Containerization's Power Use Overhead in Video Streaming — and the Case for Optimizing Scheduler Power in Tickless Kernels

Etienne-Victor Depasquale
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
e-mail: edepa@ieee.org

Saviour Zammit
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
e-mail: saviour.zammit@um.edu.mt

Abstract— **Containerization of a service enables live migration and, thereby, consolidation of running service instances onto as few host platforms as possible. However, containerization's operational overhead must be investigated to determine overall viability. One dimension of this overhead is that of power use, and this is investigated here. An architecture for a video cache service at the edge of a Communications Service Provider's (CSP) network in the metropolitan area is designed, and a scaled version is implemented in a laboratory environment. A comparison is made between power used while streaming videos in both native and containerized modes of operation. Containerization is found to incur a low power overhead while streaming video, compared with streaming video from ffmpeg running directly on the host operating system. Moreover, notwithstanding advances with tickless kernels, the kernel's scheduling timer routine remains a dominant power consumer. Power use is measured using hardware instrumentation and with PowerTOP, a software power meter. Limits on the latter's accuracy have been observed.**

Keywords- containers; power; video; streaming; implementation model; tickless kernel; PowerTOP; power instrumentation.

I. INTRODUCTION

This paper expands upon our earlier study presented at the Ninth International Conference on Green Communications, Computing and Technologies (GREEN 2024) [1], where we investigated the power overheads incurred by containerized video streaming.

Content Delivery Networks (CDNs) are overlay networks that are key to controlling the growth in demand for bandwidth in long-haul communications links. By distributing content to caches in geographical regions of the world where customers are located, the number of times which a single item of content crosses long-haul links between the content origin's region and the customer's region, is reduced to just one. In turn, the content is distributed several times to customers in the region. While the function of the CDN, from the customer's perspective, is that of reducing latency and avoiding buffer underrun, the control of bandwidth growth is a function that has a strategic role in the stability of world-wide communication. The CDN's role in bandwidth control continues to gain attention [2]; a variety of CDN implementations has been investigated [3], [4] and surveyed [5], [6] and generalized surveys are of ongoing interest [7], [8]. The importance of the CDN seems to grant sufficient ground for study of the impact of its Point of Presence (PoP) on the information and communication technology of its environs.

This study seeks to compare power use in containerized deployment of the media server in a CDN PoP. It focuses on the power use of the media server as it processes a representative set of tasks. The media selected for study is video (henceforth, the media server will be referred to as the video server), and two reasons support this choice. Video dominates traffic, whether in the access, aggregation, metro-core, or long-haul. Moreover, some of the tasks, such as transcoding, are processor-intensive and serve to indicate the power capacity required to support CDN PoPs.

The rest of this paper is structured as follows. Section II is a brief treatise of some of the foundational works in modelling power use in computer systems. In Section III, the objective is stated. In Section IV, the implementation model is presented. This supports reproduction of the test environment. In Section V, the method is elaborated upon. Section VI presents the results, and Section VII supports interpretation through analysis of these results. Section VIII draws a succinct conclusion on the impact that containerization of a video service has on power use overhead.

II. BACKGROUND

A. Power models

A better grasp of the impact of containerization on a CDN PoP's power use requires understanding of containerization's impact on the media server's use of power. Media servers are deployed on general purpose computing systems (or: Commercial-Off-The-Shelf (COTS) computer systems), and a basic understanding of these systems' power characteristics must support further study.

Power used by a computer system has an idle (static, or leakage) part, an active (dynamic) part and overhead. The power use referred to here is a system metric: it is an aggregate that sums all consumers' (system components) power use, whether it be of dynamic, static or overhead type. Idle power's (P_{idle}) relevance depends on perspective. On the one hand, idle power use is a real and significant cost: from the perspective of facility managers and sustainability advocates, it is of interest. On the other hand, it is irrelevant to the way in which processes exercise a computer system: it is of

secondary importance in a study of the power required to deliver a service. It follows, then, that a study of the power used to operate a CDN PoP's server systems must primarily engage with service power use, or, using general terms, with dynamic power use.

A simple, yet useful, classification of power models divides them into two: (a) one that treats the computer system as a black box, and uses workload to predict power in real time, and (b) another that exploits knowledge of microarchitecture and architecture [9]. They represent two different levels of abstraction (see [10, p. 42] for a more detailed distinction) of a computer system.

1) The affine relationship between aggregate power and utilization

The affine relationship is a well-known example of the black-box class. A typical representation of workload may be one or more parameters of utilization (e.g., MIPS (millions of instructions per second) and IOPS (input/output operations per second). The affine relationship between power use and utilization [9] is well suited to describing legacy network equipment [11]. The general form is reproduced as equation (1):

$$P(\rho) = P_{idle} + \big(P(\rho = 1) - P(\rho = 0)\big)\,\rho \qquad (1)$$

where P(ρ) expresses power at utilization ρ.

Equation (1) expresses power in terms of generalized utilization, but particular forms like processor load in MIPS or switching throughput in bits per second may be used (where the model holds true). Note that equation (1) refers to the idle power (P_{idle}), but not to the overhead. The static part and overhead are significant and cannot be ignored. However, idle power use has no correlation with the computer system's load. Furthermore, while the overhead (such as fan power use) can indeed be expected to relate to load (it is not a constant type of overhead), power used by these overheads can be expected to have much longer response times than that of power used by silicon. For example, a fan's speed will increase when the temperature in a thermally instrumented zone increases; heat capacity is clearly a factor that will affect temperature rise, as well as temperature drop. Therefore, fan speed will not follow silicon loading and inclusion of fans' power use in measurements will obfuscate the dynamics of power use by silicon under load.

2) Limitations of the affine relationship

Accuracy of the affine relationship has been shown to worsen when the processor does not dominate dynamic power [9]. Apart from processor utilization, system power models have been developed to handle other system components like primary (silicon dynamic random-access memory) and secondary storage (disks) [12]. Furthermore, processor power and frequency are quadratically related [13]. One approach to handling frequency variability is given in [14], where the affine relationship is modified and takes the form shown in equation (2):

$$P(\rho) = P_{idle}^{(f)} + \Big(P^{(f)}(\rho = 1) - P^{(f)}(\rho = 0)\Big)\,\rho \qquad (2)$$

In equation (2), the frequency index in $P_{idle}^{(f)}$ and $\Big(P^{(f)}(\rho = 1) - P^{(f)}(\rho = 0)\Big)$ serves to denote the dependence of intercepts (static/leakage/idle power) and gradients on frequency of operation. Evidently, the affine model expressed in equation (1) does not describe a computer system's processor's power use when the processor is operating under dynamic adaptation of voltage and frequency (Dynamic Voltage and Frequency Scaling (DVFS)). However, system power measurements must be adjusted by a baseline that includes $P_{idle}^{(f)}$. A corresponding measure will be dealt with in considerations of measurement methodology.

3) Relevance of the architectural models

Power architectural models predict power use as a linear function of several activity indicators. These indicators regard the activity of aspects of architecture and microarchitecture of a system. Therefore, models that harness microarchitectural activity indicators tend to be bound to specific hardware models. Activity indicators are commonly referred to as performance counters. Architectural models are well suited to the task of measuring dynamic power use. Performance counters compiled by the operating system are architectural; those compiled by the hardware in dedicated registers, are microarchitectural. System software can abstract microarchitectural counters by a layer that returns these counters through method calls.

A particularly useful class of these counters obtains power use directly. Examples include Intel's Running Average Power Limit (RAPL) and AMD's AMD Energy Driver (amd_energy). The feature addresses power use through hardware support and can form part of a measurement methodology. For example, RAPL's MSR_PKG_ENERGY_STATUS register provides a running, cyclic total of energy used by the CPU (Central Processing Unit) package (all cores included).

B. Isolation and attribution of dynamic power use

Dynamic power is used during both service idle time and service delivery time. The power used during service idle time is not the P_{idle} in equation (1). Rather, it is the dynamic power used by system software (whether in user or kernel mode) to maintain system operation. Service idle time's power use will be subtracted from service delivery (during video streaming) time's power use, to obtain the differential relevant to this research. Service idle time's power use is a tangible justification of the requirement to use minimalist general-purpose systems. Since user application and system software processes and threads are many, then minimization of such root causes simplifies the process of attribution of dynamic power use. An illustration of the multiplicity of subsystems of the computer that are in the scope of power use measurement may be found in [15]; clearly, detailed inspection is required to correctly isolate and attribute power use. Two broad classes of process and thread are identifiable and are briefly described below. Sub-sections II-B-1 and II-B-2 concern the video server, but the same considerations readily hold for the virtual switch's server too.

1) Kernel operations

There are several categories of operation carried out by the kernel to support the operations of the video service. These include:

- processing of hardware interrupts when packets arrive, and concomitant activities like the onerous requirement for the system call to return to userspace;
- managing the timer, to schedule processor allocation to processes and threads;
- memory and cache management, and
- processing of system power monitoring instructions.

The detail of which processes to monitor is expected to be captured in the baseline (see approach-baselining, below). The power used during service idle time will be subtracted from power used during service delivery (during video streaming), to obtain the differential relevant to this research.

2) Service operations

The video streaming service may be tersely described as one in which:

- a source file is encoded (or transcoded) using a video codec and an audio codec;
- the codecs' output is packetized and
- transmitted over a network interface as the payload of a communication protocol that handles:
    - the correct sequencing of the received content (payload) and
    - adaptation of the video quality of the content to network conditions.

These operations must be matched to specific computing entities (components such as processes, threads, main memory and cache) in the computer system and the power use thereof is to be monitored. In particular, the specific computing entities are expected to include the video server process obtained by running the principal executable, and library functions which it calls to support the three major categories of operation listed above (encoding, packetization and sequencing into a stream of adaptable quality).

C. Service scaling

Service scalability is essential to cope cost- and energy-efficiently with short-term fluctuations in demand. These fluctuations are commonly referred to as the daily diurnal and nocturnal crests and troughs In Internet service demand. Figure 1 [16] illustrates service scaling in a virtualization infrastructure. The range of service supply varies from minimum scaleLevel to maximum scaleLevel, stepping with the size corresponding to a virtual network function component (VNFC). Higher demand (load) can be met by spawning one service instance per client. The service instance may consist solely of a single VNFC.

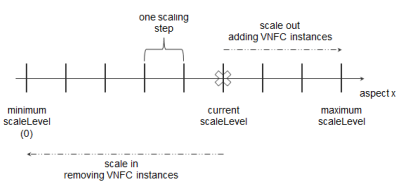

Figure 1. [16, Fig. 5.1–1] Demand is met by deploying over a range from minimum to maximum scaleLevel

III. OBJECTIVE

A. Principle

An overhead is expected in the containerized implementation, and its quantification is sought. The objective can be articulated in terms of a comparison between two types of deployment:

- power use in a computer system that runs the service within containers, with
- power use in a computer system that runs the service directly on the operating system.

Quantification is sought to control a tradeoff between native and containerized deployment. The tradeoff may be succinctly summarized as one of greater operating power per unit (physical host) versus potential for lower number of operating units (physical hosts). The following sub-sections elaborate on this summary.

B. Greater operating power per unit

1) Quiescent operating power

A host (physical server) computer system uses power in its quiescent state. Quiescence is the condition where the host has an active operating system and is running a minimal set of services. Levels of quiescence can be defined, in accordance with different specifications of the set of services. In all levels, quiescent power use consists of an idle/leakage/static component, due solely to physical properties of the hardware, and a dynamic component, due to execution of software processes on the hardware. A containerized deployment uses more power in the quiescent state because its minimal set of services is a superset of that used by a native deployment. Therefore, even when no video clients are served, a containerized deployment has greater operating power. Moreover, to grasp the difference between operating power of the two deployments, a service process deployment strategy must be defined.

2) Full load operating power

As clients appear, service processes must be started to handle the workflow. Minimally, the video service workflow consists of the following cyclical process:

- fill a memory buffer queue by copying some initial large chunk of the file from storage;
- transmit the queue head;
- repeat queue head transmission until some fraction of the queue is empty;
- re-fill the memory buffer queue from memory ramdisk and
- repeat the second, third and fourth steps until all of the file has been read into the queue.

The process, which can be tersely summarized as streaming, is independent of the file's encoding format, but will be extended should real-time transcoding be necessary to meet the client's constraints. These observations prompt the identification of load units, comprising the full amount of work (in Joules) required to process the workflow. A topmost classification divides the set of load units into two branches:

one for the case where only streaming is needed, and another for the case where both streaming and real-time transcoding is needed. Below this topmost classification, load units can be identified for every encoding type and bitrate preset. Each such load unit corresponds to a single resource unit, which is the bundle of computing and networking resources required to serve the load unit. Specification of a load unit supports the analysis of full load operating power, as the latter is used when no more load units can be taken (subject to some quality of service (QoS) condition, as described below). This limit can be articulated better in terms of bin packing, where each physical host is represented as a bin capable of serving load. When a load unit is served, the bin is partially filled, and the corresponding resource unit is removed from the aggregate of the host's available resources. As more load units are served, the bin is progressively filled until no more load units can be added. This is full load, and the power used under this condition is the full load operating power.

3) Bin packing

The condition of full load corresponds to the operating principle of maximization of capacity utilization without degrading key indicators of quality of service (QoS). That is, if each server represents a bin of some service capacity C, then the server is loaded until its capacity is fully utilized without degrading the QoS. The process of filling the server suggests modelling using bin-packing algorithms; hence, depiction of the server as a bin.

Since both capacity, C, and QoS are complex, a simplification is sought to manage the tractability of the problem. Let the capacity, C, be the number L of load units U that a host H in a set of homogenous hosts, can serve without degrading the received bit rate at any client, below the preset for U. This specification of C and QoS key performance indicator (KPI) serves to support specification of other, different loading conditions for both types of deployment. This is reserved for future study.

C. Potential for lower number of operating units

Consider the condition of consolidation, obtained by deploying video service process to the minimum number of hosts possible. Ideal consolidation is obtained when all N hosts (bins) in service except for the Nth are packed. Here, bin-packing corresponds to loading a server until the bit rate served at one or more of the clients falls below the preset.

Such an idealized consolidation is depicted in Figure 2. The top part (a) shows ideal consolidation, at some time t = 0. $N(@t = 0)$ (henceforth denoted by $N(0)$) servers are shown, of which $N(0) - 1$ are filled and 1 is partially filled. One white segment represents one utilized resource unit. One context in which this consolidation is achievable is when an initial set of load units is presented to a dispatching subsystem for distribution onto a set of idle servers. This context applies to both the case where the service is running as a User Application (UA) directly on the host Operating System (OS) (henceforth shortened to "running as a UA") and the case where the service is running in a container.

Over time, clients drop out (the black gaps represent unutilized resource units) as their viewing sessions end. While running as a UA, the service instance supporting dropped cl-

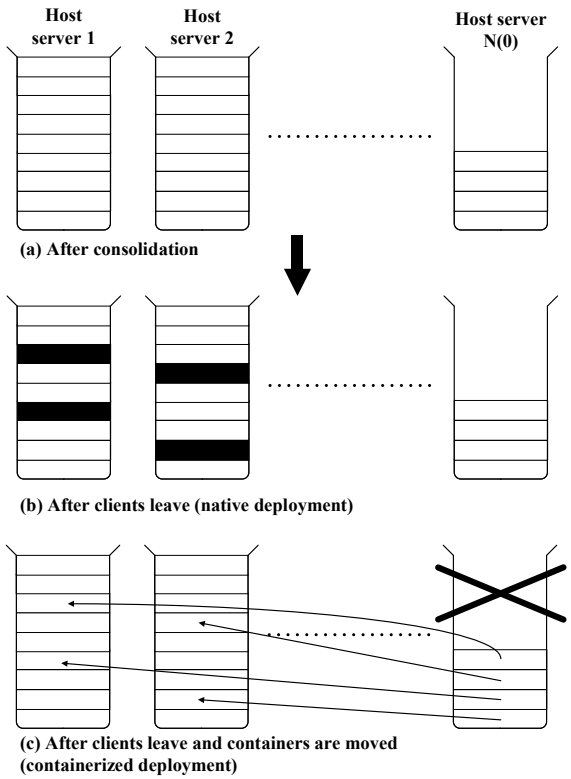

Figure 2. Simplified view of power control enabled by containerization of service application

-ients terminates and leaves a resource gap. However, these gaps cannot be filled with running instances on other servers, since UA state cannot be migrated as easily as when it runs within a container. At some arbitrary time, t, after service starts, it may not be possible to consolidate the service running as a UA, but it should always be possible to consolidate the service running containerized. Therefore:

- while server $k \in \{1, 2, \ldots N(0)\}$, draws $P_k^{(c)} > P_k^{(ua)}$, where $P_k^{(c)}$ represents power drawn while serving a capacity-sized subset from containers and $P_k^{(ua)}$ is the native counterpart, and
- while $N^{(c)}(0) \geq N^{(ua)}(0)$, since $L^{(c)} \leq L^{(ua)}$, where $L^{(c)}, L^{(ua)}$ are the respective capacities of the containerized and native service deployments,

there is no predetermined relationship between $N^{(c)}(t)$ and $N^{(ua)}(t)$. Moreover, while all but the last of the $N^{(c)}(t)$ hosts can (at least periodically) be subjected to consolidation and thus use power amounting to $P_k^{(c)}$ W, the operating state of the $N^{(ua)}(t)$ hosts, and their individual power uses, is unknown. It thus follows that the relationship between $P_{N^{(c)}(t)}^{(c)} + \left(N^{(c)}(t) - 1\right)P_k^{(c)}$ and $\sum_{k=0}^{N^{(ua)}(t)-1} P_k^{(ua)}$ is not evident, and **quantification** of the overhead $P_k^{(c)} - P_k^{(ua)}$ due to containerization, is a necessary prerequisite to understanding the scale and usage pattern at which

containerized deployment is energy efficient compared with native deployment.

IV. IMPLEMENTATION MODEL

An edge cache of a video streaming service is deployed. A high-level view of the implementational model is shown in Figures 3 and 4.

- Figure 3 shows an implementation that is easily portable to a cloud-native infrastructure (henceforth referred to as the cloud-native implementation), and
- Figure 4 shows an implementation that is a hybrid of physical (the video server) and virtual network functions (the switch).

The cloud-native implementation uses containers to host the video server. Both implementations host a virtual layer 2 switch in the intermediate node.

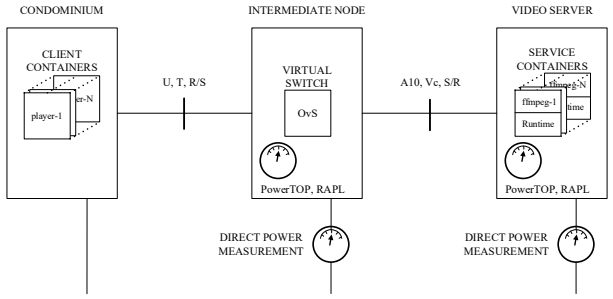

Figure 3. Physical topology of the video streaming service, deployed in containers. Video Server located in local exchange or Access Node (AN); Intermediate Note located in street cabinet (subtended AN [17]).

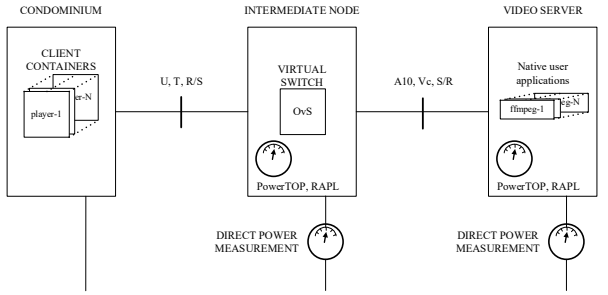

Figure 4. Physical topology of the video streaming service, deployed on a host operating system.

A. Hardware

The hardware used in this testbed consists of a set of three HPE (Hewlett Packard Enterprise) ProLiant BL460c Gen9 blade servers [18], hosted in an HPE c7000 blade enclosure. Connectivity between server and client blades is obtained through pass-through interconnect bay modules, patched with single-mode optic fibre cables. These latter modules support the goal of bypassing c7000 ecosystem interconnect-bay physical networking devices. Bypass is necessary to introduce separate, **virtual** switching hardware. The virtual switch is implemented on a third HPE Gen9 blade server. The links to the switch are of type 10GBASE-SR. The video server has a single Intel® Xeon® CPU E5-2640 v3 (2.60GHz) processor package. Dynamic Voltage and Frequency Scaling (DVFS) is under system firmware control.

B. Software

The software consists of:

- an FFmpeg [19]video server. This is representative of the access node at the edge of the metro-core network;
- a TSDuck [20] receiver. This is representative of end-user's video player, and is also used to measure received bitrate to ensure that Quality of Service (QoS) (see Section IV-C) is respected;
- the virtual switch software is Open vSwitch [21].

A minimalist operating system was selected for the video server, to support isolation and attribution in power measurements. While minimalist operating systems do not necessarily correlate with minimal noise in power measurement, it seems useful to reduce the number of possible sources from the outset. For this reason, Alpine Linux [22] Standard distribution version 3.19 was chosen.

The container system software selected is Docker [23]. Docker is a mature containerization platform and it is modular: the runtime daemon (containerd) supports other user interfaces apart from the Docker user interface (dockerd). For example, Kubernetes [24] can be used to manage containers created through the Docker Command-Line Interface (CLI).

V. METHOD

A. Instrumentation

Near-real time measurement of power use can be obtained from two sources of instrumentation. The blade servers are equipped with a management processor (known as "integrated lights-out", or iLO) that logs a power measurement every 10 seconds and stores a 20-minute history that can be read through a Redfish®[25] - compliant RESTful (Representational State Transfer) Application Programming Interface (API). Selectivity in aggregate power use measurement is afforded by blade systems, since these separate power supply to the (blade) computer system from power supply to two major overhead power drains. Blade servers use blade chassis services for power supply (where ac – dc conversion losses occur) and cooling (where blowers use power as they ventilate from chassis front to chassis rear). Thus, measurement of power used by the blade server at the supply voltage rails is free of the problematic, variable contribution from overheads, and idle power can be measured to the accuracy afforded by these blade system power measurement instruments. The measurement datum is of integer type, obtained by truncation of the decimal part of the actual measurement. Moreover: since the iLO is not part of the System Under Test (SUT), it does not alter power measurement.

While the iLO provides an aggregate power measurement, process- and thread- level granularity is obtained through

software power meters. Hardware extensions for power measurement are available in processor models that support the Intel Running Average Power Limit (RAPL) feature. PowerTOP [26] is software that enables this level of power attribution, and it is indeed capable of exploiting RAPL. This tool complements the aggregate power measurement obtained by blade sensor instrumentation. PowerTOP uses a top-down approach [27], (it divides the power measurement over a period amongst processes and threads in proportion to their core utilization) and precedes the measurement period by one of calibration (the utility was run in calibration mode for several hours before starting the first experiment) in which it obtains weighting parameters for the attribution process. Calibration is further refined with use, and PowerTOP saves its parametric refinement to persistent storage for future exploitation [28]. PowerTOP was used in its logging mode of operation, with 10 (ten) – second averaging intervals. However, PowerTOP has several significant limitations, as follows: it only measures dynamic power, it does not capture all power use, and it increases the SUT's aggregate power use. These must be mitigated.

B. Baselining

It is necessary to distinguish power used by the video service from power used by other consumers. This requires measurement of static (/idle) power use. It is also necessary to distinguish between dynamic power used during video service operation time, from dynamic power used when the service is idle. In essence: service power use can be thought of as an amount added above that used by the operating system and system software, which in turn is added above that used to operate electronic components (static/idle) power. Hence, it is possible to perceive a baseline to which service power is added to obtain the total power. Formally:

$$P_{b_1}^{(video)} = P_{idle}^{f_1} + P_q^{(os)}$$

where $P_q^{(os)}$ is the **dynamic** power corresponding to the OS's operation **without** container system software and without running User Applications (UAs), and $P_{idle}^{f_1}$ is the **idle/static** power at the frequency f_1 at which the OS is quiescent.

A second baseline, $P_{b_2}^{(video)}$, is required to ensure experimental reproducibility: it defines known starting and ending points for each run of experimentation.

$$P_{b_2}^{(video)} = P_{idle}^{f_2} + P_q^{(os+dockerd+containerd)}$$

Here, $P_q^{(os+dockerd+containerd)}$ is the **dynamic** power corresponding to the OS's operation **with** container system software but without running User Applications (UAs), and $P_{idle}^{f_2}$ is the **idle/static** power at the frequency f_2 at which the Operating System (OS) is quiescent. The state of quiescence is defined below (see V-D-2).

C. Mitigating errors

The principal source of error is measurement uncertainty at the iLO, as the iLO rounds to the nearest integer. Since the iLO rounds $[n-0.5, n+0.5)$ to $n, \in \mathbb{N}$, then, without further information on the probability density function (pdf) of the error, a fair representation of each measurement is the value n obtained by the iLO. This contrasts with the floor (round down/truncation) function, where a fair representation of a measurement n would be $n+0.5$, or the ceil (round up) function, where $n-0.5$ would be fair. Of the three conversions from real to natural number representation, rounding to the nearest integer has the least maximum error, and this corresponds to 0.5 W.

The ideal statistical distribution of errors is that of a uniform Probability Density Function (PDF). If measurement errors were indeed so distributed, then the mean of actual measurements can be obtained as the mean of the set of errored measurements. However, for the specific operating context of a quiescent operating system, the probability of a non-uniform distribution cannot be neglected because the dynamic power is low enough to keep the total power's range within half a watt. This is prone to persistent positive bias in error or persistent negative bias. In such non-uniform PDFs, the actual mean cannot be obtained; only a range of values within which the actual mean lies, can be obtained.

Both baselines regard quiescent states. If bias is detected, mitigation can be pursued through the less biased of the two baselines. The better baseline can be used to compute the affected baseline as the arithmetic combination (addition/subtraction) of the better baseline and the difference in dynamic power between the two baselines. Therefore, each measurement of baseline power must be accompanied by a measure of dynamic power, to support evaluation of the error in the means obtained through the iLO's measurements.

This approach notwithstanding, it may still not be possible to reconcile the two baselines in this manner. In such an eventuality, the ranges of values within which the actual means lie can be combined with the difference in dynamic power between the two baselines. The objective remains that of reconciling all measurements, within the margin of error anticipated.

D. Quality of Service

QoS is considered to be satisfied as long as there is sufficient capacity in the links to keep the overall average received bitrate of every video stream at or above the video file's overall bitrate.

E. Experiments

1) Test conditions

Video service will be delivered from both containerized and native deployments. The test conditions pertinent to the video server will be the following.

1. Implementation
   a. During containerized operation, each video service process and the libraries on which it depends will be operated from a container. One service process serves one client.
   b. During native operation, a new instance of the video service process will be started for every new client.
2. Load unit: This will consist of the work required to process a workflow based upon a video with the following technical specifications:

a. Overall bitrate = 457 kb/s, = video bitrate of 326 kb/s + audio bitrate of 127 kb/s + mp4 container metadata rate (overhead)
b. Duration = 1h 32m 2.19s (5522.19 s), of which 30 minutes are played, starting at a randomly selected point in the video.
c. H.264 video codec, Main profile
   i. Resolution = 1280 x 720
   ii. Frame rate ≈ 23.98 frames/second (fps)
d. Advanced Audio Coding (AAC) audio codec, Low Complexity profile
   i. Sampling rate = 44.1 kHz
e. Client supports same video and audio codec; hence server does not need real-time transcoding.

2) Procedure

The power used by the video server is measured at progressively higher load levels. Two sets of experiments are carried out: the first set uses containerized video server instances and the second set uses native video server instances. A containerized service instance consists of a container carrying ffmpeg. A single container is created to deliver a single stream and is destroyed immediately thereafter. When the container is created, ffmpeg is executed and listens on a Transmission Control Protocol (TCP) port, through which it streams 30 minutes of video. A native service instance is a single instance of the ffmpeg process; it follows the same lifecycle as the containerized instance.

Management of operations is not trivial, even at the minimum load level, as it involves the following steps:

1. Reboot the video server, to obtain a common and reproducible initial state.
2. Wait until the video server quiesces. This is the time required for server power use to fall to the state where the iLO measurement persistently shows baseline 2 usage. Persistence was empirically found to be ascertained 20 minutes after rebooting.
3. Start the power meters for both total and dynamic power, for both the video server and the virtual switch.
4. Wait for a fifteen-minute interval, to capture behaviour before video streaming.
5. Instantiate and start a container carrying the ffmpeg listener, poised for real-time playback with randomized starting point and 30-minute play time.
6. Start a TSDuck client to connect to the container and measure the bitrate, averaged over 5-second intervals.
7. Once 30 minutes of video have been played, destroy the container.
8. Wait for a fifteen-minute interval, to capture behaviour after video streaming.

For several concurrent streams, steps 5 and 6 must be repeated for each one of the additional streams. For the native service instance, step 5 involves the ffmpeg process only and there is no equivalent to step 7.

It seems evident that manual management is highly prone to error and is therefore unsuitable. Automated management using Python scripts and Ansible [29] is employed to handle the **orchestration** of the various roles: power meters, container runtime managers and video clients. This enables the experiment to be scaled out to levels that are well beyond the physical limitations of a single human operator.

VI. RESULTS

Denote:

- mean dynamic power measured by PowerTOP by $\overline{p_{dyn}^{(ptop)}}$
- mean total power measured by the iLO during a time period T_x by $\overline{p^{(iLO)}}([T_x])$.

A. Video server's baseline 1

Figure 5 shows the power used by the video server over an hour period of measurement, post-onset of quiescence. Since the iLO truncates decimals in $[n, n+1)$ to n, then the computation of the mean will count the incidences of 45 W and 44 W and use them as weights to compute a lower limit to the range of values which the average can take. An upper limit is obtained by adding the maximum possible error (equal to 1W) and the mean of the possible range obtained by adding the mean error (0.5W) to the lower limit of the range. Using this premise, the mean power measured by the iLO, under the condition of a quiescent operating system (see Figure 5) is as follows:

$$P_{b_1}^{(video)} = P_{idle}^{f_1} + P_q^{(os)} = \overline{p^{(iLO)}}([10:31:49, 11:37:01]) = 45.4198\ \text{W} \cong 45.4\ \text{W}$$

B. Mitigation of PowerTOP's limitations

PowerTOP captures neither static nor dynamic power used by Hard Disk Drives (HDDs) and Solid-State Disks (SSDs); this was observed and confirmed through discussion with PowerTOP's developers [30]. Indeed, our experiments under baseline 1 conditions show that if PowerTOP is operated in logging mode with HDD as destination, iLO aggregate power use is more than 0.5 W greater on the SUT than the figure obtained while logging to a RAM (Random-Access-Memory) disk (see Figure 6). Average aggregate power use increases to 46.05W, compared with 45.4W (see Section VI-A, above). On the other hand, while logging to RAM disk (under baseline 1 conditions), average aggregate power use only increases to 45.5W (see Figure 7), compared with 45.4W (see Section VI-A, above) when measurements are taken solely through use of the iLO's instrumentation.

C. Video server's baseline 2

The difference in average dynamic power is added to baseline 1, to obtain baseline 2:

$$\Delta\overline{p_{dyn}^{(ptop)}} = \overline{p_{dyn}^{(ptop)}}(baseline\_2) - \overline{p_{dyn}^{(ptop)}}(baseline\_1) = 0.7727 - 0.1851 = 0.5876\ \text{W}$$

$$\therefore P_{b_2}^{(video)} = P_{idle}^{f_2} + P_q^{(os+dockerd+containerd)} = P_{idle}^{f_1} + P_q^{(os)} + \Delta\overline{p_{dyn}^{(ptop)}} = 45.4 + 0.5876 \cong 45.99\ \text{W}$$

This is consistent with the graphical summarization of iLO measurements shown in Figure 8. This baseline (the graph of power against time) is essential to obtain a reproducible start-

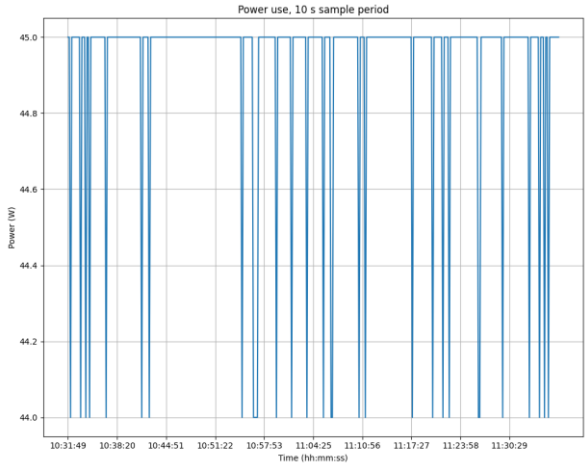

Figure 5. Power used by the video server, with a quiescent OS.

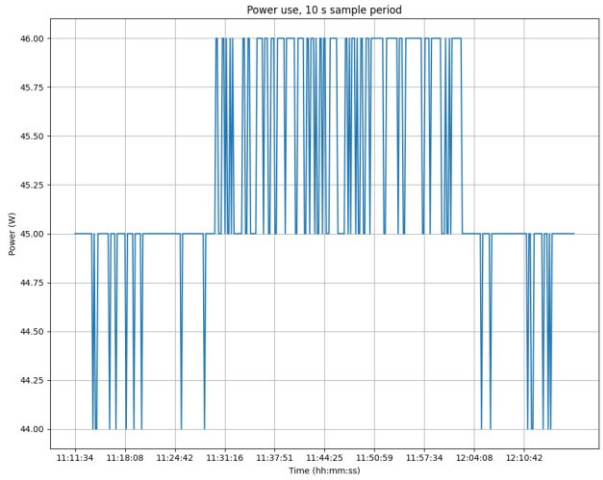

Figure 6. PowerTOP logging [11:30:02,12:02:12] to HDD has a discernable impact on power use

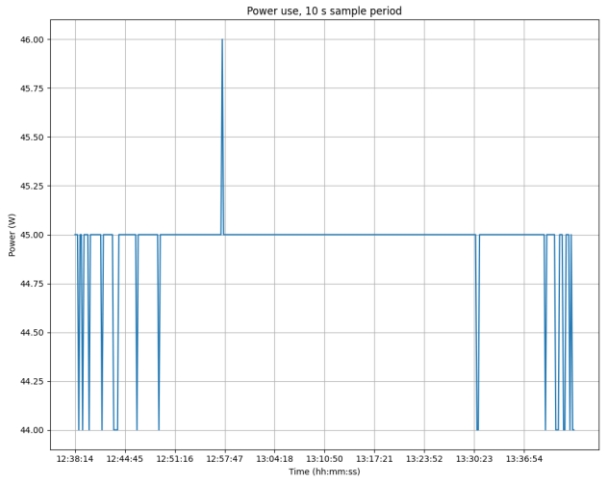

Figure 7. PowerTOP logging [12:57:27,13:29:30] to ramdisk has a lower impact than logging to HDD.

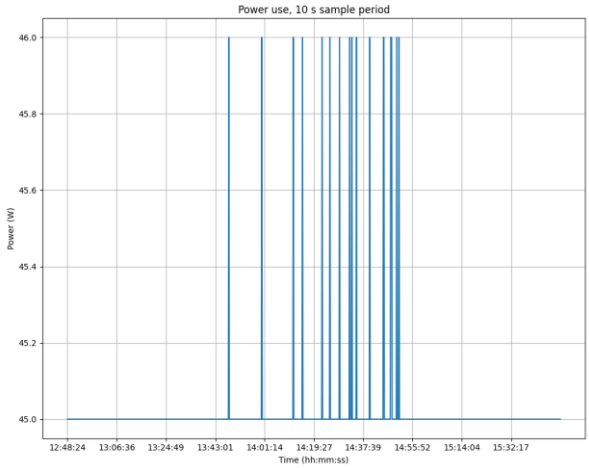

Figure 8. Baseline 2 video server aggregate power

-ing state for all video service operation experiments.

D. *Orchestration of containerized streaming*

Results from running experiments on 1, 2, 5, 10, 20, 40 and 80 instances are presented. The result items consist of:

1. Mean aggregate power use (iLO instrumentation). Due to the integer type of the measurement, actual average iLO power use can lie in the range of ± 0.5 W of the reported result.
2. Mean dynamic power use (PowerTOP instrumentation). Dynamic power data is added to baseline 1 and the sum is plotted on the same Cartesian axes as the total power data.

PowerTOP was used to attribute dynamic power to processes, and these were sorted in descending order. Graphical representations of the power used were produced too. These results are presented in the Github online repository at [31] and in Section VI-F (containerized operation only). Measurements of received stream bitrates are also available in this repository.

1) *Single instance*

Table I shows the mean power use; Figure 9 shows PowerTOP's measurements offset by baseline 1 and laid over the iLO's measurements. Time is shown in the format hh:mm:ss, where hh, mm and ss stand for hour-of-day, minutes in the hour and seconds in the minute, respectively. The larger post-operation (post-op) average power is due to activity undertaken by an instance of containerd (the container runtime) after the container is destroyed (post-ops). However, well after operations end, the iLO's measurements return to the baseline 2 profile. Pre-operations (pre-ops), both meters (iLO and PowerTOP) are in good agreement (PowerTOP's measurements would all be rounded down to 45W). Moreover, the average power used during operations as estimated by PowerTOP is 46.99 W (baseline_1, = 45.4, + 1.5940), whereas the iLO estimates 47.03W. The ten-second

averages' dissimilarity increases during and post-operations but is still good. Notably, the spike in power use at the beginning and end of operations is captured by both meters, albeit not being measurements of the same magnitude.

2) Two instances

Table II and Figure 10 show the results pertinent to two containerized video server instances. As is the case with the single instance, for pre-ops and post-ops, both meters are in good agreement (the spike at about 09:29:00 is probably due to HDD input/output operations while loading PowerTOP). During operations, the average total power estimated by PowerTOP is 48.02 W (baseline_1 + 2.6162), whereas the iLO estimates 47.06W. The discrepancy is an overestimate by about 1W.

An interpretation of the discrepancy between operating period averages is visible in the graph (Figure 8) showing real time measurements. When the iLO measures 46W, the actual value is in the range [46,47), and the rate of change between 46W and 47W is much larger than the single-instance case.

TABLE I. MEAN POWER USE – SINGLE SERVICE INSTANCE

Power type	Description	Avg[a] (W)
$\overline{p^{(iLO)}}[14:47:05,15:03:00]$	Before starting the service instance	45.65
$\overline{p^{(iLO)}}[15:03:00,15:33:05]$	During the service instance's operation	47.03
$\overline{p^{(iLO)}}[15:33:05,15:52:17]$	After the service instance ended	46.17
$\overline{p_{dyn}^{(ptop)}}[14:48:17,15:03:00]$	Mean dynamic power before service instance operation	0.8593
$\overline{p_{dyn}^{(ptop)}}[15:03:00,15:33:05]$	Mean dynamic power during service instance operation	1.5940

a. Average.

TABLE II. MEAN POWER USE – TWO SERVICE INSTANCES

Power type	Description	Avg (W)
$\overline{p^{(iLO)}}[09:24:01,09:44:27]$	Before starting the service instance	45.60
$\overline{p^{(iLO)}}[09:44:27,10:14:37]$	During the service instance's operation	47.06
$\overline{p^{(iLO)}}[10:14:37,10:29:23]$	After the service instance ended	46.12
$\overline{p_{dyn}^{(ptop)}}[09:29:27,09:44:27]$	Mean dynamic power before the service instances' operation	0.9693
$\overline{p_{dyn}^{(ptop)}}[09:44:27,10:14:37]$	Mean dynamic power during the service instances' operation	2.6162

PowerTOP's real time measurements are consistently higher than 47W, revealing that several of the 10-second measurement intervals are in certain disagreement, albeit small (< 2/46, i.e., < 5%).

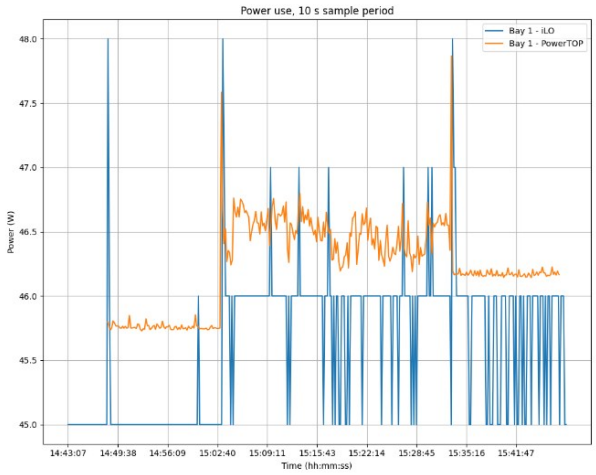

Figure 9. One instance. Video server's power use during containerized service operation. Baseline 1 added to powertop measurements.

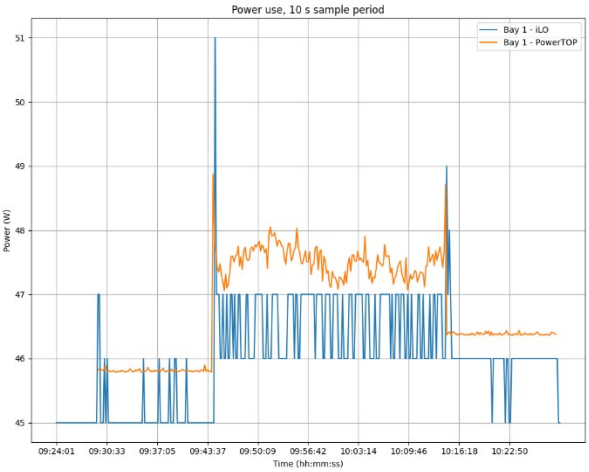

Figure 10. Two instances. Video server's power use during containerized service operation. Baseline 1 added to powertop measurements.

3) Five, ten, twenty, forty and eighty instances

The results for five (Table III, Figure 11), ten (Table IV, Figure 12), twenty (Table V, Figure 13), forty (Table VI, Figure 14) and eighty instances (Table VII, Figure 15) are shown below.

Conditions pre-operations are similar, but PowerTOP's average error estimation increases as power use increases. The numbers shown in the list are PowerTOP's estimate vs iLO's maximum estimate, for N instances (Ni):

- 5i: 50.14 vs 48.66W
- 10i: 54.38 vs 50.10W
- 20i: 60.77 vs 51.74W
- 40i: 64.59 vs 53.90W
- 80i: 60.92 vs 56.80W

Inspection of the online supplementary data on process – level power attribution suggests that PowerTOP overestimates across all processes on our test platform.

TABLE III. MEAN POWER USE – FIVE SERVICE INSTANCES

Power type	Description	Avg, (W)
$\overline{p^{(iLO)}}[11:29:41,11:49:59]$	Before starting the service instance	45.79
$\overline{p^{(iLO)}}[11:49:59,12:20:15]$	During the service instance's operation	48.16
$\overline{p_{dyn}^{(ptop)}}[11:35:00,11:49:59]$	Mean dynamic power before service instances' operation	0.9970
$\overline{p_{dyn}^{(ptop)}}[11:49:59,12:20:15]$	Mean dynamic power during the service instances' operation	4.7421

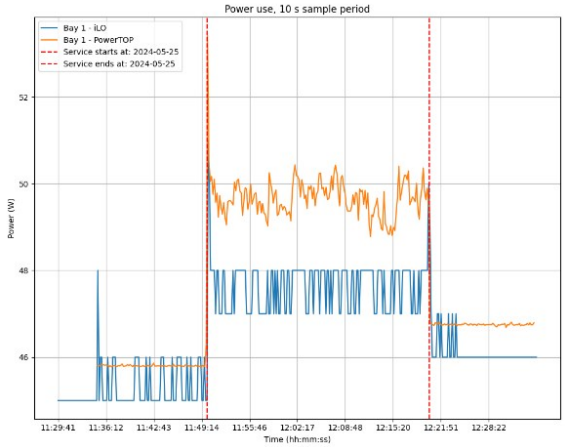

Figure 11. Five instances, containerized operations, baseline 1.

TABLE IV. MEAN POWER USE – TEN SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[15:06:49,15:27:03]$	Before starting the service instance	45.60
$\overline{p^{(iLO)}}[15:27:03,15:57:27]$	During the service instance's operation	49.60
$\overline{p_{dyn}^{(ptop)}}[15:12:03,15:27:03]$	Mean dynamic power before service instances' operation	0.8759
$\overline{p_{dyn}^{(ptop)}}[15:27:03,15:57:27]$	Mean dynamic power during the service instances' operation	8.9781

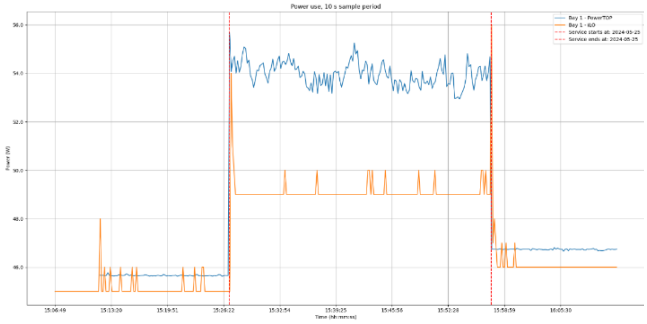

Figure 12. Ten instances, containerized operations, baseline 1.

TABLE V. MEAN POWER USE – TWENTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[17:56:58,18:17:08]$	Before starting the service instance	45.76
$\overline{p^{(iLO)}}[18:17:08,18:48:00]$	During the service instance's operation	51.24
$\overline{p_{dyn}^{(ptop)}}[18:02:08,18:17:08]$	Mean dynamic power before service instances' operation	0.8913
$\overline{p_{dyn}^{(ptop)}}[18:17:08,18:48:00]$	Mean dynamic power during the service instances' operation	15.3720

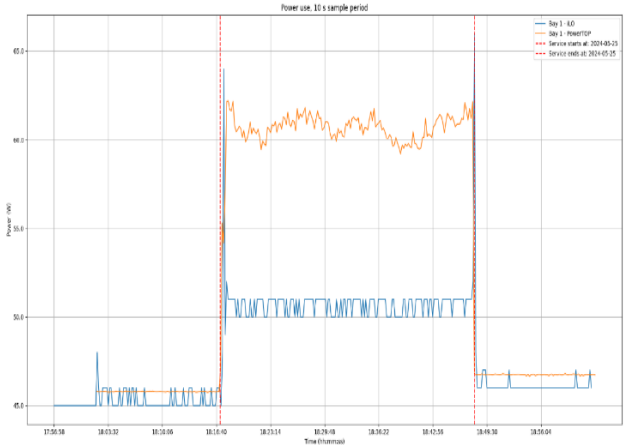

Figure 13. Twenty instances, containerized operations, baseline 1.

TABLE VI. MEAN POWER USE – FORTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[12:57:37,13:17:40]$	Before starting the service instance	45.56
$\overline{p^{(iLO)}}[13:17:40,13:49:15]$	During the service instance's operation	53.40
$\overline{p_{dyn}^{(ptop)}}[13:02:42,13:17:40]$	Mean dynamic power before service instances' operation	0.7206
$\overline{p_{dyn}^{(ptop)}}[13:17:40,13:49:15]$	Mean dynamic power during the service instances' operation	19.1873

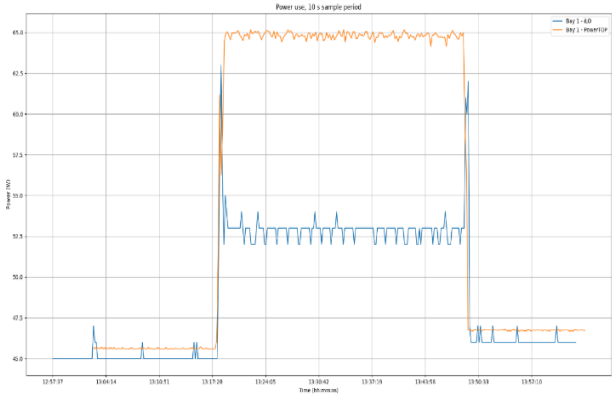

Figure 14. Forty instances, containerized operations, baseline 1

TABLE VII. MEAN POWER USE – EIGHTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[18:19:21,18:39:20]$	Before starting the service instance	45.53
$\overline{p^{(iLO)}}[18:39:30,19:13:53]$	During the service instance's operation	56.30
$\overline{p_{dyn}^{(ptop)}}[18:24:31,18:39:30]$	Mean dynamic power before service instances' operation	0.7435
$\overline{p_{dyn}^{(ptop)}}[18:39:30,19:13:53]$	Mean dynamic power during the service instances' operation	15.5243

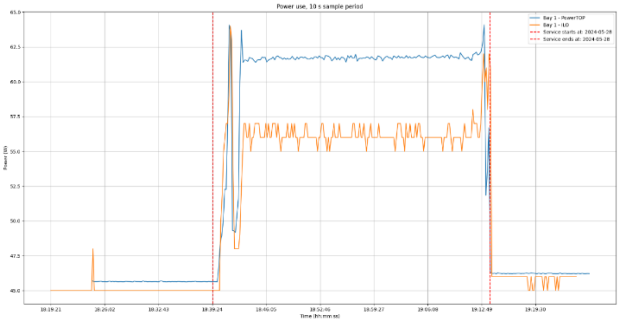

Figure 15. Eighty instances, containerized operations, baseline 1.

E. Orchestation of native streaming

A similar set of experiments was run for native video servers. The results are presented in this section, and are structured in the same manner as that used in Section VI-D.

1) Single instance

TABLE VIII. MEAN POWER USE – ONE SERVICE INSTANCE

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[13:17:42,13:37:55]$	Before starting the service instance	45.54
$\overline{p^{(iLO)}}[13:37:55,14:08:02]$	During the service instance's operation	46.38
$\overline{p_{dyn}^{(ptop)}}[13:22:55,13:37:55]$	Mean dynamic power before service instances' operation	0.2080
$\overline{p_{dyn}^{(ptop)}}[13:37:55,14:08:02]$	Mean dynamic power during the service instances' operation	0.8675

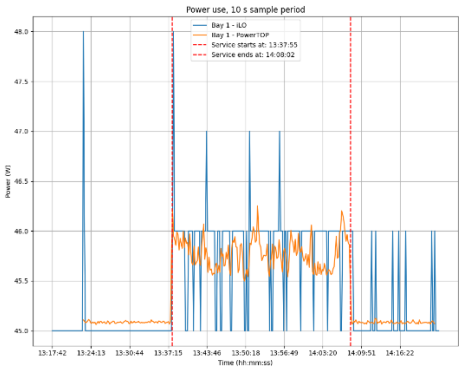

Figure 16. One instance, native operation, baseline 1

2) Two instances

TABLE IX. MEAN POWER USE – TWP SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[15:28:42,15:48:48]$	Before starting the service instance	45.525
$\overline{p^{(iLO)}}[15:48:48,16:18:56]$	During the service instance's operation	46.79
$\overline{p_{dyn}^{(ptop)}}[15:33:47,15:48:48]$	Mean dynamic power before service instances' operation	0.2390
$\overline{p_{dyn}^{(ptop)}}[15:48:48,16:18:56]$	Mean dynamic power during the service instances' operation	1.6653

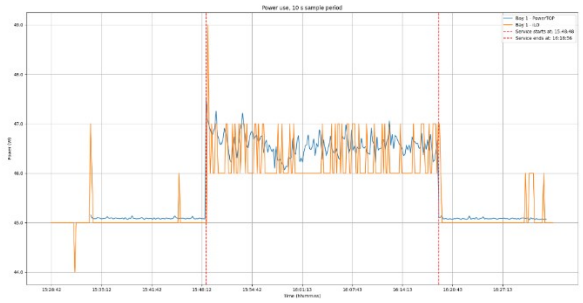

Figure 17. Two instances, native operation, baseline 1

3) Five, ten, twenty, forty and eighty instances

The results for five (Table X, Figure 18), ten (Table XI, Figure 19), twenty (Table XII, Figure 20), forty (Table XIII, Figure 21) and eighty instances (Table XIV, Figure 22) are shown below.

TABLE X. MEAN POWER USE – FIVE SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[17:49:44,18:09:58]$	Before starting the service instance	45.5
$\overline{p^{(iLO)}}[18:09:58,18:40:10]$	During the service instance's operation	47.71
$\overline{p_{dyn}^{(ptop)}}[17:54:57,18:09:58]$	Mean dynamic power before service instances' operation	0.2546
$\overline{p_{dyn}^{(ptop)}}[18:09:58,18:40:10]$	Mean dynamic power during the service instances' operation	3.7906

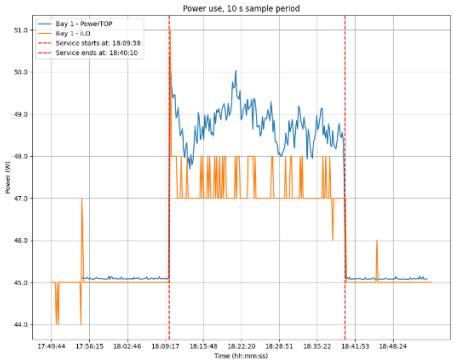

Figure 18. Five instances, native operation, baseline 1

TABLE XI. MEAN POWER USE – TEN SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[19:21:06, 19:41:12]$	Before starting the service instance	45.54
$\overline{p^{(iLO)}}[19:41:12, 20:11:32]$	During the service instance's operation	49.33
$\overline{p_{dyn}^{(ptop)}}[19:26:11, 19:41:12]$	Mean dynamic power before service instances' operation	0.2466
$\overline{p_{dyn}^{(ptop)}}[19:41:12, 20:11:32]$	Mean dynamic power during the service instances' operation	7.4315

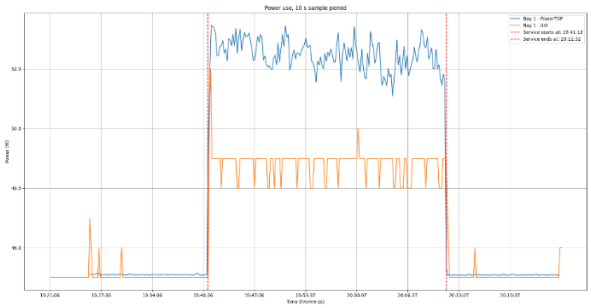

Figure 19. Ten instances, native operation, baseline 1

TABLE XII. MEAN POWER USE – TWENTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[20:52:00, 21:12:09]$	Before starting the service instance	45.52
$\overline{p^{(iLO)}}[21:12:09, 21:42:48]$	During the service instance's operation	51.27
$\overline{p_{dyn}^{(ptop)}}[20:57:09, 21:12:09]$	Mean dynamic power before service instances' operation	0.1819
$\overline{p_{dyn}^{(ptop)}}[21:12:09, 21:42:48]$	Mean dynamic power during the service instances' operation	14.3948

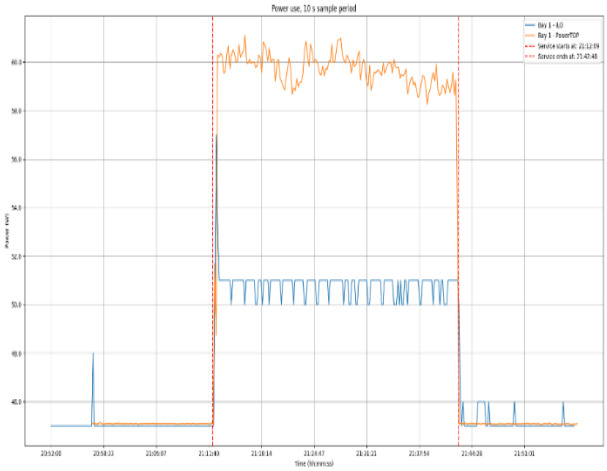

Figure 20. Twenty instances, native operation, baseline 1

TABLE XIII. MEAN POWER USE – FORTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[22:26:05, 22:46:20]$	Before starting the service instance	45.54
$\overline{p^{(iLO)}}[22:46:20, 23:17:43]$	During the service instance's operation	53.27
$\overline{p_{dyn}^{(ptop)}}[22:31:19, 22:46:20]$	Mean dynamic power before service instances' operation	0.1780
$\overline{p_{dyn}^{(ptop)}}[22:46:20, 23:17:43]$	Mean dynamic power during the service instances' operation	19.2853

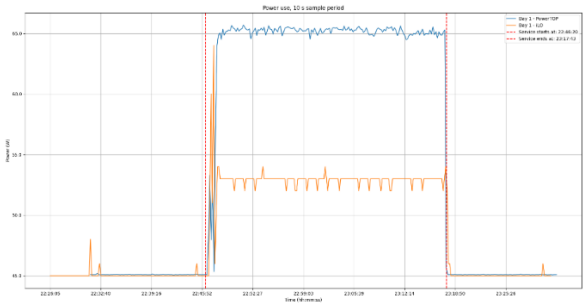

Figure 21. Forty instances, native operation, baseline 1

TABLE XIV. MEAN POWER USE – EIGHTY SERVICE INSTANCES

Power type	Description	Avg. (W)
$\overline{p^{(iLO)}}[00:08:35, 00:28:47]$	Before starting the service instance	45.52
$\overline{p^{(iLO)}}[00:28:47, 01:02:39]$	During the service instance's operation	55.81
$\overline{p_{dyn}^{(ptop)}}[00:13:46, 00:28:47]$	Mean dynamic power before service instances' operation	0.1874
$\overline{p_{dyn}^{(ptop)}}[00:28:47, 01:02:39]$	Mean dynamic power during the service instances' operation	15.1443

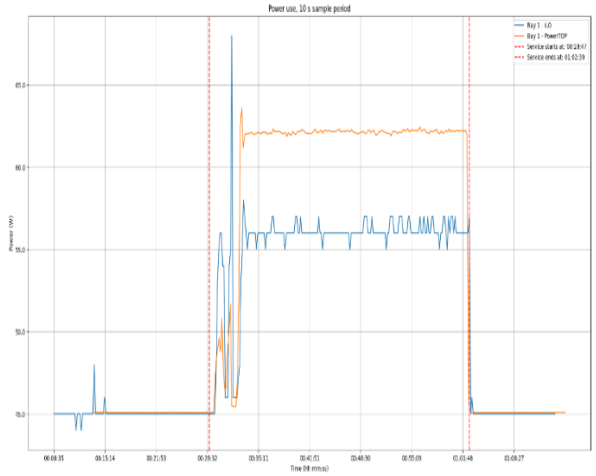

Figure 22. Eighty instances, native operation, baseline 1

F. Breakdown of dynamic power by process, containerized operation

Process power components are sorted in descending order of the mean process power over the period of measurement, until some percentage of the mean total dynamic power over the period of measurement, is obtained. Typical percentages are 50 (the 50th percentile) and 80 (80th percentile). Higher percentiles are avoided, as plots showing the largest power users up to, say, 90% are too dense. Plots and tables are presented to summarize these results.

1) Single instance

Figure 23. 1 instance – process power use, up to 50th percentile

Figure 24. 1 instance – process power use, up to 75th percentile

TABLE XV. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 94TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
[PID 3893] containerd --config /var/run/docker/containerd/containerd.toml	368.73
tick_sched_timer	322.31
[PID 4376] ffmpeg -re -ss 841 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.54
[3] net_rx(softirq)	61.61
[PID 3922] containerd --config /var/run/docker/containerd/containerd.toml	59.21
[PID 3920] containerd --config /var/run/docker/containerd/containerd.toml	53.03
[PID 3898] containerd --config /var/run/docker/containerd/containerd.toml	50.45
[PID 3894] containerd --config /var/run/docker/containerd/containerd.toml	40.07
[PID 3907] containerd --config /var/run/docker/containerd/containerd.toml	39.92
toggle_allocation_gate	38.15671

2) Two instances

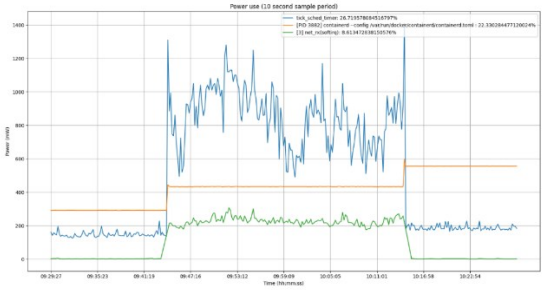

Figure 25. 2 instances – process power use, up to 50th percentile

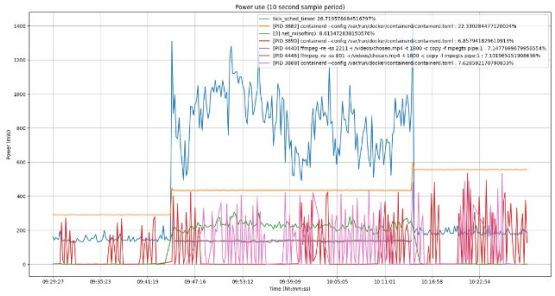

Figure 26. 2 instances – process power use, up to 75th percentile

TABLE XVI. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 87TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	511.72
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	427.66
[3] net_rx(softirq)	114.67
[PID 3890] containerd --config /var/run/docker/containerd/containerd.toml	69.27
[PID 4440] ffmpeg -re -ss 2211 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	68.03
[PID 4446] ffmpeg -re -ss 801 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	67.18
[PID 3888] containerd --config /var/run/docker/containerd/containerd.toml	66.81
[PID 3884] containerd --config /var/run/docker/containerd/containerd.toml	64.70
[PID 18] [rcu_preempt]	58.79
[PID 3887] containerd --config /var/run/docker/containerd/containerd.toml	54.53
[PID 3895] containerd --config /var/run/docker/containerd/containerd.toml	42.70
[PID 3897] containerd --config /var/run/docker/containerd/containerd.toml	38.57
toggle_allocation_gate	38.15
[PID 3898] containerd --config /var/run/docker/containerd/containerd.toml	32.94
[PID 3900] containerd --config /var/run/docker/containerd/containerd.toml	22.35

3) *Five instances*

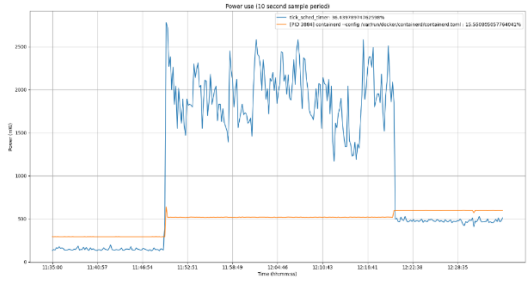

Figure 27. 5 instances – process power use, up to 50th percentile

Figure 28. 5 instances – process power use, up to 80th percentile

TABLE XVII. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 85TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	1123.59
[PID 3884] containerd --config /var/run/docker/containerd/containerd.toml	479.49
[3] net_rx(softirq)	249.06
[PID 17] [rcu_preempt]	99.97
[PID 3886] containerd --config /var/run/docker/containerd/containerd.toml	73.31
[PID 4810] ffmpeg -re -ss 1950 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	67.32
[PID 4808] ffmpeg -re -ss 1629 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.53
[PID 4806] ffmpeg -re -ss 2297 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.24
[PID 4804] ffmpeg -re -ss 1746 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	66.13
[PID 4812] ffmpeg -re -ss 2792 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	65.24
[PID 4098] containerd --config /var/run/docker/containerd/containerd.toml	60.39
[PID 3895] containerd --config /var/run/docker/containerd/containerd.toml	59.62
[PID 3889] containerd --config /var/run/docker/containerd/containerd.toml	49.85
[PID 3894] containerd --config /var/run/docker/containerd/containerd.toml	47.33
[PID 3893] containerd --config /var/run/docker/containerd/containerd.toml	45.44
[PID 3887] containerd --config /var/run/docker/containerd/containerd.toml	44.97

4) *Ten instances*

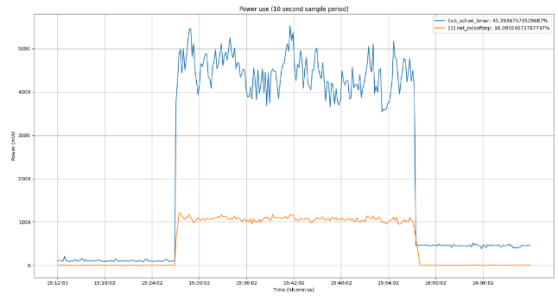

Figure 29. 10 instances – process power use, up to 50th percentile

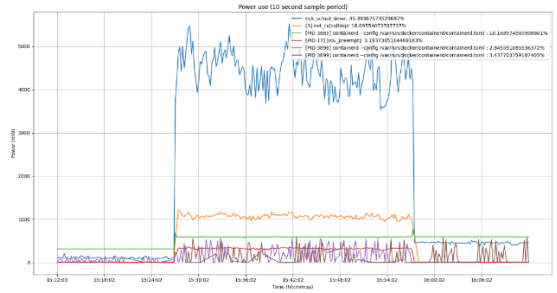

Figure 30. 10 instances – process power use, up to 80th percentile

TABLE XVIII. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 83RD PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	2340.38
[3] net_rx(softirq)	528.77
[PID 3885] containerd --config /var/run/docker/containerd/containerd.toml	524.29
[PID 17] [rcu_preempt]	164.66
[PID 3896] containerd --config /var/run/docker/containerd/containerd.toml	83.20
[PID 3899] containerd --config /var/run/docker/containerd/containerd.toml	76.19
[PID 4102] containerd --config /var/run/docker/containerd/containerd.toml	72.04
[PID 5403] ffmpeg -re -ss 589 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	65.28
[PID 5400] ffmpeg -re -ss 202 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	65.14
[PID 5407] ffmpeg -re -ss 2453 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.91
[PID 5411] ffmpeg -re -ss 121 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.74
[PID 5395] ffmpeg -re -ss 1012 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.58
[PID 5401] ffmpeg -re -ss 235 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.48
[PID 5397] ffmpeg -re -ss 2679 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.42
[PID 5405] ffmpeg -re -ss 1856 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	64.37

5) Twenty instances

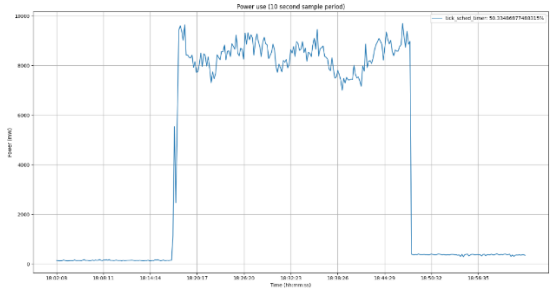

Figure 31. 20 instances – process power use, up to 50th percentile

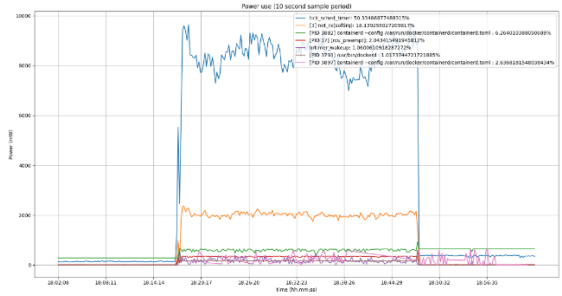

Figure 32. 20 instances – process power use, up to 80th percentile

TABLE XIX. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 78TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	4230.06
[3] net_rx(softirq)	980.63
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	526.42
[PID 17] [rcu_preempt]	171.73
hrtimer_wakeup	89.09
[PID 3793] /usr/bin/dockerd	85.50
[PID 3897] containerd --config /var/run/docker/containerd/containerd.toml	72.29
[PID 6622] ffmpeg -re -ss 2972 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	61.28
[PID 6542] ffmpeg -re -ss 2862 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.94
[PID 6592] ffmpeg -re -ss 893 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.90
[PID 6586] ffmpeg -re -ss 990 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.68
[PID 6550] ffmpeg -re -ss 1550 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.62
[PID 6562] ffmpeg -re -ss 99 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.57
[PID 6616] ffmpeg -re -ss 924 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.53
[PID 6610] ffmpeg -re -ss 2989 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	60.52

6) Forty instances

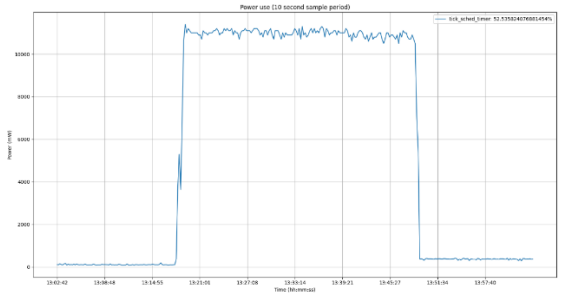

Figure 33. 40 instances – process power use, up to 50th percentile

Figure 34. 40 instances – process power use, up to 80th percentile

TABLE XX. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 80TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	5402.63
[3] net_rx(softirq)	1265.36
hrtimer_wakeup	663.73
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	405.74
[PID 18] [rcu_preempt]	104.85
[PID 3795] /usr/bin/dockerd	100.95
[PID 3899] containerd --config /var/run/docker/containerd/containerd.toml	44.12
[PID 3884] containerd --config /var/run/docker/containerd/containerd.toml	42.31
[PID 3887] containerd --config /var/run/docker/containerd/containerd.toml	40.81
[PID 7927] ffmpeg -re -ss 1413 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.60
[PID 7969] ffmpeg -re -ss 237 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.52
[PID 3898] containerd --config /var/run/docker/containerd/containerd.toml	36.22
[PID 7933] ffmpeg -re -ss 60 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.21
[PID 7993] ffmpeg -re -ss 2988 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.15
[PID 7913] ffmpeg -re -ss 1714 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	36.07

7) Eighty instances

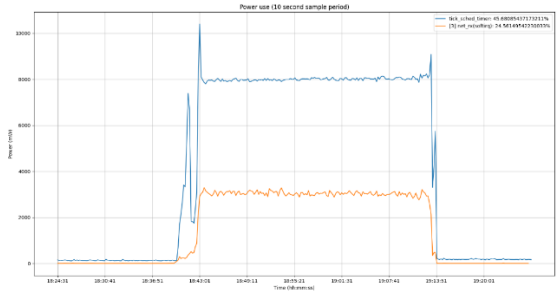

Figure 35. 80 instances – process power use, up to 50th percentile

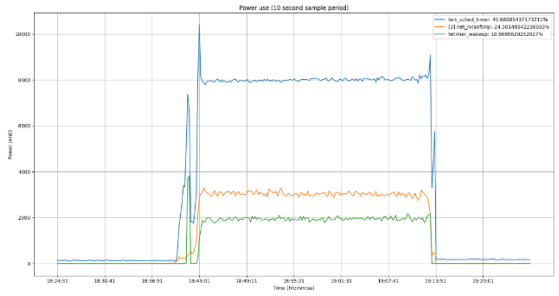

Figure 36. 80 instances – process power use, up to 80th percentile

TABLE XXI. SINGLE INSTANCE: PROCESSES IN DESCENDING ORDER OF MEAN POWER USE, UP TO 79TH PERCENTILE OF TOTAL

Description	PW Estimate (mW)
tick_sched_timer	4038.35
[3] net_rx(softirq)	1449.76
hrtimer_wakeup	935.01
[PID 3882] containerd --config /var/run/docker/containerd/containerd.toml	251.19
[PID 3794] /usr/bin/dockerd	103.59
[PID 18] [rcu_preempt]	68.98
[PID 3881] containerd --config /var/run/docker/containerd/containerd.toml	37.20
[PID 3892] containerd --config /var/run/docker/containerd/containerd.toml	29.13
toggle_allocation_gate	28.56
[PID 3897] containerd --config /var/run/docker/containerd/containerd.toml	24.14
[7] sched(softirq)	23.67
[PID 8262] ffmpeg -re -ss 1711 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	20.19
[PID 3893] containerd --config /var/run/docker/containerd/containerd.toml	19.72
[PID 9089] ffmpeg -re -ss 247 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	19.58
[PID 11433] ffmpeg -re -ss 415 -i /videos/chosen.mp4 -t 1800 -c copy -f mpegts pipe:1	19.02

VII. ANALYSIS

Various characterizations of power use are considered and plotted in Figure 37. In the notation shown below, the (n) symbol indicates dependence of power used on number of streaming containers.

1. total power during operations, $P_{ops}^{iLO}(n)$, and
2. differential total power, where the difference is between operations and quiescence, $P_{ops}^{iLO}(n) - P_{q}^{iLO}$.

Figure 37 illustrates the results in graphical form. The top row of graphs compares total power and differential total power, respectively, for containerized and native operations. The bottom row shows the difference between total power and differential total power. The non-monotonic behaviour seen in the bottom row is due to the error introduced by the rounding of iLO instrumentation.

$P_{b_1}^{(video)}$ (45.4W) was used as the offset for dynamic power obtained using PowerTOP, at every instance count. The value of $P_{b_2}^{(video)}$ (45.99W), obtained by adding the increment in dynamic power inferred by PowerTOP (see Section VI-C), was larger than that measured as $\overline{p^{(iLO)}}[service\_operation\_time]$ for any of the instance counts. If, however, $P_{b_2}^{(video)}$ is obtained in the same way as $P_{b_1}^{(video)}$, by averaging $p^{(iLO)}$ over the relevant period of time, attainment of $P_{b_2}^{(video)}$ supports well the purpose of good known starting and ending events for a run of experimentation.

Dynamic power measurements as a function of streaming videos are not shown in Figure 37, as PowerTOP's measurements do not produce consistent, intelligible results on our platform. Estimates are insufficiently accurate. PowerTOP is capable of capturing power change behaviour (see, notably, Figure 36), but it requires further development before its estimates can be used for quantitative analysis.

On the other hand, notwithstanding PowerTOP's problematic scaling, its capability to capture power change suggests that its relative attribution of power consumption to processes is sound. Basing upon this understanding, it is possible to detect that, notwithstanding the potential to save power consumption through use of tickless kernels, the tick_sched_timer function is easily the largest power consumer (see Figures 27, 29, 31, 33 and 35).

VIII. CONCLUSION AND FUTURE WORK

The objective set out in Section III was to quantify the overhead incurred by operating the video service containerized, instead of as an application running directly on the host operating system (native operation). An access network of the Active Ethernet type was constructed and a video cache deployed in an access node to stream videos to the access node's service area. An implementation model describing the access network was included.

The results obtained have shown that the overhead is negligible and that the benefit of running the video source in a container comes at little cost. The possibility of

consolidating video streaming containers can be pursued with confidence.

No discernable cause for concern was found in the power measurement instrumentation embedded in the HPE Gen9 platform. Documentation on interfacing with the Integrated Lights-Out (iLO) server management was readily available. For detail beyond typical interest, HPE readily divulged information on this tool when contacted for help, including, for example, the method used to round the power measurement into an integer [32].

On the other hand, PowerTOP's accuracy poses a problem. The various graphs of power against time have shown that it captures changes well but significantly overestimates them. In the light of these errors, works that have investigated containerization's overhead with the use of this tool (e.g., [15]) may need to be reviewed for the implications of inaccuracies introduced by the tool, perhaps by using external, physical power meters to calibrate PowerTOP's measurement.

Baselines have been obtained for both the video server and the virtual switch. In particular, P_{b2}^{video} has been found useful in obtaining a reproducible starting point for experiments; to a lesser extent, P_{b1}^{video} has been found useful in providing an offset for power obtained through tools that measure dynamic power. This segues well into an observation that merits particular attention. Even with 80 concurrent streams, the static power has dwarfed the dynamic power. The importance of this observation pertains to the importance of the benefit of containerization as an enabler of consolidation of physical hosts. It can readily be stated that the overhead incurred in providing the **service framework** of containerization poses no obstacle to exploration of exploitation of this benefit.

PowerTOP's limitations invite researchers to explore its causes, as the demand for software power meters is pressing in multi-tenant hosting. Future work would do well to assess the relative accuracy of PowerTOP with Scaphandre [33] prior to embarking on the use of either within power instrumentation.

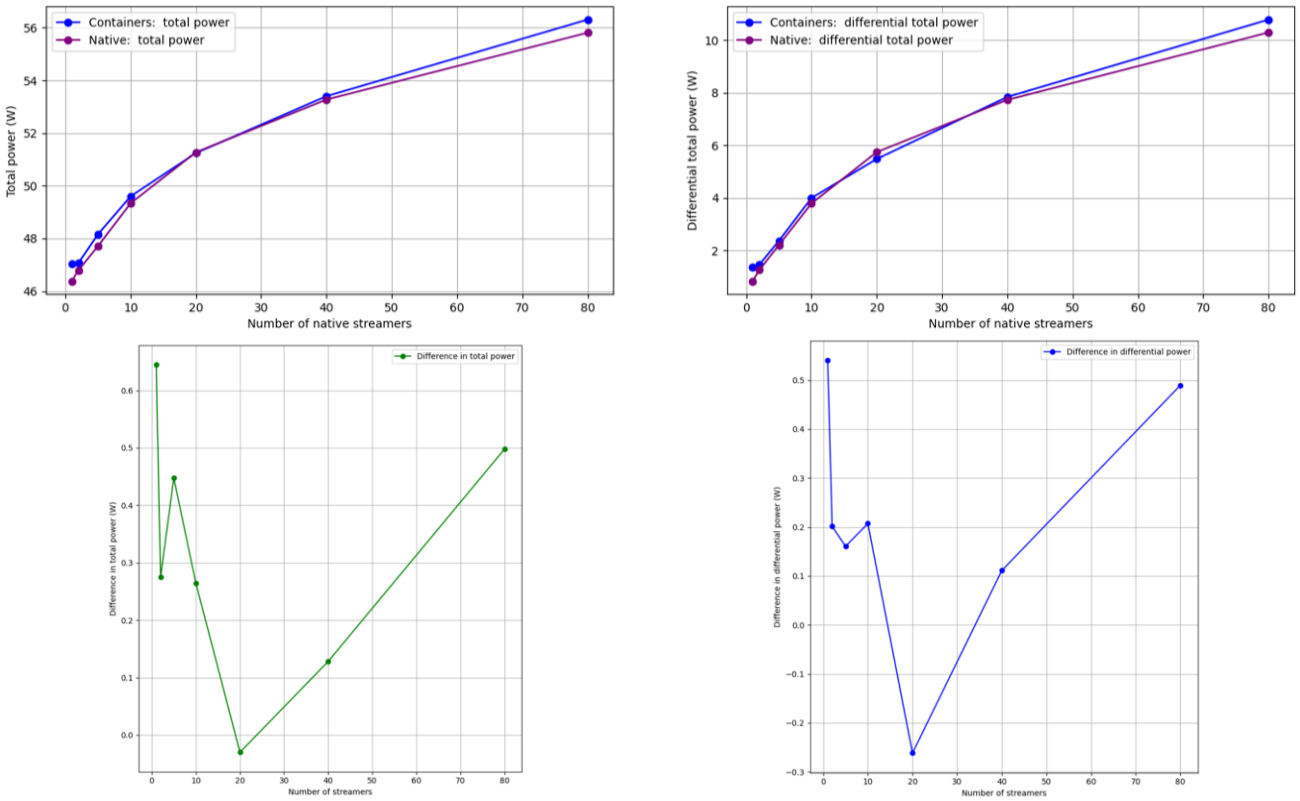

Figure 37. Comparison: native vs containerized streaming. Clockwise from top left: $P_{ops}^{iLO}(n)$, $P_{ops}^{iLO}(n) - P_{q}^{iLO}$, $P_{ops}^{iLO}(n_{cont}) - P_{ops}^{iLO}(n_{native})$ and $\left(P_{ops}^{iLO}(n_{cont}) - P_{q}^{iLO}{}_{cont}\right) - \left(P_{ops}^{iLO}(n_{native}) - P_{q}^{iLO}{}_{native}\right)$.

REFERENCES

[1] E.-V. Depasquale and S. Zammit, 'Containerization's Power Use Overhead in Video Streaming', in Proceedings of the Ninth International Conference on Green Communications, Computing and Technologies (GREEN 2024), Nice, France: IARIA, 2024, pp. 1–9. [Online]. Available: https://www.thinkmind.org/articles/green_2024_1_10_80008.pdf

[2] A. Teker, A. H. Örnek, and B. Canberk, 'Network Bandwidth Usage Forecast in Content Delivery Networks', in 2020 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom), Jul. 2020, pp. 1–6. doi: 10.1109/CoBCom49975.2020.9174180.

[3] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, 'FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers', IEEE Trans. Inf. Theory, vol. 59, no. 12, pp. 8402–8413, Dec. 2013, doi: 10.1109/TIT.2013.2281606.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, 'Cache in the air: exploiting content caching and delivery techniques for 5G systems', IEEE Commun. Mag., vol. 52, no. 2, pp. 131–139, Feb. 2014, doi: 10.1109/MCOM.2014.6736753.

[5] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, 'A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications', IEEE Access, vol. 5, pp. 6757–6779, 2017, doi: 10.1109/ACCESS.2017.2685434.

[6] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, 'A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges', IEEE Commun. Surv. Tutor., vol. 20, no. 1, pp. 416–464, 2018, doi: 10.1109/COMST.2017.2771153.

[7] J. Zhao, P. Liang, W. Liufu, and Z. Fan, 'Recent Developments in Content Delivery Network: A Survey', in Parallel Architectures, Algorithms and Programming, H. Shen and Y. Sang, Eds., Singapore: Springer, 2020, pp. 98–106. doi: 10.1007/978-981-15-2767-8_9.

[8] B. Zolfaghari et al., 'Content Delivery Networks: State of the Art, Trends, and Future Roadmap', ACM Comput. Surv., vol. 53, no. 2, p. 34:1-34:34, Apr. 2020, doi: 10.1145/3380613.

[9] S. Rivoire, P. Ranganathan, and C. Kozyrakis, 'A comparison of high-level full-system power models', in Proceedings of the 2008 conference on Power aware computing and systems, in HotPower'08. USA: USENIX Association, Dec. 2008, p. 3.

[10] E.-V. Depasquale, F. Davoli, and H. Rajput, 'Dynamics of Research into Modeling the Power Consumption of Virtual Entities Used in the Telco Cloud', Sensors, vol. 23, no. 1, Art. no. 1, Jan. 2023, doi: 10.3390/s23010255.

[11] K. Hinton, F. Jalali, and A. Matin, 'Energy consumption modelling of optical networks', Photonic Netw. Commun., vol. 30, no. 1, pp. 4–16, Aug. 2015, doi: 10.1007/s11107-015-0491-5.

[12] W. Lin, H. Wang, and W. Wu, 'A power monitoring system based on a multi-component power model', Int. J. Grid High Perform. Comput., vol. 10, no. 1, pp. 16–30, Jan. 2018.

[13] B. Rountree, D. K. Lowenthal, M. Schulz, and B. R. de Supinski, 'Practical performance prediction under Dynamic Voltage Frequency Scaling', in 2011 International Green Computing Conference and Workshops, Jul. 2011, pp. 1–8. doi: 10.1109/IGCC.2011.6008553.

[14] T. Guérout, Y. Gaoua, C. Artigues, G. Da Costa, P. Lopez, and T. Monteil, 'Mixed integer linear programming for quality of service optimization in Clouds', Future Gener. Comput. Syst., vol. 71, pp. 1–17, Jun. 2017, doi: 10.1016/j.future.2016.12.034.

[15] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, and N. S. Martinelli, 'Analyzing the Power Consumption in Cloud-Native 5/6G Ecosystems', in 2023 IEEE International Conference on Communications Workshops (ICC Workshops), May 2023, pp. 611–617. doi: 10.1109/ICCWorkshops57953.2023.10283755.

[16] Environmental Engineering (EE); Green Abstraction Layer (GAL); Power management capabilities of the future energy telecommunication fixed network nodes; Enhanced Interface for power management in Network Functions Virtualisation (NFV) environments, Sophia Antipolis, France., Jan. 2024.

[17] Multi-service Broadband Network Architecture and Nodal Requirements, TR-178 Issue 2, Broadband Forum., Sep. 2017. [Online]. Available: https://www.broadband-forum.org/technical/download/TR-178_Issue-2.pdf

[18] Hewlett Packard Enterprise, 'HPE ProLiant BL460c Gen9 Server Blade', PSNow. Accessed: Jun. 17, 2024. [Online]. Available: https://www.hpe.com/psnow/doc/c04347343

[19] 'FFmpeg'. Accessed: Jun. 17, 2024. [Online]. Available: https://ffmpeg.org/

[20] 'TSDuck'. Accessed: Jun. 17, 2024. [Online]. Available: https://tsduck.io/

[21] 'Open vSwitch'. Accessed: Jun. 17, 2024. [Online]. Available: https://www.openvswitch.org/

[22] 'index | Alpine Linux'. Accessed: Jun. 17, 2024. [Online]. Available: https://alpinelinux.org/

[23] 'Docker: Accelerated Container Application Development'. Accessed: Jun. 17, 2024. [Online]. Available: https://www.docker.com/

[24] 'Kubernetes Documentation', Accessed: Sep. 03, 2019. [Online]. Available: https://kubernetes.io/docs/home/

[25] DMTF Redfish Forum, Redfish Specification, DSP0266, Apr. 03, 2024. Accessed: Jun. 17, 2024. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.20.1.pdf

[26] 'Powertop - ArchWiki'. Accessed: Feb. 20, 2024. [Online]. Available: https://wiki.archlinux.org/title/powertop

[27] A. van de Ven, 'PowerTOP running average interval and display update interval', Apr. 08, 2024.

[28] A. van de Ven, 'PowerTOP running average interval and display update interval (update)', Apr. 08, 2024.

[29] Ansible Community Documentation, 'User Guide'. Accessed: Jun. 17, 2024. [Online]. Available: https://docs.ansible.com/ansible/latest/user_guide/index.html

[30] A. van de Ven, 'PowerTOP running average interval and display update interval (2nd update)', Apr. 30, 2024.

[31] edepa, edepa/video_streaming_power_use. (Jun. 18, 2024). Accessed: Mar. 15, 2025. [Online]. Available: https://github.com/edepa/video_streaming_power_use

[32] J. Sultana, 'Power measurement - Gen9', May 24, 2024.

[33] 'Introduction - Scaphandre documentation'. Accessed: Mar. 15, 2025. [Online]. Available: https://hubblo-org.github.io/scaphandre/book/