An Empirical Study of Performance, Power Consumption, and Energy Cost of Erasure Code Computing for HPC Cloud Storage Systems

Hsing-bung Chen, Gary Grider, Jeff Inman, Parks Fields, Jeff Alan Kuehn
Los Alamos National Lab
Los Alamos, New Mexico 87545, USA
hbchen@lanl.gov

Abstract — *Erasure code storage systems are becoming popular choices for cloud storage systems due to cost-effective storage space saving schemes and higher fault-resilience capabilities. Both erasure code encoding and decoding procedures are involving heavy array, matrix, and table-lookup compute intensive operations. Multi-core, many-core, and streaming SIMD extension are implemented in modern CPU designs. In this paper, we study the power consumption and energy efficiency of erasure code computing using traditional Intel x86 platform and Intel Streaming SIMD extension platform. We use a breakdown power consumption analysis approach and conduct power studies of erasure code encoding process on various storage devices. We present the impact of various storage devices on erasure code based storage systems in terms of processing time, power utilization, and energy cost. Finally we conclude our studies and demonstrate the Intel x86's Streaming SIMD extensions computing is a cost-effective and favorable choice for future power efficient HPC cloud storage systems.*

Keywords - *Erasure code, SIMD Vectorization, Power measurement, Energy cost, Power consumption, Cloud storage*

I. Introduction

Erasure code technologies [1][2][3] are primary deployed on high speed telecommunication networks, unreliable wireless, mobile, and deep space communication channels, consumer's multimedia electronics such as Blue-Ray discs, CDs and DVDs, and computer applications such as RAID-6 based storage systems. Many erasure code encoding and decoding implementations are using hardware based solution such as micro-processor, DSP [30], and FPGA [31]. Erasure coding hardware solutions work well on real-time data streaming applications however the majority of solutions can only handle limited problem size due to the limitation of computing power and available memory size needed in erasure coding processes. FPGA solution can provide efficient CPU coding bandwidth but it normally comes with high deployment and development cost. Furthermore it supports finite encoding methods, fixed data chunks and code chunks ratio because of its restricted programmable space.

Besides the application areas of consumer electronics and wireless communications, erasure code based storage systems are becoming popular choices for cloud storage systems due to cost-effective storage space saving schemes and higher fault-resilience capabilities. Customizable redundancy schemes implemented in erasure code are more storage-efficient alternative to substitute the triple or multiple data replications solution implemented in most of today's large-scale data centers and HPC storage systems [2].

Efficient and effective erasure coding needs support from heavy CPU bound computing for finite field's matrix-vector multiplication, and log/anti-log table lookup operations and memory bound referencing and updating for log and anti-log table lookup operations. Research studies and reports from commercial systems [1][2][3][26][29] have showed that software-based erasure coding can be deployed on distributed storage system. Long processing time and slow coding bandwidth is a major road block when we are attempting to employ software based erasure coding technique on large data set [22]. Intel x86 and SIMD based erasure code open-source software are available for research studies and commercial applications [18][19][20]. The x86 based instruction sets cannot efficiently execute data parallelism mathematical computing property in Finite field matrix-vector multiplications and parallel table lookup operations. SIMD's vector computing capability allows parallel data processing and matrix computing by multiple homogeneous cores inside a single CPU chip. Modern general purpose microprocessors feature multimedia extensions that support SIMD parallelism such as Intel SSEx, AVX, AVX-2, and AVX-512 SIMD instruction sets and ARM Neon SIMD instruction sets. Performance benefits from applying SIMD vectorization depend on the problem domain data model and programming structure [21][41]. Motivations of using SIMD platform on erasure code storage systems are exploit data level parallelism and avoid concurrency/combine with concurrency, come at no extra cost, it is available in modern processors design and implementation, it has lower amount of memory load instructions through instruction and data prefetching support, and it is easy to use. There are also some disadvantages to use SIMD such as backward compatible issue between old and new platforms, extra software porting overhead and cost, may have performance penalty without adequate porting and optimization efforts.

Power, scalability, and reliability are three major challenging areas we need in order to construct future Exa-scale computing and storage systems [4][5][6][7][17].

Available energy resources, power consumption, and energy cost are dominating factors to support, operate, and administrate Exascale systems. Understanding properties and characteristics of power consumption in computer systems can assist us to improve system design and increase power utilization. Modeling, software profiling, and measurement are three universal approaches to understand power consumption of systems and applications. Power modeling is limited to inter-node variability in power draw. Software power profiling is used to evaluate power optimization techniques and to make power/performance trade-off. It is also an important approach to supply critical power information for operating systems and power-aware software. Power measurement is a direct method to estimate application power consumption on running systems. Respectively power measurement can be done at system level, processor level, core level, CPU function unit level, process level, and virtual machine levels. Research and studies of power measurement issue are focusing mostly on computing territory [8] [10] [12] [13] [14] [15] [16]. Power consumption study on erasure code based storage domain is not well covered in HPC and cloud research communities [9][11].

Many erasure code research works are only focusing on encoding and decoding algorithm design and performance studies on small size data set [37][38][39][40][41] that can be fit into memory based coding operations. They did not reflect real world usage of erasure code application on storage systems such as shared disk storage systems or SSD/Flash array systems. The impact of real storage I/O operations was not considered in those research studies. In this paper, we mainly concentrate on studying power consumption of erasure code on large size data sets normally generated in HPC cloud storage environment. To do this we design data workload generally seen in HPC computing environment and adopt breakdown measurement and performance analysis approach. We decouple power consumption from various testing cases and provide a reasonable estimating model for energy cost. We compare erasure encoding power consumption on a large data set then measured x86 vs. SIMD extension platforms in terms of processing time, coding bandwidth, and energy cost. We also study the performance impact of using various storage devices during erasure code computing such as SATA hard disk and Flash memory storage device.

The rest of this paper is organized as follows. We present related research works in section II. In Section III, we present the methodology used in power measurement and the test bed setup for power consumption studies. In Section IV, we provide performance results and analysis. We do a survey of related works in Section V. In section VI, we conclude our findings and outline future research and development activities in this area.

II. RELATED WORKS

In this section, we provide background information of power related researches and studies. Modeling, profiling, and Vmeasurement are three common used methodologies in power studies on computer systems. The work in [5] describes a tri-level HPC power measurement methodology for large scale High performance computing systems. Level-1 only measure core phase average power consumption. Level-2 measures 10 average power measurement in the core phase, covers full range of power consumption, and also measures idle power consumption. Level-3 covers level-2 measurement scope with continuously integrated energy. In [6], author present a power measurement framework and obtain a per-node (socket) granularity at frequencies of up to 100 samples per second. This frame work has been tested on SNL's Catamount Light Weight Kernel OS. It also can quantify the amount of energy used by applications and to contrast application energy use between a Light Weight and General Purpose operating system. In [8], authors use a power consumption model to predict the power utilization and usage information on computing server and storage system in data centers. The proposed power consumption model covers CPU, memory, hard disk, mainboard, fan, and power supply unit. In [9], authors describe a power model called "TEMPO" and The proposed "TEMPO" power cost estimating method is used to accurately estimate the disk power consumption using a trace of all requests issued to the disk with a real-time streaming workload. In [7], authors present a study of power consumption in real-world, large-scale, enterprise, disk based backup storage systems. They also provide several key observations on power consumption variations across platforms and approaches for future development of power management technologies. A power consumption breakdown study on a modern laptop is mentioned in [12]. Authors performed multi-phases power measurement on CPU, Hard drive, LCD display. In [14], a simple power consumption model was proposed and used to study HDD and SSD storage devices. Authors discuss how to apply their power consumption model and select a target storage server so that the total power consumption can be reduced.

III. EXPERIMENTAL TEST-BED SETUP

A. *storage node platform and storage devices*

A HP Z620 workstation is configured as the storage node in testing. Table-1 shows the testing hardware information. We use both SSD/flash storage and SATA disk based storage in testing. We study the impact of I/O access latency on erasure coding process. Figure-1 shows the erasure coding processing diagram.

B. *Open soruce eraure code softwate used in testing*

We select three open-source erasure code software systems in our erasure code computing power study. They are zfec [29], Jerasure 1.2 [18], and Jerasure 2.0 [19] [20][21].

"zfec" is based on the forward error correction code (FEC) software designed by Luigi Rizzo in 1998 [26]. Rizzo's FEC software was one of the earlier software based Reed-Solomon erasure code implementations. There are three purposes of Rizzo's FEC software: (1) to put software FEC in the right perspective by presenting the principle of operation of erasure codes and showing that software based FEC is not exceedingly expensive, (2) to provide the research community with a high performance C implementation of erasure code, and (3) to demonstrate that software based erasure code can despite its overhead be used to actually improve performance on slow wireless networks and fast wired networks as well as unicast and multicast communication protocols. "zfec" has made several changes from the original FEC package, including addition of the Python API, refactoring of the C API to support zero-copy operation, a few clean-ups and optimizations of the core code itself, and the addition of a command-line tool named "zfec". "zfec" provides command line interface and support API access interface in C, Python, and Haskell. "zfec" has been used in erasure code storage research projects such as Tahoe-LAFS [33] and NCCloud [32]. Current the zfec software only run on Intel X86 platforms.

Jerasure-1.2 (aka EC-1) has implemented Vandermonde based Reed-Solomon code and Cauchy based Reed-Solomon code erasure encoding and decoding methods. The mathematical properties are based on Galois-field word size support of 8, 16 or 32 bits. It also implements Minimal Density RAID-6 encoding/decoding method. Jerasure-1.2 is implemented as a single thread/process erasure coding library using C/C++ programming interface. The main target application is storage systems. Jerasure-1.2 technology has been deployed in commercial products such as Scality Ring Storage system CEPH distributed file system [34], and OpenStack's SWIFT object store [35]. Jerasure 1.2 is support Intel X86 platform only.

Jerasure-2.0 (aka EC-2) [21] basically is a software enhancement of Jerasure-1.2 with Intel stream SIMD extension instruction sets support and some flexible and optimization improvement. Jerasure-2.0 is implemented in C++ language and has added generalized EVENODD (Double Disk Failure) and RDP (Row-Diagonal Parity) to its library. Jerasure-2.0 supports arbitrary Galois-field word size 8, 16, 32, 64, and 128 bits. Jerasure 2.0 supports both Intel SIMD (MMX, SSEx, and AVX) and ARM SIMD (NEON) platforms.

Hardware-HP Z620	Description
CPU	Intel Xeon E5-2620 v2 Processor, 2.1GHZ. 6 cores
Memory	32GB PC-3 1600 DDR3 memory
OS	Fedora 20, 64-bit OS
SATA HDD	4TB Hitachi 4TB Sata HDD
PCI-E based Flash storage	OCZ Revo-350 960 GB PCI-E Flash Storage

Table-1: storage node configuration and setup

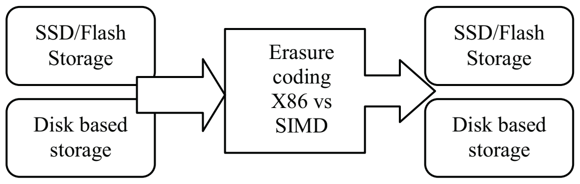

Figure-1: Source and destination devices used in erasure coding process

Coding platform	Methods	Description
1-x86	zfec	zfec erasure code
2-x86	EC1-A	Jerasure-1.2: Reed-Solomon-Vandermonde
3-x86	EC1-B	Jerasure-1.2:Cauchy-Original based
4-x86	EC1-C	Jerasure-1.2:Cauchy-Good based
5-SIMD	EC2-A	Jerasure-2.0: Reed-Solomon-Vandermonde
6-SIMD	EC2-B	Jerasure-2.0:Cauchy-Original based
7-SIMD	EC2-C	Jerasure-2.0:Cauchy-Good based

Table-2: Erasure coding methods

C. *Power Measurement device*

"Watts Up/.net" [23] is the device we used in measuring erasure code computing power consumption. We can simply plug any device into "Watts Up/.net" and the meter instantaneously displays the wattage (power) being used, as well as the cost in dollars and cents. "Watts Up/.net" provides useful power consumption information such as current watts, minimum watts, maximum watts, power factor, cumulative watt hours, average monthly kilowatt hours, tier 2 kilowatt hour threshold (used to calculate secondary kWh rates), elapsed time, cumulative energy cost, average monthly cost, line volts, minimum volts, maximum volts, current amps, minimum amps, and maximum amps. The minimum sampling period is one second.

A Built-in Non-volatile memory is used to record continuous and cumulative power consumption data. Clear "Non-volatile memory" is use to emulate a start monitoring process. "Watts/Up" software runs on a connected PC and is used to read logged power data from "No-Volatile memory". It displays selected timing graph for all recorded data such as Watts, Volts, Amps, WattHour, and Energy Cost. The data and graphs can then be exported to Excel spreadsheet and word processor programs for further analysis.

Table-2 lists the erasure coding methods used in power measurement and energy cost studies.

IV. METHODOLOGICAL APPROACH

A. *Methodoligical Approach*

Our methodological approach is to apply a breakdown based power measurement and analysis method. We use a "stand-alone" inline meter" type power measurement device and take physical power measurement on designed active workloads and system idle state that are applied to large data set erasure code encoding process. We concentrate on a single storage node level power measurement and we monitor runtime power utilization time line graph. We observe the correlated time-line graph of erasure code computing, I/O operations, and power consumption at varying system utilization (or workload) levels. Power measurement and power consumption monitoring provide us insight to select the appropriate computing platform(s) to reach cost-effective and energy-efficiency optimization.

B. Testing workload desings

Erasure coding bandwidth has been studied and compared [1][2][18][[19][20][21][26] but are limited to small scale data sets and are restricted to memory access operations. Hence coding process is only done within memory range and cache buffer is used to speed up the coding bandwidth. Those coding bandwidth tests are focusing on comparing different type of coding methods. Those performance results cannot truthfully reflect practical usage of erasure code process on storage systems. Typically large HPC data sets in the range of gigabytes to terabytes are constantly created dy-to-day. Realistic testing workload should not be limited to memory only access range and they should include read and write operations on storage devices in power measurement studies. To minimize memory caching or buffering impact on selected testing workload and incorporate real operation impact from data access latency, we choose to use a 40GB data set and design three different testing workloads.

The encoding is parameterized by two integers, K and M. K is the total number of data chunks produced, and M is the total number of code chunks produced. Any K chunks of data and code are necessary to reconstruct the original data. The coding ratio is defined as M/K and the storage overhead is defined as (K+M)/K. Three designed workloads are covered 50%, 25% and 20% storage overhead cases in erasure code computing. Table-3 lists is three designed testing workloads.

Data Size : 40GB	K: data chunk	M: code chunk	Overhead	Read chunk	Write chunk
Workload-1	160	80	50%	160	240
Workload-2	160	40	25%	160	200
Workload-3	160	32	20%	160	192

Table-3: testing work loads

C. Breakdown power measurement phases

We adopt a breakdown analysis in power measurement process. We measure power consumption in different system states: system idle, read-then-write data access, and erasure code computing. We then calculate power consumption for Erasure-computing. We collect wall clock processing time, encoding bandwidth, and energy cost data generated from a "Watts/UP" power meter. Each phase's activity is described as follows.

- Phase-1: Baseline power measurement in system idle state

A baseline power consumption data is measured when a system is not running any application type workload. Only system related tasks are active. The system idle state is in a state where a computer system is ready to accept any workload [4] [5]. The system idle state is not a sleep or a hibernation state. This measurement is calculated as constant power consumption per time unit of the system.

  - P_{idle} *- power consumption/hour in Idle state*
- Phase-2: Read/Write data power measurement

Apply read-then-write operations on a 40 Gigabyte data set and calculate the average read-then-write energy cost.

  - $P_{read\text{-}write}$ *- Read-then-write Power consumption/hour*
  - P_{read} *- Read Power consumption/hour*
  - P_{wrtie} *- Write Power consumption/hour*
  - $Cost_{read\text{-}write}$ *- Read-then-write Power consumption/hour*
  - $Cost_{read}$ *- Read Power consumption/hour*
  - $Cost_{wrtie}$ *- Write Power consumption/hour*
- Phase-3: Erasure code computing power measurement – total processing time

Launch an erasure coding process with a data set and selected coding method.

  - T_{total} *- total processing time*
  - T_{ec} *- Total erasure coding time*
  - T_{IO} *- Total IO time*
  - T_{etc} *- misc time, inter viability events system or kernel related*
  - $T_{total} = T_{ec} + T_{IO} + T_{etc}$
  - P_{ec} *- erasure computing power consumption/hour*
  - *ecOverhead - storage overhead ratio = (K+M)/K*
  - P_{total} *- Total power consumption*
  - $P_{total}=P_{idle}*T_{total}+P_{read\_write}*T_{IO}+ P_{ec}*T_{ec}$
- Phase-4: calculate erasure code power energy cost

We then calculate the energy cost for processing erasure encoding.

  - $P_{ec}=P_{total}-P_{idle}*T_{total}+P_{read\text{-}write} * T_{IO}$
- Phase-5: Energy cost estimating calculation

We use energy cost generated from "Watts/UP" meter measurement and convert it to a real estimated energy cost based on local energy price.

  - *TestingObjectSize: In real world applications, storage device I/O access latency can contribute signification overhead on erasure coding process. The benefit of memory caching may offset the influence of I/O latency and can't truthfully represent real world application workloads. To understand the realistic impact of I/O access latency on power consumption, we try to minimize the effect of memory caching and emphasize the impact of I/O access delay from various storage devices. We choose a data object which size is bigger than the system memory size. We reference to large scale Checkpoint & Restart and visualization data sets commonly seen in many large scale HPC computing environments [36][42][43][44] and choose a set of 40/80/120 GB data objects and use them for our erasure code computing power measurement cases.*
  - *FinalObjectSize: TestingObjectSize*ecOverhead*
  - *TripleCopyObjectSize: TestingObjectSize * 3*
  - $ENG_{ReadthenWrite}$*: AvergaeCost per GB Read- then-Write*
  - *TargetObject Size: TOS, the total amount of erasure coded data*
  - *FinaltargrObjectSize: FTOS, FTOS = TOS * ecOverhead*
  - $Unit_{eng}$ *- Energy cost per* kilowatt-hour
  - ENG_{cost}*- Power Consumption/kW ** $Unit_{eng}$ **Px*
    - $ErasureComputing(ENG_{cost})=(Pec*Tec/kW\}*ENG_{Cost}$
    - $IO(ENG_{cost})=(P_{IO}*T_{IO}/kW\}*ENG_{Cost}$
    - $SystemBasic(ENG_{cost})=(P_{idle}*T_{idle}/kW\}*ENG_{Cost}$
  - ENG_{test}*: $Unit_{eng}$ cost is set to \$10/KWh in the "Watts/UP" power meter. The actual Energy cost will be prorated to an actual energy cost in* ENG_{real}*.*

- ENG_{test} *is generated from Watts/UP spread sheet.*
- ENG_{local}*: Local energy cost is about 10.7cent/KWh in New Mexico State. we use* ENG_{local} *to calculate real estimation in production systems.*
- ENG_{real} *: Actual energy cost spending is calculated as* $(((FTOS)/TestingObjectSize)* ENG_{test})/(10/ ENG_{local})$
- $ENG_{TripleCopy}$*: TripleCopyObjectSize ** $ENG_{ReadthenWrite}$

V. Testing results and Performance Analysis

Analyzing power consumption data from selected seven erasure coding methods, we conduct a sequence of lengthy testing cases. Each designed testing cases are repeated three times and we take average of results. We focus on three performance metrics: processing time, power consumption, and energy cost of erasure coding process. Processing time is measured using "time" command. Power consumption is measured using "WattsUp" power meter and energy cost is calculated based on the consumed power and unit energy cost defined in power meter. Power consumption is associated with prorated energy cost. In our performance analysis, we exclusively present processing time and energy cost in performance studies.

A. Baseline system Idle state power consumption and accociated energy cost

For Idle state power consumption, we collect one four-hour period of a system in the Idle state circumstance. From "Watts UP" logging data, we check hourly data. We compare consistency of data value changed over contiguous hours and we then calculate P_{idle} value.

B. Storage Device Read-then-Write result

For $P_{read\text{-}wrtie}$ value, we use Linux "dd" command (a data copy and data convert command) and launch a read from "device A" and write to "device B" testing case on various size data object. We obtain the read-then-write processing time and energy cost record from the spread-sheet logged in "Watts UP" power meter. We calculate the $P_{read\text{-}write}$ value. To simplify the read-then-write power cost estimation, we assume that read and write operations equally share processing time and energy cost.

Single DD testing is used on 40GB, 80GB, and 120GB, data sets to collect processing time and energy cost information on storage devices. SATA HDD read-then-write bandwidth is in the range within 48.49MB/sec and 72.39MB/sec. The range of PCI-E SSD read-then-write bandwidth is from 392.93 MB/sec to 504.12MB/sec. Figure-2A and Figure-2B expose the advantage using PCI-E SSD in terms of processing time and energy cost reduction. Energy cost data is reflecting the length of processing time on read-then-write data access operation. The I/O access patterns and variability of blended I/O and computing operations constitute final processing time. The influence of I/O read-then-write processing time plays an important factor when applications incorporated heavy I/O operations during processing. Results from Figure-2A also point out that decomposing a larger object into several smaller objects can reduce overall I/O access time even they are handled sequentially. For example, we barely spend 485 seconds on six individual 40GB objects (total 6x40GB=240 GB size) Read-then–Write but we have to use 625.46 second to handle a single 240 GB object.

C. Processing time, encoding bandwidth, and energy cost

We show processing time comparison using zFec, EC1{A,B,C}, and EC2{A,B,C} on HDD and SSD storage devices in Figure-3A and Figure-4A. We show that employing SIMD support can reduce significant portion of processing time by 85.33% and 90.25% respectively.

We also illustrate that the choice of storage devices has impacted on EC2-B and EC2-C coding methods. In Figure-3A the SIMD advantage is counteracted by the long media access time. Slower data access latency on HDD device has caused the procrastination of overall processing time, slow encoding bandwidth (Figure-3A) and increased energy cost increase. Some x86 based EC1-B/EC1-C methods can out-perform SIMD based EC2-B/EC2-C methods when HDD device is used as testing storage device on all three testing workloads.

The result is turned toward other direction when PCI-E SSD device is deployed as the storage media (Figure-4A). EC2-B/EC2-C can reduce processing time (28.01%/28.15%, 27.01%/25.64%, 27.17%/25.85%) on three different testing workloads. They also can reduce power consumption (28.29% / 26.69%, 28.19% / 24.54%, 27.54% / 25.29%) when experimented on three different workloads. Figure-3B and Figure-4B show the encoding bandwidth when applying erasure computing on SATA/HDD and SSD/Flash storage devices. SIMD's erasure coding approach can outperform the other x86 based methods when using SSD as the storage devices. Using SSD/FLASH device can significantly reduce data access time due to its lower I/O latency and high IOPs features. It also contributes to minimize total processing time and reduce energy spending cost (Figure-5A and Figure-5B) .

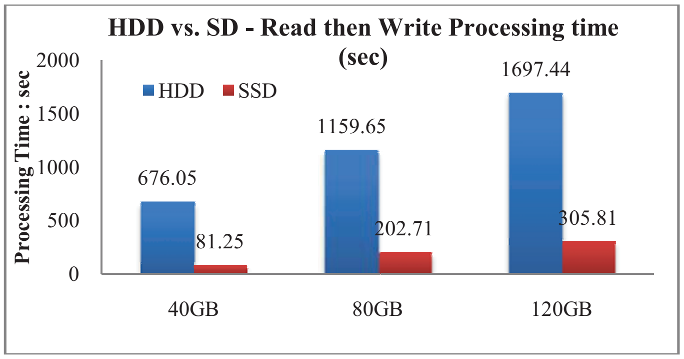

Figure-2A: Read-then-write - HDD vs PCI-E Flash SSD – processing time

"zfec" software only runs on x86 platform. We use "zfec" testing results as reference performance metrics. We notice that $EC2_{SIMD}$ coding methods have outperform "zfec" coding method on every testing workload. However "zfec" can achieve shorter processing time and consume less power compared to the EC1-A method on HDD and SSD storage devices.

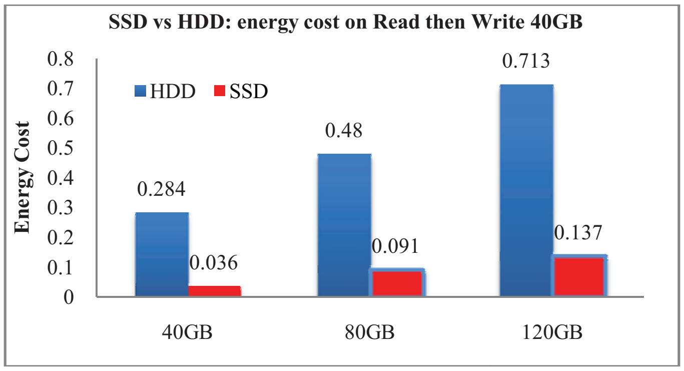

Figure-2B: Read-then-write on HDD vs PCI-E Flash SSD – energy cost

D. Impact of Storage device on Erasure coding process

Our testing results illustrate that EC1-A and EC2-A coding methods can demonstrate the significant performance impact of adoption (1) x86 platform vs. SIMD platform and (2) various storage devices. Figure-6A shows that energy cost reduction of x86 vs. SIMD using SATA HDD is 90.01%, 87.85%, and 87.04%, Figure-6B shows energy cost reduction x86 vs SIMD using PCI-E SSD is 56.02%, 42.96%, and 52.42%. SIMD based erasure coding method can drastically ameliorate coding throughout and cut back power consumption. The SIMD's data parallelism capability proves to be a superior enhancement on erasure code computing.

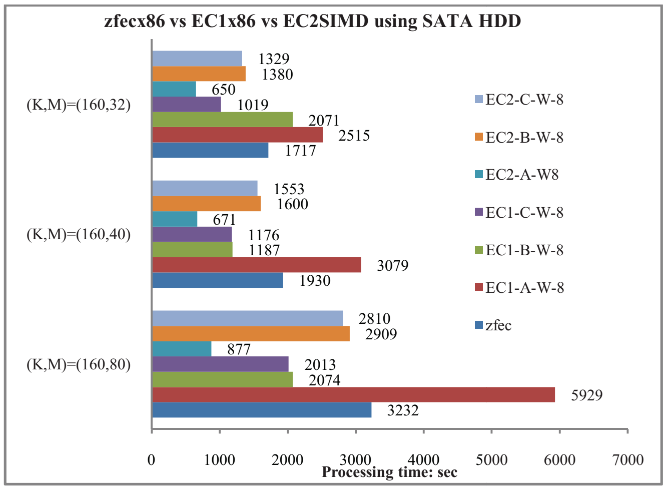

Figure-3A: zfec$_{x86}$ vs. EC1$_{x86}$ vs EC2$_{SIMD}$ – Processing time using SATA HDD

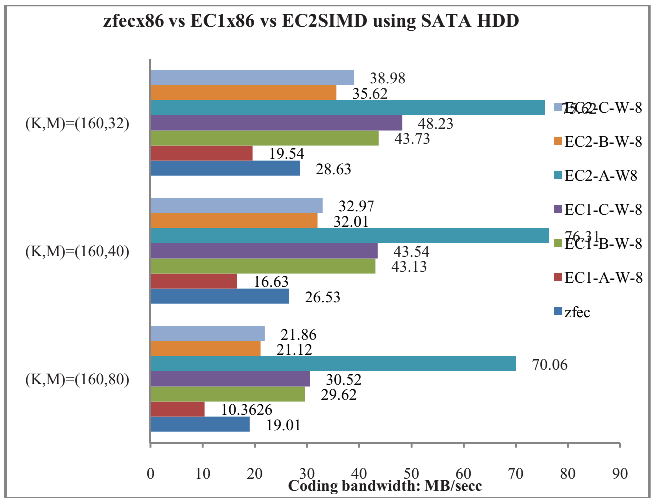

Figure-3B: zfec$_{x86}$ vs. EC1$_{x86}$ vs EC2$_{SIMD}$ – Encoding bandwidth using SATA HDD

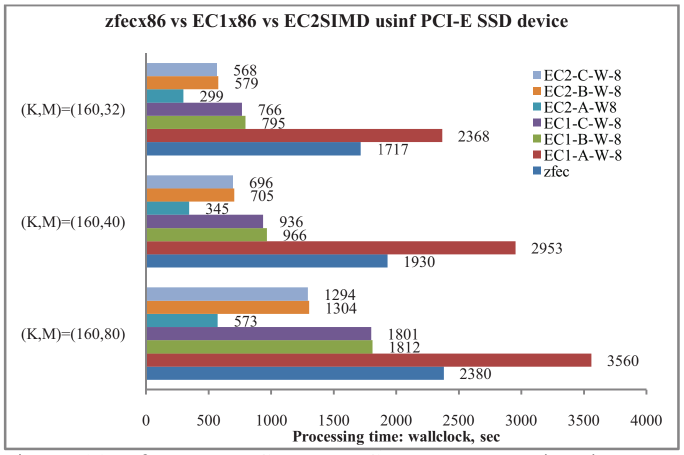

Figure-4A: zfec$_{x86}$ vs. EC1$_{x86}$ vs EC2$_{SIMD}$ – processing time using PCI-E SSD

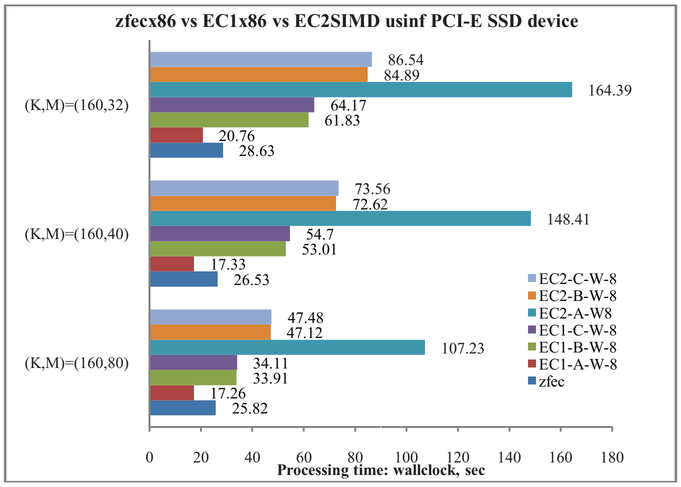

Figure-4B: zfec$_{x86}$ vs. EC1$_{x86}$ vs EC2$_{SIMD}$ – Encoding bandwidth using PCI-E SSD

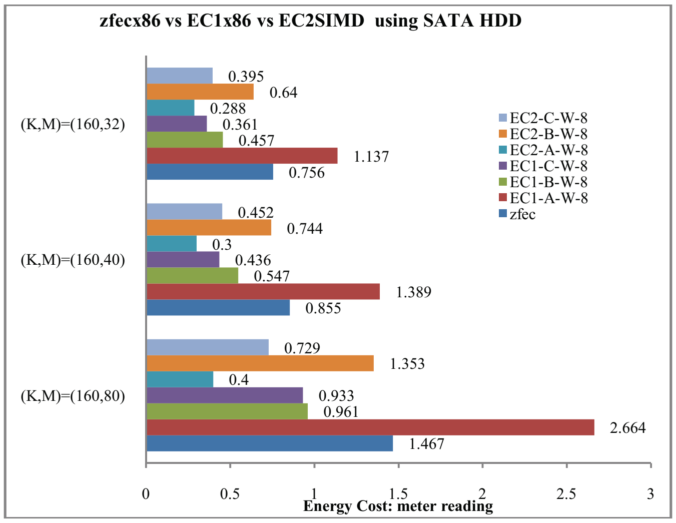

Figure-5A: zfec$_{x86}$ vs. EC1$_{x86}$ vs EC2$_{SIMD}$ – energy cost using SATA HDD

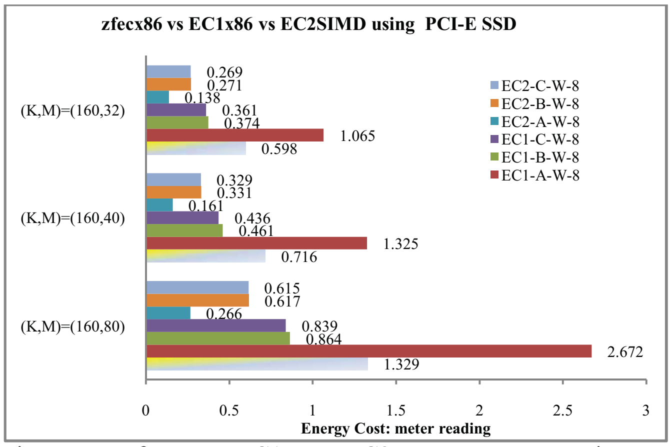

Figure-5B: zfec$_{x86}$ vs. EC1$_{x86}$ vs EC2$_{x86}$ – energy cost using PCI-E SSD

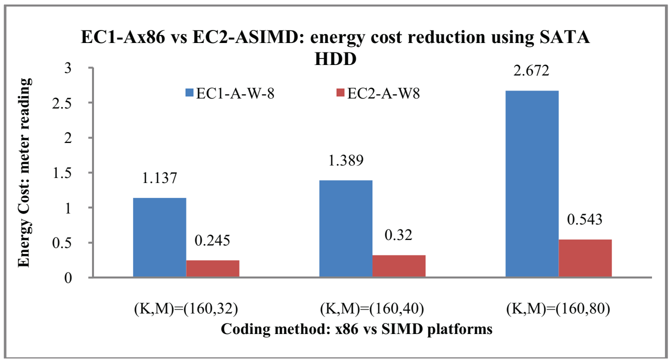

Figure-6A: Energy cost reduction of x86 vs SIMD using SATA HDD

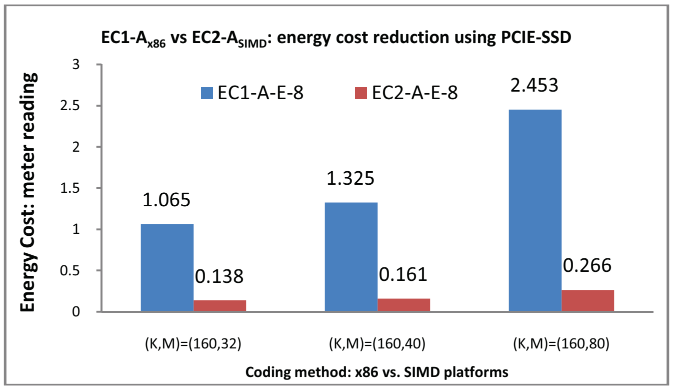

Figure-6B: Energy cost reduction v86 vs SIMD using PCI-E SSD

E. Energy cost analysis from Mixed storage media testing

Figure-7 presents the mixed storage medias EC2-A erasure coding test results. All SSD based system can reduce 50.75% energy cost compared to all HDD based system. Mixed medias or hyper-converged storage system are used in modern data center and HPC storage system. SSD/HDD storage system can save 33.33% energy cost and HDD/SSD storage system can save 24.15% energy cost compared to HDD/HDD system.

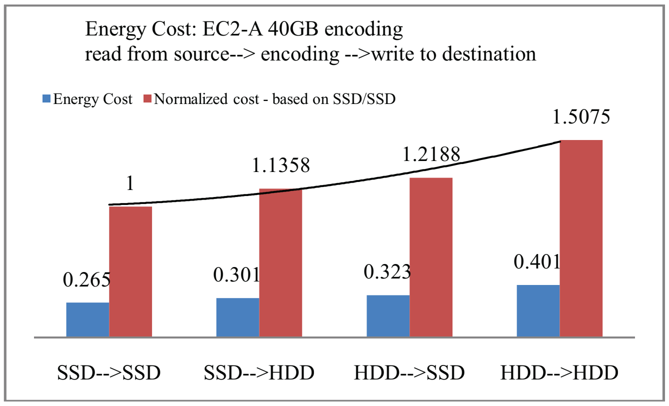

Figure-7: Mixed Media Erasure encoding energy cost: from source device -to destination device- EC2-A coding method

Results have demonstrated that storage device's access latency and IOPS play an important factor of reducing processing time and saving energy cost on erasure code computing.

F. Energy cost reduction from using SIMD extension on EC1-A$_{x86}$ vs. EC2-A$_{SIMD}$ method

LANL's 2015/2016 Trinity machine is expecting to create around three peta bytes active archival data every month. Applying Phase-E formula and using measured data from Figute-6A and Figure-6B, we calculate energy cost using EC1-A and EC2-A methods for three continuous production years period. We expect to process 120 PB data. To simplify the energy cost estimation, we ignore energy cost of data transmission over network link in our calculation.

We list the energy cost comparison and energy cost reduction ratio between EC1-A_{x86} and EC2-A_{SIMD} coding methods in Figure-8 and Figure-9. We compare three coding overhead ratios: 50%, 25%, and 20%. In Figure-8 we obtain 67.25% to 84.95% energy cost reduction using SATA HDD storage device. In Figure-9, we see energy cost reduction from 87.04% to 90.04% when utilizing the build-in SIMD extension in x86 CPU core and SSD/Flash storage device .

We use $ENG_{TripleCopy}$ formula defined in the section IV and calculate energy cost for Triple-Replication approach. The data transmission cost over network is ignored here.

In Figure-11, we compare cost saving between the x86 based erasure coding approach and the triple-Replication approach. x86 based erasure code solution does not show energy cost saving advantage compared to the triple replication approach both on using HDD and SDD storage devices. The SATA HDD I/O access time and x86 computing time is the main root-cause of prolonging erasure coding processing time. The consequence is the high energy cost.

In Table-4, we compare cost saving between SIMD erasure coding approach and Triple-Replication approach. The SIMD based erasure coding methods show significant energy cost reduction compared to the triple replication methods. In addition the deployment of erasure code method on cloud storage systems not only reduce energy cost but also cut the spending on infrastructure required for setting up cloud storage systems such as storage devices, data network facility, floor space, etc.

G. Energy cost breakdown analysis

Table-4 presents the energy cost distribution among baseline IDLE state, I/O operation, and erasure code computing. The baseline/IDLE state energy cost has contributed 77% to 92% of overall power consumption from all coding methods. The large portion of energy cost goes to IDLE/baseline state is due to longer processing time from using x86 platform and SATA HDD storage device. It indicates that we need a better approach to boost system utilization, increase coding throughput, and reduce energy cost on a modern multi-core compute system. To handle same amount of workload, shorter total processing time means better throughput and better energy efficiency because the IDLE/baseline energy cost was amortized from high data access bandwidth and faster erasure coding process.

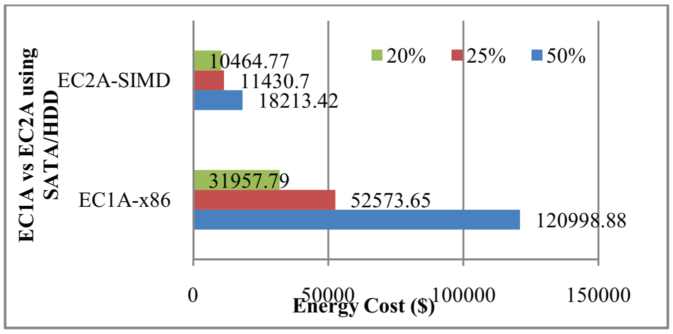

Figure-8: Energy cost comparison - EC1-A_{x86} vs EC2-A_{SIMD} using HDD

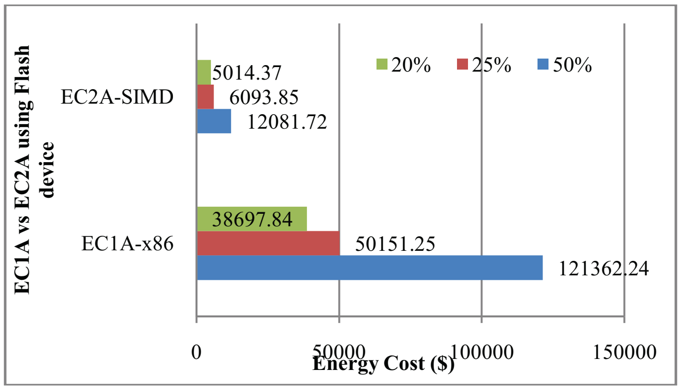

Figure-9: Energy cost comparison - EC1-A_{x86} vs EC2-A_{SIMD} using SSD

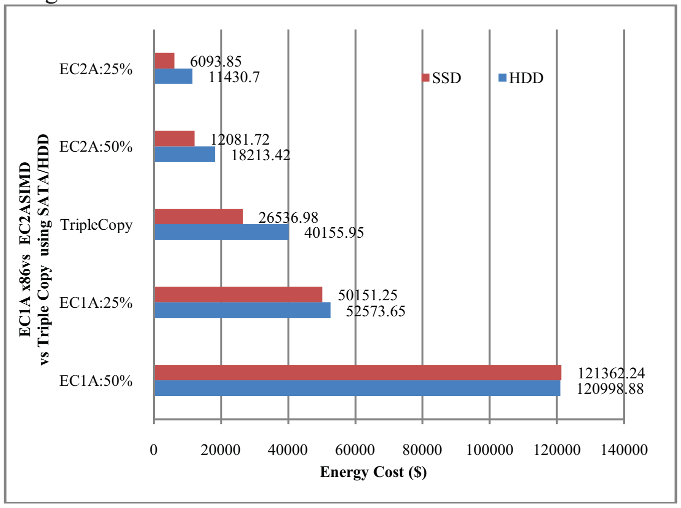

Figure 10: Energy cost comparison – EC1-A_{x86} vs EC2A_{SIMD}vs Triple copies using HDD and SSD

Coding method	(K.M) ratio	I/O	Erasure coding	IDLE baseline
EC1-A	(160,80)	1%	13%	86%
EC1-A	(160,40)	2%	8%	90%
EC1-A	(160,32)	2%	6%	92%
EC2-A/SIMD	(160,80)	10%	11%	79%
EC2-A/SIMD	(160,40)	13%	9%	78%
EC2-A/SIMD	(160,32)	16%	7%	77%

Table 4: Energy cost distribution: EC1-A vs. EC2-A using HDD

VI. Conclusion and Future Works

In this paper, we investigated the advantage of deploying SIMD platform on erasure code based storage system in term of processing time and energy cost. We applied a breakdown power measurement approach on three open source erasure code software. We applied multi-phase measurement on three designed workloads and collected power measurement information from a stand-alone power meter and storage sever. We observed the mixing I/O access behavior with compute bound erasure computing process on SATA HDD and PCI-E based SSD devices. Our testing results have demonstrated that using SIMD support on erasure code computing can help to reduce erasure coding processing time and significantly reduce energy cost.

We also noticed that System IDLE state has dominated the overall processing time and energy cost. When we consider applying erasure code on a very big data set, a single encoding process is not capable of providing enough coding bandwidth. We need to utilize complete SIMD computing power from each CPU core in a multi-core system and explore its potential parallel I/O capability.

The initial power measurement work on erasure code computing has inspired additional efforts in many related areas. Large objects in gigabyte to terabyte range are commonly seen in today's cloud big data computing environment. From read-the-write testing cases, we presented the advantage of partitioning a big object into several small objects and handle them sequentially. Current erasure coding software solution used a single process approach. To effectively apply erasure code on very large data set and sustain a reasonable coding bandwidth, we cannot merely reply on single process approach. In addition to employ SIMD data parallelism, applying function parallelism on a multi-core computer system is a promising feature to boost system utilization and increase productivity in terms of reducing energy cost and coding throughput. To relief the bottleneck of using single thread erasure encoding approach, our future plan for this work will leverage open source erasure code software and implement a parallelizing erasure coding system software system. We intend to utilize all N-Way SIMD computing resource from an N-core/CPU based system and extend it with cluster capability. Furthermore, we are planning to apply this parallelizing erasure coding software on very large scale cluster and cloud storage systems.

ACKNOWLEDGMENT

This work was supported in part by the US Department of Energy and LANL ASCI funding. We thank the technical supports from LANL's HPC-5 group and HPC-3 group.

REFERENCES

[1] John D. Cook, Robert Primmer, AD de Kwant. Compare Cost and Performance of Replication and Erasure Coding, In Hitachi Review Vol. 63, July 2014

[2] Hakim Weatherspoon and John D. Kubiatowicz. Erasure Coding vs. Replication: A Quantitive Comparison, In IPTPS 2006 - The 5th International Workshop on Peer-to-Peer Systems

[3] Osama Khan, Randal Burns , James Plank, William Pierce, Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads, In Usenix 2012 FAST conference

[4] Chung-Hsing Hsu and Stephen W. Poole, Power Measurement for High Performance Computing: State of the Art, In 2011 International Conference Green Computing Conference and Workshops (IGCC)

[5] Thomas R. W. Scogland , and etc., A Power-Measurement Methodology for Large-Scale, High-Performance Computing, In Proceedings of the 5th ACM/SPEC international conference on Performance engineering (2014)

[6] James H. Laros III and etc., Topics on Measuring Real Power Usage on High Performance Computing Platforms, In Cluster Computing and Workshops, 2009, CLUSTER '09. IEEE International Conference

[7] Zhichao Li, Kevin M. Greenan, Andrew W. Leung, Erez Zadok, Power Consumption in Enterprise-Scale Backup Storage Systems, In Usenix 2012 FAST Conference

[8] Robert Basmadjian and etc, A Methodology to Predict the Power Consumption of Servers in Data Centres, In Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking (2011)

[9] Donald Molaro, Hannes Payer, Damien Le Moal , Tempo: Disk Drive Power Consumption Characterization and Modeling, In ISCE '09. IEEE 13th International Symposium on Consumer Electronics

[10] Dimitris Economou , Suzanne Rivoire , Christos Kozyrakis. Full-system power analysis and modeling for server environments, In 2006 Workshop on Modeling Benchmarking and Simulation (MOBS)

[11] Zhuo Liu, Jian Zhou, Weikuan Yu, Fei Wu, Xiao Qin, and Changsheng Xie, MIND: A Black-Box Energy Consumption Model for Disk Arrays, In International Green Computing Conference 2011

[12] Aqeel Mahesri and Vibhore Vardhan, Power Consumption Breakdown on a Modern Laptop, In Lecture Notes in Computer Science Volume 3471, 2005, pp 165-180

[13] Hung-Ching Chang, Erik Kruus, Thomas J Barnes, Abhishek R. Agrawal, and Kirk W. Cameron, Storage Power Optimizations for Client Devices and Data Centers, In Intel Technology Journal on Energy and Sustainability, 2012.

[14] Takuro Inoue, Makoto Ikeda, Tomoya Enokido, Ailixier Aikebaier, and Makoto Takizawa, A Power Consumption Model for Storage-based Applications, In 2011 International Conference on Complex, Intelligent, and Software Intensive Systems

[15] Meikel Poess and Raghunath Othayoth Nambiar, Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results, In Proceedings of the VLDB Endowment Journal

[16] James William Smith, Ali Khajeh-Hosseini, Jonathan Stuart Ward, and Ian Sommerville, CloudMonitor: Profiling Power Usage, In 2012 IEEE 5th International Conference on Cloud Computing

[17] Jack Dongarra, Hatem Ltaief, Piotr Luszczek, and Vincent M. Weaver, Energy Footprint of Advanced Dense Numerical Linear Algebra Using Tile Algorithms on Multicore Architectures, In 2012 Second International Conference on Cloud and Green Computing (CGC)

[18] Jerasure1.2: A Library in C/C++ Facilitating Erasure Coding for Storage Applications version 1.2 -

http://web.eecs.utk.edu/~plank/plank/papers/CS-08-627.html

[19] Jerasure erasure code open source software from UTK - "Jerasure.org"

[20] Open Source Encoder and Decoder for SD Erasure Codes - University of Tennessee at Knoxville tehnical report CS-13-704

[21] James S. Plank , Kevin M. Greenan , and Ethan L. Miller , Screaming Fast Galois Field Arithmetic Using Intel SIMD Instructions, In FAST 2013: 11th USENIX Conference on File and Storage Technologies,

[22] Gaurav Mitra, Beau Johnston. Alistair P. Rendell, and Eric McCreath , and Jun Zhou, Use of SIMD Vector Operations to Accelerate Application Code Performance on Low-Powered ARM and Intel Platforms, In 2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum

[23] "Watts Up/.net" - WattsUP Metters INC

[24] Chung-hsing Hsu and Stephen W. Poole. Power Signature Analysis of the SPECpower_ssj2008 Benchmark, Proceeding ISPASS '11 Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software

[25] Guilherme Calandrini, Alfredo Gardel, Ignacio Bravo, Pedro Revenga, José L. Lázaro, and F. Javier Toledo-Moreo. Power Measurement Methods for Energy Efficient Applications, Sensors Journal, Special Issue State-of-the-Art Sensors Technology in Spain 2013)

[26] Luigi Rizzo, "On the feasibility of software FEC", University di Pisa, Italy, 1998

[27] Maria A. Kazandjieva, Brandon Heller, Philip Levis, Christos Kozyrakis. Energy Dumpster Diving, Second Workshop on Power Aware Computing (HotPower), November 2009

[28] Chung-Hsing Hsu, Jacob Combs, Jolie Nazor, Fabian Santiago, Rachelle Thysell, Suzanne Rivoire. Application Power Signature Analysis, HPPAC 2014 - The Tenth Workshop on High-Performance, Power-Aware Computing

[29] zfec – a fast erasure code used in command line c, python and Haskell: zfec python open source code

[30] TI Reed Solomon Decoder: TMS320C64x Implementation. Application Report SPRA686 - December 2000

[31] Lilian Atieno, Jonathan Allen, Dennis Goeckel and Russell Tessier, An Adaptive Reed-Solomon Errors-and-Erasures Decoder. ACM 2006 FPGA International Conference, February 22–24, 2006, Monterey, California, USA

[32] Yuchong Hu, Henry C.H. Chen, and Patrick P.C. Lee, NCCloud: Applying Network Coding for the Storage Repair in a Cloud-of-Clouds. 2012 the 10th USENIX Conference on File and Storage Technologies (FAST) confernence.

[33] Tahoe-LAFS- Tahoe-LAFS web page

[34] CEPH Erasure code pool document - CEPH Erasure code information from "ceph.com"

[35] Paul Luse and Kevin Greenan, Swift Object Storage: Adding Erasure Code. SNIA Eduction September 2014

[36] Nosayba El-Sayed and Bianca Schroeder, Checkpoint / Restart in Practice: When 'Simple is Better', 2014 IEEE Cluster Conference

[37] Catherine D. Schuman James S. Plank, A Performance Comparison of Open-Source Erasure Coding Libraries for Storage Applications, University of Tennesse Technical Report Technical Report UT-CS-08-625

[38] James S. Plank, Jianqiang Luo, and Catherine D. Schuman, A Performance Evaluation and Examination of Open-Source Erasure Coding Libraries For Storage, USENIX 2009 FAST Conference

[39] John D. Cook Robert Primmer Ab de Kwant Compare Cost and Performance of Replication and Erasure Coding", Hitachi Review Vol 63, July 2014

[40] Jianqiang Luo, Lihao Xu, and James Plank, An Efficient XOR-Scheduling Algorithm for Erasure Codes Encoding, 2009 IEEE International Conference on DependableSystems and Networks

[41] Aleksei Marov and Sergei Platonov, Effective method for coding and decoding RS codes using SIMD instructions, 2014 High performance Extreme Computing Conference

[42] Philip Carns, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley, 24/7 Characterization of Petascale I/O Workloads, 2014 Parallel Processing: 20th International Conference Euro-Par

[43] Youngjae Kim, Raghul Gunasekaran, Galen M. Shipman, David A. Dillow, Zhe Zhang, and Bradley W. Settlemyer, Workload Characterization of a Leadership Class Storage Cluster, SC 2009, PDSW workshop

[44] James Ahrens, Implications of Numerical and Data Intensive Technology Trends on Scientific Visualization and Analysis, The 2015 SIAM Computational Science and Engineering (CSE)