Energy Prediction for I/O Intensive Workflow Applications

Hao Yang, Lauro Beltrão Costa, Matei Ripeanu
University of British Columbia, Vancouver, BC, Canada

ABSTRACT

As workflow-based data-intensive applications have become increasingly popular, the lack of support tools to aid resource provisioning decisions, to estimate the energy cost of running such applications, or simply to support configuration choices has become increasingly evident. Our goal is to design techniques to predict the energy consumption of these workflow-based applications, evaluate different optimization techniques from an energy perspective, and explore energy/performance tradeoffs. In this paper, we propose a methodology to predict the energy consumption for workflow applications that follows a two-step approach: First, we build an analytical energy consumption model to link the energy characteristics of the underlying platform and those of the workflow application. Second, we augment an application performance predictor our group has built with the energy model to enable energy consumption predictions. We use this methodology to create an energy predictor and demonstrate its utility by: (1) exploring the energy impact of performance- and power-centric tuning, and (2) exploring energy/performance tradeoffs for representative workflow applications.

1. INTRODUCTION

Scientific investigation relies increasingly on workflow-based data-intensive applications. These workflow applications are typically assembled using different standalone binaries, which communicate via temporary files stored on a distributed storage system [15]. A workflow management system schedules the tasks resulted from the execution of these binaries based on completion of dependent tasks [18].

In this setup, the performance of the storage system plays a key role in the overall workflow performance [8, 9]. In fact, the storage systems have evolved to incorporate advanced techniques that enable trade-offs over interrelated performance metrics such as throughput, reliability, and generated I/O overhead, to best match the application/deployment scenario at hand [4]. At the same time, user decisions involve allocating resources (e.g., total number of nodes) and configuring the storage system (e.g., choosing the chunk size or the data placement policy). While the trade-off space over traditional performance metrics has been extensively studied over the past decades [9], the focus on energy efficiency is relatively new. Moreover, this aspect has grown in importance due its impact on the cost or even on the feasibility of building large-scale data-centers/supercomputers [6].

This context presents us with two main questions: First, *In what scenarios, if any, existing optimization techniques lead to energy savings?* Second, *What is the performance impact, in terms of time-to-solution, of energy-centric tuning?* Our goal is to answer these questions and additionally build tools to support the user/administrator in the complex and relatively unexplored task of determining the desired balance between application's time-to-solution and its energy bill.

Specifically, the focus of this paper is a mechanism to predict energy consumption of an workflow application given the resource allocation and storage system configuration. This work focuses on RAM-based shared storage system that harnesses node-local storage space. The contributions of this work are: First, we propose a simple analytical energy consumption model that enables adequately accurate energy consumption predictions. This makes it possible not only to estimate energy consumption but also to reason about the relative benefits different system configuration and provisioning decisions offer. Second, we carry out an empirical evaluation of energy consumption using both synthetic benchmarks and real workflow applications. This evaluation quantifies the energy savings of performance optimizations for the distributed storage system as well as the energy and performance impact of power-centric tuning techniques. Third, we demonstrate the predictor's ability to expose energy/performance tradeoffs. Overall, the average prediction accuracy is higher than 85% and the median accuracy is 90% across different scenarios. This accuracy is sufficient to use the predictor to make system configuration choices.

The rest of the paper is organized as follows: Section 2 presents the background for this work. Section 3 discusses the predictor's requirements, the energy consumption model, and our implementation. Section 4 presents the evaluation using synthetic benchmarks and real workflow applications. Section 5 surveys related research, and Section 6 presents discussion and summary.

2. BACKGROUND

This section presents the context in which workflow applications are executed and the performance predictor we build upon.

2.1 Intermediate Storage System

One widely adopted approach to support workflow applications is the *many-task* approach [15]. With this approach workflows use a loose task-coupling model: the standalone executables that compose the workflow communicate through temporary files stored in a shared storage system. Because of the loosely task-coupled model, the tasks can be scheduled in a distributed systems easily as long as the storage is shared.

To avoid the latency to access the backend storage system (e.g., NFS, GPFS) during workflow application executions, recent work [4, 19] proposes using an in-memory shared storage layer as an *intermediate* storage system for inter-task communication. Each compute node that participates in the workflow execution contributes its local storage to form a shared intermediate storage. The workflow runtime engine schedules workflow tasks to the compute nodes that produce intermediate files to the shared storage, and the files are consumed by later workflow tasks.

This relatively simple execution model has allowed assembling complex workflow applications and executing them on large shared-nothing infrastructures. This execution model, however, offers many configuration choices (e.g., data placement poli-

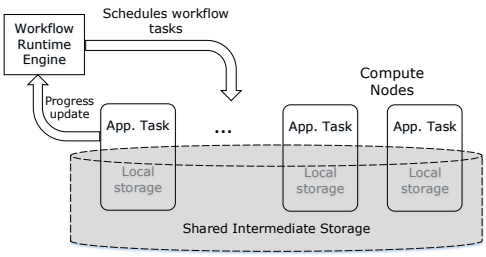


Figure 1: High level architecture of a workflow system. The shared intermediate storage harnesses the storage space of the participating compute nodes and provides a low latency shared storage space. The input/output data is staged in/out from the backend storage.

cies, file chunk size, number of nodes to allocate to storage) [8]. Different workflow applications, however, achieve optimal performance with different configuration choices [4, 9]. Exploring the configuration space via application runs is time consuming and costs substantial computational resources. As a result, there has been increasing demand for optimized configuration and resource provisioning decisions via time-efficient and lightweight approaches. To this end, we have built a performance predictor to efficiently predict the application performance given a certain resource and storage configuration [9]. This paper extends this work and to support energy-oriented decision making.

2.2 The Discrete-event Performance Predictor

This section briefly presents the performance predictor our energy prediction mechanism builds upon (Costa et al. [9] present the full details). The performance predictor takes as inputs a description of the overall workflow composition, a characterization of the workload, the storaged system configuration (e.g., the replication level, system-wide chunk size), and the performance characteristics of the hardware platform. The workload description drives a discrete-event simulation to obtain runtime estimates for each stage of the workflow and for the aggregate runtime. This work augments the predictor to output the detailed runtime breakdown of different states (§3.4).

Workload Description. The workload description is an application trace logged by the distributed storage. The trace contains I/O operations with timestamps, operation size, offset and type as well as application compute times between these operations. The predictor uses this description to simulate the application by replaying the I/O operations and inferring compute times during simulation.

The System Model. Each participating system component is modeled using a service module with its in- and out- queues. An *application driver* emulates the workflow scheduler and replays the application traces for all workflow stages. The *manager* component is responsible to store metadata operations. The *storage* component stores file chunks. The *client* component provides an I/O interface to the application and coordinates the storage components and the metadata manager.

Model Seeding. The predictor uses a lightweight identification process for its key performance-related parameters. A non-intrusive procedure at the client-level (i.e., no changes required to the underlying system) identifies the value to seed storage system's parameters. A script runs a network utility tool (e.g., iperf) to measure the network service times.

3. DESIGN OF THE ENERGY PREDICTOR

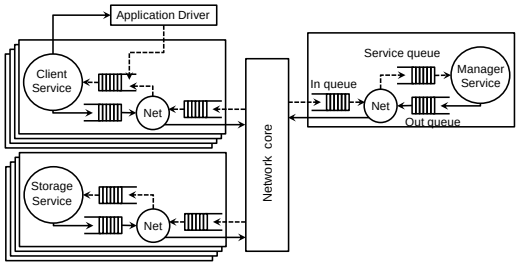


Figure 2: The queue-based model: The application driver replays the workflow trace to emulate the workflow execution [9].

3.1 Requirements

Our goal is to design energy prediction tools that enable evaluating performance/energy tradeoffs and exploring the configuration and provisioning space from an energy perspective. To this end predicting energy consumption should not be time-consuming and should provide adequate accuracy (so that the relative estimates of energy performance tradeoffs are valid among different configurations, thus making it possible to evaluate the tradeoffs among different configuration choices). Additionally, the prediction tools should be simple to use: we aim for tools that do not require complex system instrumentation or seeding measurements.

These considerations make the performance predictor described in §2.2 a good starting point for our energy prediction tools. It is: (i) based on a simple model and seeding mechanism, (ii) effective identification of the desired system configuration, (iii) scales to model a workflow application run on an entire cluster while using over 2000x less resources [9].

3.2 Energy Model Description

A typical stage of a workflow progresses as follows: (i) each node brings the input data from the intermediate storage to memory (likely through multiple I/O operations), (ii) the processor loads the data from the memory and processes it, (iii) the output is pushed back to the intermediate storage.

Based on this observation our energy model associate different phases of the workload with different power profiles: (1) *Idle state* - part of the power is spent simply to keep the node on; (2) *Application processing state* - the node runs application task binaries on the CPU with the data already in memory (once it has already been fetched from storage); (3) *Storage servicing state* - serving read/write requests; and (4) *Network IO state* - performing inter-node data transfers.

We guide the energy usage modeling according to the power consumption profile of each of these states: idle (P^{idle}), CPU processing (P^{App}), storage operations ($P^{storage}$), and network transfers (P^{net}). As we use RAMDisks for the shared storage[1], the I/O operations are mainly operations over RAM.

With this mindset, the total energy spent during a workflow application execution can be expressed as the sum of the energy consumption of individual nodes: $E_{cluster} = \sum_{i}^{N} E_i^{total}$, where N is the total number of nodes, and E_i^{total} is the total energy consumption of node i. For each node the energy usage during a workflow execution is $E_i^{total} = E_i^{base} + E_i^{App} + E_i^{WS}$. The *base energy* is the energy spent to maintain the node active:

[1]This is a common setup for running workflow applications sometimes imposed by the infrastructure (e.g., IBM BG/P nodes are not equipped with hard drives.)

$E_i^{base} = P_i^{idle} * T^{total}$, where P_i^{idle} is the node's idle power and T^{total} is the predicted application runtime. This base portion of the total energy consumption accounts mainly for the non energy-proportionality of the hardware platform. As platforms become increasingly energy proportional, the share of this portion in the total energy envelope will decrease.

The *application energy* is the additional energy the workflow stages consume once the data needed has been fetched. It is modeled by $E^{App} = (P^{App} - P^{idle}) * T^{App}$.

The *workflow system energy* (E_i^{WS}) is modeled by $E_i^{WS} = E_i^{storage} + E_i^{net}$, which is the sum of the energy spent on reading/writing from/to the local storage and sending/receiving data to/from other compute nodes. $E_i^{storage}$ is modeled by $E_i^{storage} = (P^{storage} - P^{idle}) * T_i^{storage}$, where $P^{storage}$ is the node power consumption performing storage operations and $T_i^{storage}$ is the time spent on these operations. Similarly, $E_i^{net} = (P^{net} - P^{idle}) * T_i^{net}$ is the estimation for the energy spent on network transfers. As the performance predictor tracks the network events, it is feasible to estimate the time spent on each read/write data from/to the network.

This linear model requires both power and time input. §3.3 explains how the power parameters are gathered, and §3.4 explains the extensions made to the performance predictor [9] to generate the input for the energy model.

3.3 Energy Model Seeding

To seed the parameters in the energy model, one needs to get the power characteristics of the nodes. We use synthetic workloads that resemble different phases of the workflow application execution to get the power states: (i) P_i^{idle}, we retrieve the power samples when the nodes are idle; (ii) P_i^{App}, we use *stress* [1] to impose load on CPU and measure power; (iii) $P_i^{storage}$, we use local write and read throughput tests; and (iv) P_i^{net}, we perform remote writes from a client to a storage service, and measure the power at the client side. Table 1 shows the gathered power parameters and average values for the cluster we use. These values change when we apply power tuning techniques (described §4.3).

3.4 Implementation

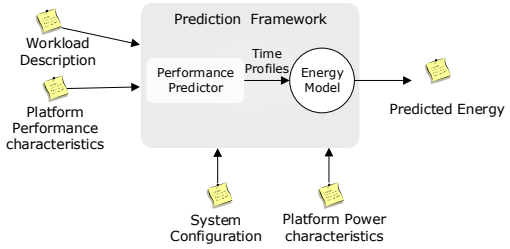


Figure 3: The predictor receives the application description, platform performance characteristics, estimates the time for several events in the system and passes this information to a module that uses power characteristics of the platform and uses the energy model to estimate the energy consumption.

Figure 3 presents the performance predictor and the energy-related additions. We have augmented the predictor to track the time each node spends on the different phases of a workflow task: executing the compute intensive part of the application, writing to/reading from storage, and receiving/sending network data. The energy module added to the performance predictor [9] estimates the energy consumption. This module implements the model described in §3.2, receives the parameters that describes the power consumption of the platform (§3.3) in each different phase of a workflow task, and obtains the estimated time that is spent on different power states from the performance predictor.

4. EVALUATION

We use synthetic benchmarks and real workflow applications to evaluate the accuracy of our energy consumption predictor. The synthetic benchmarks mimic the patterns previous work has identified in real applications [16, 19] and is used to evaluate the prediction accuracy for each pattern independently. The two representative applications we use incorporate multiple patterns to demonstrate the accuracy of the predictor. Additionally, we evaluate the predictor using different configuration choices, power tuning techniques (§4.3) and analyze energy-performance tradeoffs when one varies the size of the resource allocation (§4.4).

Experimental setup. We use an open source distributed storage system [4, 8] for the evaluation because it has multiple configuration knobs (e.g., replication level, chunk size, data placement policy) and we can evaluate their impact on energy consumption. We use the label *DSS* for experiments running a (Default Storage System) configuration: data chunks are striped across storage nodes in a round-robin fashion and no optimization data-placement optimization is used. We label *WOSS* (Workflow Optimized Storage System) the experiments where the system is optimized for a specific workflow pattern (including location-aware scheduling, data-placement, or replication) [3, 8]. The goal of showing results for these two configurations is two-fold: (i) demonstrate the accuracy in a default configuration setting (§4.1.1), and (ii) show its ability to predict energy savings when performance oriented configurations are used (§4.1.2).

Table 1: Platform power parameters for Taurus cluster (These values change when we apply power tuning techniques and evaluate other clusters)

idle power	P_i^{idle}	91.6W
power when stressing CPU only	P_i^{App}	125.2W
power when performing storage operations	$P_i^{storage}$	129.0W
power when doing network transfer	P_i^{net}	127.7W
peak power	P_i^{peak}	225.0W

Testbed and Power Meters. We use 11 nodes from Grid5000 'Lyon' Taurus cluster [7]. Each node has two 2.3GHz Intel Xeon E5-2630 CPUs (each with six cores), 32GB memory and 10 Gbps NIC (the size of infrastructure with power meters limits our evaluation scale). Table 1 shows the gathered power profiles at Taurus. A dedicated node runs the metadata manager and workflow scheduler, while each of the others run the storage service, the I/O client service, and application processes. Each node is connected to a SME Omegawatt power-meter, which provides 0.01W power resolution at 1Hz sampling rate. We aggregate the power consumption, for each node, for the duration of the workflow execution to measure the total energy consumption. The evaluation does not consider the energy consumed by the node that runs metadata service and workflow scheduler as these are fixed and not subject to the configuration changes that we target (e.g., replication level, number of nodes).

Evaluation Metrics. The evaluation focuses on prediction accuracy by comparing the energy consumption and execution time of actual runs and predictions. We report prediction inaccuracy as defined by $I(E) = |1 - E_{pred}/E_{actual}|$. Plots report average of 10 trials with error bars showing the standard deviation.

4.1 Synthetic Benchmarks: Workflow Patterns

The workflow patterns used to evaluate are: pipeline, reduce and broadcast. The synthetic benchmarks involve multiple concurrent clients and storage nodes, and each client performs intensive 'read-process-write' procedures mimicking workflow stages.

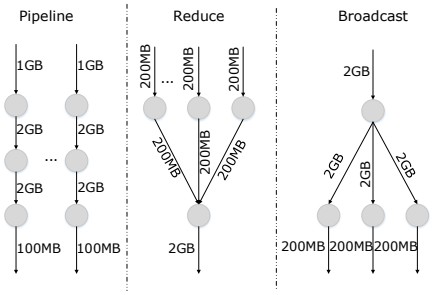


Figure 4: Pipeline, Reduce and Broadcast benchmarks. Circles represent a workflow task performing CPU processing using *stress* [1] and arrows represent data transfers among stages. The labels on arrows represent the file sizes used.

Pipeline benchmark models a set of compute tasks assembled in parallel sequences in such a way that the output of a previous task is the input of a next task in a chain (Figure 4). 10 application pipelines run concurrently on 10 compute nodes and perform three processing stages that read/write files from/to the distributed shared storage and also stress CPU.

Reduce benchmark represents a single task that consumes the outputs produced by multiple computations. 10 processes run in parallel on different nodes and each produces an intermediate result. A following reduce task consumes the intermediate files, and produces the final output.

Broadcast benchmark has a single task producing an output file that is consumed by multiple concurrent tasks. 10 processes run in parallel and consumes the file produced in an earlier stage.

4.1.1 Evaluating Energy Prediction Accuracy on DSS

We first evaluate the predictor's accuracy when the workflow patterns are running on a default storage system (DSS). Left plots in Figure 5(a), 5(b), and 5(c) present the predicted and actual energy consumption for the synthetic patterns. The pipeline benchmark exhibits the best accuracy (only 5.2% inaccuracy) (Figure 5(a)). For reduce the average inaccuracy is 16.4%, while for broadcast it is 15.9%. Overall, the predictions have an average of 12.5% inaccuracy and typically close to one standard deviation interval. Importantly, the predictor response time is 20-30x times faster than running the actual benchmark, resulting in the usage of 200x-300x less resources (machines × time) and showing that our results satisfy the objectives presented in §3.1.

4.1.2 Evaluating Energy Prediction Accuracy on WOSS

In the default storage system configuration (DSS), a round-robin data placement policy is used: the files produced by workflow tasks are striped across all the nodes of the shared storage. Thus, when one task consumes the input files, it needs to connect to other compute nodes and receive file chunks, which generates high network contention and results in suboptimal performance. Past work [3] proposes using file system custom metadata to enable workflow optimizations including moving computation near data, and location aware scheduling. Hence, we evaluate the predictor's ability to capture the energy savings

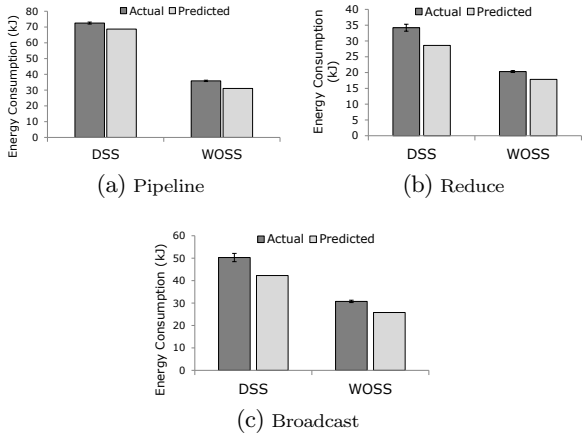


Figure 5: Actual and predicted energy consumption for synthetic benchmark for DSS (left bars) and WOSS (right bars) storage systems.

when using a workflow optimized storage system (WOSS).

In the pipeline scenario, WOSS stores the intermediate pipeline files on the storage node co-located with the application. The workflow scheduler later places the task that consumes the file on the same node to improve performance via data locality. The right plot in Figure 5(a) shows the actual and predicted energy: the predictor achieves 13.4% inaccuracy. In the reduce pattern scenario, WOSS co-places the output files on a single node and exposes their location, so that the scheduler can schedule the reduce task on the machine. The predictor achieves 12.2% inaccuracy. In the broadcast pattern scenario, parallel tasks consume the same file concurrently, which creates an access bottleneck. WOSS creates multiple replicas of the bottleneck file so that the parallel tasks have multiple data access points. Figure 5(c) shows the results using 4 replicas: the predictor achieves 16% inaccuracy. For performance prediction using WOSS configuration the predictor achieves 2.2 - 4.5% inaccuracies.

4.1.3 Summary

The predictor captures the energy consumption for both DSS and WOSS configurations with adequate accuracy and, more importantly, accurately predicts the energy savings brought by WOSS. As a result, the predictor can help users to make storage system configuration decisions based on the energy consumption metric.

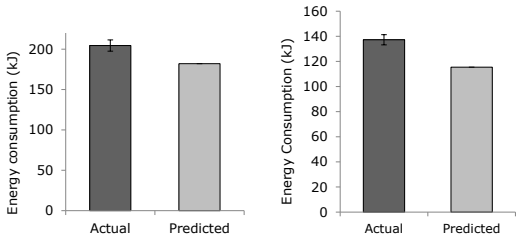
4.2 Energy Predictions for Real Applications

This section evaluates the framework's prediction availability when it is used for real workflow applications: BLAST and Montage. BLAST [5] is a DNA search tool. Each node receives 8 DNA sequence queries as input (a file for each node) and all nodes search the same database file (i.e., BLAST has the broadcast pattern). Montage [14] is a complex astronomy workflow composed of 10 different stages, and a highly variable I/O communication intensity among the workflow stages (Table 2). Additionally, it has a number of distinct workflow patterns (e.g., mProject, mDiff and mBackground are pipelines; mConcatFit and mAdd have reduce). In total, the workload contains around 2000 tasks.

Overall, the energy predictions are more accurate for the real applications than for synthetic benchmarks. This happens because the synthetic benchmarks are designed to produce a high stress on the I/O subsystem, which results in contention

Table 2: Characteristics of Montage workflow stages

Stage	Data	#Files	File Size
stageIn	320MB	163	1.7MB-2.1MB
mProject	1.3GB	324	3.3MB-4.2MB
mImgTbl	50KB	1	50KB
mOverlaps	54KB	1	54KB
mDiff	409MB	895	100KB - 3MB
mFitPlane	1.8MB	449	4KB
mConcatFit	21KB	1	21KB
mBgModel	8.3KB	1	8.3KB
mBackground	1.3GB	325	3.3MB - 4.2MB
mAdd	1.3GB	2	503MB
mJPEG	15MB	1	15MB
stageOut	518MB	2	15MB-503MB



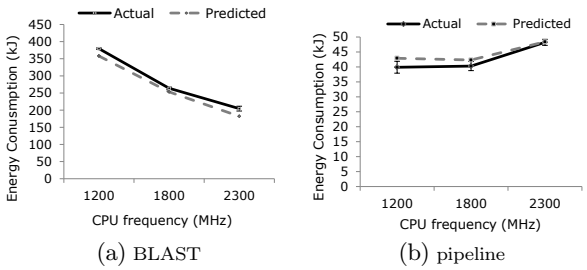
(a) BLAST (b) Montage

Figure 6: Actual and predicted average energy consumption for BLAST and Montage.

and higher variance that is harder to capture when modeling the storage system. Specifically, Figure 6(a) shows 5.2% inaccuracy in energy for BLAST and Figure 6(b) shows 15.9% inaccuracy in energy for Montage. Although the evaluation is not large-scale due to the platform limitation, the workloads discussed are I/O intensive and representative. We previously evaluated the performance prediction using large-scale platform and demonstrated satisfying accuracy [9]. We expect adequate energy prediction accuracy when moving to a larger scale testbed.

4.3 The Impact of Power-centric Tuning

CPU frequency scaling (a.k.a. CPU throttling) is an important energy conservation technique where processors run at less-than-maximum frequency. To evaluate the predictor's ability to predict the energy impact of CPU frequency scaling, we use two types of representative applications: (i) BLAST, with same workload as in the previous section, representing a mix of I/O and CPU intensive applications; (ii) the pipeline benchmark, performing just I/O operations (we reduced to a minimum CPU stress stage), representing an I/O intensive application. We set the processors at different frequencies (1200MHz, 1800MHz and 2300MHz), and for each frequency perform independent seeding.



(a) BLAST (b) pipeline

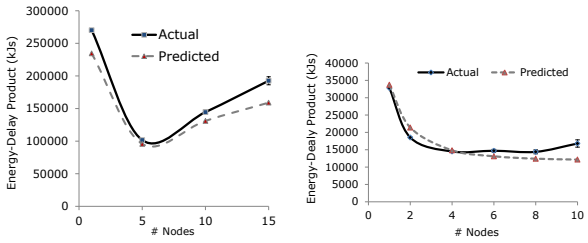
Figure 7: Actual and predicted average energy consumption for BLAST and pipeline for various CPU frequencies. Energy model seeding is performed at each frequency level.

Figures 7 show the actual and predicted energy consumption for BLAST and, respectively for the pipeline benchmark, for different frequencies. Since BLAST is more CPU intensive, using the minimum frequency (1200MHz) just prolongs the runtime and leads to 85.5% more energy consumed than when using the maximum frequency. The predictor accurately estimates the much higher energy cost. For pipeline, using minimum frequency does not increase runtime. In fact, since the instantaneous power is reduced, CPU throttling actually brings energy savings, which is partially captured by the predictions. The actual runs show 17% energy savings, while the predictor estimates 11% savings.

The results for the two workloads highlight that, depending on the computational and I/O characteristics of the workflow application, CPU throttling can bring energy savings or lead to additional energy costs. The predictor provides an effective mechanism to predict the energy consumption when the platform enables power tuning techniques like frequency scaling and can be used in practice to make configuration decisions.

4.4 Predicting Energy-performance Tradeoffs

Another important decision available to users/administrators is the allocation size. As one allocates more compute and storage nodes for executing workflow applications, the performance should improve because of the extra resources. However, due to scale overheads and non-energy proportionality the total energy cost will likely increase. For instance, the Montage workload we evaluated has the lowest energy footprint when using only one node (yet in this case it displays the highest time to solution). A popular hybrid metric that finds a compromise between these two metrics is the energy-delay product (EDP).



(a) (b)

Figure 8: Montage energy-delay product (EDP). (a) Actual and Predicted EDP at Sagittaire. (b) Actual and Predicted EDP at Taurus.

We estimate energy-delay product while varying the number of allocated nodes. Due to platform size limit, we run the same Montage workload used in previous sections on up to 10 nodes at Taurus cluster. We also uses 15 nodes (each of which has two 2.4GHz AMD Opteron CPUs (each with one core), 2GB RAM and 1 Gbps NIC) from Grid5000 Sagittaire cluster. Figure 8 shows the accuracy of the predictor. Our experiments suggest that the predictor can be used to make resource allocation decisions: for Sagittaire the actual and predicted runs both indicate that 5 nodes is the best choice; for Taurus the actual runs indicate that using 8 nodes gives the best EDP for the workload we use, and the predictor suggests that 8 - 10 nodes are good choices.

5. RELATED WORK

Previous work addressed modeling the full system power. For example, Economou et al. [10] presents a non-intrusive method for modeling full-system power consumption based on the idle

power and utilization metrics. Soft-Watt [13] provides low-level analytical models tied to architectural events and uses simulations to predict power consumption. These more granular models, however, typically lead to a longer prediction time and are often highly coupled to the underlying architecture. Additionally, some parameters of the model are hard to obtain unless one executes the application in every possible configuration.

Some work use analytical models to evaluate the energy efficiency of distributed scientific applications. Feng et al. [11] developed a predictor for profiling power and energy characteristics of parallel scientific applications. Ge et al. [12] have similar scope as this paper. The main difference from this paper is that Ge et al. [12] focus on parallel applications while we focus on the distributed storage layer of workflow applications that have various patterns and much more I/O operations. We use discrete event based simulation to obtain the runtime and obtain higher accuracies.

Similar to our approach, past work uses simulation instead of only analytical modeling. For instance, OMNeT++ [17] provides general and accurate network modeling, while DiskSim [2] can model the storage devices at the hardware level. These tools could be modified and integrated to build a detailed simulator. However, due to the low component level simulation, they often lack fast time to solution. Our approach has achieved reasonable accuracy while remains lightweight and provides a simple and effective way to seed the model [9].

6. DISCUSSION AND SUMMARY

What are the Causes of Inaccuracies? Although some inaccuracy is expected from the simplicity of the model and its seeding mechanism as mentioned in §3 (e.g., the energy model does not capture scheduling overleads and metadata operations, which could lead to underprediction), it is important to discuss in more depth the sources of inaccuracies. They fall in three main categories: first, the cluster used in the evaluation shares the same networking switch with two other clusters, thus interference can impact the accuracy of the seeding measurements. This factor can be addressed by having more exclusive reservations to limit network interference. Second, despite that the nodes in the cluster used for evaluation are homogenous machines, they can have different performance and power profiles. For instance, the idle power can vary around 5%. The seeding process can be performed on multiple nodes to obtain the average profiles. The third source, and more important in this context, is attributing inaccuracies precisely to time or to energy modelling. One approach to validate the inaccuracy source is to compare the energy prediction results between giving real time inputs to the energy predictor and giving predicted times to the predictor. Consider the synthetic benchmark results on DSS, the energy predictor achieves 5% inaccuracy for the pipeline benchmark, while for the reduce benchmark the inaccuracy is 16%. For the pipeline scenario, the predicted times and actual times are close (within 1%), thus the final 5% inaccuracy can be attributed to the energy model. For the reduce benchmark, the overall predicted time is underestimated by 6%, which leads to energy underprediction. When giving the actual time inputs to the energy model, the prediction inaccuracy is reduced to 9%. Thus, out of the 16% inaccuracy, 9% can be attributed to the energy model and 7% can be attributed to the time prediction.

What to do to improve accuracy? As discussed previously, the nodes in a cluster can have different characteristics. One approach to increase accuracy is to perform performance and power seeding process on multiple nodes and obtain the average power value per state. The energy model captures the major execution states, however, it does not include the energy spent on metadata path and workflow scheduling. Implementation the energy cost of those operations can improve the prediction accuracy. Since the energy predictor requires time estimates from the performance predictor, and good time estimates lead to accurate energy prediction. As suggested by Costa et al. [9], currently the performance predictor does not capture workflow scheduler overheads, the workflow task launch overheads. Modeling these overheads can improve the accuracy of the time to solution, as well as the spent energy.

Optimizing for Time vs. Optimizing for Energy To optimize for time, one can use optimized storage configuration or add more resources. The former approach usually exploits data locality and location-aware scheduling, which generally reduces the amount of data transfers, thus reduces the energy costs as well. Due to non-energy proportionality of the state-of-the-art platforms, idle power remains a large portion of the total power consumption. Increasing the allocation size of a workload could improve performance at the cost of spending more energy. The experiments (§4.3) demonstrate that for a subclass of applications, it is additionally possible to optimize for energy only by using power-tuning techniques like CPU throttling. However, these techniques need to be carefully considered, as they can bring energy savings or lead to additional costs depending on the specific application patterns. The proposed energy prediction tool is useful to support this type of decisions.

Summary This work augments a performance predictor to obtain energy consumption estimates for workflow-based applications. We evaluate the accuracy of the predictor using synthetic benchmarks that represent common data access patterns and two real world workflow applications. Additionally, we demonstrate the ability of the predictor to support choosing between different storage configurations, to support configuration decisions for processor frequency scaling, and for resource provisioning. Overall, our experiments demonstrate that the predictor is a low cost, time-efficient tool for evaluating power-tuning techniques that target a multitude of scenarios and success metrics (e.g., energy, energy-delay product).

7. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This project was supported in part by the DoE ASCR XStack program (ER26013) and by the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

[1] Stress. `http://people.seas.harvard.edu/~apw/stress`.

[2] DiskSim. `http://www.pdl.cmu.edu/DiskSim/`.

[3] S. Al-Kiswany et al. The case for cross-layer optimizations in storage: A workflow-optimized storage system. *CoRR*, abs/1301.6195.

[4] S. Al-Kiswany et al. The Case for a Versatile Storage System. *SIGOPS Oper. Syst. Rev.*, 44, March 2010.

[5] S. F. Altschul et al. Basic local alignment search tool. *Journal of molecular biology*, Oct. 1990.

[6] C. L. Belady. In the Data Center, Power and Cooling Costs more than the IT Equipment it Supports.

[7] F. Cappello et al. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *Grid Computing Workshop*, 2005.

[8] L. Costa et al. The case for workflow-aware storage: An opportunity study. In *Journal of Grid Computing*.

[9] L. B. Costa et al. Supporting Storage Configuration for I/O Intensive Workflows. In *ICS2014*.
[10] D. Economou et al. Full-system power analysis and modeling for server environments. In *MOBS'06*.
[11] X. Feng, R. Ge, and K. W. Cameron. Power and Energy Profiling of Scientific Applications on Distributed Systems. In *IPDPS '05*.
[12] R. Ge et al. Modeling and evaluating energy-performance efficiency of parallel processing on multicore based power aware systems. In *IPDPS'09*.
[13] S. Gurumurthi et al. Using complete machine simulation for software power estimation: the softwatt approach. In *HPCA'02*, Feb. 2002.
[14] J. C. Jacob et al. Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. *Int. J. Comput. Sci. Eng.*, 2009.
[15] I. Raicu et al. Many-Task Computing for Grids and Supercomputers. In *MTAGS08*.
[16] T. Shibata et al. File-Access Patterns of Data-Intensive Workflow Applications and their Implications to Distributed Filesystems. In *HPDC '10*.
[17] A. Varga. Using the OMNeT++ Discrete Event Simulation System in Education. *Education, IEEE Trans. on*, 42(4), 1999.
[18] M. Wilde et al. Swift: A language for distributed parallel scripting. *Parallel Computing*, 37(9), 2011.
[19] J. M. Wozniak et al. Case Studies in Storage Access by Loosely Coupled Petascale Applications. In *Proc. of PDSW '09*, 2009.