Nf-PEAK: Process-Based Energy Attribution for Nextflow Workflows on Kubernetes Clusters

Philipp Thamm
Humboldt-Universität zu Berlin
Berlin, Germany
thammphx@hu-berlin.de

Somayeh Mohammadi
Freie Universität Berlin
Berlin, Germany
mohamadi.s@gmail.com

Kathleen West
University of Glasgow
Glasgow, United Kingdom
Kathleen.West@glasgow.ac.uk

Knut Reinert
Freie Universität Berlin
Berlin, Germany
knut.reinert@fu-berlin.de

Lauritz Thamsen
University of Glasgow
Glasgow, United Kingdom
lauritz.thamsen@glasgow.ac.uk

Ulf Leser
Humboldt-Universität zu Berlin
Berlin, Germany
ulf.leser@hu-berlin.de

***Abstract*—Scientific workflows are pipelines of interdependent tasks. They are increasingly executed on shared Kubernetes clusters via workflow engines such as Nextflow. Their energy consumption matters for both cost and sustainability. It is necessary to examine and optimize workflow tasks individually, because they can be very heterogeneous. However, estimating task-level energy on clusters is difficult: Intel RAPL counters report only node-level energy, access to counters and host process information is typically restricted, and concurrent workloads introduce resource contention and measurement noise.**

We present Nf-PEAK, a containerized method to attribute CPU-package and DRAM energy to individual *processes* and Nextflow *tasks*. Nf-PEAK (i) identifies workflow pods, (ii) maps pods to host processes via cgroup metadata, (iii) samples RAPL and per-process performance counters, and (iv) applies a non-linear energy-credit model before aggregating results at task level. On a Kubernetes cluster, we evaluate three nf-core workflows under controlled co-located CPU load. Nf-PEAK reaches an average Mean Absolute Percentage Error of 6.6% in isolated runs and 10.9% when an unrelated workload saturates 8 of 32 hardware threads per node, and remains stable across 2, 3, 4, and 8 nodes. Compared to the state-of-the-art Kubernetes tool Kepler, Nf-PEAK yields lower error on average, particularly under co-located load.

***Index Terms*—scientific workflows, infrastructure profiling, energy measurement, cluster computing, cloud computing, sustainable computing**

I. Introduction

Extracting information from a dataset can require multiple interdependent processing steps. Such pipelines of tasks are referred to as *scientific workflows* [17] and are used in many areas, such as genomics [9] or remote sensing [20]. Workflow configurations are commonly optimized for fast execution, but recently energy-efficient computing has received more attention due to environmental concerns and rising energy costs [4], [7], [11]. For example, one run of the workflow Sarek[1] for genome variant detection consumes more than 1.16 MJ of energy across 4 cluster nodes over a runtime of 56 minutes in our experiments with 10.6 GB of real-world input data. Of this energy, about 0.7 MJ is attributable to the workflow itself after subtracting static energy consumption of the cluster and overhead introduced by the Kubernetes scheduler, equivalent to drawing an average power of 208.3 W over the workflow's runtime.

Optimizing a workflow for energy efficiency requires feedback beyond node-level totals: workflow developers and users need to understand *which tasks* dominate energy consumption and how configuration changes affect them. A common approach to obtain energy information on Intel systems is to use Running Average Power Limit (RAPL) energy counters [6], [15], which provide low-overhead energy readings for CPU and (depending on the CPU model) DRAM. While access to RAPL is straightforward on local hardware through tools such as perf[2] or powercap[3], extracting accurate *task-level* information on managed clusters is challenging for several reasons:

- **Limited access:** User privileges are often limited, making it impossible to install supporting tools on nodes or to access RAPL counters and host-wide process information.
- **Containerization:** Cluster workloads are commonly containerized, which restricts access to RAPL and to processes running outside the current container.
- **Resource manager:** Workloads are scheduled on the cluster by a resource manager sitting between workflow engine and nodes, preventing knowledge about the location of individual tasks prior to execution.
- **Distributed execution:** Workflows are executed across multiple nodes, so measurement must be coordinated and aggregated.
- **Complex node architecture:** Multi-socket CPUs and NUMA memory require socket-aware monitoring and attribution.
- **Shared environment:** Multiple workloads may run on

[1]https://nf-co.re/sarek/3.5.1/, last accessed: March 22, 2026

[2]https://perfwiki.github.io/main/, last accessed: March 22, 2026

[3]https://docs.kernel.org/power/powercap/powercap.html, last accessed: March 22, 2026

the same nodes concurrently, but RAPL only counts at CPU package level, aggregating energy across unrelated workloads.

In this paper, we present **Nf-PEAK** (**P**rocess-based **E**nergy **A**ttribution on **K**ubernetes clusters targeting **N**ext**f**low workflows), a method to estimate the energy consumption of individual processes and workflow tasks on Kubernetes clusters. In summary, Nf-PEAK combines RAPL measurements with per-process performance counters and uses a non-linear attribution strategy inspired by EnergAt [14]. The tool is fully containerized and can be deployed through Kubernetes without the need to install special software on cluster nodes.

Our contributions are:

- **Containerized task attribution on Kubernetes:** a practical pipeline to discover Nextflow tasks, map them to host processes, and monitor CPU and DRAM energy on all involved nodes.
- **Process-level non-linear attribution:** an energy-credit model adapted to process granularity, enabling accurate CPU and DRAM energy attribution for heterogeneous tasks and concurrent cluster workloads.
- **Evaluation on real workflows:** experiments with three nf-core workflows across four cluster sizes (2, 3, 4, and 8 nodes) and under controlled co-located CPU load, including a comparison to Kepler [1], [2].

In the following, we first provide background about the technologies used in our work. In Section III, we present how Nf-PEAK attributes energy consumption and integrates with Kubernetes. Our experiments and their results are presented in Section IV and discussed in Section V. Section VI presents related work in the area of software-based energy attribution. Finally, Section VII concludes the paper.

II. Background

This section summarizes the technologies and concepts most relevant to our work: Executing scientific workflows on Kubernetes, RAPL energy counters, and energy attribution strategies.

A. Scientific Workflows on Kubernetes

Kubernetes [5] is a widely used container orchestration system that coordinates the execution of workloads, including scientific workflows, on compute clusters. It implements scheduling and re-scheduling of tasks on a distributed infrastructure, and also supports automatic resource allocation and load balancing.

Workloads are executed on the nodes through containers, packaging programs together with their dependencies and execution environment. These containers are running in isolation on top of the node's operating system's kernel. The containers utilized by Kubernetes are commonly built and managed by Docker [19]. They are organized in Kubernetes pods. Each pod contains one or multiple containers and is scheduled independently.

Scientific workflows decompose an analysis into separate logical tasks with explicit data dependencies [12]. Nextflow [8] expresses logical tasks in a DSL and orchestrates their execution in the form of physical tasks on a variety of backends, including Kubernetes. Here, each logical task is split into one or multiple physical tasks, depending on the task and workflow configuration. In a Kubernetes backend, Nextflow executes each physical workflow task in an isolated pod/container (e.g., Docker), using Kubernetes to orchestrate execution on a compute cluster [21] and to manage the available resources.

B. RAPL Energy Counters

Intel RAPL exposes energy counters for different domains (depending on CPU model), such as *Package* (CPU package) and *DRAM* [15]. RAPL readings are low-overhead and widely used for power/energy estimation [13]. However, RAPL counters are *coarse-grained*: each CPU provides only aggregate energy for all workloads on that CPU and directly associated memory. In shared clusters, this makes it impossible to directly infer task-level energy from RAPL alone. Furthermore, energy counters are bounded registers and overflow after a certain amount of counted energy, requiring periodic sampling to compute correct deltas over longer durations. On our hardware, the energy counted before an overflow is 262,143.3 J for CPUs and 65,713 J for DRAM.

C. Energy Attribution and EnergAt

Energy attribution methods allocate measured energy to specific workloads, at the level of containers, processes, or threads, based on resource usage. A common distinction is between *static* energy (idle baseline) and *dynamic* energy induced by workload execution. While dynamic energy can be attributed based on activity, attributing static energy depends on the usage scenario: according to the guidelines stipulated in the Greenhouse Gas Protocol[4], static energy should be apportioned among concurrent workloads proportionally to their resource usage in shared clusters.

EnergAt [14] is a NUMA-aware energy attribution method that uses *energy credits* derived from CPU-time fractions and a non-linear scaling factor to model the power–utilization relationship. Nf-PEAK adapts this idea to a Kubernetes setting and to *process-level* monitoring, enabling task-level aggregation for workflow engines.

III. Method

This section describes Nf-PEAK, our approach for task-level energy attribution for scientific workflows executed on compute clusters.

A. Approach Overview

Scientific workflows commonly contain multiple tasks that can be executed in parallel. Additionally, unrelated workloads are often run simultaneously on the same hardware in shared clusters. To determine the energy consumption of individual workflow tasks, Nf-PEAK performs energy attribution. An overview of Nf-PEAK is presented in Figure 1. Using three

[4]https://ghgprotocol.org/sites/default/files/2023-03/GHGP-ICTSG%20-%20ALL%20Chapters.pdf, last accessed: March 22, 2026

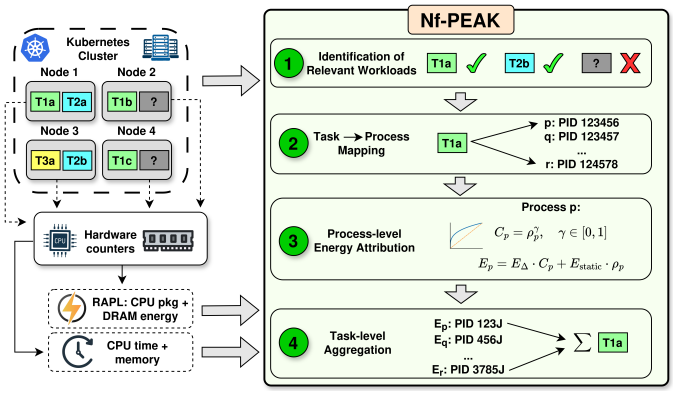


Figure 1. Overview of energy attribution with Nf-PEAK. Using pod information from Kubernetes, node-level energy data and process-level CPU time and memory statistics, Nf-PEAK attributes task-level energy in four steps.

types of input data, including information about currently running pods, node-level energy data and process-level CPU time and memory statistics, Nf-PEAK attributes task-level energy in four steps. During a workflow run, Nf-PEAK (i) identifies task pods and extracts metadata (task name, pod UID), (ii) maps each pod UID to the corresponding host PIDs via process metadata, (iii) attributes per-process energy using a non-linear credit model, and (iv) aggregates process-level energy to produce task-level results.

B. Energy Attribution

We compute energy attributions for each individual workflow task by calculating the sum of the energy attributed to each of its processes. To attribute energy to an individual process, we use the following approach:

Static and dynamic energy. Let $E^D_{\text{total},s}$ be the total energy in domain $D \in \{\text{CPU,DRAM}\}$ measured by RAPL on socket s during a sampling interval of length T. Nf-PEAK estimates static (idle) power $P^D_{\text{static},s}$ by sampling RAPL at some point before workflow execution while the node is idle and computes static energy during an interval as:

$$E^D_{\text{static},s} = P^D_{\text{static},s} \cdot T \tag{1}$$

Dynamic energy is then:

$$E^D_{\Delta,s} = E^D_{\text{total},s} - E^D_{\text{static},s} \tag{2}$$

CPU energy. For a process a, we estimate its CPU-time share on socket s within the interval as $\rho_{a,s} = T^{\text{CPU}}_{a,s} / T^{\text{CPU}}_{\text{total},s}$, where $T^{\text{CPU}}_{\text{total},s}$ is the total CPU time accrued by all processes on s. Note that unlike EnergAt [14], we are directly monitoring processes instead of individual hardware threads and compute energy credits based on CPU time share of processes. We compute an *energy credit* as:

$$C^{\text{CPU}}_{a,s} = \rho^\gamma_{a,s}, \quad \gamma \in [0,1] \tag{3}$$

where γ is a hyperparameter modeling the non-linear power–utilization relationship (e.g., due to DVFS and other hardware effects). CPU energy is attributed as:

$$E^{\text{CPU}}_a = \sum_{s \in S} \left(E^{\text{CPU}}_{\Delta,s} \cdot C^{\text{CPU}}_{a,s} + E^{\text{CPU}}_{\text{static},s} \cdot \rho_{a,s} \right) \tag{4}$$

Memory energy. To attribute energy to a process based on its memory usage, we use the same approach as for the energy used by the CPU, but leveraging the DRAM domain of RAPL and the fraction of the total memory belonging to the process instead. For a process a, we estimate its memory share on socket s within the interval as $\sigma_{a,s} = M_{a,s} / M_{\text{total},s}$, where $M_{\text{total},s}$ is the total memory space accrued by all processes on s. An energy credit is then again computed as:

$$C^{\text{M}}_{a,s} = \sigma^\gamma_{a,s}, \quad \gamma \in [0,1] \tag{5}$$

similar to the credit for CPU time. Memory energy is attributed as:

$$E^{\text{M}}_a = \sum_{s \in S} \left(E^{\text{M}}_{\Delta,s} \cdot C^{\text{M}}_{a,s} + E^{\text{M}}_{\text{static},s} \cdot \sigma_{a,s} \right) \tag{6}$$

Total process energy. The final energy attributed to a process is the sum of the energy attributed to CPU package and memory:

$$E_a = E^{\text{CPU}}_a + E^{\text{M}}_a \tag{7}$$

C. Hyperparameters

Nf-PEAK has three main parameters: the non-linearity exponent γ, the polling interval for detecting new tasks/processes, and the RAPL sampling interval.

The non-linearity exponent γ controls how strongly resource shares are transformed into energy credits and thereby captures the non-linear relationship between power and CPU time/memory utilization. In our experiments, we set $\gamma = 0.3$. This value was calibrated on the target CPUs by running a representative synthetic workflow and attributing energy using Nf-PEAK with $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The workflow contains a series of tasks executed sequentially that use `stress-ng` to introduce varying load on the CPU, ranging from completely idle to full load on all cores in steps of 10% CPU load. We evaluated attribution quality using the MAPE defined in Section IV-C, which was lowest for $\gamma = 0.3$. To avoid overfitting γ to the synthetic workflow used for calibration, we did not conduct additional fine-tuning experiments with values around $\gamma = 0.3$ to further optimize γ on this workflow. The synthetic workflow used for calibration is not part of our experimental evaluation. Figure 2 shows the calibration experiment used to select $\gamma = 0.3$.

The polling interval determines how often Nf-PEAK checks for newly started tasks/processes. In our experiments, we poll for new tasks/processes every 5 s. Section IV-D includes an experiment that quantifies the trade-off between shorter polling intervals and increased monitoring overhead.

The RAPL sampling interval determines the temporal granularity at which Nf-PEAK reads the RAPL energy counters. In our experiments, we sample RAPL every 2 s. This RAPL sampling interval was evaluated in the same experiment as the polling interval.

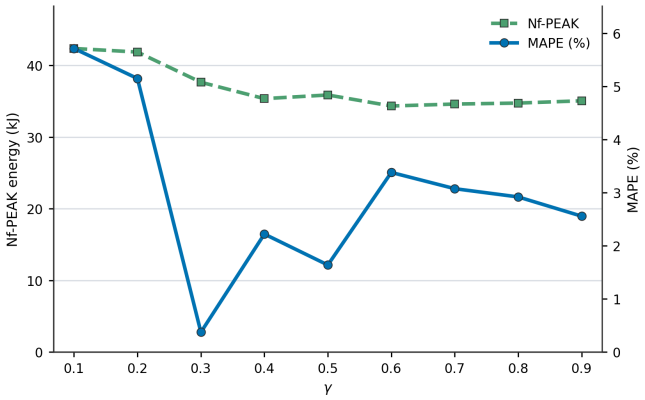


Figure 2. Sensitivity of attributed Nf-PEAK energy and MAPE to the non-linearity exponent γ on the synthetic calibration workflow. The minimum MAPE occurs at $\gamma = 0.3$, which is used for all experiments.

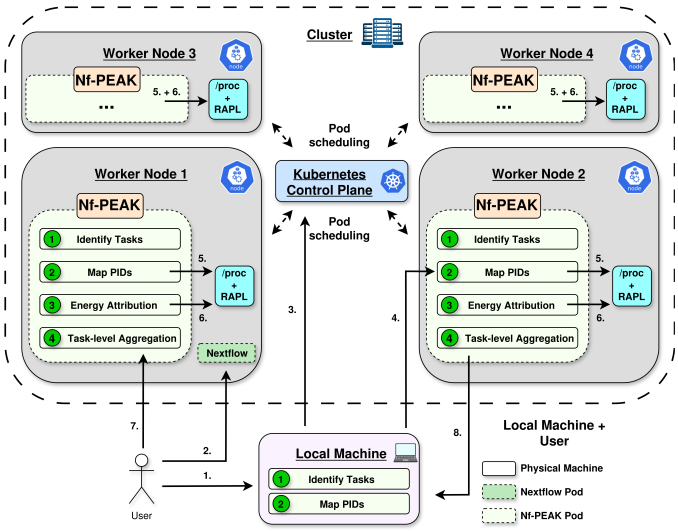


Figure 3. Actors and components of Nf-PEAK and their communication during attribution. Steps are numbered chronologically. Energy monitoring may start any time before workflow execution. Numbered steps include the following actions: 1. User starts energy monitoring. 2. User starts workflow execution. 3. Query currently running pods. 4. Filter associated pod UIDs and send the UID list to the cluster. 5. Map pods to associated processes using `/cgroup`. 6. Attribute energy for newly identified PIDs by reading RAPL counters from `powercap` and process metrics from `/proc`. 7. Consolidate process-level data to task-level using PID→UID mapping. 8. Transfer results for post-processing.

D. Architecture and Deployment

Nf-PEAK is deployed as a set of containers through Kubernetes. The deployment separates a user-side controller from node-local monitoring components. The controller starts and stops monitoring, queries the Kubernetes API for running workflow pods, filters pods that belong to the monitored Nextflow execution, and collects the final result files. The node-local monitoring components run on the cluster nodes where the workflow pods are executed. They map pod UIDs to host PIDs, sample RAPL counters and process metrics, and write process-level attribution data for later aggregation. Figure 3 summarizes the actors and the data flow of Nf-PEAK.

During execution, the controller periodically discovers newly started task pods and forwards their UIDs to the monitoring components on the nodes where the pods are scheduled. Each monitoring component inspects cgroup metadata on its respective node in `/proc` to identify the host PIDs associated with the received pod UIDs. For each discovered PID, per-process CPU time and memory statistics are sampled and combined with RAPL Package and DRAM energy deltas from the same node. After the workflow finishes, the collected PID-level records are mapped to the UID of their physical workflow tasks and aggregated to physical tasks, logical tasks, and workflow-level summaries.

Only the monitoring pods require visibility of all PIDs on the host, realized with `hostPID: true`. Workflow pods remain unmodified and do not require additional privileges, instrumentation, or changes to the Nextflow workflow definition. This separation keeps the monitored workflow portable while limiting access to host-level information to a small set of known monitoring components. The security implications of this deployment model are discussed in Section V.

IV. Evaluation

We evaluate (i) attribution accuracy in isolated executions, (ii) stability across cluster sizes, (iii) robustness under co-located CPU load, and (iv) performance relative to Kepler.

A. Hardware and Software Setup

We use a Kubernetes cluster consisting of 26 nodes in total, of which 8 were available exclusively for our experiments. Eight nodes are sufficient to provide distributed environments for experiments under conditions equivalent to a real-world cluster environment. At the same time, using more than 8 nodes would reduce node-level task parallelism in the tested workflows. Since providing accurate energy attribution for parallel workflow tasks is the primary goal of our work, creating conditions where parallel tasks occur regularly is important for a thorough evaluation. Each node contains an Intel(R) Xeon(R) Silver 4314 CPU (2.40 GHz base, 3.40 GHz max) and supports the RAPL Package and DRAM domains. The cluster uses Kubernetes for orchestration and deploys Docker containers. Workflows are implemented and executed in Nextflow.

B. Workflows

We evaluate three nf-core workflows [18] that cover different task granularities and parallelism patterns. Table I provides a qualitative overview of the workflows and summarizes key execution metrics in our experimental configuration.

RNASeq[5] is a bioinformatics workflow used to analyze RNA sequencing data with a reference genome. RNASeq is characterized by few physical tasks and low task parallelism.

[5]https://nf-co.re/rnaseq/3.18.0/, last accessed: March 22, 2026

Table I

KEY METRICS OF THE WORKFLOWS IN OUR EXPERIMENTS FOR AN EXECUTION ON 2 NODES. CPU UTILIZATION METRICS ARE GIVEN PER TASK. AVGCPU IS THE AVERAGE CPU UTILIZATION OVER ALL INDIVIDUAL TASKS OF THE WORKFLOW, WHILE MAXCPU IS THE MAXIMUM AVERAGE CPU UTILIZATION OF ANY SINGLE TASK. ALL MEMORY AND I/O METRICS ARE TOTALS OVER THE WHOLE WORKFLOW AND ACROSS ALL USED NODES.

Workflow	Goal	Task Len.	Makespan	AvgCPU	MaxCPU	Memory	Read	Write	Phys. Tasks	DoP
RNASeq	RNA sequencing	Long	27 min	850%	1306%	168.2 GB	134.6 GB	85.3 GB	9	2
Sarek	Genome var. calling	Medium	65 min	415%	2274%	1485.5 GB	454.4 GB	141.6 GB	152	24
Rangeland	Land-cover analysis	Short	48 min	190%	1246%	1340.2 GB	95.1 GB	21.3 GB	2665	100

Sarek[6] is another bioinformatics workflow that detects variants on whole genome or targeted sequencing data. Many tasks execute in parallel, causing high sustained CPU load.

Rangeland[7] is a remote sensing workflow that processes satellite imagery to assess land-cover changes. It features numerous very short tasks, representing a worst case for the interval-based monitoring of Nf-PEAK.

C. Experimental Design

We evaluate the accuracy of Nf-PEAK by comparing its task-level attributions (aggregated to workflow level) against node-level RAPL measurements under controlled execution conditions.

For each workflow run, we collect in parallel: (i) node-level energy by sampling the Package and DRAM RAPL counters on all involved nodes, and (ii) task energy attributed by Nf-PEAK. Because RAPL measures energy across all activity on a node, we compare primarily at the workflow level using Mean Absolute Percentage Error (MAPE), detailed below. To make the comparison meaningful, we account for (a) static energy not attributed by Nf-PEAK and (b) platform/measurement overhead.

Unless stated otherwise, we repeat each experiment three times and report means.

We conduct three sets of experiments that target complementary aspects of Nf-PEAK: (i) accuracy in an isolated setting, (ii) stability when increasing cluster size, and (iii) robustness when unrelated workloads run concurrently.

(1) Isolated runs. We execute RNASeq, Sarek, and Rangeland on 2 nodes without additional user workloads. During each run, we measure node-level energy by sampling the Package and DRAM RAPL counters on all involved nodes and, in parallel, run Nf-PEAK to attribute energy to workflow tasks. This isolated setup allows obtaining precise RAPL-based reference values for a direct comparison with Nf-PEAK. By collecting RAPL measurements and Nf-PEAK attributions in parallel on the same runs, we ensure that individual comparisons are not affected by changes in overhead or variance between workflow runs.

(2) Scaling across cluster sizes. We execute the same workflows on 2, 3, 4, and 8 nodes using identical Nextflow configurations. Using the same experimental methodology as in experiment (1), we examine how the attribution accuracy of Nf-PEAK changes for different cluster sizes. Note that it is not sufficient to compare the results produced by Nf-PEAK across cluster sizes, since the differences in available computational resources cause changes in task scheduling and parallelism. Therefore, the same workflow with the same configuration and input data has a different runtime and energy consumption for different cluster sizes.

(3) Co-located CPU load. To emulate co-located workloads, we run `stress-ng`[8] CPU stressors on $n \in \{1, 2, 4, 6, 8\}$ hardware threads on 4 nodes during workflow execution. We choose a maximum of 8 loaded threads (of 32) so that at least 24 threads remain available. Since none of the tested workflows uses more than 24 threads for a single task, workflows remain executable, enabling evaluation of performance under co-located workloads.

The mean absolute percentage error (MAPE) quantifies the deviation between the workflow energy attributed by Nf-PEAK and a reference value based on RAPL. Let E_{RAPL} be the total CPU+DRAM energy measured by RAPL on all nodes during a workflow run. Let E_{static} be the static CPU and DRAM energy the same nodes would consume while idle for the same duration, estimated from a pre-run idle-power measurement as described in Section III-B. Because Nf-PEAK already attributes part of the static energy to workflow processes, we add only the remaining unattributed static energy before comparing against RAPL:

$$\begin{aligned} E_{\text{cmp}} &= A_{\text{PEAK}} + (E_{\text{static}} - A_{\text{static}}) \\ E_{\text{ref}} &= \beta_{\text{overhead}} \cdot E_{\text{RAPL}} \\ \text{MAPE} &= 100 \cdot \left| 1 - \frac{E_{\text{cmp}}}{E_{\text{ref}}} \right| \end{aligned} \tag{8}$$

Here, A_{PEAK} is the total CPU and DRAM energy attributed by Nf-PEAK to workflow tasks, including dynamic energy and the static energy fractions assigned by Eq. (4) and Eq. (6). A_{static} denotes the subset of A_{PEAK} that corresponds to the attributed static energy. E_{cmp} is therefore the workflow estimate based on Nf-PEAK after adding the remaining unattributed static energy. E_{ref} is the reference value based on RAPL after discounting the average non-workflow overhead. We measured $\beta_{\text{overhead}} = 0.97$ for our experimental setup by running an empty workflow that only executes `sleep`, corresponding to an average overhead of 3%. A MAPE value close to zero indicates that the corrected Nf-PEAK estimate closely matches the reference based on RAPL.

[6]https://nf-co.re/sarek/3.5.1/, last accessed: March 22, 2026

[7]https://nf-co.re/rangeland/1.0.0/, last accessed: March 22, 2026

[8]https://manpages.ubuntu.com/manpages/jammy/man1/stress-ng.1.html, last accessed: January 12, 2026

D. Experimental Results

Correlation with RAPL in isolated runs. Figure 4 summarizes isolated 2-node results. Since the workflows are executed in isolation with no other workloads on the node, we expect the total energy attributed by Nf-PEAK to the workflow to be equal to the dynamic energy of the nodes, as measured by RAPL during execution, plus the fraction of static energy attributed to the tasks, depending on their CPU time. When we add the remaining static energy that we previously determined using RAPL to the energy attributed by Nf-PEAK and account for overhead, the result should match the energy measured with RAPL during execution.

For RNASeq, Nf-PEAK attributes only a fraction of total RAPL energy, because most tasks run on few CPU cores and there is no parallel task execution. The remaining energy that has not been attributed by Nf-PEAK consists of static energy unassigned due to low CPU time of the workflow tasks and overhead. Because RNASeq tasks do not overlap on a node in our configuration, we can compare per-task RAPL deltas directly with the energy attributed to each task. Across the entire workflow, the energy attributed by Nf-PEAK results in 4.6% $MAPE$.

Investigating the individual tasks of the RNASeq workflow, as visible in Figure 4a, reveals that Nf-PEAK is accurate across all tasks in the workflow. For each task, the attributed energy results in a single-digit MAPE. This result confirms that Nf-PEAK is capable of accurately attributing energy across a range of different workflow tasks, including tasks with low CPU usage, such as `samtools`, and tasks with high CPU usage, such as `star_index`.

For Sarek, the high sustained load on the nodes caused by task parallelism leads to attribution values that are close to the total energy used during workflow execution as measured with RAPL (Figure 4b). Accounting for static energy and overhead, Nf-PEAK achieves 0.8% $MAPE$, showing that Nf-PEAK is capable of attributing energy accurately in scenarios with parallel tasks and high CPU utilization.

Rangeland contains many sub-second and few-second physical tasks and therefore represents the main failure mode of Nf-PEAK. Polling-based discovery and interval-based sampling observe task start, process lifetime, and RAPL energy deltas in different sampling windows. For very short-lived processes, the resulting process statistics are insufficient to accurately determine which process should receive which part of an interval's energy. In our experiments, this limitation of Nf-PEAK leads to a high attribution error of 46.1% $MAPE$, as shown in Figure 4c. Note that the energy attributed to Rangeland by Nf-PEAK is higher than the total energy measured by RAPL, which is physically impossible. This is because energy is attributed for each process individually during runtime and then aggregated to tasks. If over-attribution occurs for multiple processes, the result can be an over-attribution for the total energy used. Such an error can be detected after workflow execution, but post-run correction is difficult, because it is not known which attributions are inaccurate. The high attribution error of Nf-PEAK on Rangeland highlights the limitations of the used polling- and interval-based attribution strategy. If the workflow contains multiple short tasks, as Rangeland does with over 2000 tasks with a length of less than ten seconds in our experiments, attributions for individual physical tasks are less precise, reducing overall accuracy significantly.

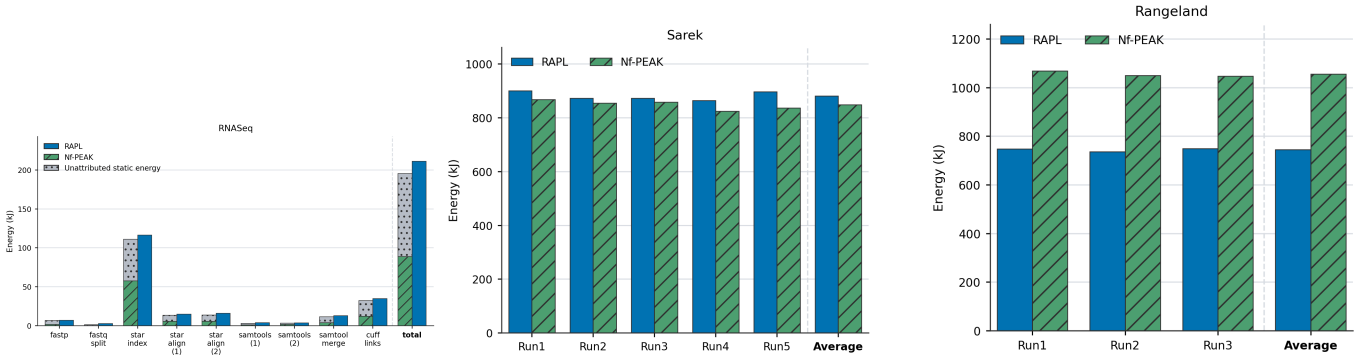
Scaling across cluster sizes. To examine how Nf-PEAK behaves when the cluster size increases, we repeated the same set of experiments that we previously conducted on 2 nodes, using 3, 4, and 8 nodes of the same cluster. We expect the overall energy consumption of the workflows to be similar to that on 2 nodes, with any differences mainly being caused by runtime differences due to the execution of more tasks in parallel rather than sequentially.

RNASeq does not make use of more resources than are available already on 2 nodes in the configuration used in our experiments, resulting in very similar runtime and energy consumption. The energy attributed by Nf-PEAK is very stable as well, resulting in an error of 0.8% $MAPE$ for an execution on 4 nodes and 4.1% $MAPE$ for an execution on 8 nodes on average.

Unlike RNASeq, Sarek includes parallel workflow tasks and benefits from additional resources. As a result, the runtime on 4 nodes decreases to 84.6% in comparison to execution on 2 nodes, and attributed energy decreases to 82.3%. The resulting error remains low at 4.2% $MAPE$. On 8 nodes, further reductions in parallel task execution lead to a low error of 0.8% $MAPE$.

Rangeland scales well across multiple nodes. When executed on 4 nodes, its runtime is equal to 64.8% of the runtime on 2 nodes. At the same time, energy attributed by Nf-PEAK also decreases, resulting in a lower error of 14.9% $MAPE$ for 4 nodes and 12.4% $MAPE$ for 8 nodes. While the error of Nf-PEAK for Rangeland on 4 and 8 nodes is smaller due to reduced over-attribution, the MAPE remains significantly larger compared to RNASeq and Sarek, highlighting the problem of attributing energy for short tasks.

Behavior under co-located CPU load. Our third experiment aims to quantify the accuracy of Nf-PEAK under co-located, unrelated CPU load. While executing each workflow on 4 and 8 nodes using the same configuration and input data as in the second experiment, we execute $n \in \{1, 2, 4, 6, 8\}$ hardware threads of a `stress-ng` CPU benchmark, fully loading one logical CPU core each. We expect the impact on individual workflow tasks to be small, because none of the tested tasks use more than the 24 remaining unoccupied logical cores of our CPUs. Consequently, the energy attributed to each workflow by Nf-PEAK should be nearly the same. Figure 5 shows total energy as measured by RAPL, energy attributed by Nf-PEAK, and workflow runtime under increasing co-located CPU load on 4 nodes. As expected, total RAPL energy increases with additional load because previously idle threads now consume dynamic energy. Meanwhile, the runtime and attributed energy for the workflows themselves remain stable, with moderate increases at higher load levels due to increased runtime, matching our expectations.



(a) RNASeq (per-task breakdown with unattributed static energy)

(b) Sarek (aggregated energy over five runs showing a close match with RAPL)

(c) Rangeland (aggregated energy over three individual runs showing over-attribution)

Figure 4. Isolated results on 2 nodes for RAPL and Nf-PEAK. For RNASeq, unattributed static energy is shown separately to highlight that the combination of attributed energy and unattributed static energy match the energy measured with RAPL.

For RNASeq, total RAPL energy increases by 35.6% when 8 threads are loaded compared to no additional load, while runtime increases by 6.5% and Nf-PEAK attributed energy increases by 26%. The increase in attributed energy results in 6.4% $MAPE$. While this value indicates a larger error than in the experiments without co-located load, the attribution of Nf-PEAK remains relatively stable with a limited error even when a quarter of all hardware threads are fully used by an unrelated, but co-located workload. On 8 nodes, the error in attributed energy is again lower with 2.1% $MAPE$.

For Sarek, RAPL energy increases by 50.7%, runtime by 5.8%, and attributed energy by 18.6% in an execution on 4 nodes with co-located load. Similar to RNASeq, the resulting 7.3% $MAPE$ is slightly higher than in scenarios without co-located workload. This is also true on 8 nodes, where the error is 4.8% $MAPE$.

The attributed energy of Rangeland changes by only 3.6%, while runtime increases by 11.8% and total RAPL energy by 36.1%. This results in a similar increase in MAPE as for the other workflows. With 19.1% $MAPE$ on 4 nodes and 26.2% $MAPE$ on 8 nodes, the error remains significantly higher than for the other workflows we tested.

Impact of monitoring intervals. Shorter polling and sampling intervals make it more likely to collect sufficient data for accurate energy attribution, but they also increase the overhead of each individual per-process monitor.

We evaluate this trade-off by gradually shortening both the interval of the polling loop and the interval for reading RAPL energy counters over multiple, otherwise identical workflow runs. On average, the energy attributed by Nf-PEAK for Rangeland increases by 7.7% when shortening the polling interval from 10 s to 0.2 s and the interval for reading RAPL from 4 s to 0.05 s. Note that there is already an over-attribution by Nf-PEAK to Rangeland with the longer intervals used in our experiments. This result highlights that insufficient polling intervals are not the only cause for the reduced accuracy on short workflow tasks. For the other workflows, no significant changes in the average attributed energy occur for shorter polling intervals. This indicates that the longer polling intervals used previously are sufficient to collect the data needed for accurate attribution. At the same time, shorter polling intervals increase the overhead. Between the longest and the shortest intervals we tested, the total energy measured with RAPL during workflow execution increases by 15%. We therefore conclude that shorter measurement intervals have no measurable advantage for the accuracy of Nf-PEAK on any of the tested workflows, but significantly increase the overhead. Task-discovery and RAPL sampling intervals of 5 s and 2 s, respectively, are sufficient for accurate attribution while ensuring low overhead.

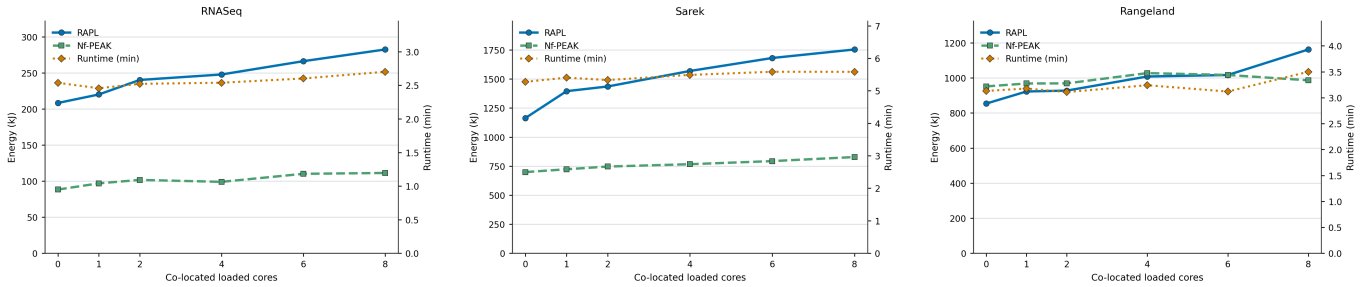
Comparison to Kepler. Kepler is a widely used Kubernetes energy exporter that also provides energy attribution on container/task-level granularity. For task-level energy attribution, Kepler uses a linear CPU-time heuristic[9]:

$$\textit{Workload Power} = \left(\frac{\Delta \textit{Workload CPU Time}}{\Delta \textit{Node CPU Time}} \right) \cdot \textit{Power} \quad (9)$$

To enable a direct comparison of the accuracy of both approaches, we ran Kepler's energy attribution alongside that of Nf-PEAK for all three tested workflows, with and without co-located load, on 4 nodes. For our experiments, we used Kepler v0.8.0. Table II summarizes MAPE for both approaches in isolated runs and under load on 4 nodes. Across the three workflows, Nf-PEAK achieves an average MAPE of 6.6% without additional load and 10.9% under load. In comparison, Kepler achieves an average MAPE of 17.4% without load and 22.5% with load. Our results indicate that Nf-PEAK on average outperforms Kepler without and with co-located load. Kepler performs competitively in some tests, but degrades more under co-located load for Sarek and Rangeland in our experiments.

Kepler's energy attribution model for physical tasks is based on a linear function with CPU-time share as input. When unrelated co-located load increases the node-level CPU time

[9]https://sustainable-computing.io/archive/design/power_model/, last accessed: February 5, 2026



(a) RNASeq (b) Sarek (c) Rangeland

Figure 5. Workflow-level energy and runtime on 4 nodes under increasing co-located CPU load. Energy measured by RAPL increases with the co-located load for all workflows. Meanwhile, the tested workflows are only slightly affected by the co-located load, as indicated by the mostly stable runtimes. Energy attributed by Nf-PEAK remains stable as well, indicating that Nf-PEAK is capable of dealing with co-located load during energy attribution.

Table II
MAPE FOR NF-PEAK AND KEPLER IN ISOLATED RUNS AND UNDER LOAD (8 THREADS) ON 4 NODES. LOWER IS BETTER. DEVIATION DENOTES THE SIGNED DEVIATION UNDER LOAD RELATIVE TO THE ISOLATED RESULT IN PERCENTAGE POINTS.

Workflow	Tool	No Load	Load	Deviation
RNASeq	Nf-PEAK	**0.8%**	6.4%	**5.6 pp**
	Kepler	13.5%	**3.5%**	-10.0 pp
Sarek	Nf-PEAK	4.2%	**7.3%**	**3.1 pp**
	Kepler	**4.0%**	16.7%	12.7 pp
Rangeland	Nf-PEAK	**14.9%**	**19.1%**	**4.2 pp**
	Kepler	34.8%	47.4%	12.6 pp

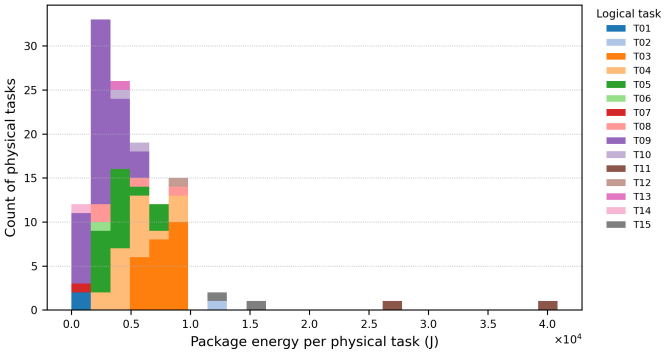


Figure 6. Histogram of the energy consumption of each physical task in the Sarek workflow. Task types of each physical task are marked by colors.

and RAPL energy, this changes the denominator used for attribution and reduces the energy attributed to a physical task, even if the physical task itself is unchanged. Nf-PEAK also uses process-level CPU time and memory usage, but transforms them through a non-linear credit model before assigning dynamic energy, which reduces the sensitivity to changes in aggregate node utilization.

Task-Level Analysis for Parallel Workflows. The task-level results produced by Nf-PEAK allow insights into the distribution of energy consumption even for highly parallel scientific workflows.

Figure 6 shows the distribution of energy consumption for Sarek's physical tasks and groups them by logical task. It highlights that the total energy consumption of logical tasks is not always correlated with the energy consumption of individual physical tasks. While T9 consumes about 12,000 J of energy and contains many physical tasks, most of these tasks use relatively little energy. Meanwhile, T11 consumes less total energy (7,500 J), but contains only 2 physical tasks with much higher energy consumption.

To identify opportunities for optimization in individual workflow tasks and task scheduling, it is important to consult data regarding both the logical workflow tasks as well as individual physical tasks. For both categories of tasks, energy consumption can be analyzed using Nf-PEAK data, making it useful not only for reporting energy consumption, but also for identifying opportunities for workflow optimization.

V. DISCUSSION

Accuracy and robustness. Across three real workflows from two scientific domains, Nf-PEAK produces accurate task-level energy estimates, outperforming Kepler on average. It remains stable across cluster sizes and under co-located load, showing a smaller increase in deviation from the expected result than Kepler.

Deployment and security. Nf-PEAK is designed for managed clusters where users cannot install node software. The workflow pods are executed unchanged, but the monitoring pods require host-level visibility. Concretely, they use `hostPID: true` to inspect host processes and a read-only mount of the host `powercap` interface to read RAPL counters. These privileges weaken the isolation normally provided by Kubernetes. Thus, Nf-PEAK should be treated as an administrator-approved observability component, not as an ordinary user workload. Clusters that enforce the Baseline or Restricted Kubernetes Pod Security Standards disallow host namespaces and hostPath volumes[10]; such clusters require a policy exception or an alternative administrator-provided telemetry interface. In comparison, Kepler has similar deployment constraints because it also requires host-level visibility,

[10]https://kubernetes.io/docs/concepts/security/pod-security-standards/, last accessed: May 12, 2026

which it obtains through eBPF [1]. This method is efficient, but requires access to kernel hooks of the host machine. Kepler produces measurement results for whole nodes as well as attributions for individual tasks/containers. The installation requirements for Nf-PEAK and Kepler are very similar, except that Kepler requires Go 1.21+[11] to implement its eBPF kernel extensions.

Limitations. The main accuracy limitation of Nf-PEAK is per-physical-task attribution for very short tasks. Nf-PEAK relies on periodic polling for task discovery and interval-based sampling. For very short tasks, the collected performance and energy data can be insufficient, resulting in inaccurate attributed energy. This can be observed in our experiments with the Rangeland workflow, where the energy attributed by Nf-PEAK is almost 50% higher than expected and even exceeds the total energy measured with RAPL for the involved nodes. Shortening the task-discovery and RAPL sampling intervals does not remove this limitation in our experiments: it increases the total energy measured during workflow execution by up to 15%, while not improving attribution quality for short tasks. We therefore consider Nf-PEAK reliable for workflows whose physical tasks are observed over multiple sampling intervals, but not for accurate per-physical-task attribution in workflows dominated by sub-second or few-second tasks. For the sampling intervals used in this work, energy can not be reliably attributed to tasks with a length of less than 15 seconds.

Furthermore, RAPL measures only CPU and DRAM energy. Additional energy used by storage, network, and accelerators is not covered by our current implementation. GPU support could be especially relevant, since GPU accelerators are often critical components of modern data-intensive workflows. The NVIDIA Management Library (NVML), which also underlies `nvidia-smi`, exposes information about current board power draw, power limits and GPU utilization[12]. An Nf-PEAK extension could add a GPU monitor that maps CUDA processes to Kubernetes pod UIDs, integrates sampled GPU power, and aggregates the resulting GPU energy to the same physical Nextflow tasks as the current attribution based on CPU and DRAM. Due to a lack of access to compute infrastructure with modern NVIDIA accelerators, such an extension remains future work.

VI. Related Work

Energy measurement and attribution have been studied from several angles, including low-level hardware counter access, process/container attribution, and higher-level carbon accounting.

Kubernetes energy monitoring and attribution. Kepler [1] is a widely used Kubernetes energy exporter that provides container/workload-level attribution and can additionally train machine-specific models [2]. Other Kubernetes-focused tools (e.g., Scaphandre[13]) expose node energy and attribute it to containers primarily through linear resource shares. These systems are popular because they integrate with cluster observability stacks (e.g., Prometheus[14]), but their attribution heuristics are typically linear and do not model the non-linear power behavior in many utilization regimes.

Process- and container-level estimation on single nodes. SmartWatts [10] estimates power consumption of processes/containers using calibrated models across CPU frequencies and can improve accuracy compared to purely linear CPU-time models. In a different but related direction, CodeCarbon [16] and Carbontracker [3] aim to estimate energy and carbon emissions of software runs, often focusing on developer usability and carbon intensity rather than attributing energy to fine-grained tasks in distributed cluster runs. Such tools are useful for reporting and awareness, but generally do not solve the workflow-task attribution problem.

Fine-grained attribution in multi-tenant settings. EnergAt [14] proposes a thread-level, NUMA-aware method for CPU and DRAM attribution that introduces non-linear energy credits. It enables energy attribution for individual processes running on single nodes. METRION [22] further studies fine-grained attribution and monitoring for shared environments. While these approaches address important systems challenges (multi-socket, NUMA, concurrent workloads), they do not target a containerized deployment model and distributed execution in compute clusters, and do not provide workflow-task aggregation for engines such as Nextflow.

Positioning of Nf-PEAK. Nf-PEAK complements prior work by combining (i) a containerized deployment on Kubernetes, (ii) process-level monitoring using a non-linear credit model, and (iii) task-level aggregation for Nextflow workflows. This enables practical energy attribution for distributed workflow executions in managed cluster environments where users cannot install node-level tooling.

VII. Conclusion

We presented Nf-PEAK, a containerized method for process- and task-level attribution of RAPL-visible CPU and DRAM energy for Nextflow workflows executed on Kubernetes clusters. By combining RAPL energy counters with per-process performance metrics and a non-linear credit model, Nf-PEAK enables fine-grained energy feedback in managed and multi-tenant environments. Across three nf-core workflows, Nf-PEAK achieves an average Mean Absolute Percentage Error of 6.6% in isolated runs and 10.9% under co-located CPU load on 4 cluster nodes, outperforming Kepler.

In the future, we plan to go beyond measurements and attribution by predicting task-level energy consumption for workflows to enable fine-grained energy and carbon-aware scheduling.

[11] https://sustainable-computing.io/kepler/installation/guide/#prerequisites, last accessed: February 27, 2026

[12] https://developer.nvidia.com/management-library-nvml, last accessed: May 13, 2026

[13] https://github.com/hubblo-org/scaphandre, last accessed: March 22, 2026

[14] https://prometheus.io/, last accessed: March 22, 2026

Data Availability

Code and usage instructions are available on GitHub at https://github.com/Nf-PEAK/Nf-PEAK_Artifacts.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 414984028 – SFB 1404 FONDA.

References

[1] Marcelo Amaral, Huamin Chen, Tatsuhiro Chiba, Rina Nakazawa, Sunyanan Choochotkaew, Eun Kyung Lee, and Tamar Eilam. Kepler: A Framework to Calculate the Energy Consumption of Containerized Applications. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*, pages 69–71, Chicago, IL, USA, July 2023. IEEE.

[2] Marcelo Amaral, Huamin Chen, Tatsuhiro Chiba, Rina Nakazawa, Sunyanan Choochotkaew, Eun Kyung Lee, and Tamar Eilam. Process-Based Efficient Power Level Exporter. In *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*, pages 456–467, Shenzhen, China, July 2024. IEEE.

[3] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models, July 2020. arXiv:2007.03051 [cs].

[4] Taimyra Batz Liñeiro and Felix Müsgens. Pay-back time: Increasing electricity prices and decreasing costs make renewable energy competitive. *Energy Policy*, 199:114523, April 2025.

[5] Carmen Carrión. Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Computing Surveys*, 55(7):1–37, July 2023.

[6] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. RAPL: memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, pages 189–194, Austin Texas USA, August 2010. ACM.

[7] Laure de Roucy-Rochegonde and Adrien Buffard. AI, Data Centers and Energy Demand: Reassessing and Exploring the Trends. *Ifri Papers*, February 2025.

[8] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4):316–319, April 2017.

[9] James A. Fellows Yates, Thiseas C. Lamnidis, Maxime Borry, Aida Andrades Valtueña, Zandra Fagernäs, Stephen Clayton, Maxime U. Garcia, Judith Neukamm, and Alexander Peltzer. Reproducible, portable, and efficient ancient genome reconstruction with nf-core/eager. *PeerJ*, 9:e10947, March 2021.

[10] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 479–488, May 2020. arXiv:2001.02505 [cs].

[11] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S. Blair, and Adrian Friday. The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns*, 2(9):100340, September 2021.

[12] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the Challenges of Scientific Workflows. *Computer*, 40(12):24–32, December 2007.

[13] Daniel Hackenberg, Robert Schone, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 896–904, Hyderabad, India, May 2015. IEEE.

[14] Hongyu Hè, Michal Friedman, and Theodoros Rekatsinas. EnergAt: Fine-Grained Energy Attribution for Multi-Tenancy. *Energy Informatics Review*, 4(3), 2024.

[15] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 3(2):1–26, June 2018.

[16] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the Carbon Emissions of Machine Learning, November 2019. arXiv:1910.09700 [cs].

[17] Bertram Ludäscher, Mathias Weske, Timothy McPhillips, and Shawn Bowers. Scientific Workflows: Business as Usual? In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *Business Process Management*, volume 5701, pages 31–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. Series Title: Lecture Notes in Computer Science.

[18] Philip A. Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilm, Maxime Ulysse Garcia, Paolo Di Tommaso, and Sven Nahnsen. The nf-core framework for community-curated bioinformatics pipelines. *Nature Biotechnology*, 38(3):276–278, March 2020.

[19] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, 17(3):228–235, 2017.

[20] Martin Sudmanns, Dirk Tiede, Hannah Augustin, and Stefan Lang. Assessing global Sentinel-2 coverage dynamics and data availability for operational Earth observation (EO) applications using the EO-Compass. *International Journal of Digital Earth*, 13(7):768–784, July 2020.

[21] Maarten Van Steen and Andrew S. Tanenbaum. A brief introduction to distributed systems. *Computing*, 98(10):967–1009, October 2016.

[22] Benjamin Weigell, Simon Hornung, and Bernhard Bauer. METRION: A Framework for Accurate Software Energy Measurement, December 2025. arXiv:2512.06806 [cs].