Modeling Power Consumption of Lossy Compressed I/O for Exascale HPC Systems

Grant Wilkins
Holcombe Department of Electrical
and Computer Engineering
Clemson University
Clemson, South Carolina 29634
Email: gfwilki@clemson.edu

Jon C. Calhoun
Holcombe Department of Electrical
and Computer Engineering
Clemson University
Clemson, South Carolina 29634
Email: jonccal@clemson.edu

***Abstract*—Exascale computing enables unprecedented, detailed and coupled scientific simulations which generate data on the order of tens of petabytes. Due to large data volumes, lossy compressors become indispensable as they enable better compression ratios and runtime performance than lossless compressors. Moreover, as (high-performance computing) HPC systems grow larger, they draw power on the scale of tens of megawatts. Data motion is expensive in time and energy. Therefore, optimizing compressor and data I/O power usage is an important step in reducing energy consumption to meet sustainable computing goals and stay within limited power budgets. In this paper, we explore efficient power consumption gains for the SZ and ZFP lossy compressors and data writing on a cloud HPC system while varying the CPU frequency, scientific data sets, and system architecture. Using this power consumption data, we construct a power model for lossy compression and present a tuning methodology that reduces energy overhead of lossy compressors and data writing on HPC systems by 14.3% on average. We apply our model and find 6.5 kJs, or 13%, of savings on average for 512GB I/O. Therefore, utilizing our model results in more energy efficient lossy data compression and I/O.**

I. Introduction

As high-performance computing (HPC) approaches exascale, computing centers consume significant amounts of power, on the scale of tens of MWs [1]. These HPC systems run highly complex applications that evolve large volumes of data. Scientific workloads now require moving data for analysis and processing, yet large-data I/O is a computationally expensive operation. For example, Hardware/Hybrid Accelerated Cosmology Code (HACC) [2], can generate enough snapshots to require 10 hours to transmit at a 500 GB/s bandwidth [3]. To mitigate I/O bottlenecks, applications use lossy compressors such as SZ [4]–[7] and ZFP [8] to compress floating point data, reduce the storage size, and save I/O time. Lossy compressors have the advantage of better space-savings and runtime efficiency over lossless compressors, making them more desirable for compressing petabytes of floating-point data.

However, compressing large datasets can require a non-trivial amount of time. Consider the situation of needing to write a large amount of data. For dumping data one can compress and then write the compressed data, or simply transmit without compressing. I/O runtime savings on data-dumping can be up to 25% with lossy compression, yet there are cases where the compression itself can outweigh the runtime for reading and writing the compressed data [3]. A smaller, compressed file transports faster, but how long it takes to compress that data can mitigate those advantages.

How does this affect the energy involved in the I/O? That is, if compression takes significant runtime, how does that affect how much energy is drawn for the I/O operation? Assuming single CPU compression and I/O, the CPU usage and clock frequency of the cores directly correlates with the energy-draw of that CPU [9]. Since power draw is a given for any system, our goal is then to minimize the compression and data transit energy usage by lowering the clock frequency of the chip.

The motivation for focusing on compression and I/O with respect to CPU frequency is that in HPC workloads, I/O is not as performance-sensitive as a simulation running. As one lowers CPU frequency, applications execute more slowly despite saving energy. When a user runs simulations, one needs the full CPU power as these jobs are computationally exhaustive and time can be limited due to competitive resource schedules. On the other hand, compression and I/O can afford a longer runtime in many use cases, as this data transit process is several orders of magnitude shorter than most scientific simulations. Therefore, finding where runtime and energy consumption are optimized via trade-off based on the CPU frequency is possible in the context of compression and data dumping.

In this paper, we measure the energy usage of SZ and ZFP compressing floating point data on a cloud HPC system with a network file system (NFS). Also we measure the energy usage of moving different-sizes data on an NFS. With these results we construct models of energy and average power that provide a CPU frequency tuning framework to optimize energy usage for lossy compression and data transmission, characterizing an I/O workflow. Using our models we demonstrate energy-savings in the use-case of lossy compression and then transmitting data.

Our contributions in this paper are as follows:

- We derive and compare power consumption models of SZ and ZFP to enable CPU-frequency tuning to find optimal power and energy consumption.

- We provide recommendations to reduce energy usage of lossy compression in I/O by 19.4% on average by lowering the CPU frequency by 12.5%.
- We construct a model for data transit power consumption and achieve reduced energy usage by 11.2% on average by lowering the CPU frequency by 15%.
- We utilize our frequency tuning models for energy-optimizing lossy compression and data transmission and demonstrate 14.3% overall energy savings.
- We apply our model to a real-world example of compressing and writing 512GBs of floating point data on an NFS, saving 6.5 kJs on average, which equates to 13% energy savings.

We organize our paper as follows. In Section II we describe related works to our project, as well as motivation concerning past studies in energy-efficiency, lossy compression, and data movement. In Section III we outline the different methods, softwares, and datasets we use in our experiments. In Sections IV, V we develop our model of power consumption for lossy compression and data transit, respectively. In Section VI we discuss how we optimize the power consumption of I/O using our models. Section VII then shows use-case driven examples of tuning an HPC system for energy-savings.

II. Related Work

A. Lossy Compression and Data Dumping

As HPC systems approach exascale, the question of how to transport data on cloud systems for analysis becomes important. Lossy compressors, such as SZ [4]–[7] and ZFP [8] provide the advantage of decreasing size with impressive compression ratios. Lossy compressors achieve this by eliminating redundancies in chunks of scientific, floating-point data via encoding the binary representation of each datum. Based on the granularity required, an error bound can be set to preserve more or less data. These space savings make lossy compression a valuable tool in scientific computing, particularly in data movement.

Previous work [3] has shown the possibility for lossy compression to decrease transit time on an I/O-bound operation. Liang et al. focused on the different configurations of SZ and ZFP in the context of runtime optimization for transporting large datasets, finding time-savings when compressing before disk writing. A primary interest in data dumping is sending a small number of larger files over a large number of smaller files due to I/O bottlenecking in HPC systems [10]. Lossy compression on large scientific datasets achieves this by retaining the number of files and reducing their size, rather than increasing the number of files with smaller size. Using lossy compression on large scientific datasets, our study looks to take a similar approach, however taking energy savings in place of time-savings for transporting large volumes of data via tuning CPU frequency.

B. Energy Optimization Techniques in HPC

Large supercomputers in the age of exascale computing can have a power rating on the order of tens of MWs [11]–[13]. One approach in particular is CPU frequency scaling, as previous work in [11], [12] have shown to be effective. These studies focus on how dynamic voltage and frequency scaling are used to achieve energy savings, much like what we work to achieve in our study. Mòran et al., have proven success in not only optimizing the energy consumption of checkpoint/restart systems but also parameterizing it. Using CPU frequency as an independent variable, they explore different experimental settings to then characterize the system's energy consumption. Other work [13] demonstrates similar results to Mòran, finding a critical power slope. This curve shape denotes a sharp exponential increase in power over the range of CPU frequency, meaning energy savings can be achieved by lowering CPU frequency.

Similarly, estimation of energy consumption proves to be a useful tool in optimizing the energy efficiency of an HPC system [14]. Studies which perform this analysis focus on surveying a large number of energy optimized algorithms to then select, per-system, which algorithm is the best fit for a specific use case [14]. With this study we add the perspective of energy estimation in the context of lossy compression and data dumping towards exascale systems, adding a technique for that area of software.

In our work, we use frequency scaling for energy reduction in the context of lossy compression. We focus on parameterization and also a discussion about how to optimize I/O energy usage with efficient lossy compression. We look to evaluate the impact that the CPU clock frequency can have on the power usage of the HPC system.

III. Methods

The total energy, E_{total}, of a process is a product of average power, P_{avg}, and runtime, t_{run}:

$$E_{total} = P_{avg} \cdot t_{run}. \tag{1}$$

To establish a model for power consumption, one needs to measure the energy and runtime of that operation over a range of frequencies. Using `perf`, a Linux performance-measurement tool, we sample the total energy and runtime of compression. In our case, we model power consumption of I/O by recording energy and runtime of compression and data transmission.

We generalize our power experiments for compression with the usage of (1) the SZ and ZFP lossy compressors with different error bounds, (2) CPU frequency scaling, (3) data size and dimension to compress, and (4) hardware specifications for our experimental platforms.

A. Lossy Compressors and Error Bound

We use two leading HPC lossy compressors SZ and ZFP. ZFP [8] compresses by transforming floating-point data to fixed-point values block-by-block and adopts an embedded coding to encode generated coefficients. SZ [4]–[6] compresses blocks of data using a series of steps: data prediction, error quantization, Huffman encoding, and lossless compression. An important component of lossy compression is the

error bound: the tolerance of how well the data should be reconstructed at decompression. A smaller error bound in general yields lower compression ratios and is more runtime expensive.

We use the absolute error bound in SZ and the ZFP fixed-accuracy mode, both of which bound the compression error at a fixed level. The error-bounds we target for both compressors are $1e{-}1, 1e{-}2, 1e{-}3, 1e{-}4$. These bounds are used to generalize compressor use-cases for the energy-tuning model, as different users utilize different granularity depending on the level of accuracy needed in data reconstruction.

B. CPU Frequency

We set the CPU frequency for all CPU cores using the Linux system call, `cpufreq-set`. Changing the CPU frequency affects the amount of voltage supplied to the chip. The range of frequencies we consider in this study goes from the the minimum clock speed (800MHz) to the maximum clock frequency for the given CPU with a step size of 50MHz. This step size was chosen as it would provide a sufficient granularity to notice trends in the total energy measured for each compression job conducted.

C. Data Characteristics

The dimensionality and size of data has a large impact on compression and data I/O. For example, a 10GB 4-D array of floating-point data is generally more difficult to compress than a 100MB 1-D array [15], because there are more points and dimensions to encode and process. SDRBench [16] provides floating-point, scientific data sets to assist in benchmarking compressor performance. In Table I we summarize the characteristics of the data we compress.

Domain	Dimensions	Size of Fields
CESM-ATM [17]	$26 \times 1800 \times 3600$	673.9MB
HACC [2]	1×280953867	1046.9MB
NYX [18]	$512 \times 512 \times 512$	536.9MB

TABLE I
DATA SETS CONSIDERED IN STUDY

We utilize diverse data as it better generalizes the runtime and total energy the system consumes for a given compression operation. This step ensures greater ubiquity of results for scientific, floating-point data.

Note that for data transmission, only the size of the data matters. Therefore, for our experiments involving transmitting different chunks of data, we vary the size without worrying about dimension or domain.

D. CloudLab Hardware

We utilize different hardware platforms to better generalize our power consumption results. CloudLab is a cloud system which provides HPC nodes with remote, root access [19]. All experiments are run using CloudLab on the m510 and c220g5 node types. We summarize the pertinent, single-core hardware specifications in Table II.

CloudLab	CPU	CPU Min - Base Clock	Series
m510	Xeon D-1548	0.8GHz - 2.0GHz	Broadwell
c220g5	Xeon Silver 4114	0.8GHz - 2.2GHz	Skylake

TABLE II
HARDWARE UTILIZED

IV. MODELING POWER CONSUMPTION

In this Section, we develop models of lossy compressor and data writing power consumption using regression. This model has the purpose of providing an objective function to minimize the amount of energy lossy compression and data dumping consume. With a model parameterized in terms of CPU frequency, we predict and therefore simulate compressor behavior *ab initio*. We then understand how a system's power and runtime behaves with respect to compression and data writing, informing energy-savings for HPC system-users. First, using the data collected from compressing data at different frequencies with SZ and ZFP, we develop models for the power draw of lossy compression. Then we perform the same analysis for the power draw of writing data on an NFS.

A. Proposed Models for Compression Power Draw

Our goal is to create a general model for lossy compressor power consumption that provides the ability to estimate trade-offs in power and runtime for energy-efficiency. To achieve this we collect energy and runtime results for lossy compression. In our experimental setup, we performed single-core SZ and ZFP compression on the datasets in Table I on the two nodes in Table II. We perform compression on the range of CPU frequencies in Table II with a step size of 50MHz. To ensure our model accounts for many different use cases we compress the data with four error bounds, as outlined in Section III-A. Finally, to ensure validity we also repeat each compression 10 times per frequency step and average the results.

Since we use two compressors on two sets of hardware, we evaluate the five models of power draw which combine these variables. For each model we either regressed one compressor with both chips, or two compressors on a singular hardware configuration. We detail our five models for power consumption in Table III.

Model Data	Compressor(s)	CPU(s)
Total	SZ, ZFP	Broadwell, Skylake
SZ	SZ	Broadwell, Skylake
ZFP	ZFP	Broadwell, Skylake
Broadwell	SZ, ZFP	Broadwell
Skylake	SZ, ZFP	Skylake

TABLE III
MODELS PRODUCED FOR TUNING

As shown in Figures 1 and 2, the power consumption data with respect to frequency is non-linear. Choosing the correct non-linear model is a matter of minimizing root-mean-squared-error (RMSE), sum-of-squared-error (SSE). Note that R^2 is not always accurate for non-linear modeling; however, it can still explain model variance [20]. From observing Figure

2, it is clear that there is a constant region with a sudden jump. We use the MATLAB™ Curve Fitting Toolbox [21] to find the model of frequency versus power, sliced along the partitions outlined in Table III. This toolbox finds the most optimal model, minimizing SSE and RMSE. In this case all equations correspond with the following equation

$$P_{fit}(f) = af^b + c, \tag{2}$$

where f is CPU-frequency, and a, b, c are model fit parameters.

In Table IV we present the models for power consumption of lossy compresssion, along with the goodness of fit metrics (GF), demonstrating how well each model predicts the power usage of compression.

Model Data	$P_{Compress}(f)$	SSE	RMSE	R^2
Total	$0.0086f^{4.038} + 0.757$	11.407	0.0442	0.5771
SZ	$0.0107f^{3.788} + 0.754$	5.964	0.0441	0.5864
ZFP	$0.0062f^{4.414} + 0.7589$	5.359	0.0440	0.5725
Broadwell	$0.0064f^{5.315} + 0.7429$	2.463	0.0279	0.8731
Skylake	$2.235\text{e}{-9}f^{23.31} + 0.7941$	1.372	0.0226	0.8185

TABLE IV
MODEL EQUATIONS AND GF FOR COMPRESSION

From the models and GF in Table IV, the Broadwell and Skylake power consumption models have a lower SSE and RMSE and an R^2 closer to 1, meaning power consumption is less dependent on the choice of lossy compressor. Therefore, power consumption results and energy-savings should turn towards hardware specifications when using the model-based tuning technique.

B. Proposed Models for Data Writing Power Draw

Here we continue our modeling of power consumption and extend these results to modeling data transit in terms of CPU frequency. With a power consumption model for data writing, we can optimize the energy usage of sending data over a NFS.

In our experiments for data transit, we measure the energy of transmitting floating point data. First, we allocate fixed sizes of data from 1GB to 16GB and then we copy that data to an NFS, using single-core data movement. During the data transport we measure the total energy and runtime. We perform this same operation over a range of frequencies for the single-core processes on both the Skylake and Broadwell chips. To reduce variance in results, we repeat each data transit at each frequency-step 10 times, averaging the results.

In our experiments, we vary the data size being written and the chip where we perform our tests. To find a regression model for power consumption, we must consider the power results from each CPU separately. Staying consistent with compression, our models are (1) all of the aggregated power results for data transit on both chips, (2) the power results from just the Broadwell chip, and (3) the power results from just the Skylake chip.

Regression of the power comes from plotting frequency and power from these different data partitions in the MATLAB™ Curve Fitting Toolbox [21]. In the case of our models for power consumption of data transit, this was of the same form as Eqn. 2.

Model Data	$P_{Data}(f)$	SSE	RMSE	R^2
Total	$0.0133f^{3.379} + 0.7985$	0.8446	0.05631	0.4361
Broadwell	$0.0261f^{3.395} + 0.7097$	0.03423	0.01675	0.9578
Skylake	$9.095\text{E}{-9}f^{20.9} + 0.888$	0.07875	0.02355	0.5992

TABLE V
MODELS AND GF DATA TRANSIT

From Table V we notice that the SSE and RMSE are minimized for the CPU specific models, as compared to combining the results. This implies that data transit power savings should be modeled on a hardware to hardware basis. The lower values of error do show that one can find the optimal frequency for power over the range of available frequencies per chip. The low R^2 for the Skylake data transit model is persistent among repeated tests, and shows that R^2 is an inconsistent metric for non-linear regression, as statistics like SSE and RMSE better characterize the error in a given model.

Using the models found in Tables IV and V, we are able to optimize data I/O. Treating I/O in two steps as compression and then data writing, different optimal frequencies are chosen, creating a piecewise model for I/O optimization.

V. POWER OPTIMIZATION OF COMPRESSED I/O

We propose the use case of tuning a CPU to achieve a more efficient power-state during compression and I/O for an HPC workflow. As noted in Sections IV-A and IV-B, the choice of the hardware has the greatest influence on the model of power. In this Section, we analyze the energy consumption and runtime results to show how one can use the power model to improve I/O on a per CPU basis.

A. Characteristic Plot Results Analysis

Energy consumption and power draw can vary in magnitude on different systems. For example, the Intel Xeon D-1548 has a thermal design power (TDP) of 45W, as compared to the Intel Xeon Silver 4114's 85W TDP [19]. To make power and energy results comparable, we scale power-usage and energy consumption down by the max clock frequency's power consumption and total energy, respectively. Scaling down the power results has the advantage of putting all of the power on the same range as a percentage. Power as a percentage better illustrates the change in power over increasing clock frequency.

We note that for consistency in comparison, we also scale the runtime by the compression runtime at the max clock frequency. This has the added advantage of discerning the impact of a lower clock frequency on the runtime. Therefore, all results presented are in terms of their scaled value, not their magnitude, easing the analysis of changing clock frequency.

We also note that in Figures 1 and 2 we display the error bounds we compressed at, yet the trends are close to indiscernible as we found no significant difference between these results when scaling the power and runtime by the peak

values on their range. However, one should note that there is a difference based on error bound in the magnitude of energy consumed, as shown in Figure 6. Similarly, we found no significant difference in the power consumption or runtime based on data size for Figures 3 and 4 after scaling results. Therefore, we do not show the different lines for data size.

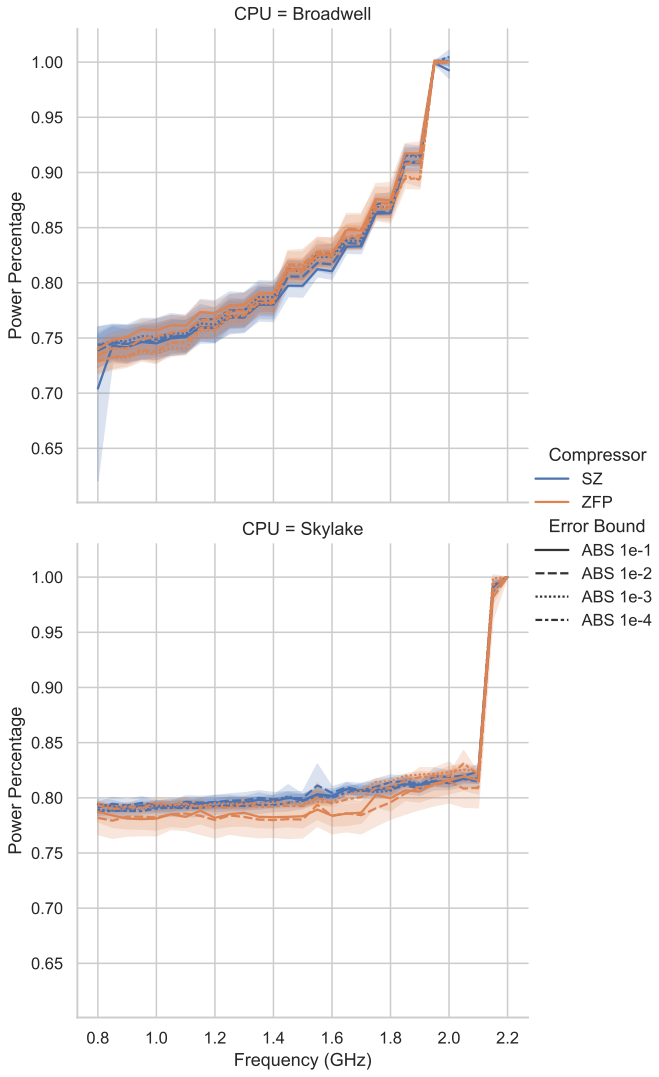

Fig. 1. Compression Scaled Power Characteristics.

Figures 1–4, summarize our scaled power and runtime result by plotting the scaled power consumption and runtime over frequency for compression and data writing. The two graphs also show different lines for the CPU and compressor. Around each trend we shade a 95% confidence interval, to account for possible noise or error in results.

1) Power Dissipation: The characteristic in Figure 1 represent the scaled power consumption of SZ or ZFP on the Broadwell and Skylake architectures. Since we seek power savings, any percentage below 1 is considered power savings. Therefore, when optimizing for only power consumption we look for where the plot reaches its minimum. This occurs at the

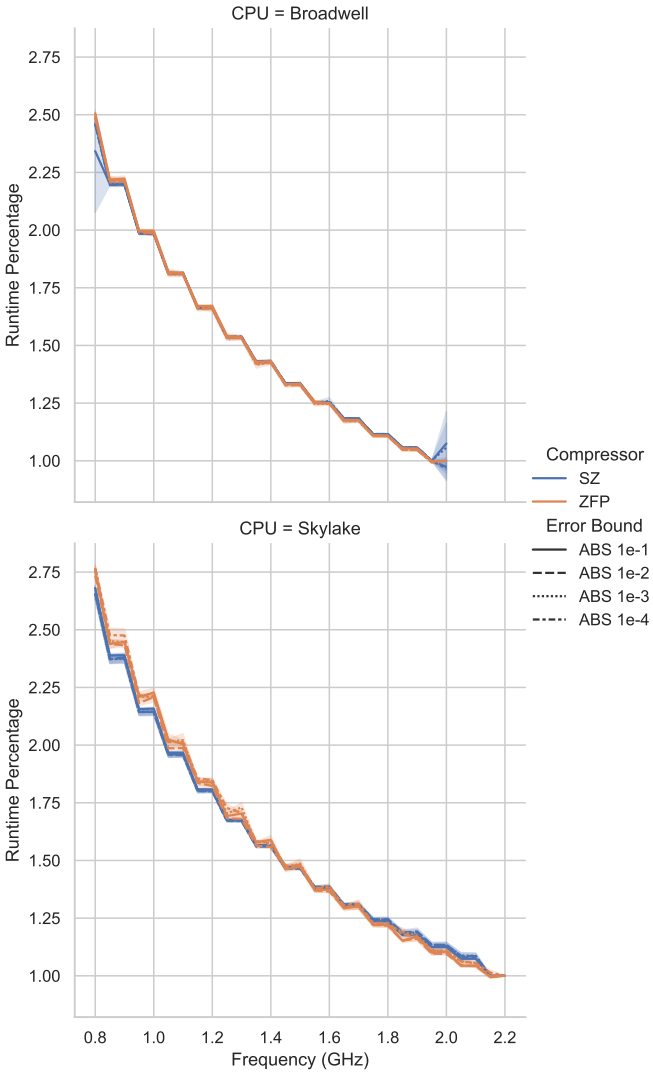

Fig. 2. Compression Scaled Runtime Characteristics.

lowest frequency for both chips and all compressors. However, after analyzing runtime, we find that this is not the best choice for CPU frequency, as this is when the runtime is the greatest.

The error bounds that are included for the compression results do not greatly influence the scaled power consumption of either compressor or CPU. However, in Figure 6, one can see the effect of error bound on the magnitude of the energy without scaling. Based on the method of scaling, the calculation of power from Eqn. 1 factors out the magnitude of difference in runtime and energy that occurs between error bounds. One should note that with a more fine error bound, compression will typically take longer and have a lower compression ratio [5]. To represent different error bounds in the data transit case, we vary the size of the data being sent.

Similarly, in Figure 3 we present scaled power consumption of writing different sizes of data on Broadwell and Skylake architectures of a series of frequencies. We find that the lowest power consumption comes at the lowest frequency value. The

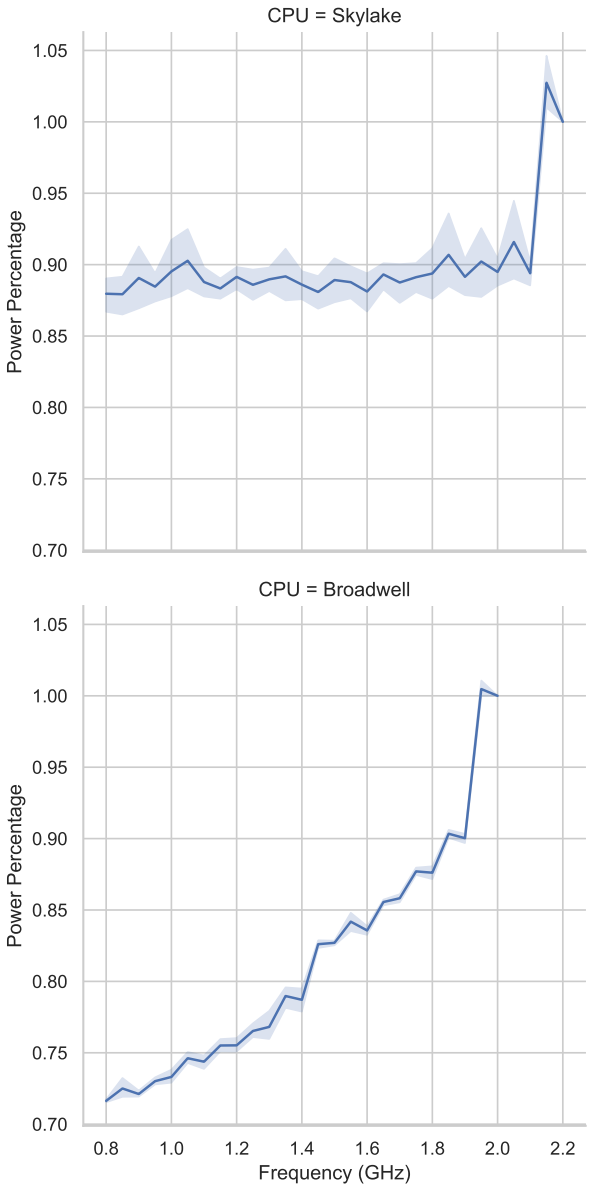

Fig. 3. Data Transit Scaled Power Characteristics.

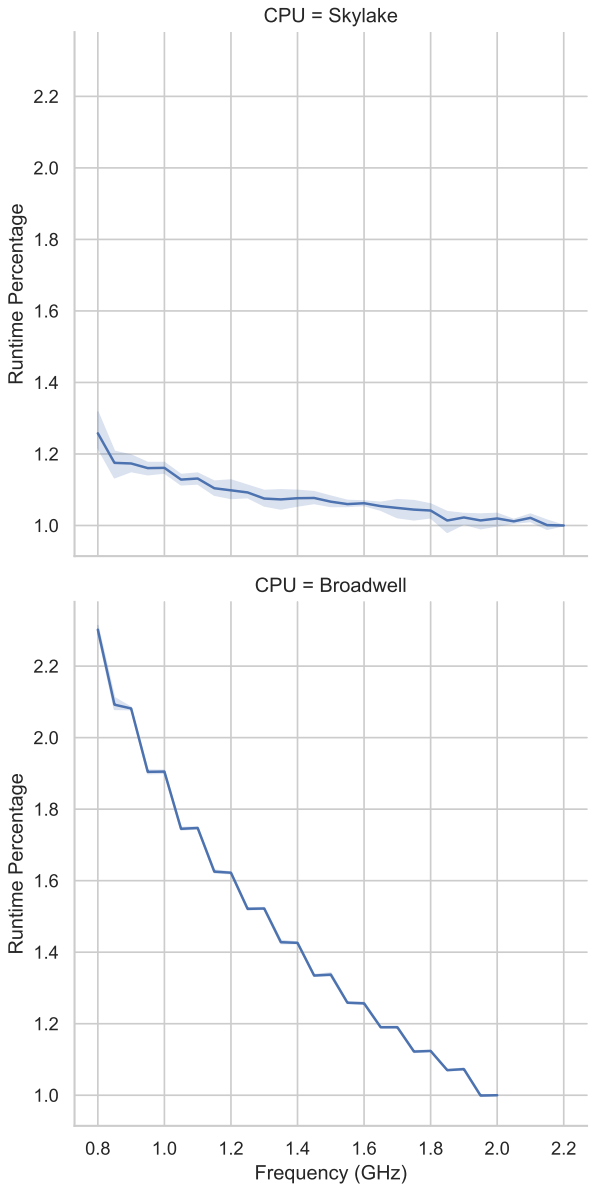

Fig. 4. Data Transit Scaled Runtime Characteristics.

primary difference between the Skylake and Broadwell is the Skylake doesn't have as large of a range as the Broadwell. This is consistent with the compression results. The reason for a power percentage of 0.9 in data writing as compared to 0.8 during compression is that data writing can be more intensive than the steps to compress. Therefore, we do not see as significant of power savings during data writing.

Figures 1 and 3 demonstrate the critical power slope which is discussed in [13]. The critical power slope is a sharp, increasing curve of CPU power draw over a set of frequencies. In this model, the values in the curve are nearly constant for a large range of frequencies. This is the same behavior seen in Figures 1 and 3.

By utilizing the models found in Table IV and the data in Figure 1, we compute that for compression on both CPUs and both compressors we achieve approximately 19.4% power savings during compression, by lowering frequency by 12.5%. Similarly, using the data in Figure 3 and models from Table V, we achieve approximately 11.2% power savings on average during data writing, by lowering the CPU frequency of both chips by 15%.

2) Runtime: The values in Figure 2 represent the scaled runtime of SZ or ZFP compressing data over a range of clock frequencies. In this graph, we interpret a percent runtime below 1 as runtime savings. Notice that the best compressor runtime in Figure 2 comes at the highest clock frequency, meaning that if a user desires very fast compression, they should utilize the max clock on a CPU. However, this is not the

minimal power consumption, as this occurs at the lowest clock frequency. Therefore, we find where power is minimized and runtime is minimized. This is at the point of 12.5% frequency reduction for compression.

Yet again, the included error bounds for the scaled runtime in compression do not influence the trend. This is again due to the technique of scaling, as the magnitude of runtime is higher for finer error bounds.

Similarly, in Figure 4 we show the scaled runtime of writing different sizes of data over a range of frequencies. The lowest runtime still occurs at the max clock frequency. Yet, as mentioned above this is not where power is minimized as well. This point then occurs for data transit at 15% frequency reduction during data writing.

We note also that in Figure 2, the trends overlap showing consistent runtimes between SZ and ZFP. We find that the runtime is stagnant in data writing for the Skylake processor, as this is similar to the power in Figure 3 not scaling during this operation. This is likely indicative of the load data writing puts on a single-core for the Skylake generation. Also we note that energy and power analysis of the Skylake chips have not shown major improvement in energy efficiency over older generations, despite being newer and having better performance [22]. Therefore, the stagnant scaling is indicative of this lack of energy efficient scaling.

An important note about reducing clock speeds is with a lower clock speed, there is an increase in the runtime of an application, as seen in Figure 2. The equation for average power, Eqn. 1, factors out time, meaning those results are invariant of runtime. Therefore, one must notice the affect that the intertwined metrics have on the energy of the system.

3) Energy-Aware Trade-off: As mentioned previously, the goal of this study is to optimize the energy consumption of lossy compression and writing data with CPU frequency for HPC I/O. In Figures 1–4 we present an approach for monitoring the system power and time usage during compression. In our analysis of the runtime and power dissipation separately, we find that the best power and time savings are at opposite ends of the frequency spectrum. The most power is saved at the lowest clock frequency, whereas runtime is optimal at the highest clock frequency.

Decreasing energy through lowering power with clock frequency is a trade-off. Would a user benefit from faster compression? or less energy-consumed? This trade-off is a user and system consideration, yet the decreased runtime nets significant energy-savings during data I/O. We achieve a maximal 19.4% power savings with a net 7.5% increase in runtime, by decreasing the CPU clock speed by 12.5% in both models of compression. For data writing we achieve 11.2% average power savings with a net 9.3% increase in runtime, by decreasing the CPU frequency by 15%. Symbolically, we represent our recommendations for tuning the CPU frequency f in Eqn. 3, with respect to the maximum clock frequency f_{max}.

$$f_{I/O} = \begin{cases} 0.875 f_{max} & \text{lossy compression} \\ 0.85 f_{max} & \text{data writing} \end{cases} \quad (3)$$

Averaging these two savings, we find that on average these savings are equivalent to 14.3% energy savings with a net increase of 8.4% in runtime during lossy compression and data writing.

VI. Frequency Tuning for Energy Savings

In this Section, we utilize the findings of Sections IV and V to show how our power model and recommendations for tuning in Eqn. 3 are used to predict and lower energy usage in HPC I/O. We present two examples that demonstrate the versatility of our model and solution against non-optimized clock frequency compression and data dumping. In the first example, we show our model for the Broadwell chip against different datasets not included in model regression, to demonstrate how well our work estimates new results. In the second, we measure energy consumption for data dumping on a file system (NFS) through Cloudlab, utilizing our optimized CPU frequency recommendations.

A. Estimating Power Consumption Model

To test our model for new results, we perform the same experiment with the Hurricane-ISABEL dataset from SDR-Bench [16]. Hurricane ISABEL represents a weather simulation for different weather metrics by the National Center for Atmospheric Research. The uncompressed floating-point data snapshots have dimension $100 \times 500 \times 500$; we compress six 95MB fields (PRECIP, P, TC, U, V, W) using both SZ and ZFP with a $1e{-}4$ error bound.

Measuring the average power across all frequencies, we produce similar results to Figures 1 and 2. In Figure 5 we demonstrate how well our power-model fits against the power dissipation in the new results. We calculate SSE $= 0.1463, \text{RMSE} = 0.0256$ for these curves, demonstrating that the model estimates the data well with little error . From this, we determine that our model estimates power behavior well, even with data not factored into our model.

B. Data Dumping with Lossy Compression

Data dumping is a popular method for transferring data on an HPC system. Lossy compression is often used to reduce size and expedite data movement. However, compression can take significant energy before data movement. As the impetus for this study, we use our tuning framework in Eqn. 3 as a test against non-optimized data movement.

As an experiment we simulate compressing and transmitting 512 GBs of NYX data, attained by concatenation, over an NFS with and without tuning the clock frequency. In particular, we lower the clock frequency by 12.5% for SZ compression and then by 15% to tranport the data, as per Eqn. 3. In both our frequency scaling and the base clock we record the total energy dissipated in the experiment. We compress using SZ over several error bounds $(1e{-}1, 1e{-}2, 1e{-}3, 1e{-}4)$ to demonstrate how the difference in compressed data size affects

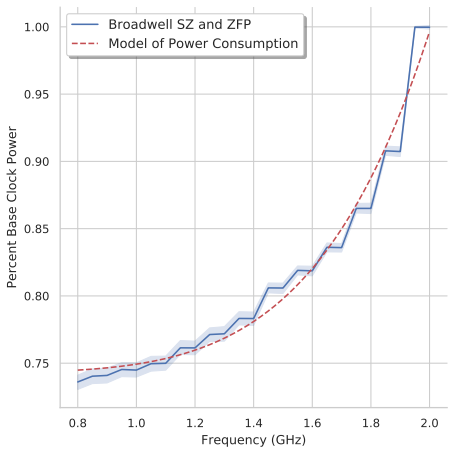

Fig. 5. Broadwell Chip Model for Power Consumption

data transit and the magnitude of runtime results for large datasets. In this example, we compress and transmit a 512GB velocity-x field of the NYX dataset on a 10Gbps ethernet connection.

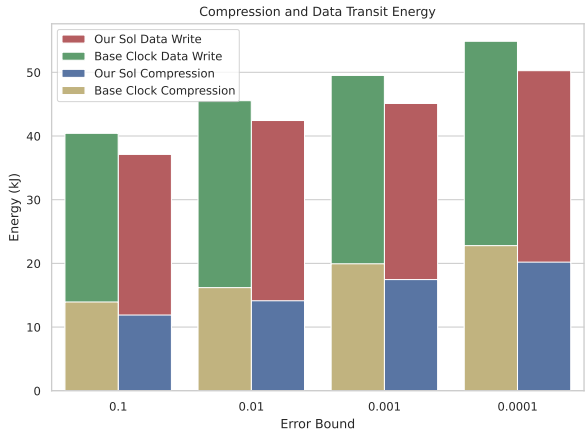

Fig. 6. Energy Dissipation for Data Dumping

As shown in Figure 6, our solution always reduces the amount of energy consumed in data compression and movement. The non-scaling compression and data transit is represented by the tag Base Clock. We show that compression energy usage and data movement is reduced by lowering the frequency. On average this is 6.5kJ, or 13%, of energy saved over all of the error bounds using our solution. While future studies will strive to address whether these trends hold on different CPUs, with exascale data, these initial results show promise in reducing the energy consumption of data transit with a compression stage.

VII. Conclusion

In this study, we present a framework for modeling the power consumption of lossy compressors and I/O on HPC systems. We recommend lowering the CPU frequency by 12.5% during compression and by 15% during data writing to minimize energy usage. Through our analysis of CPU power-results, we optimize energy-usage of I/O bound jobs by 14.3% on average, with only a net 8.4% increase in runtime. We find this to be consistent across different CPU architectures, datasets, error bounds, and compressors. We also demonstrate that our model estimates power consumption accurately, and saves 6.5kJs on average for compressing and writing 512GB of floating point data. Applications of these findings in HPC computing centers will help meet green-computing initiatives and assist sustainable computing goals.

Acknowledgements

Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR, and the U.S. National Science Foundation (NSF) This material is based upon work supported by the National Science Foundation under Grant No. SHF-1910197.

References

[1] D. Ellsworth, T. Patki, M. Schulz, B. Rountree, and A. Malony, "A unified platform for exploring power management strategies," in *Proceedings of the 4th International Workshop on Energy Efficient Supercomputing*, ser. E2SC '16. IEEE Press, 2016, p. 24–30.

[2] S. Habib and et al., "HACC: extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

[3] X. Liang, S. Di, D. Tao, S. Li, B. Nicolae, Z. Chen, and F. Cappello, "Improving performance of data dumping with lossy compression for scientific simulation," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, 2019, pp. 1–11.

[4] S. Di, D. Tao, X. Liang, and F. Cappello, "Efficient lossy compression for scientific data based on pointwise relative error bound," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, pp. 331–345, 2019.

[5] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 438–447.

[6] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017, pp. 1129–1139.

[7] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, and F. Cappello, "Significantly improving lossy compression for hpc datasets with second-order prediction and parameter optimization," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 89–100. [Online]. Available: https://doi.org/10.1145/3369583.3392688

[8] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[9] A. Jaiantilal, Y. Jiang, and S. Mishra, "Modeling cpu energy consumption for energy efficient scheduling," in *Proceedings of the 1st Workshop on Green Computing*, ser. GCM '10, 2010, p. 10–15.

[10] Y. Liu, Z. Liu, R. Kettimuthu, N. Rao, Z. Chen, and I. Foster, "Data transfer between scientific facilities – bottleneck analysis, insights and optimizations," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2019, pp. 122–131.

[11] J. Corbalan, O. Vidal, L. Alonso, and J. Aneas, "Explicit uncore frequency scaling for energy optimisation policies with ear in intel architectures," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, 2021, pp. 572–581.

[12] M. Morán, J. Balladini, D. Rexachs, and E. Luque, "Prediction of energy consumption by checkpoint/restart in hpc," *IEEE Access*, vol. 7, pp. 71 791–71 803, 2019.

[13] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: Understanding the runtime effects of frequency scaling," in *Proceedings of the 16th International Conference on Supercomputing*, ser. ICS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 35–44. [Online]. Available: https://doi.org/10.1145/514191.514200

[14] J. M. Montañana Aliaga, A. Cheptsov, and A. Hervás, "Towards energy efficient computing based on the estimation of energy consumption," in *Sustained Simulation Performance 2019 and 2020*, M. M. Resch, M. Wossough, W. Bez, E. Focht, and H. Kobayashi, Eds. Cham: Springer International Publishing, 2021, pp. 21–33.

[15] H. Luo, D. Huang, Q. Liu, Z. Qiao, H. Jiang, J. Bi, H. Yuan, M. Zhou, J. Wang, and Z. Qin, "Identifying latent reduced models to precondition lossy compression," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 293–302.

[16] Scientific Data Reduction Benchmark, https://sdrbench.github.io/, online.

[17] J. Kay and et al., "The community earth system model (CESM), large ensemble project," *Bulletin of the American Meteorological Society*, vol. 96, no. 8, pp. 1333–1349, 2015.

[18] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukic, and E. V. Andel, "Nyx: A massively parallel amr code for computational cosmology," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, feb 2013.

[19] D. D. et al., "The design and operation of CloudLab," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, Jul. 2019, pp. 1–14. [Online]. Available: https://www.flux.utah.edu/paper/duplyakin-atc19

[20] A. Colin Cameron and F. A. Windmeijer, "An r-squared measure of goodness of fit for some common nonlinear regression models," *Journal of Econometrics*, vol. 77, no. 2, pp. 329–342, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304407696018180

[21] MATLAB, *version 9.11.0 (R2021b)*. Natick, Massachusetts: The MathWorks Inc., 2021.

[22] R. Schöne, T. Ilsche, M. Bielert, A. Gocht, and D. Hackenberg, "Energy efficiency features of the intel skylake-sp processor and their impact on performance," in *2019 International Conference on High Performance Computing Simulation (HPCS)*, 2019, pp. 399–406.