Title

Quantifying Interconnect Energy Efficiency on Perlmutter: pJ/bit Measurements of NVLink, PCIe, Slingshot NICs, and Rosetta Switches

Authors

Zhao, Zhengji
Williams, Samuel
Antepara, Oscar
et al.

Publication Date

2026-04-28

Peer reviewed

Quantifying Interconnect Energy Efficiency on Perlmutter: pJ/bit Measurements of NVLink, PCIe, Slingshot NICs, and Rosetta Switches

Zhengji Zhao, Samuel Williams, Oscar Antepara, Leonid Oliker, Brian Austin, Nicholas J. Wright

Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
{ZZhao, SWWilliams, OAntepara, LOliker, BAustin, NJWright}@lbl.gov

***Abstract*—In the exascale era, comprehensive energy accounting is critical for sustainable HPC. Standard monitoring captures CPU and GPU power, but the energy footprint of interconnects, including switches, NICs, and PCIe/NVLinks, remains hidden due to limited hardware sensors. We address this with a software-centric methodology using targeted microbenchmarks on the Perlmutter (HPE Cray EX) system at NERSC. By correlating controlled stress on specific network components with job- and rack-level power telemetry, we isolate each component's energy consumption, quantifying dynamic energy per bit (pJ/bit) for active communication while separating constant-power overhead. Integrating network bandwidth measurements from vendor tools such as NVIDIA DCGM and CrayPat enables fine-grained, application-level estimates of network energy use, unattainable with standard monitoring. This approach establishes a generalizable framework for evaluating network energy, providing actionable insights for designing and operating power-efficient, sustainable supercomputing systems.**

***Index Terms*—Network energy, pJ/bit, PCIe, NVLINK, Slingshot-12 NIC, Rosetta Switches, NVIDIA A100 GPU**

I. Introduction

In the exascale era, energy consumption is a primary constraint for High-Performance Computing (HPC) sustainability. Therefore, optimizing the use of every watt becomes essential to sustaining the future of supercomputing. While modern architectures like the HPE Cray EX provide high-resolution telemetry for CPUs, GPUs, and memory, the energy footprint of high-speed interconnects—such as Slingshot NICs, Rosetta switches, and PCIe/NVLinks—remains largely opaque. Due to a lack of dedicated hardware sensors, network energy is often either hidden within aggregate node totals or excluded from job-level accounting entirely. As a result, the network's energy footprint has remained largely hidden or neglected, and is frequently excluded from energy optimization strategies and sustainability analyses.

To address this limitation, we develop a software-centric methodology based on a suite of carefully designed microbenchmarks that generate controlled and communication patterns. We evaluate this methodology on Perlmutter, an HPE Cray EX system based on NVIDIA A100 GPUs at the National Energy Research Scientific Computing Center (NERSC). By correlating specific communication intensities with observed changes in job-level and rack-level power measurements, we systematically decouple and characterize the power overhead associated with individual network components. Our results demonstrate that this approach effectively exposes the previously obscured energy consumption of interconnect, quantifying energy per bit for each network component. By incorporating network bandwidth measurements from complementary vendor tools such as NVIDIA DCGM and CrayPat, our methodology further provides fine-grained estimates of application-level network energy consumption that are not achievable with standard monitoring alone.

This work establishes a robust and generalizable framework for evaluating network energy consumption on Cray systems and platforms with similar telemetry capabilities, providing actionable insights to support the design and operation of power-optimized, environmentally sustainable supercomputing systems.

The remainder of this paper is organized as follows. Section II reviews related work and situates our contributions. Section III describes the system configuration, experimental setup, and our software-centric benchmark strategies, and Section IV presents the experimental designs for component-wise evaluation of network energy efficiency. Section V presents the results, including pJ/bit measurements for network components and application-level network energy estimates. Section VI provides further discussion, and Section VII concludes the paper.

II. Related Work

As the exascale era arrives, the energy cost of data movement has emerged as a primary bottleneck, fulfilling early predictions that communication power would eventually overshadow floating-point computations [1], [2]. Although modern compute nodes still exhibit substantial idle power, their processing units (CPUs and GPUs) employ sophisticated dynamic power management, such as dynamic voltage and frequency scaling (DVFS) and power capping, to continuously modulate energy consumption based on workload demands. In contrast, high-performance interconnects lack this flexibility. To maintain ultra-low latency and peak bandwidth readiness, they act as "always-on" links that draw near-maximum power regardless of actual data traffic [3], [4]. Because of this rigidity, the interconnection network's share of the overall

high-performance computing (HPC) energy budget has grown significantly [4]–[6].

To combat this growing energy footprint, considerable effort has been directed toward hardware and topological innovations. At the physical layer, researchers have explored replacing power-hungry electrical links with co-packaged silicon photonics to dramatically lower the energy-per-bit cost of data movement [7]. Architecturally, network energy proportionality has been pursued through novel, low-diameter topologies. Designs such as the flattened butterfly [8] and the Slim Fly topology [9] aim to approach theoretical optimal network diameters, inherently reducing the physical number of active routers, transceivers, and necessary network hops without sacrificing latency.

Alongside hardware and topological improvements, significant research has focused on protocols, communication libraries, and runtime tuning. While in-depth performance studies of modern interconnects like HPE Slingshot [10] exist to comprehensively characterize routing and congestion under heavy workloads, explicit energy optimization efforts have largely relied on dynamic link power management [6]. These strategies attempt to transition idle network components—such as InfiniBand links or Network-on-Chip (NoC) routers—into low-power sleep states during idle cycles [11]. Furthermore, software-hardware co-design efforts have attempted to shape MPI communication traffic to reduce burstiness [3], while frameworks like PerfBound [4] aim to save energy by dynamically managing these link states based on runtime performance [4]. However, practical implementations of these dynamic routing and sleep-state strategies in production HPC environments remain limited due to the severe latency penalties associated with waking up links. Consequently, as modern intra-node fabrics (like NVLink) and cluster interconnects face unprecedented bandwidth demands, mitigating network energy usage in practice remains an urgent challenge.

Crucially, optimizing network energy requires accurate, component-level measurement. While modern telemetry captures CPU, GPU, and switch power, it fails to isolate the energy footprint of individual network components [12]. This lack of empirical data forces a reliance on vendor specifications—such as 1.93 pJ/bit for PCIe Gen4 PHYs [13], 8 pJ/bit for early NVLink [14], and 250 W for the 64-port Rosetta switch [10]—which may diverge from real-world behavior.

This work presents the first methodology and benchmark suite for measuring component-wise network energy efficiency (pJ/bit) for NVLink, PCIe, and Slingshot NICs on a production Cray EX system. Unlike broader system-level monitoring, our approach enables job-level energy decomposition by pairing targeted benchmarks with existing HPE Cray power telemetry. Notably, we incorporate Rosetta switch power data, a resource that has rarely been utilized in prior studies [15]–[18]. The resulting framework provides a generalizable approach for quantifying interconnect efficiency, highlighting neglected areas for network optimization to help guide the design and operation of power-constrained exascale systems.

III. System Configuration and Experiment Setup

A. Perlmutter System Configuration

This work was performed on the Perlmutter [19] supercomputer at NERSC. Perlmutter is a heterogeneous system that integrates 1,792 GPU-accelerated nodes and 3,072 CPU-only nodes within a single HPE Cray EX system. Each GPU-accelerated node contains one AMD EPYC 7763 "Milan" processor, 256 GB of DDR4 memory, four NVIDIA A100 Tensor Core GPUs, and four HPE Slingshot "Cassini" Network Interface Cards (NICs). Each A100 GPU has peak performance of 9.7 TFLOPS for FP64 vector operations, 19.5 TFLOPS for FP64 tensor operations, respectively, 40 GB of HBM2 memory, and peak memory bandwidth of 1.5 TB/s. The thermal design power (TDP) of an A100 GPU is 400 W.

Each Perlmutter GPU rack (or cabinet) contains 128 AMD Milan host nodes with NVIDIA A100 accelerators, interconnected via an all-to-all dragonfly topology. More information about Perlmutter is available online [19].

B. Power and Energy Measurement

NERSC's Operations Monitoring and Notification Infrastructure (OMNI) [20] gathers and manages operational data from across the NERSC data center. The data includes power measurements from sensors distributed throughout the system. The Cray Power Monitoring (PM) interface [21] runs on all compute nodes and provides access to power measurements for key components of the node (the CPU, each GPU, and the DDR memory), and the total node power (which includes the aforementioned components plus peripherals such as NICs). Cray power telemetry is also available each Rosetta switches of the Slingshot interconnect on the system. These measurements are forwarded to OMNI's data store using the Lightweight Distributed Metric Service (LDMS) [22]. LDMS samples the measurements at one-second intervals.

The node and its component power were obtained from OMNI using the query scripts [23] or Grafana queries. Power usage was also monitored in real time through Grafana dashboards during the benchmarking runs. The total energy consumption was computed as the product of the average power and the application runtime.

C. Benchmarking Strategies

Isolating the relatively small power contribution of network components within a high-power compute node is challenging. The total node power is large and exhibits significant variability, which can easily obscure the subtle dynamic signals associated with PCIe, NVLink, and NIC activities. Moreover, node power measurements include PCIe, NVLink, and NIC power but exclude the energy consumed by the Slingshot fabric and switches. Our experimental design uses a differential measurement approach combined with strict control mechanisms to accurately quantify network power.

1) Differential Measurement: Network component power is measured as the difference between an *active* state, where the component is fully utilized by a micro-benchmark, and a *baseline* state, where it is minimally used while other node components remain active. This subtraction removes the large, shared baseline power, isolating the dynamic power attributable to the component under test.

2) Mitigating Manufacturing Hardware Variability: Hardware variations can introduce node-to-node and device-to-device power differences [15], often larger than network signals. Active and baseline measurements were conducted on the same node, and when two nodes were needed, their roles were swapped to reduce residual variability. Tests were repeated on additional nodes to confirm robustness and generality. The same procedure is applied to GPUs.

3) Microbenchmark Design and Execution: In addition to using standard benchmarks such as the OSU Microbenchmarks [24], we designed micro-benchmarks to isolate and stress each network component to its peak practical bandwidth [25]. To mitigate short-duration transients and ensure stable measurements, benchmarks ran for a sustained period, repeatedly executing the core communication operation until power consumption stabilized. Pinned (page-locked) host memory was used to maximize CPU-to-device transfer bandwidth, and large transfers (on the order of gigabytes) were employed to bypass cache interference and achieve full bandwidth saturation. Additionally, to improve measurement accuracy, we execute multiple identical instances of these microbenchmarks across multiple GPUs or nodes, with appropriate process and thread pinning to amplify network power usage where applicable.

4) Energy-Per-Bit Modeling: Consistent with prior work [26]–[28], we assume dynamic energy scales linearly with data volume. We model this using a constant rate, e `(pJ/bit)`, multiplied by the data volume, where e is derived from measured dynamic power and bandwidth (detailed in the next section).

5) Evaluation Precision: The precision of the dynamic power measurement ($\Delta P = \bar{P} - \bar{P}_{\texttt{idle}}$) was evaluated using standard error propagation principles [29], [30]. Because the absolute measurements were taken using the same instrument (e.g., Cray power monitoring counters) under identical conditions, the systematic error (instrument bias) remains constant and is entirely canceled out during the differential calculation. Consequently, the standard uncertainty of ΔP depends solely on the standard error of the mean (SEM) of the averaged measurements, denoted as $\sigma_{\bar{P}}$. For independent random variables, variances are additive under subtraction; therefore, the combined standard uncertainty can be calculated in quadrature as:

$$\sigma_{\Delta P} = \sqrt{\sigma_{\bar{P}}^2 + \sigma_{\bar{P}_{\texttt{idle}}}^2} \tag{1}$$

6) Evaluating Network Power Usage for Applications: On Perlmutter, GPU activity metrics are collected via NVIDIA DCGM, including the volume of data sent and received through network devices. By combining these measured network activities, such as `nvlink_{rx,tx}_bytes` and `pcie_{rx,tx}_bytes`, with the evaluated network power per bit, we can accurately estimate the dynamic power consumed by NVLinks and PCIe and, in turn, the energy used at the job level. CrayPat measures data volumes traversing the Slingshot NICs and Rosetta switches, which can be multiplied by the corresponding energy-per-bit values to estimate NIC and switch energy consumption.

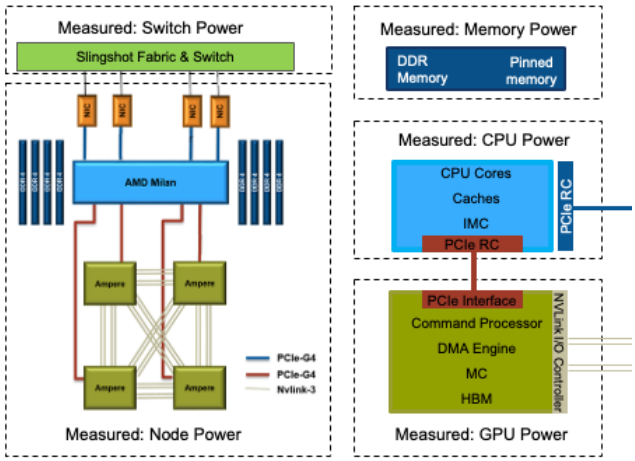


Fig. 1: Power measurement domains available on Perlmutter. Each dashed box denotes a distinct power telemetry domain (switch, node, memory, CPU, and GPU), and lists the hardware components included in that measurement.

IV. Experimental Design for Network Energy Evaluation

On Perlmutter, an HPE Cray EX system, the power consumption of each node and its major components, CPU, DDR memory, and four GPUs, are directly measured using Cray power monitoring sensors. Figure 1 summarizes the available power measurement domains and the hardware components included in each. Network component power, however, is not exposed as an independent node-level measurement and may be aggregated within other reported domains. Therefore, we design targeted benchmarks to quantify its contribution. Additionally, while switch power is measured directly, switches are typically shared across jobs, requiring further methodology for application-level attribution.

A. Evaluating Energy Cost of NVLink

Because NVLink accounts for only a small fraction of total node power, isolating its dynamic energy consumption requires a precise differential analysis. To achieve this, we execute two `cudaMemcpyAsync` benchmarks on a single Perlmutter node: a peer-to-peer (P2P) data transfer between two GPUs via NVLink, and a device-to-device (D2D) copy within a single GPU (illustrated in Figure 2). We utilize large data payloads (≥ 4 GB) to ensure a high signal-to-noise ratio during measurement.

During these transfers, the total dynamic GPU power can be logically decomposed into the energy consumed by the hardware endpoints (memory and DMA engines) and the interconnect itself. The measured dynamic power of the D2D test, $P^{(\text{GPU})}_{\text{dyn, D2D}}$, captures the baseline power required for HBM reads/writes and internal DMA operations without activating the NVLink. Because dynamic memory power scales linearly with data throughput, we project the D2D baseline power to match the bandwidth achieved during the P2P transfer (β_{P2P}):

$$P_{\text{data\_scaled}} = P^{(\text{GPU})}_{\text{dyn, D2D}} \times \frac{\beta_{\text{P2P}}}{\beta_{\text{D2D}}} \tag{2}$$

With the baseline power of the D2D test scaled to match the P2P test, the measured dynamic GPU power of the P2P transfer, $P^{(\text{GPUs})}_{\text{dyn, P2P}}$ (the combined power of both the sender and receiver GPUs), is expressed as the sum of this scaled memory cost and the NVLink overhead:

$$P^{(\text{GPUs})}_{\text{dyn, P2P}} = P_{\text{data\_scaled}} + P_{\text{NVLink}} \tag{3}$$

Substituting Equation 2 into Equation 3 and rearranging the terms allows us to isolate the pure dynamic power of the NVLink interconnect:

$$P_{\text{NVLink}} = P^{(\text{GPUs})}_{\text{dyn, P2P}} - P^{(\text{GPU})}_{\text{dyn, D2D}} \times \frac{\beta_{\text{P2P}}}{\beta_{\text{D2D}}} \tag{4}$$

Finally, we calculate the NVLink energy efficiency, e_{NVLink}, in picojoules per bit (pJ/bit). We divide the isolated NVLink power, P_{NVLink} (in Watts, or J/s), by the measured P2P bandwidth, β_{P2P} (in GB/s). Converting Joules to picojoules (a factor of 10^{12}) and Gigabytes to bits (a factor of 8×10^9) introduces a simplified scaling multiplier of $10^3/8$, yielding our final calculation:

$$e_{\text{NVLink}}\ [\text{pJ/bit}] = \frac{P_{\text{NVLink}}}{\beta_{\text{P2P}} \times 8} \times 10^3 \tag{5}$$

B. Evaluating Energy Cost of PCIe Link (CPU – GPU)

To isolate the dynamic energy of the PCIe link, we first measure the total CPU and GPU power during a Host-to-Device (H2D) data transfer (illustrated in Figure 3). We then subtract the power overheads of the GPU memory endpoints, the host integrated memory controller (IMC), and the DMA engines, which we determine through auxiliary experiments by measuring their power in isolation and scaling it to match the H2D transfer bandwidth.

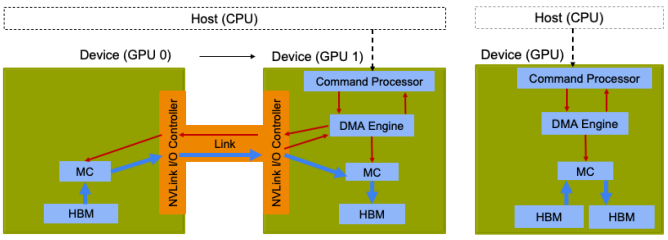
Our experimental phases are defined as follows:

- **Primary H2D Transfer:** The core experiment pushes large data volumes ($D \geq 4$ GB) using `cudaMemcpyAsync` from pinned host memory to the device (Figure 3). The measured total dynamic transfer power across the CPU ($P^{(\text{CPU})}_{\text{dyn,H2D}}$) and GPU ($P^{(\text{GPU})}_{\text{dyn,H2D}}$) encompasses the PCIe link, the host IMC, the GPU HBM write operations, the GPU memory controllers (MC), and the DMA engines. Additionally, we measure the achieved PCIe bandwidth (β_{PCIe}).
- **Auxiliary 1: GPU HBM-Write Baseline:** To characterize the device memory overhead, a custom CUDA kernel executes continuous write-only operations to the GPU's High Bandwidth Memory (HBM). This saturates the device memory controllers without activating the PCIe interface, yielding the maximum dynamic HBM write power ($P^{(\text{GPU})}_{\text{dyn,HBM\_write}}$) at bandwidth ($\beta_{\text{HBM}}$).
- **Auxiliary 2: Host RAM Read:** To characterize the host memory overhead, a CPU thread continuously reads a multi-gigabyte array from host memory. This forces data through the Integrated Memory Controller (IMC), the Infinity Fabric, and the CPU cores, measuring $P^{(\text{CPU})}_{\text{dyn,RAM\_read}}$ at bandwidth β_{RAM}.
- **Auxiliary 3: Host Cache Read:** To isolate pure CPU execution, the exact same read instructions from Auxiliary 2 are executed, but targeting a 1 MB array that fits entirely within the L2/L3 cache. This isolates the CPU core execution power ($P^{(\text{CPU})}_{\text{dyn,Cache\_read}}$) at bandwidth β_{Cache}, keeping the IMC and Infinity Fabric completely idle.

To extract the PCIe link power, we subtract the dynamic power of the endpoints from the total dynamic power observed during the H2D transfer. Because dynamic memory power scales linearly with data bandwidth, we first isolate the dynamic HBM power from Auxiliary 1 and scale it down to match the bandwidth of the PCIe transfer:

$$P_{\text{HBM\_scaled}} = P^{(\text{GPU})}_{\text{dyn,HBM\_write}} \times \frac{\beta_{\text{PCIe}}}{\beta_{\text{HBM}}} \tag{6}$$

Second, to isolate the power overhead of the DMA engine, we analyze the dynamic power difference between the



(a) P2P (NVLink) data transfer (b) D2D copy

Fig. 2: This figure shows the data flow (blue) and control flow (red) for peer-to-peer (P2P) data transfers (left) over NVLinks and device-to-device (D2D) copies (right) across internal fabrics using `cudaMemcpyAsync`.

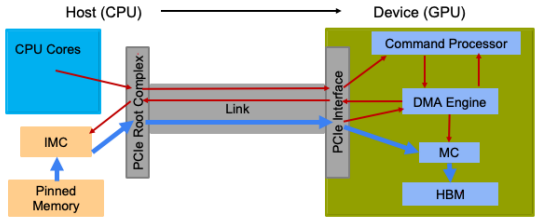


Fig. 3: Host-to-Device (H2D) data transfers across the PCIe link using `cudaMemcpyAsync`. Blue arrows indicate data flow; red arrows indicate control flow.

two GPUs (the sender and the receiver) during the Peer-to-Peer (P2P) transfer detailed in Section IV-A. Assuming the HBM read/write operations and the NVLink transmit/receive physical layer costs are largely symmetrical, the residual dynamic power difference between the sending and receiving devices provides an estimation of the asymmetric DMA engine overhead at the P2P bandwidth (β_{P2P}). We define this baseline DMA power as:

$$P_{\text{DMA, P2P}} = \left| P^{(\text{GPU})}_{\text{dyn, sender}} - P^{(\text{GPU})}_{\text{dyn, receiver}} \right| \tag{7}$$

Because the DMA engine's dynamic power scales linearly with data throughput, we project this overhead down to match the bandwidth achieved during the PCIe Host-to-Device (H2D) transfer (β_{PCIe}). The scaled DMA overhead is calculated as:

$$P_{\text{DMA\_scaled}} = P_{\text{DMA, P2P}} \times \frac{\beta_{\text{PCIe}}}{\beta_{\text{P2P}}} \tag{8}$$

Third, we isolate the CPU IMC power using a Cache-Hit Subtraction strategy based on Auxiliaries 2 and 3. On modern architectures like AMD EPYC, the IMC and the PCIe Root Complex share the same SoC power domain. To isolate the IMC, we project the dynamic power of the CPU cores performing pure execution (via the Cache Read) by scaling it to match the bandwidth of the RAM read phase (β_{RAM}). This scaled core power is calculated as:

$$P_{\text{core\_scaled}} = P^{(\text{CPU})}_{\text{dyn,Cache\_read}} \times \frac{\beta_{\text{RAM}}}{\beta_{\text{Cache}}} \tag{9}$$

We then subtract the projected core execution power from the total RAM read power to isolate the dynamic uncore (IMC and Infinity Fabric) power, scaling it down to the PCIe bandwidth:

$$P_{\text{dyn,IMC}} = P^{(\text{CPU})}_{\text{dyn,RAM\_read}} - P_{\text{core\_scaled}} \tag{10}$$

$$P_{\text{IMC\_scaled}} = P_{\text{dyn,IMC}} \times \frac{\beta_{\text{PCIe}}}{\beta_{\text{RAM}}} \tag{11}$$

With the endpoint memory subsystem and DMA power cleanly isolated, the total dynamic power of the PCIe link—encompassing the PCIe Root Complex, the motherboard physical routing, and the GPU PCIe PHY—is calculated by subtracting these scaled overhead costs from the primary H2D power:

$$\begin{aligned} P_{\text{PCIe,Link}} = {} & P^{(\text{CPU})}_{\text{dyn,H2D}} - P_{\text{IMC\_scaled}} \\ & + P^{(\text{GPU})}_{\text{dyn,H2D}} - P_{\text{HBM\_scaled}} - P_{\text{DMA\_scaled}} \end{aligned} \tag{12}$$

Finally, we calculate the PCIe link energy efficiency ($e_{\text{PCIe,Link}}$) in pJ/bit. This is derived by dividing the isolated PCIe dynamic power, $P_{\text{PCIe,Link}}$ (in Watts, or J/s), by the measured PCIe bandwidth, β_{PCIe} (in GB/s). Applying the appropriate unit conversions yields:

$$e_{\text{PCIe,Link}}\ [\text{pJ/bit}] = \frac{P_{\text{PCIe,Link}}}{\beta_{\text{PCIe}} \times 8} \times 10^3 \tag{13}$$

C. Evaluating Energy Cost of Casini NIC

This section isolates the dynamic energy cost of the NIC (e.g., Cassini) and its dedicated host PCIe interface. Because the NIC and its host PCIe link operate in lockstep during DMA transfers, we evaluate this CPU-to-NIC interconnected subsystem as a single cohesive unit, denoted as NIC+PCIe.

We measure the total power consumption and bandwidth during a bulk data transfer using the OSU Microbenchmarks (`osu_bw`) between two nodes (illustrated in Figure 4). Because `osu_bw` is a unidirectional test, we focus our telemetry strictly on the active sender (Node 1) to accurately capture the data injection cost.

To extract the pure NIC+PCIe hardware power, we must isolate the power of the data payload moving through the system and subtract the overheads of the memory subsystem. We achieve this in three steps: factoring out the CPU core software overhead, removing the physical DRAM power, and subtracting the CPU Integrated Memory Controller (IMC) power.

First, to capture the software overhead, we execute an **MPI Polling Baseline** without data movement. During an `osu_bw` DMA transfer, the MPI library keeps the CPU cores in a continuous busy-wait state, polling the network completion queues. By taking the difference in total node power between the active `osu_bw` transfer ($P^{(\text{Node1})}_{\text{BW}}$) and this MPI Polling Baseline ($P^{(\text{Node1})}_{\text{MPI\_poll}}$), we cleanly isolate the node-level dynamic power dedicated strictly to moving the payload ($P^{(\text{Node1})}_{\text{payload}}$), naturally canceling out the busy-wait CPU core overhead without relying on estimated CPU RAPL registers:

$$P^{(\text{Node1})}_{\text{payload}} = P^{(\text{Node1})}_{\text{BW}} - P^{(\text{Node1})}_{\text{MPI\_poll}} \tag{14}$$

Second, we account for the CPU IMC overhead. The IMC resides on the CPU die and is highly active during bulk DMA transfers. Using the isolated dynamic IMC power ($P_{\text{dyn,IMC}}$) derived via the Cache-Hit Subtraction strategy in the previous section, we project this uncore overhead down to match the measured network bandwidth:

$$P_{\text{IMC\_scaled}} = P_{\text{dyn,IMC}} \times \frac{\beta_{\text{BW}}}{\beta_{\text{RAM}}} \tag{15}$$

Finally, with the software overhead already removed in Equation 14, we calculate the isolated power of the NIC+PCIe

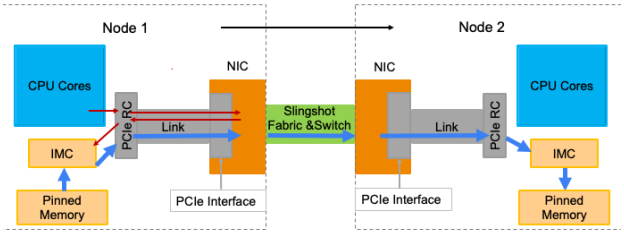


Fig. 4: This figure illustrates the data flow during a point-to-point communication between two nodes (Node 1 → Node 2). The blue arrows indicate the data transfer path, while the red arrows represent the control flow.

hardware stack. We subtract the scaled IMC power and the dynamic physical host memory power ($P^{(\text{Mem1})}_{\text{dyn,BW\_payload}}$) from our node-level payload power:

$$P_{\text{NIC+PCIe}} = P^{(\text{Node1})}_{\text{payload}} - P^{(\text{Mem1})}_{\text{dyn,BW\_payload}} - P_{\text{IMC\_scaled}} \tag{16}$$

By substituting power (P) and bandwidth (β), we calculate the combined hardware energy efficiency (pJ/bit) for the NIC and its dedicated PCIe link:

$$e_{\text{NIC+PCIe}}\ [\text{pJ/bit}] = \frac{P_{\text{NIC+PCIe}}}{\beta_{\text{BW}} \times 8} \times 10^3 \tag{17}$$

D. Evaluating Energy Cost of Rosetta Switches

The HPE Cray EX Slingshot network employs a 3-hop dragonfly topology. Each GPU rack comprises eight chassis, each with eight compute and four switch blades. Figure 5 illustrates the interconnect between nodes on a single chassis. Every compute blade hosts two GPU-accelerated nodes, each equipped with four NICs. The 32 switch blades per rack form two 16-switch dragonfly groups. Network connectivity is provided by rear-mounted switch blades, and all switches within a group are fully interconnected.

Figure 5 also illustrates the network's three physical link levels: copper L0 (Host) links connecting nodes to switches, copper L1 (Group) links connecting switches within a group, and optical L2 (Global) links interconnecting switch groups. Because inter-node traffic must traverse different combinations of these physical cables, we establish our own custom experimental terminologies to represent these routing costs. Specifically, we define three distinct **communication paths**—labeled the **L0, L1, and L2 paths** (Figure 6)—and experimentally determine the aggregate energy per bit associated with traversing each one.

On Perlmutter, aggregate switch power is captured via Cray telemetry. Because the Slingshot interconnect routes traffic across three distinct link levels, we design a series of targeted experiments to isolate the dynamic power cost of each tier. We execute the OSU Microbenchmarks MPI All-to-All (`osu_alltoall`) among selected nodes within a single 128-node rack to progressively exercise these links. To bypass Cray MPICH's intra-node shared-memory optimizations, we assign exactly one MPI task per node, forcing all communication out to the network fabric. Furthermore, we fix the peer-to-peer message size to ensure the data volume per path remains constant across all scaling tests.

To isolate the tiers, we capture the aggregate dynamic switch power across three progressive configurations. First, we isolate intra-chassis traffic by running an *Intra-Chassis All-to-All* among 16 nodes within a single chassis. To amplify the power signal, we run this simultaneously across all 8 chassis in the rack, capturing the total dynamic power ($P_{\text{dyn,L0}}$) and aggregate bandwidth (β_{L0}). Second, we expand to an *Intra-Group All-to-All* across the 4 chassis (64 nodes) comprising a single switch group, exercising both L0 and L1 links. We run this concurrently on both switch groups within the rack, capturing $P_{\text{dyn,L0+L1}}$ and $\beta_{\text{L0+L1}}$. Finally, we run an *Inter-Group All-to-All* across all 128 nodes in the rack, exercising all three link levels simultaneously to capture $P_{\text{dyn,L0+L1+L2}}$ and $\beta_{\text{L0+L1+L2}}$.

By correlating the dynamic power and achieved bandwidth during each progressive experiment with the exact number of active paths at each tier, we can isolate the energy per bit for each link level. Assigning one MPI task per node yields the exact combinatorial path counts (N) for each link level within a single cabinet, as detailed in Table I.

Variable	Formula	Calculated Value
N_{L0}	$8 \times \binom{16}{2}$	960
$N_{\text{L0+L1}}$	$2 \times \binom{64}{2}$	4032
N_{L1}	$2 \times \binom{64}{2} - N_{\text{L0}}$	3072
$N_{\text{L0+L1+L2}}$	$\binom{128}{2}$	8128
N_{L2}	$\binom{128}{2} - N_{\text{L0}} - N_{\text{L1}}$	4096

TABLE I: Calculation of Network Paths in a Perlmutter Rack

Using the measured dynamic power and bandwidth (in GB/s), the resulting energy per bit for the pure L0 path is calculated as:

$$e_{\text{L0}}\ [\text{pJ/bit}] = \frac{10^3}{8} \times \frac{P_{\text{dyn,L0}}}{\beta_{\text{L0}}} \tag{18}$$

To derive the L1 efficiency, we isolate the L1 energy by subtracting the L0 contribution from the combined intra-group test. Assuming uniform bandwidth distribution, the ratio of data transferred through each path type corresponds directly to the ratio of active paths ($D_{\text{L0}}/D_{\text{L1}} = N_{\text{L0}}/N_{\text{L1}}$). Substituting the scaled dynamic power and bandwidths yields:

$$e_{\text{L1}}\ [\text{pJ/bit}] = \frac{10^3}{8}\left(\frac{P_{\text{dyn,L0+L1}}}{\beta_{\text{L0+L1}}} \times \frac{N_{\text{L0+L1}}}{N_{\text{L1}}} - \frac{P_{\text{dyn,L0}}}{\beta_{\text{L0}}} \times \frac{N_{\text{L0}}}{N_{\text{L1}}}\right) \tag{19}$$

Following the same logic, we isolate the L2 optical links by subtracting the scaled L0+L1 power from the full rack all-to-all measurement:

$$e_{\text{L2}}\ [\text{pJ/bit}] = \frac{10^3}{8}\left(\frac{P_{\text{dyn,L0+L1+L2}}}{\beta_{\text{L0+L1+L2}}} \times \frac{N_{\text{L0+L1+L2}}}{N_{\text{L2}}} - \frac{P_{\text{dyn,L0+L1}}}{\beta_{\text{L0+L1}}} \times \frac{N_{\text{L0+L1}}}{N_{\text{L2}}}\right) \tag{20}$$

V. Results

A. Isolation of Network Power Usage

We demonstrate our methodology by isolating the power consumption of NVLink, PCIe, Cassini NIC, and the Slingshot interconnect. Figure 7 shows node- and component-level power when running our microbenchmarks targeting NVLink, PCIe, and NIC. Differences in power across these tests, caused by communication workloads, allow the extraction of energy attributable solely to communication via a

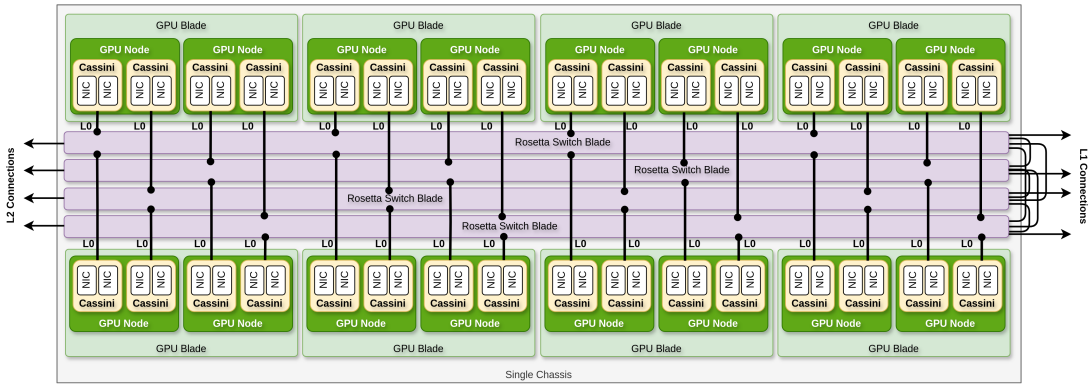


Fig. 5: Diagram of a single GPU compute chassis [19].

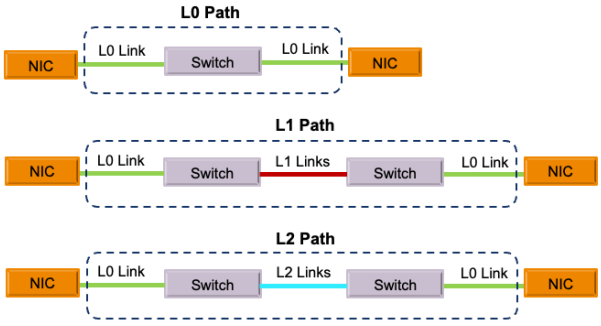


Fig. 6: Data transmission paths via L0, L1, and L2 links between two Perlmutter nodes (assuming minimal paths). We define the L0 path as L0 $\rightarrow$ switch $\rightarrow$ L0; the L1 path as L0 $\rightarrow$ switch $\rightarrow$ L1 $\rightarrow$ switch $\rightarrow$ L0; and the L2 path as L0 $\rightarrow$ switch $\rightarrow$ L2 $\rightarrow$ switch $\rightarrow$ L0. We experimentally determine the energy per bit associated with each of these communication paths.

differential approach, clearly separating the marginal overhead of NVLink, PCIe, and NIC. Figure 8 illustrates the aggregate power consumption of a Rosetta switch within a 128-node Perlmutter cabinet under varying all-to-all traffic conditions. During each link-level test, we evaluated four message sizes: 1 KB, 16 KB, 64 KB, and 1 MB. As depicted, although power consumption varies with message size, the power draw for the 1 KB messages is practically indistinguishable from the idle baseline. Consequently, we excluded the 1 KB results from our analysis, focusing instead on the larger message sizes that produce clear, measurable deviations from the idle state.

Table II displays the power and bandwidth measurements for a single, representative run of these microbenchmarks, highlighting two key observations. First, the data validates our assumption in Section IV-B that the energy consumption for reading from and writing to the HBM is similar. This is evidenced by the comparable GPU power usage recorded during the Host-to-Device (HBM write) and Device-to-Host (HBM read) data transfer experiments (see GPU0 in the H2D and D2H rows). Second, we can estimate the power consumption of the DMA engine by taking the difference between the two GPU power readings during the peer-to-peer (P2P) tests. This yields approximately 36 W at a bandwidth of $\beta_{\text{P2P}} = 93.55$ GB/s, which equates to an energy cost of 48 pJ/bit.

The last two columns of the table report the bandwidths achieved during the various microbenchmarks alongside the theoretical peak bandwidths of the Perlmutter GPU architecture [19]. As shown, these microbenchmarks achieved a high percentage of their respective theoretical peaks. This high utilization was essential for elevating the network's dynamic power consumption, rendering these otherwise subtle signals measurable.

B. pJ/bit evaluation

The pJ/bit for the network components can be obtained using the measured power and bandwidths in each test using Eqs. 5, 13, 17, 18, 19, 20. Table III shows the calculated pJ/bit for the tested network components on Perlmutter. Figure 9 visualize the same data. NVLink and PCIe demonstrated highly stable power profiles, yielding 70.9 pJ/bit and 167.9 pJ/bit, respectively. Isolating the dynamic energy of the NIC+PCIe subsystem was challenging due to the low signal-to-noise ratio in power telemetry and the need to mix node and component power measurements to isolate the NIC power. Because these node and component measurements rely on different types of sensors, inherent measurement discrepancies occur. Consequently, the sum of the component powers can exceed the recorded node power, resulting in negative calculated values for individual runs. To address this, we recorded high-resolution Cray energy counter data across 20 distinct node executions. Averaging these normally distributed measurements mitigates the impact of this sensor variance, yielding a positive mean. The derived average energy efficiency for the active sender NIC+PCIe is 4.1 pJ/bit (standard error: 3.2 pJ/bit). This value reflects the energy footprint of the network hardware when decoupled from CPU and memory overheads.

These numbers are notably higher than those provided in vendor specifications. This discrepancy can likely be attributed to two factors: first, the parameters available in the literature

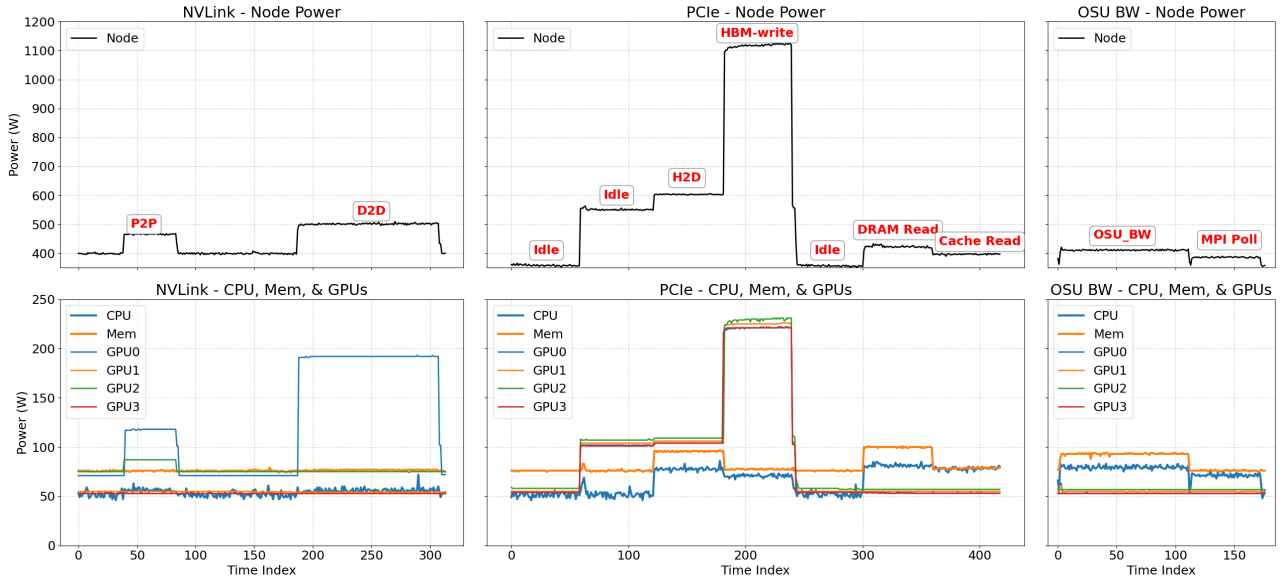


Fig. 7: Node and component-level power usage during experiments designed to measure NVLink, PCIe, and NIC energy efficiency. The top row illustrates the total node power (W) over time, with critical operational phases annotated for each test (e.g., P2P and D2D for NVLink; OSU_BW and MPI Poll for the inter-node point-to-point bandwidth test). For the PCIe test, annotations include H2D, HBM-write, DRAM Read, Cache Read, and three distinct idle phases: an external idle, a GPU-side idle immediately following data array initialization, and a CPU-side idle post-initialization. The bottom row details the corresponding power draw for the CPU, memory, and four individual GPUs (GPU0–GPU3). The x-axis represents the time index. Most tests were executed with four concurrent instances to ensure the resulting power draw was clearly measurable.

may be outdated; and second, the specific hardware components included in prior measurements are rarely explicitly defined. Consequently, direct comparisons between our results and theirs may not be apples-to-apples. In contrast, our methodology explicitly defines these boundaries. For instance, our NVLink energy incorporates both the NVLink PHY and the I/O controllers at both ends. Similarly, our PCIe link energy accounts for the interfaces on both the host and the device. Furthermore, for each link level, we define comprehensive data paths that encompass both the physical links and the intervening switches, as illustrated in Figure 6. Specifically, the L0 path includes one switch and two L0 links; the L1 path includes two switches, two L0 links, and one L1 link; and the L2 path includes two switches, two L0 links, and one L2 link.

The measured dynamic energy cost of 4.1 pJ/bit for the combined PCIe and NIC subsystem is notably low, especially when contrasted with the 167.9 pJ/bit required for PCIe transfers to the GPU. This stark contrast is rooted in two primary architectural behaviors of modern High-Performance Computing (HPC) nodes.

First, the GPU PCIe data path incurs a massive "Active Tax." Waking the NVIDIA A100 GPU's command processor, memory fabric, and PCIe PHY from a deep idle state to an active state (P0) requires approximately 50 W of uncore power. In contrast, the host-to-NIC data path entirely bypasses the GPU. The data flows directly from the host DRAM, through the highly efficient CPU Integrated Memory Controller (IMC) and PCIe Root Complex, and into the NIC's DMA engine, avoiding the massive GPU uncore wakeup penalty entirely.

Second, modern high-speed interconnects such as the Cassini NIC exhibit a heavily static power profile. Maintaining the physical link state—powering the optical transceivers, SerDes circuits, and internal ASIC routing at 200 Gbps—requires substantial continuous power. This high baseline power is completely captured in the node's idle power ($P_{\text{idle}}^{(\text{Node1})}$). Consequently, the *dynamic* cost—the marginal power strictly required to inject payload bits through an already-active PCIe Root Complex and network link—is highly efficient. Therefore, the 4.1 pJ/bit result accurately isolates the pure physical dynamic cost of data movement across the host PCIe wires and the NIC transmission engines, devoid of static link-maintenance overheads or unrelated accelerator wake-up taxes.

C. Network Energy Attribution in Applications

By combining the energy-per-bit estimates derived in the previous section with recorded network activity, we can perform a fine-grained attribution of application energy consumption across various hardware components, including network devices. The data traffic traversing the NVLink and PCIe interfaces is captured via always-on DCGM metrics, while the data volume processed by the NICs is measured using the CrayPat tool.

The left panel of Figure 10 presents node-averaged (spatial average) DCGM metrics collected during the MILC Generation execution, specifically highlighting the data volumes

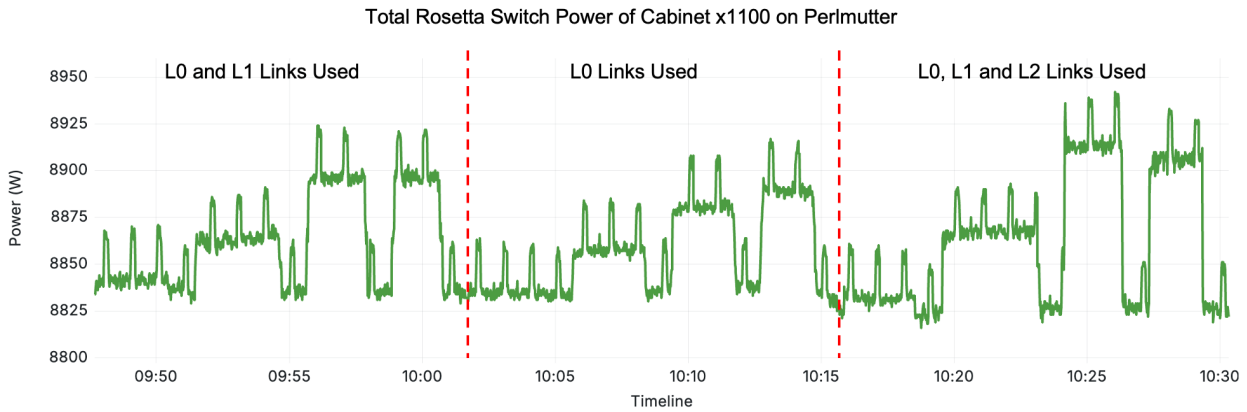


Fig. 8: Total switch power of a Perlmutter GPU cabinet, aggregated across 32 Rosetta switches, during OSU MPI All-to-All (osu_alltoall) benchmarks. The experiment exercises different network link levels, denoted by the three zones separated by red dashed lines. Within each link-level zone, we evaluated message sizes of 1 KB, 16 KB, 64 KB, and 1 MB. Periodic power spikes, which occur approximately every 60 seconds regardless of workload or idle state, were attributed to system noise and filtered from our analysis. The precise source of these spikes remains a subject for future investigation. The baseline idle power, measured when the network is inactive, is approximately 8.8 kW.

Experiment	**Power (W)**							**BW (GB/s)**	**Peak BW (GB/s)**
	Node	**CPU**	**Mem**	**GPU0**	**GPU1**	**GPU2**	**GPU3**		
P2P (GPU0 → GPU2)	102.43	5.73	0.11	68.87	0.00	30.00	0.03	93.55	100
P2P (GPU2 → GPU0)	97.70	4.83	-0.12	31.00	0.00	64.00	-0.03	93.55	100
D2D	137.47	4.95	0.59	143.75	0.00	0.00	-0.03	1383.16	1555
H2D	67.37	8.56	4.93	55.00	0.00	0.00	0.03	24.94	32
D2H	68.31	8.32	5.90	55.50	0.00	0.00	0.05	24.53	32
HBM-Write	136.9	5.9	0.4	166.7	0.0	0.0	0.0	1423.56	1555
RAM Read	11.8	8.0	4.2	0.0	0.0	-0.1	0.0	17.45	204.8
Cache Read	5.8	6.3	0.5	0.1	0.0	-0.1	0.0	18.14	
OSU_BW (Node 1)	13.7	8.4	4.2	0.7	0.0	0.0	0.0	24.27	25
MPI Polling Baseline	5.7	6.2	0.1	-0.2	0.0	-0.1	0.0	-	-
Idle (Node 1)	385.6	52.3	74.4	55.8	53.3	52.2	53.0	-	-
Idle (Node 2)	359.73	52.80	75.06	55.00	55.00	57.00	53.00	-	-

TABLE II: Dynamic power consumption across system components and achieved bandwidth for various data transfer microbenchmarks. The last two rows show the idle power for each component. The last column reports the peak bandwidth for the NVLink, HBM, PCIe, NIC, and DDR memory exercised during the P2P, D2D, H2D/D2H, NIC, and DRAM Copy experiments, respectively.

transmitted and received via NVLink (NVLTX/NVLRX) and PCIe (PCITX/PCIRX). Integrating these volumes over time yields the total data transferred across each interface. By multiplying these totals by our measured energy efficiencies (in pJ/bit), we isolate the energy consumed by NVLink and PCIe during the job. To estimate the Slingshot network energy consumption, we use CrayPat to measure the total data volume processed by the NIC.

Due to current instrumentation limitations, we cannot track the exact distribution of messages across specific link-level paths at runtime. Therefore, we estimate the combined energy consumption of the Slingshot fabric and Rosetta switches by multiplying the total NIC data volume by the average energy cost (pJ/bit) across all link levels. This methodology is applied to calculate both dynamic and total energy per bit.

In typical production environments, jobs share network switches, making it difficult to decouple isolated switch energy from the broader fabric energy at job levels, although the individual switch energy is directly measurable. However, for the MILC Generation benchmark, we utilized all 128 nodes within a full cabinet. This exclusive access ensured the switches were not shared with competing workloads (assuming minimal routing paths), rendering the switch power directly measurable (as shown in the right panel of Figure 10). Integrating this directly measured switch power over time provides the precise total switch energy for the MILC job.

Table IV details this energy attribution across various hardware components for select NERSC-10 benchmark applications. Examining intra-node communication, the data reveal that while the energy consumed by NVLink is largely negli-

	NVLink	PCIe	Cassini NIC+PCIe	Rosetta Switch		
				L0 path	L1 path	L2 path
	(pJ/bit)	(pJ/bit)	(pJ/bit)	(pJ/bit)	(pJ/bit)	(pJ/bit)
Dynamic	70.9 ± 1.0	167.9 ± 10.5	4.1 ± 3.2	3.9 ± 0.4	5.9 ± 0.3	5.2 ± 1.6
Total (Dynamic + Idle)	-	-	-	596.2 ± 42.7	796.1 ± 170.6	482.5 ± 105.1

TABLE III: Interconnect energy efficiency. This table presents the dynamic and total energy per bit for individual network components and end-to-end communication paths, including the standard error across multiple runs. Total energy per bit is provided for reference where applicable (e.g., for the Slingshot interconnect, where measured idle power dominates the total network power). The communication paths, which comprise multiple links and switches, are defined in Figure 6.

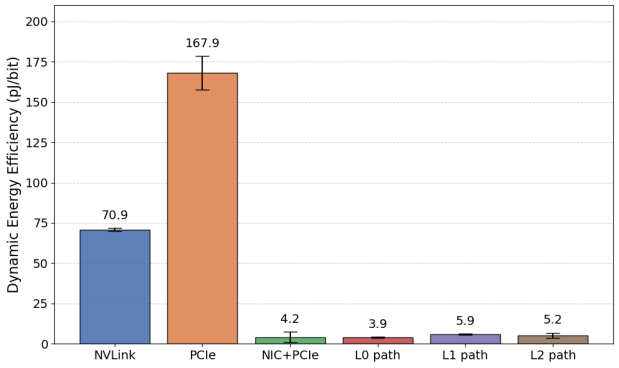


Fig. 9: Dynamic energy per bit for all network components on Perlmutter.

gible, PCIe energy can be substantial; for instance, the MILC code expends approximately 5.8% of its total node energy on PCIe transfers. Turning to inter-node communication, the node-averaged network switch energy constitutes a significant measurable overhead. Furthermore, as shown in Figure 11, switch power—a factor frequently overlooked in prior studies—accounts for approximately 8% of the overall power consumption.

VI. Discussions

The software-centric methodology in this work provides a much-needed lens into the power dynamics of the HPC interconnects. By isolating the dynamic energy per bit from constant power overheads on the Perlmutter system, our findings not only shed light on current hardware inefficiencies but also pave the way for more comprehensive application-level energy accounting.

A. High Idle Power and the Inefficiency of "Always-On" Networks

A major observation from our component-level energy isolation is the disproportionately high idle power consumption of network switches. In current HPC network architectures, interconnects operate under an "always-on" paradigm. To maintain clock synchronization, lane alignment, and link stability, continuous data scrambling and the transmission of idle characters occur in the background - even when no useful application data is being routed [8], [32]. In essence long-sequences of zero-value data or no data still result in the transmission of effectively random sequences of (scrambled) data. Consequently, the PHYs are effectively constantly switching, thus maximizing power draw. Empirical observation suggests this dominates the network energy balance and as a result, the network draws near-peak power regardless of the actual communication volume. This massive constant-power overhead severely limits overall energy efficiency when networks are underutilized.

This high idle power represents a critical area for future energy optimization. To achieve true energy-proportional computing, future network architectures must move away from continuous background scrambling on idle links. Addressing this requires hardware-software co-design: architectures could implement ultra-fast sleep/wake transitions (similar to advanced PCIe L-states or Energy Efficient Ethernet (EEE), but adapted for low-latency HPC fabrics [4], [33]) that allow individual links or entire switch ASICs to dynamically power down during communication pauses. Workload profiling and experimentation could quantify the maximum recovery time (deepest power state) with the minimal impact on workload performance. Additionally, future routing protocols could be made energy-aware, dynamically consolidating traffic to fewer active links and putting unused ports into deep sleep without suffering synchronization penalties upon wake-up. In summary, solving the interconnect power wall calls for more energy-efficient network design that spans hardware, network protocols, communication libraries, and workload-adaptable tuning.

B. Uncovering Hidden Energies: From System Optimization to Holistic Job Accounting

Traditionally, HPC energy accounting has been heavily CPU- and GPU-centric. Lacking fine-grained hardware telemetry for individual interconnects, the energy footprints of NVLink, PCIe interfaces, and NICs have historically been obscured—lumped indiscriminately into overarching node-level power metrics. By successfully isolating the dynamic energy per bit for these discrete pathways, our methodology finally unmasks these hidden costs. Revealing the true physical cost of data movement provides the missing link for accurate energy attribution across the entire system. This newfound visibility serves a dual purpose: it equips HPC network architects with the empirical insights needed to design the next generation of energy-efficient network devices, and it empow-

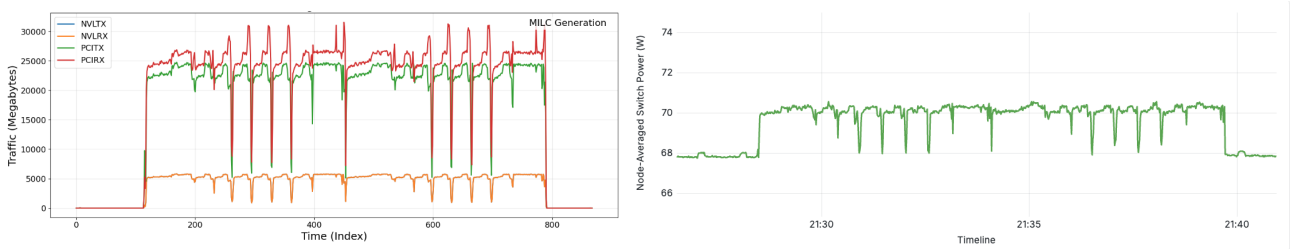


Fig. 10: Node-averaged traffic (left panel) and switch power (right panel) during a full-cabinet MILC generation execution. GPU metrics (NVLTX, NVLRX, PCITX, and PCIRX), which represent data transmitted and received via the NVLink and PCIe interconnects, were collected using NVIDIA DCGM. Switch power data was obtained via Cray power telemetry.

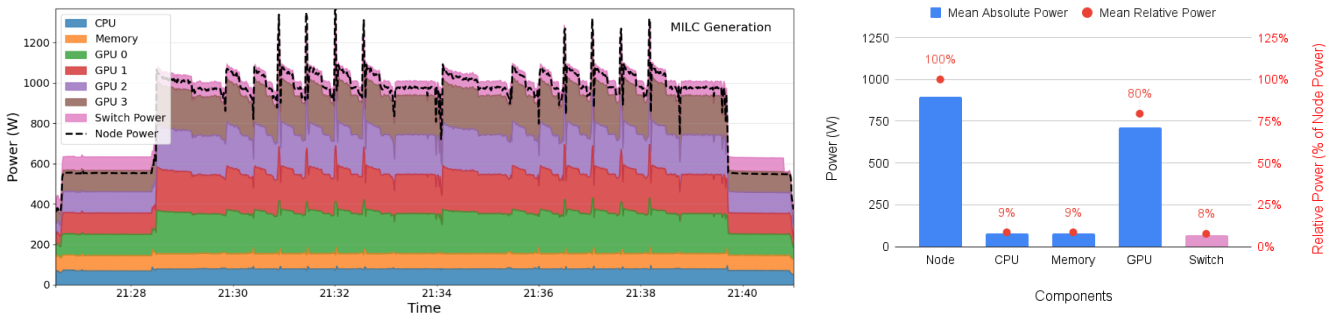


Fig. 11: Average power consumption of a compute node and its individual components (CPU, memory, and GPUs) during the execution of the MILC Generation code on a full 128-node Perlmutter cabinet. The right panel displays the time-averaged mean power for each component, both in absolute terms and relative to the total node power. The per-node switch power—highlighted by the pink stack in the timeline (left panel) and the rightmost pink bar (right panel)—is calculated as the total power of all 32 cabinet switches divided by the 128 nodes.

Application	E_{Node} **(Megajoules)**	**(% of E_{Node})**							**Benchmark Size/ Node Count**
		CPU	**Memory**	**GPU**	**NVLink**	**PCIe**	**Cassini NIC+PCIe**	**Rosetta Switch**	
BGW Epsilon	0.16	9.08	8.37	77.69	0	0.63	0.00	7.21	4x(Medium/32)
BGW Sigma	0.50	6.86	7.13	79.47	0	0.45	0.00	6.68	4x(Medium/32)
LAMMPS	0.37	4.85	5.44	79.77	0.00	0.04	0.00	5.17	Reference/128
MILC Gen	0.74	8.53	8.52	79.54	0.54	5.85	0.00	8.09	Reference/128
MILC Spec	0.32	9.15	9.32	78.91	0.56	5.93	0.00	8.92	Reference/128

TABLE IV: Energy attribution across system components for various HPC applications, expressed as a percentage of average node energy (E_{Node}). Note that E_{Node} includes the energy consumed by PCIe, NVLink, and the NIC, but excludes the Slingshot fabric and switch energies. Values in columns 2–4 and 8 are directly measured via Cray power telemetry, whereas the values in columns 5–7 are derived by multiplying the measured data volumes by their corresponding energy-per-bit metrics. The final column details the benchmark sizes and the number of nodes utilized for each run (for further details, refer to the NERSC-10 Benchmark suite [31]).

ers software developers to precisely profile communication-bound workloads. By making these costs explicit, it guides targeted code optimizations and strongly incentivizes the adoption of energy-aware programming paradigms, such as communication-avoiding algorithms.

Furthermore, this methodology represents a crucial step toward accurate, holistic energy accounting at the job level. Historically, job-level energy metrics have excluded shared infrastructure, like network switches, because their power consumption is difficult to attribute to individual workloads. By systematically profiling the switch power tiers (L0, L1, L2), we demonstrate a viable path to assigning shared network energy costs to specific jobs based on their actual network utilization. While our current study focuses on compute and interconnect hardware—excluding shared facility overheads such as cooling energy, parallel file system I/O, and AC-to-DC conversion losses—the differential scaling methodology presented here can be generalized. Ultimately, bringing these remaining components into the accounting framework will pave the way for complete and precise cost-to-solution metrics

in modern data centers.

C. Revisiting Exascale Predictions: Empirical Realities of Network Energy

The dynamic power of the network represents a major predictive success for exascale hardware scaling. Foundational exascale studies established a strict 20 Megawatt system power cap, dictating an aggressive active data-routing budget of just 2 to 10 pJ/bit [2], [34]. Our dynamic measurements for the Rosetta Switch (4.5 to 10.5 pJ/bit) and the Cassini NIC+PCIe (2.1 pJ/bit) land almost exactly within this target zone. This proves that hardware architects successfully delivered on these early active efficiency goals.

However, the "always-on" static penalty of the network exposes a massive structural blind spot in those early projections. Our data demonstrates that static idle power actually dominates the energy profile. As terabit-scale SerDes circuits and optical transceivers must remain lit continuously, the total energy cost (dynamic plus idle/scrambler) for the switch ranges from 600 to over 1345 pJ/bit. This exorbitant idle tax shatters the theoretical 20-100 pJ/bit (copper) or 1-2 pJ/bit (photonic) exascale system predictions, but are in line with the earlier roadmap based predictions by Shalf et al. [2]. However, these predictions all missed the fact that these energies are omnipresent and scaled by peak bandwidth (not utilized bandwidth) into omnipresent power, highlighting a impediment in achieving true energy-proportional networking.

Perhaps the most surprising divergence from early projections is the severe energy penalty of intra-node links, exposing an unexpected high-speed copper wall. Early models assumed short chip-to-board data movements would be extremely cheap, targeted anywhere from 10 pJ/bit [34] to 1000 pJ/bit [2]. Our measurements show industry made substantial gains attaining anywhere from 50 pJ/bit (NVLink) to 200 pJ/bit (PCIe), but underestimated some of the physical power impediments (e.g., signal equalization, retimers, and Forward Error Correction) required to push terabit-per-second bandwidths over standard copper traces.

Ultimately, these empirical realities offer a profound validation of the international exascale software roadmap [1]. Early software blueprints emphasized that architectures could no longer rely on hardware improvements alone to mask the cost of data movement. Our data provides the definitive proof: with local PCIe transits dynamically costing $\sim$200 pJ/bit and total network transits exceeding 600 pJ/bit, developers simply cannot afford to move data unnecessarily. These results strongly justify the past decade of investment in CPU-GPU integration, advanced signaling technologies, and communication-avoiding algorithms. Furthermore, as recently emphasized by Shalf and colleagues, the persistent challenge of data locality remains the defining bottleneck for the post-exascale era [35]. Our component-level measurements empirically confirm this reality, proving that future progress relies critically on radically energy-aware software design that strictly minimizes data movement.

D. Limitations

While our methodology successfully isolates the dynamic cost of data movement, operating within a restricted user-space environment introduces two primary limitations. First, our approach evaluates only the *dynamic* energy per bit. Accurately profiling or manipulating the static power overheads of individual interconnects requires root access to low-level hardware power states. Therefore, we were unable to evaluate the static energy overhead for NVLink, PCIe, and NICs, except for Rosetta switches. Second, our link-level energy calculations assume minimal path routing. In live production systems, adaptive routing may redirect traffic along non-minimal paths to bypass congestion. Forcing deterministic, minimal path routing requires administrative privileges over the network fabric. Without this root access, our measurements represent a baseline topological assumption, which may slightly underestimate the energy costs of adaptively routed traffic.

VII. Conclusion

Our results demonstrate that this approach effectively reveals previously obscured network energy consumption, enabling the quantification of dynamic energy per bit for each network component. By incorporating network bandwidth measurements from complementary vendor tools such as NVIDIA DCGM and CrayPat, our methodology further provides fine-grained estimates of application-level network energy consumption that are not achievable with standard monitoring alone.

This work establishes a robust and generalizable framework for evaluating network energy efficiency on HPE Cray systems and platforms with similar telemetry capabilities, providing insights for the design and operation of power-optimized, environmentally sustainable supercomputing systems. Moving forward, we plan to extend this methodology to systems with InfiniBand interconnects and emerging generations of NVIDIA and AMD GPUs. This expansion is necessary to demonstrate the framework's broad applicability across diverse network topologies to accurately quantify the shifting energy-per-bit costs on modern accelerator-based architectures.

Acknowledgement

The authors thank Steven Martin at HPE for help obtaining Rosetta switch power from the Cray telemetry data, and thank many NERSC and HPE staff who assisted in making this work possible, especially Jon Stile and Chris Samuel. This work was supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231. This work used the resources of the National Energy Research Scientific Computing Center (NERSC) at the Lawrence Berkeley National Laboratory.

The authors acknowledge the use of Google's Gemini 3.1 Pro for assistance with code generation and editorial suggestions. All AI-assisted content was iteratively guided, reviewed,

and finalized by the authors, who bear full responsibility for the published work.

REFERENCES

[1] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig *et al.*, "The international exascale software project roadmap," *The International Journal of High Performance Computing Applications*, vol. 25, no. 1, pp. 3–60, 2011.

[2] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *International Conference on High Performance Computing for Computational Science (VECPAR 2010)*. Springer Berlin Heidelberg, 2011, pp. 1–25.

[3] T. Hoefler, "Software and hardware techniques for power-efficient hpc networking," *Computing in Science & Engineering*, vol. 12, no. 6, pp. 30–37, 2010.

[4] K. P. Saravanan and P. M. Carpenter, "Perfbound: Conserving energy with bounded overheads in on/off-based hpc interconnects," *IEEE Transactions on Computers*, vol. 67, no. 7, pp. 960–974, 2018.

[5] M. Sánchez de la Rosa, F. J. Andújar, J. Escudero-Sahuquillo, J. L. Sánchez, and F. J. Alfaro-Cortés, "On the power saving in high-speed ethernet-based networks for supercomputers and data centers," *Journal of Systems Architecture*, vol. 176, p. 103786, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762126001049

[6] G. Georgakoudis, N. Jain, T. Ono, K. Inoue, S. Miwa, and A. Bhatele, "Evaluating the impact of energy efficient networks on hpc workloads," in *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, 2019, pp. 301–310.

[7] K. Bergman, J. Shalf, G. Michelogiannakis, S. Rumley, L. Dennison, and M. Ghobadi, "PINE: An energy efficient flexibly interconnected photonic data center architecture for extreme scalability," in *2018 IEEE Optical Interconnects Conference (OI)*, 2018, pp. 25–26.

[8] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 338–347.

[9] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 348–359.

[10] D. De Sensi, S. Di Girolamo, K. H. McMahon, D. Roweth, and T. Hoefler, "An in-depth analysis of the slingshot interconnect," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–14.

[11] F. Triviño, D. Bertozzi, and J. Flich, "A fast algorithm for runtime reconfiguration to maximize the lifetime of nanoscale nocs," in *Proceedings of the 2013 Interconnection Network Architecture: On-Chip, Multi-Chip*, ser. IMA-OCMC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 1–4. [Online]. Available: https://doi.org/10.1145/2482759.2482760

[12] L. G. León-Vega, N. Tosato, and S. Cozzini, "A comprehensive analysis of process energy consumption on multi-socket systems with gpus," in *High Performance Computing*, G. Guerrero, J. San Martín, E. Meneses, C. J. Barrios Hernández, C. Osthoff, and J. M. Monsalve Diaz, Eds. Cham: Springer Nature Switzerland, 2025, pp. 52–67.

[13] W. Bae, S.-Y. Cho, and D.-K. Jeong, "A 1.93-pj/bit PCI express gen4 phy transmitter with on-chip supply regulators in 28 nm cmos," *Electronics*, vol. 10, no. 1, p. 68, 2021.

[14] AleksandarK, "Nvidia is preparing co-packaged photonics for NVLink," *TechPowerUp*, 2020. [Online]. Available: https://www.techpowerup.com/276139/nvidia-is-preparing-co-packaged-photonics-for-nvlink

[15] Z. Zhao, B. Austin, E. Rrapaj, and N. J. Wright, "Understanding VASP power profiles on NVIDIA A100 GPUs," in *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, USA, 2024, pp. 1496–1505.

[16] F. Acun, Z. Zhao, B. Austin, A. K. Coskun, and N. J. Wright, "Analysis of power consumption and gpu power capping for milc," in *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024, pp. 1856–1861.

[17] E. Rrapaj, S. Bhalachandra, Z. Zhao, B. Austin, H. Nam, and N. J. Wright, "Power consumption trends in supercomputers: A study of NERSC's cori and Perlmutter machines," in *ISC High Performance 2024 Research Paper Proceedings*, 2024.

[18] Z. Zhao, E. Rrapaj, S. Bhalachandra, B. Austin, H. A. Nam, and N. Wright, "Power analysis of nersc production workloads," in *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023)*. Denver, CO, USA: ACM, 2023, p. 9.

[19] "Perlmutter, a HPE Cray EX system," 2023. [Online]. Available: https://docs.nersc.gov/systems/perlmutter/architecture/

[20] E. Bautista, M. Romanus, T. Davis, C. Whitney, and T. Kubaska, "Collecting, monitoring, and analyzing facility and systems data at the national energy research scientific computing center," in *Proceedings of the 48th International Conference on Parallel Processing: Workshops*, 2019, p. 10.

[21] S. J. Martin and M. Kappel, "Cray xc30 power monitoring and management," in *Proceedings of Cray User Group Meeting*, Lugano, Switzerland, 2014.

[22] A. Agelastos *et al.*, "The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 154–165.

[23] S. Bhalachandra, "Perlmutter OMNI Power Analysis Scripts," 2023. [Online]. Available: https://gitlab.com/NERSC/perlmutter-omni-analysis/

[24] The MVAPICH Project, Network-Based Computing Laboratory, The Ohio State University, "OSU Micro-Benchmarks (OMB) Suite, Version 7.5.1," http://mvapich.cse.ohio-state.edu/benchmarks/, 2025.

[25] Z. Zhao, "Network Energy Evaluation," https://gitlab.com/NERSC/network_energy_evaluation.git, 2024.

[26] X. Zhang, X. Qin, and J. Sun, "A high-level energy consumption model for heterogeneous data centers," in *Proceedings of the 2nd International Workshop on Energy-Aware Software Engineering and Architecture*, ser. EASEA '13, 2013, pp. 27–32.

[27] Y.-J. Huang, Y.-L. Liu, and J.-L. Hsu, "Prediction of overall energy consumption of data centers in different locations," *Sustainability*, vol. 14, no. 9, p. 5629, 2022.

[28] D. Tiwari and S. S. Vazhkudai, "Reducing data movement costs using energy-efficient, active computation on SSD," in *Proceedings of the 4th USENIX Workshop on Hot Topics in Power Management (HotPower)*. Berkeley, CA, USA: USENIX Association, 2012.

[29] J. R. Taylor, *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*, 2nd ed. Sausalito, Calif.: University Science Books, 1997.

[30] JCGM, "Evaluation of measurement data — Guide to the expression of uncertainty in measurement (GUM 1995 with minor corrections)," Joint Committee for Guides in Metrology, Report JCGM 100:2008, 2008.

[31] National Energy Research Scientific Computing Center (NERSC), "N10-benchmarks: NERSC-10 Benchmark Suite," https://gitlab.com/NERSC/N10-benchmarks, 2024, accessed: 2026-03-19.

[32] K. Christensen, P. Reviriego, B. Nordman, M. Shoemaker, M. Bennett, and J. A. Maestro, "Ieee 802.3 az: the road to energy efficient ethernet," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 50–56, 2010.

[33] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, "A performance perspective on energy efficient HPC links," in *Proceedings of the 28th ACM International Conference on Supercomputing*, ser. ICS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 313–322. [Online]. Available: https://doi.org/10.1145/2597652.2597671

[34] R. Lucas, J. Ang, K. Bergman, S. Borkar, W. Carlson, L. Carrington, G. Chiu, R. Colwell, W. Dally, J. Dongarra *et al.*, "DOE advanced scientific computing advisory subcommittee (ASCAC) report: Top ten exascale research challenges," USDOE Office of Science (SC), Advanced Scientific Computing Research (ASCR), Tech. Rep., Feb 2014. [Online]. Available: https://www.osti.gov/biblio/1222713

[35] D. Unat, A. Dubey, E. Jeannot, and J. Shalf, "The persistent challenge of data locality in the post-exascale era," *Computing in Science & Engineering*, vol. 27, no. 4, pp. 19–27, 2025.