com) Nithish Mahalingam
(nithishm@microsoft.com)

Microsoft Corporation

Abstract— AI and High-Performance Computing (HPC) growth is pushing data centers against their scarcest resource: power. To hit 99.99–99.999% availability, operators add redundant infrastructure, so power is provisioned hierarchically (Colo/Cell → Row → Rack → Server → CPU) yet rarely fully used. Because workloads load asynchronously, this slack compounds at every tier — cell slack exceeds row slack, which exceeds rack slack — stranding large amounts of power. Benchmark-based rack elevation, oversubscription, and zero-reserve-power designs can reclaim this slack and lift utilization by roughly 33%, consistent with published zero-reserve-power results. But reclaimed headroom stresses Power Distribution Units (PDUs) and breakers: a utility outage combined with redundant-UPS failure can drain batteries in seconds, and breaker trip curves leave only tens of seconds — or less — to respond. Power capping is the safety net that prevents brownouts and breaker trips during failover and maintenance events.

Reclaimed power is only safe if capping is proven, not assumed. The novelty here is not power capping itself but a method to prove it: a cross-layer Test and Validation framework built on four measurable SLAs — power-limit accuracy (± 5 –10%), settling time, overload detection and trigger time, and recovery time. Recovery matters because capping temporarily throttles customer workloads. Where milliseconds count, manual PROCHOT assertion delivers fast throttle, and its response is measured too. We then stress the feature of negative tests that mirror real operations, not ideal labs: Online Insertion and Removal (OIR) of field-replaceable units, power-cycle and reboot persistence, firmware upgrades and VM churn, fault injection, and injected power spikes. Each result is tabulated by impact, duration, and recovery

for operators. Guided by these findings, we recommend validated design practices spanning dedicated hardware control, telemetry efficiency, sensor accuracy, tighter IO knobs, and debug hooks.

The payoff is direct: every watt of headroom that validated capping unlocks needs no extra PDU, rack, or row — cutting both operational and embodied carbon per unit of compute. As GPU demand surges, rigorously validated power capping is what makes aggressive oversubscription safe. You can only cap what you can trust, and you can only trust what you have tested.

Keywords—*data-center power management; power capping; test and validation; SLA metrics; oversubscription; stranded power; sustainability; embodied carbon*

I. Introduction

Artificial Intelligence and High-Performance Computing workloads are growing faster than the electrical infrastructure that serves them, and the shortage now expresses itself as a power constraint rather than a compute constraint. Availability targets make the constraint tighter still. To meet 99.99–99.999% availability, operators build redundancy into every electrical tier, and power is provisioned hierarchically from the Colo or Cell, down through the Row, the Rack, the Server, and finally the CPU sockets and components. At each of those tiers a portion of the provisioned power is held in reserve, and the remainder is allocated to equipment.

Servers, however, rarely draw the power allocated to them. The difference between allocated and consumed power is slack, or stranded power, and because workloads load asynchronously across machines the slack compounds as it rolls up the hierarchy: cell-level slack exceeds row-level slack, which exceeds rack-level slack, which exceeds chassis-level slack. The Top side of Fig. 1 shows the traditional picture, in which reserve power and stranded power together account for a substantial share of a facility's electrical capacity while utilization power remains modest.

Three families of technique are widely deployed to reclaim that slack. The first estimates node power with industry-standard benchmarks and uses the measured figure, rather than the nameplate figure, for rack elevation. The second is

power harvesting or oversubscription, which adds servers, racks, or rows inside the existing electrical footprint. The third pushes toward zero reserve power by putting the reserved portion to work; published results report roughly a 33% increase in power utilization from this approach [1]. The bottom side of Fig. 1 shows the effect: stranded power shrinks and utilization power expand into the space it vacates.

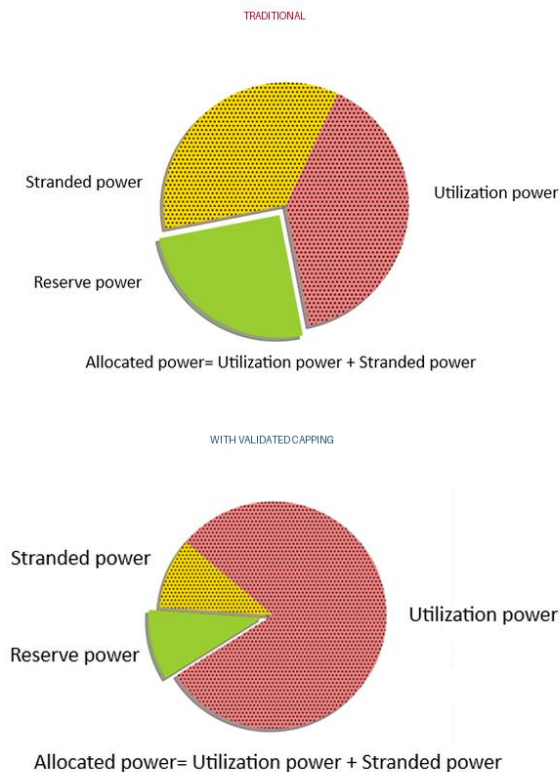


Fig. 1. Reclaiming stranded power — traditional data-center utilization (top) versus improved utilization after validated oversubscription and power capping (bottom).

Reclaiming headroom is beneficial, but it transfers risk onto the electrical distribution path. Running closer to the rated capacity of PDUs and breakers leaves less margin for redundant feed failure, UPS failover, planned power maintenance, and synchronous workload bursts that briefly exceed provisioned power. The window for a response is short, and it narrows as the overload deepens. Circuit-breaker trip curves, illustrated in Fig. 2, show trip time falling steeply as current rises above the rated value; in data-center practice the available window is typically on the order of tens of seconds, and in the worst case — a

utility outage coincident with a redundant UPS failure — battery discharge is faster still, compressing the time available to regulate power back within the allocated envelope.

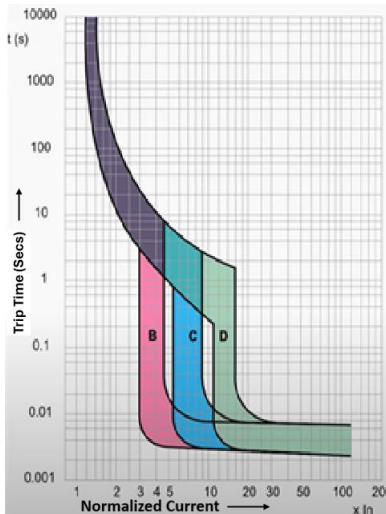


Fig. 2. Typical circuit-breaker trip behavior; the response window shrinks sharply as current rises above the rated value [3].

Power capping is the safety net that closes this gap. It regulates consumption within the allocated limit during failover and maintenance events, preventing brownouts and breaker trips at every level of the distribution hierarchy. The premise of this paper is that reclaimed power is only safe if capping is proven, not assumed. Our contribution is therefore not the capping mechanism itself but a method to prove it: a cross-layer test and validation framework built on four measurable service level agreements, a negative-test suite drawn from real operational events rather than ideal laboratory conditions, and a set of validated design practices that follow from what the testing exposes.

II. Power Capping Feature and Components Involved

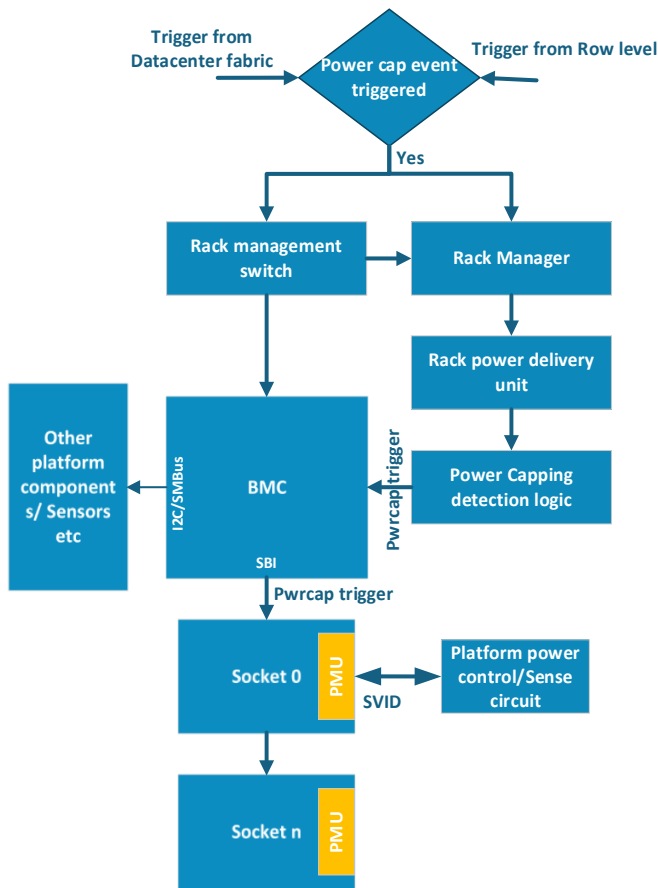


Figure 4: Typical components involved in power capping implementation

The diagram above illustrates the typical components involved in the implementation of power capping in datacentres. Power capping events can be triggered from various sources, such as the datacentre fabric, row-level managers, or other similar origins.

One scenario involves the rack manager receiving the power capping signal and subsequently passing it on to the Board Management Controller (BMC), which is responsible for all control activities at the server or node level. The BMC can then send the power cap limit to the processor after analysing the power requirements of the remaining platform components. Upon receiving this limit, the processor Power Management Unit (PMU) will cap the overall processor power utilization by coordinating with platform-level sensors or power monitoring units.

In another scenario, a power capping detection circuit on the server detects the power capping events and forwards them to the BMC for further processing. Typically, the PMU

from the primary socket (or socket 0) is responsible for controlling the remaining sockets through inter-processor communication channels.

III. A Test and Validation Framework: Four Measurable SLAs

A capping implementation is trustworthy only to the extent that its behavior is bounded by numbers an operator can hold it to. We define four such numbers. They are measurable at the node, they roll up to rack and row, and together they describe the full arc of a capping event from detection through throttle, settling, and release. Table I summarizes them.

A. Power Limit Accuracy

Limits are set at the higher tiers from allocated power and workload performance needs, then driven downward to rack, node, and component. Node-level accuracy is the one that must hold at all times and across the full loading range, from light load to full load, because once the node limit is accurate the rack and row limits follow with relatively little additional effort. Accuracy depends on sensor reporting accuracy, the averaging and sampling window configured for those sensors, frequency and temperature variation, and how tightly bandwidth can be constrained on subcomponents such as memory and IO. A design accuracy of $\pm 5\%$ to $\pm 10\%$ is a reasonable target and is the band we validate against.

B. Settling Time

Settling time is the interval a node or rack takes to bring consumption down to the pre-configured limit. Its budget is derived, not chosen: the available breaker trip time, the magnitude of the credible power surge, and the UPS battery discharge time at the higher tiers together fix an end-to-end window, which is then subdivided into rack-level and node-level allowances. Where milliseconds decide the outcome, a fast throttle is required, and manual assertion of the PROCHOT signal supplied by CPU vendors provides it. When that path is used, the time the node takes to reach the PROCHOT power level is measured as a separate figure alongside the normal settling time.

C. Overload Detection and Trigger Time

This metric belongs primarily to the Power Supply Units (PSUs) inside the distribution path. It covers the time a PSU takes to detect an overload, normally through over-current warning and protection limits, and the additional time the management unit takes to sense that alert and generate the downstream trigger that starts throttling. Load is usually shared across several PSUs, and those units do not detect identically, so the spread between them must be characterized rather than assumed away. In many designs this detection and trigger time is combined with settling time to express a single end-to-end SLA, which is the form an operator can actually hold the platform to.

D. Recovery Time

Recovery time is the interval the node takes to return to its original performance state once the capping condition is removed. It is the metric that connects the electrical domain to the customer, because capping throttles live workloads and that throttling carries a cost [2]. A capping design that is accurate and fast but slow to release converts a brief electrical event into a long performance event, so recovery is validated with the same rigor as the other three metrics rather than treated as an afterthought.

Fig. 3 shows a sample server power-capping trace annotated with these metrics: idle power, the rise to maximum power under a stress workload, the capping trigger, the fast throttle, the settling interval, the capped region within which power-limit accuracy is assessed, and the recovery back to original performance after capping stops.

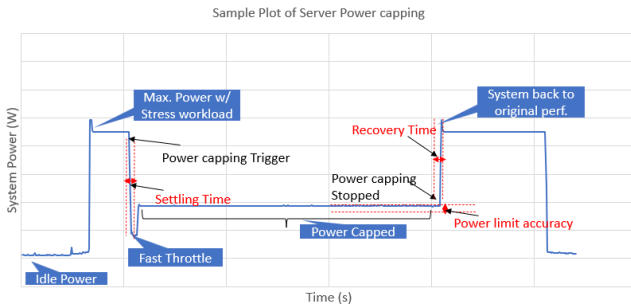


Fig. 3. Sample server power-capping behavior, annotated with the validated SLA metrics: fast throttle, settling time, power-limit accuracy, and recovery time.

TABLE I. SUMMARY OF THE FOUR VALIDATED POWER-CAPPING SLAS

SLA metric	What is measured	Validation target
Power limit accuracy	Deviation of measured node power from the configured limit, across the full load range	$\pm 5\%$ to $\pm 10\%$
Settling time	Time taken to bring node or rack power down to the configured limit	Derived from breaker trip time, credible surge and UPS discharge time; PROCHOT fast throttle measured separately
Overload detection and trigger time	PSU over-current detection plus management-unit trigger generation, including PSU-to-PSU spread	Often combined with settling time as a single end-to-end SLA
Recovery time	Time to return to the original performance state after the capping condition is removed	As short as the design allows; bounded by acceptable customer performance impact

IV. Negative Test Scenarios

Meeting the four SLAs on a healthy, quiescent platform proves very little. Capping earns trust only when it holds through the events that actually occur in a live facility, so the framework stresses the feature with negative tests that mirror real operations rather than ideal laboratory conditions. Table II lists the stimulus applied and the observables recorded for each scenario group.

A. Online Insertion and Removal of Field-Replaceable Units

Online Insertion and Removal (OIR) of field-replaceable units (FRUs) is routine maintenance, and capping must remain active and correct throughout it without creating an outage at the cell level. Each replaceable module class is exercised with capping asserted, and the behavior is recorded against the SLAs defined above.

1) AC/DC feeds: Feeds drop from the PDUs to the rack and the server and are usually redundant, so they go out for regular maintenance. Live insertion and removal of a feed

is tested with capping active to confirm the limit is preserved through the transition.

2) Power supply units: Front-end AC-DC or DC-DC PSUs are the most common FRU, are typically deployed N+1 or N+N with load sharing and are serviced regularly. Because the PSUs are also the elements responsible for overload detection, OIR testing on them with capping asserted is the most critical case in this group.

3) Management units: Rack and node management units are field-replaceable in several designs. Removing the element that carries the capping control path is precisely the case where silent loss of the feature is most likely, so it is exercised explicitly.

4) Add-in cards and drives: PCIe network cards and storage drives are replaceable in most servers, and their insertion or removal is expected to produce no power spike that disturbs the applied limit.

5) Fans: Fan modules are hot-serviceable at node or rack level in several designs. Fans are an inductive load and can generate sudden power excursions on insertion, which makes them a useful stimulus for the capping loop as well as a maintenance case.

6) Whole nodes: Some implementations support OIR of the complete node. Because the node is plugged into live AC or DC power, this case is evaluated for its effect on rack-level capping rather than node-level behavior alone.

B. Power-Cycle and Reboot Persistence

Any of the modules above may be AC or DC cycled individually, and the user may trigger a cold or warm reboot at any time. The design must guarantee that limits applied to those modules survive the cycle and the reboot, and testing verifies persistence rather than accepting a re-application by the higher tier as equivalent.

C. Firmware Upgrades and VM Churn

Every module and many of the subcomponents inside them — MCU firmware, BIOS, FPGA and CPLD images, voltage regulators — require online firmware upgrade. Some upgrades apply in-system and others take effect only after a cycle or reboot, and capping is expected to hold across both. Separately, most data-center servers host multiple virtual machines that share the underlying resources and

are inserted and removed dynamically, so VM addition and deletion is tested as its own churn case.

D. Fault Injection

Capping depends on software running on MCUs and CPUs, and that software can hang for many reasons. Those conditions are emulated deliberately and the effect on capping is measured. The suite also injects hardware and interface faults: a PSU that stops delivering output, and telemetry interfaces such as PMBus, SMBus, I2C, or IPMI that stall and stop answering requests. These are the cases that separate an implementation which fails safe from one which fails silently.

E. Injected Power Spikes

Sudden power bursts are introduced, and the throttling response is observed; the burst should be absorbed within a short interval. Bursts can be generated by additional CPU or GPU workload, or by raising the activity of IO components that sit outside direct processor control. Manually stepping fan speed up and down is a convenient and repeatable way to produce the latter.

In some of these scenarios the capping feature will be impacted, and that is a legitimate outcome rather than a test failure to be hidden. Every such case is tabulated for the operations team — the scenario, the observed impact, the duration of the impact, and the recovery behavior — so that data-center IT teams can manage and control operations knowingly during capping events.

TABLE II. NEGATIVE-TEST MATRIX: STIMULUS APPLIED AND OBSERVABLES RECORDED

Scenario group	Stimulus applied	Observables recorded
OIR of field-replaceable units	Live insertion and removal of AC/DC feeds, PSUs, management units, add-in cards and drives, fans, and whole nodes	Retention of the applied limit, deviation from limit, duration of deviation, recovery behavior
Power cycle and reboot	AC and DC cycle of individual modules; cold and warm reboot	Persistence of the limit across the cycle, time to re-establish the limit
Firmware	In-system and	Continuity of

upgrade and VM churn	reboot-required updates to MCU firmware, BIOS, FPGA/CPLD and VRs; dynamic VM addition and deletion	capping through the update, deviation during churn, recovery time
Fault injection	Hung monitoring software, faulty PSU output, stalled PMBus, I2C, SMBus or IPMI telemetry	Fault detection, fail-safe behavior, duration of any loss of capping, alerting
Injected power spikes	CPU and GPU burst workloads, elevated IO activity, forced fan-speed steps	Overload detection and trigger time, settling time, peak excursion above limit, recovery time

V. Validated Design Practices

The negative tests do more than grade an implementation; they point at the design choices that decide whether the SLAs can be met at all. The practices below are the ones our testing experience consistently rewards.

A. Move Overload Detection into Hardware

Overload detection and trigger time is the metric under the most pressure, and it is usually served by a mix of hardware and software. When power-monitoring logic and interrupt processing run as threads in kernel and user space, the response acquires both additional latency and, more damagingly, variance that depends on whatever else the MCU or CPU is running. Two remedies for work. The first dedicates a microcontroller with minimal software and operating-system overhead to continuous power monitoring and downstream trigger generation. The second places that monitoring and triggering generation in a small programmable device — a CPLD or FPGA — inside the management unit, which additionally covers the fault case in which the software stack hangs.

B. Parallelize and Group PSU Telemetry

Distribution units deliver power through several PSUs sharing load in parallel, and their telemetry usually arrives over PMBus or I2C. When those interfaces are combined onto a single master port, power and current collection become sequential, which lengthens polling time and adds

variation. Providing parallel I2C interfaces from the MCU helps, and a CPLD can run several I2C conversations concurrently, subject to the IO pin budget that forces some grouping in practice. Independently, the group PMBus command lets a single command on the shared bus start sensing on every PSU simultaneously, after which the values are read individually. That reduces the transaction count and, more importantly, captures data from all PSUs at the same instant.

C. Tune Sampling for Sensor Accuracy

Hardware and software design must select and configure the correct ADC sampling and averaging interval for each sensor, because the reported value drives both the polling cadence of the management unit and the achievable power-limit accuracy. An averaging window chosen for noise rejection alone will quietly widen the accuracy band that the platform can honor.

D. Add Tighter Control Knobs for IO Devices

Node-level limits are applied directly to CPU cores and, to a degree, to memory bandwidth, while the remaining IO interfaces are controlled loosely or only indirectly. Activity on those devices then shows up as system-level power variation that the capping loop cannot fully suppress. Three knobs close much of that gap: constraining memory bandwidth from the CPU while the limit is in force; invoking PCIe L1 ASPM on storage drives when the capping event is triggered, with exit latencies characterized because sleep states affect recovery time; and driving the PWRBRK# signal to PCIe add-in cards so the card itself participates in limiting power [4].

E. Expose Debug Hooks and Timestamps

Beyond ordinary functional logging, firmware should provide hooks that make the negative tests measurable and make a missed metric diagnosable. The CPU power management unit should emit status messages and, where possible, signal event start and end on a GPIO that can be probed externally to measure event duration directly. Software should timestamp the logs of every process and thread on the critical path, since a metric that cannot be attributed to a stage cannot be improved.

VI. Power Savings and Carbon Footprint

The payoff of validated capping is direct and structural. Every watt of headroom that capping makes safe to use is a watt of compute that requires no additional PDU, no additional rack, and no additional row. That avoided equipment is avoided embodied carbon, manufactured and shipped and installed only if the headroom cannot be trusted, and the higher utilization of the electrical plant that already exists lowers operational carbon per unit of compute at the same time. The savings are therefore counted twice, once in the facility that is not built and once in the energy that is not wasted holding capacity in reserve.

This is why validation, rather than capping itself, is the load-bearing contribution. Oversubscription is only as aggressive as the confidence an operator has in the safety net beneath it, and that confidence is bounded by evidence. As GPU demand surges and the electrical envelope tightens further, rigorously validated power capping is what makes aggressive oversubscription safe to deploy at scale.

VII. Conclusion

Data centers can reclaim the power stranded by hierarchical provisioning, and the techniques to do so are well understood. What has been missing is a disciplined way to demonstrate that the safety net holds when the reclaimed headroom is called upon. This paper has set out that method: four measurable SLAs — power-limit accuracy, settling time, overload detection and trigger time, and recovery time — measured across the full arc of a capping event including fast throttle; a negative-test suite drawn from the maintenance, upgrade, fault, and burst conditions that real facilities produce, with every impact tabulated by duration and recovery for the operations team; and the design practices those tests reward, from hardware-resident overload detection through telemetry parallelism, sensor tuning, tighter IO knobs, and debug instrumentation.

Applied together, these give operators headroom they can use with confidence and a lower carbon footprint per unit of compute. You can only cap what you can trust, and you can only trust what you have tested.

Acknowledgement

We sincerely thank Kartik Ananth for his invaluable support in reviewing this document.

References

- [1] Flex: High-Availability Datacenters with Zero Reserved Power.
- [2] Precise Power Capping for Latency-Sensitive Applications in Datacenter.
- [3] Understanding Type B MCB (Miniature Circuit Breaker). [Online]. Available: <https://igoyeenergy.com/understanding-type-b-mcb-miniature-circuit-breaker/>
- [4] Open Compute Project, Project Olympus Power and Management Distribution Unit v1.2. [Online]. Available: <https://www.opencompute.org/documents/project-olympus-power-and-management-distribution-unit-v1-2-pdf/>