2013

Energy Analysis of Hadoop Cluster Failure Recovery

Weiyue Xu
University of Nebraska – Lincoln, weiyue@cse.unl.edu

Ying Lu
University of Nebraska-Lincoln, ying@unl.edu

2013 International Conference on Parallel and Distributed Computing, Applications and Technologies
Pages: 141 - 146, DOI: 10.1109/PDCAT.2013.29

Energy Analysis of Hadoop Cluster Failure Recovery

Weiyue Xu, Ying Lu
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE, USA
{weiyue, ylu}@cse.unl.edu

***Abstract*—Energy efficiency is now used as an important metric for evaluating a computing system. However, saving energy is a big challenge due to many constraints. For example, in one of the most popular distributed processing frameworks, Hadoop, three replicas of each data block are randomly distributed in order to improve performance and fault tolerance. But such a mechanism limits the largest number of machines that can be turned off to save energy without affecting the data availability. To overcome this limitation, previous research introduces a new mechanism called covering subset which maintains a set of active nodes to ensure the immediate availability of data, even when all other nodes are turned off. This covering subset based mechanism works smoothly if no failure happens. However, a node in the covering subset may fail.**

In this paper, we study the energy-efficient failure recovery in Hadoop clusters. Rather than only using the replication as adopted by a Hadoop system by default, we investigate both replication and erasure coding as possible redundancy mechanisms. We develop failure recovery algorithms for both systems and analytically compare their energy efficiency.

I. Introduction

Enormous data centers are built to support various types of data processing services like email services and searching engine services. Data centers are becoming crucial in modern life. According to data center research organization Uptime Institute's survey in May 2011, 36 percent of the large companies surveyed were expecting to exhaust IT capacity within 18 months. That means, they must enlarge their existing data centers or build more data centers. To maintain a data center, a large amount of energy need to be consumed for both computing and cooling [2]. It is reported that in 2010 2% of electricity is used by data centers in US and 1.3% around the world [5]. As data centers continue to grow in size and number, some researchers estimated that by 2012 the cost of electricity for data centers could exceed the cost of the original capital investment [10]. As a result, how to achieve energy efficiency is a major issue for data centers [1].

One typical and effective way for saving computing energy is to shut down idle machines. According to [3] and [6], there are a large number of idle machines in data centers consuming up to 60% of the total energy. In order to inactivate as many machines as possible to save energy, researchers attempt to dynamically match the number of activate nodes with the current workload [3]. However, it is nontrivial to apply this approach to MapReduce framework which is a popular and powerful programming model for data-intensive cluster computing. First, MapReduce framework stores the data across many nodes in order to provide an affordable storage for multi-petabyte datasets with a good performance and reliability. [4] indicates that data availability requirement prohibits a MapReduce system from shutting down idle nodes even if significant periods of inactivity are observed. During such periods, energy consumed by those idle machines is wasted. Moreover, MapReduce provides mechanisms to ensure fault tolerance and load balance, which also exert a negative effect on the energy efficiency. By default, a well-known open source MapReduce framework implementation, Hadoop, employs three replications for each data block and the copies are distributed randomly in the cluster. This mechanism actually limits the number of nodes that can be turned off without affecting the data availability.

In order to address this limitation, Leverich et al. introduce a new mechanism which groups machines of a MapReduce cluster into two subsets, i.e., covering and non-covering subsets [4]. At least one replica of all data blocks must be stored in the covering subset nodes. This way, it ensures the immediate availability of the data, even when all nodes in non-covering subset are turned off. With this mechanism, non-covering subset nodes can be turned on or off according to the workload volume without affecting the data availability. It is shown that the covering subset approach can save between 9% and 50% of energy consumption in Hadoop clusters [4]. This approach overcomes the aforementioned limitation and is likely to be widely adopted.

Complementary to Leverich et al.'s work [4], this paper investigates failure recovery in Hadoop clusters and analyzes the energy consumed in the process. A node of a cluster can become unaccessible when it experiences hardware/software errors. Large clusters of commodity machines often have high failure rates, where the Mean Time Between Failure (MTBF) could be as short as 40 minutes [9]. Since a Hadoop cluster is usually constructed with commodity machines, a node failure happens frequently and cannot be neglected. In Hadoop, as well as most other well-known MapReduce implementations, replication is the default redundancy mechanism used to achieve fault tolerance. Although straightforward, replication leads to high storage overhead. There is another commonly-used redundancy technology, erasure coding (i.e., parity schema), which uses an order of magnitude less storage than replication under the same fault tolerance level. However, to create redundant fragments and to recover lost data, extra encoding and decoding efforts are required.

This paper investigates energy efficient failure recovery in Hadoop clusters, where either replication or erasure coding is adopted as the data redundancy mechanism. Similar to the covering subset mechanism [4], we divide the Hadoop cluster into several sets and always keep the set of nodes that store a copy of all data blocks online. Upon a node failure, a recovery mechanism will be invoked to restore the data availability using redundant data stored in off-line nodes. Greedy failure recovery algorithms are developed. To analyze and compare the energy efficiency of failure recovery, we build energy models and simulate node failures and recoveries in clusters of varied sizes.

II. System Structure

This section describes the system structure we create for a Hadoop cluster. All files stored in a Hadoop cluster are managed by Hadoop distributed filesystem, i.e. HDFS, which follows a master-slave architecture. In particular, there is a master node named Namenode and a number of slave nodes called Datanodes. In HDFS, Namenode maintains the file system namespace and the files' metadata. Files are divided into fixed-sized (64MB by default) blocks and distributively stored in Datanodes. We assume a homogenous Hadoop cluster where Datanodes are grouped into three sets: *Fundamental Set (FS), Extended Set (ES), and Waiting Set (WS).* FS is similar to the covering subset (CS) defined in [4] which stores only one copy of all data blocks. Redundant data blocks, generated following a redundancy mechanism (i.e., replication or erasure coding), are stored in ES nodes. The size of ES, i.e., the number of Datanodes contained in ES, is proportional to that of FS. In both FS and ES, data blocks are randomly distributed and stored in their nodes, where each node has approximately the same number of data blocks. Datanodes in WS are not assigned to store any data and they are used as backup nodes for failure recovery. When there is no node failure, only nodes in FS are required to be active to respond to data requests while nodes can be turned off in ES and WS to save energy.

In this paper, we consider FS node crash failures. In particular, we study and analyze the recovery of a single node failure, since it contributes up to 90% of node failures in typical commodity clusters [7]. Because only one replica of all data blocks are stored in FS nodes, when a FS node fails complete data availability in FS is lost. To restore it, we activate a new node in WS to replace the failed node in FS. Specifically, we recover the lost data replicas by utilizing the redundant data stored in ES nodes and copy them to the newly activated WS node, which will then be added to FS. Since not all data blocks are available before failure recovery completes, we assume that Datanodes are not serving any data request during the failure recovery process. Thus, any energy consumed in the process is considered as overhead and should be minimized.

In HDFS, replication is the default redundancy mechanism used to achieve fault tolerance. When creating a file, a user can define a replication factor r, which tells the system how many replicas should be maintained for the file's data blocks. By default, $r = 3$, i.e. three copies of each block are stored in HDFS. Unlike simple replication, erasure coding technology encodes redundant data. It transforms an object of d blocks into a set of $d + e$ fragments such that any d out of the $d + e$ fragments can be used for reconstructing the original d blocks. Erasure coding thus provides some flexibility in choosing fragments for failure recovery. Furthermore, as long as $e < (r - 1) \times d$, erasure coding uses less storage space than simple replication. In this paper, we study failure recovery in Hadoop clusters, where either replication or erasure coding is adopted as the data redundancy mechanism.

III. Failure Recovery Algorithms

To recover a FS node failure, a naive approach is to turn on all Datanodes in ES, making all redundant data available for data reconstruction. Since the process of starting up machines consumes high energy and turning on and off machines causes wear and tear, this approach is inappropriate. In this paper, we investigate and develop energy efficient methods for restoring lost data, which incorporate greedy algorithms, aiming to activate minimal number of ES nodes for failure recovery, in replication-based and erasure-coding-based Hadoop clusters respectively.

A. Failure Recovery in Replication-Based System

By default, simple replication approach is used for maintaining data redundancy in HDFS. Accordingly, we first study the case where r copies of all data blocks are stored in the cluster. As mentioned, we divide all Datanodes into three sets, FS, ES, and WS, and the size of ES is proportional to that of FS. That is, if a total m number of unique data blocks are stored in n FS nodes, then $(r-1)\times m$ data blocks are maintained in $(r-1) \times n$ ES nodes. For fault tolerance, replicas of a data block are kept in different nodes. In both FS and ES, blocks are randomly distributed and stored. Thus, on average, a node has around $k = \frac{m}{n}$ number of data blocks.

When a FS node fails, we lose k unique data blocks. To restore them, we desire to identify the smallest number of nodes that store at least one copy of these k blocks. We can reduce this failure recovery problem to the set covering problem because essentially we need to identify the smallest number of sets (i.e., sets of data blocks stored in ES nodes) whose union contains all lost data blocks. The set covering problem is known to be NP-hard. Therefore, we develop a greedy algorithm to find a near-optimal solution. The main idea of the greedy algorithm is quite simple. At each step, it activates one ES node and continues until all lost data blocks are found in the activated ES nodes. When picking the next node, it always chooses the one that will lead to the recovery of the largest number of blocks. After finding all lost blocks in the activated ES nodes, they are transferred to a newly activated WS node, which will then be added to the FS.

Since each block has $(r-1)$ replicas stored in $(r-1) \times n$ ES nodes, when a data block is lost, the probability of finding its replica in an ES node is:

$$p(replication) = \frac{r-1}{(r-1) \times n} = \frac{1}{n} \tag{1}$$

If we take the k lost data blocks into consideration, the probability of finding a large number of them in a node is small. Therefore, in order to recover from a node failure, quite a few ES nodes need to be turned on.

B. Failure Recovery in Erasure-Coding-Based System

In this section, we develop a greedy failure recovery algorithm for Hadoop clusters where erasure coding is used to generate redundant data. In such erasure-coding-based system, we treat d blocks as an erasure coding object and from them, we generate e encoded blocks and form an erasure coding set of $d+e$ blocks. Since FS nodes are kept online for serving data requests, the original d blocks of an erasure coding set are put in FS while the encoded e blocks are stored in ES. Assume there are a total m number of unique data blocks stored in n FS nodes. We have $s = \lceil \frac{m}{d} \rceil$ number of erasure coding objects and sets. To store $e \times \lceil \frac{m}{d} \rceil$ number of encoded blocks, $e \times \lceil \frac{n}{d} \rceil$ number of nodes are used in ES. For fault tolerance, blocks of an erasure coding set are kept in different nodes. Blocks are randomly distributed and stored in both FS and ES nodes. Thus, on average, a node has around $k = \frac{m}{n}$ number of data blocks.

When a FS node fails, we lose k unique data blocks from k different erasure coding objects. To restore them, we need to reconstruct the k objects. Since an object can be reconstructed using any d blocks of its erasure coding set and the set still has $d-1$ blocks stored in working FS nodes, only one more block of the set needs to be retrieved from an ES node to reconstruct the object. A set has e encoded blocks stored in ES nodes and any one of them can be used for the failure recovery. To recover the k lost blocks, i.e., to reconstruct the corresponding k objects, we would like to identify the smallest number of ES nodes that store at least one encoded block of these k erasure coding sets. This problem is NP-hard and we develop a greedy algorithm to find a near-optimal solution. The main idea of the greedy algorithm is quite simple. At each step, it activates one ES node and continues until each of the k sets has at least one block in the activated ES nodes. When picking the next node, it always chooses the one that will lead to the recovery of the largest number of objects.

After finding adequate blocks in the activated ES nodes, we need to use them for data reconstruction. As described in [11], decoding blocks and reconstructing erasure coding objects is computation-intensive and time-consuming. Rather than reconstructing the k objects on a node, we would like to leverage all working FS nodes to parallelize data transmission and reconstruction so as to shorten the recovery process. On average, each FS node is assigned to reconstruct $\frac{k}{n-1}$ number of objects. Since each object still has $d-1$ data blocks stored in FS nodes, to reduce data transfer, a FS node with a block for an object is chosen for reconstructing the object. Besides an encoded block retrieved from an ES node and a local block, to build the object, a FS node still needs to get $d-2$ original blocks from other FS nodes. Note that the decoding and reconstruction process can only be started after having d blocks of a set available locally. Therefore, we transfer data blocks following the set order. This way, we can compute for an erasure coding set while simultaneously transferring data for another. The recovered data blocks will then be sent to a newly activated WS node and added to the FS.

Since each set has e encoded blocks stored in $e \times \lceil \frac{n}{d} \rceil$ ES nodes, the probability of recovering one lost data block in FS by a random node in ES is:

$$p(erasure\_coding) = \frac{e}{e \times \lceil \frac{n}{d} \rceil} \approx \frac{d}{n} = d \times p(replication) \tag{2}$$

That is, $p(erasure\_coding)$ is about d times of $p(replication)$ (Eq. (1)). As a result, a fewer number of machines are turned on to tackle a node failure. However, it is important to notice that this smaller number of active nodes does not necessarily guarantee energy-efficient failure recovery. While the recovery in erasure-coding-based system requires a fewer number of machines to be turned on, extra data transfer and computation are needed. In order to compare energy efficiency of failure recovery in these two systems, we build energy models and give detailed analysis in Section IV.

IV. Energy Models

In this section, we build models to analyze the energy consumption of failure recovery in Hadoop clusters.

When a FS node fails, we invoke the recovery algorithm (see Section III) to restore the lost data. Some ES nodes will be activated. Data will be transmitted between nodes. In erasure-coding-based clusters, the recovery also involves computation where lost data are reconstructed. Since we consider homogenous clusters, power consumed by any two nodes to do the same job are assumed to be the same. We denote the power consumed by a node in data transfer (i.e., sending/receiving data blocks[1]), decoding (i.e., reconstructing the data), and doing both simultaneously as P_{tran}, P_{comp}, and P_{t+c} and the corresponding energy consumption as E_{tran}, E_{comp}, and E_{t+c}. Moreover, the energy and time spent by a node during the activation (inactivation) process are represented as E_{act} and T_{act} (E_{inact} and T_{inact}). P_{idle} and E_{idle} represent the power and energy consumption of an idle machine. Since our cluster consists of FS, ES, and WS and the recovery process starts with retrieving data from ES and ends with receiving data in WS, we discuss the energy

[1]In order to simplify the analysis, we assume a node consumes the same amount of energy in sending and receiving the same amount of data.

consumption of a failure recovery algorithm by analyzing energy spent in ES, FS, and WS (i.e., $E(ES)$, $E(FS)$, and $E(WS)$) during the recovery process.

A. Energy spent in ES

In ES, a set of nodes will be turned on to send out data blocks needed for data recovery and these nodes will be turned off immediately after the data transmission. We assume every activated ES node consumes the same amount of energy during the recovery. Suppose A nodes are activated and each node spends T_{ES} seconds in sending out blocks. Then, the energy consumed in ES can be described as:

$$E(ES) = A \times (E_{act} + P_{tran} \times T_{ES} + E_{inact}) \quad (3)$$

- **For replication-based system:**
  When replication is used, data blocks will be directly transferred from ES nodes to a WS node. As a result, data transfer speed is limited by the receiving capability of the WS node rather than the aggregated sending capability of ES nodes. That is, we need $\frac{64\times k}{b}$ seconds to transfer k $64MB$ data blocks, where a node's transmission speed is bMB/s. Consequently, Eq (3) becomes[2]:
  $$E(ES)_R = A_R \times (E_{act} + P_{tran} \times \frac{64 \times k}{b} + E_{inact}) \quad (4)$$
- **For erasure-coding-based system:**
  When erasure-coding is used, k encoded data blocks will be transferred from ES nodes to $n-1$ FS nodes. Assume A_{EC} number of ES nodes are activated. Their total sending capability is $A_{EC} \times b$, while the total receiving capability of working FS nodes is $(n-1)\times b$. These two set a limit on the possible transmission speed. Therefore, we need $\frac{64\times k}{min(A_{EC},(n-1))\times b}$ seconds to transfer k $64MB$ data blocks from ES to FS. Eq. (3) becomes:
  $$E(ES)_{EC} = \quad (5)$$
  $$A_{EC} \times (E_{act} + P_{tran} \times \frac{64\times k}{min(A_{EC},(n-1))\times b} + E_{inact})$$

B. Energy spent in FS

The energy consumed by FS nodes also varies with the adopted redundancy technology. If replication approach is employed, all FS nodes are idle during the recovery process. While if the erasure coding approach is applied, FS nodes are involved in data transfer and decoding. The energy consumed in FS can be described as follows:

$$E(FS) = (n-1) \times (E_{idle} + E_{tran} + E_{t+c} + E_{comp}) \quad (6)$$

- **For replication-based system:**
  In replication-based system, FS nodes are idle during the whole recovery process. They wait for $T_{act} + \frac{64\times k}{b}$ seconds when ES nodes are activated and data are transferred from ES nodes to a WS node. Thus, the energy consumed by FS nodes is:
  $$E(FS)_R = (n-1) \times P_{idle} \times (T_{act} + \frac{64 \times k}{b}) \quad (7)$$

[2]In this paper, subscripts R and EC will be added to parameters when we refer to the replication-based and erasure-coding-based systems.

- **For erasure-coding-based system:**
  In erasure-coding-based system, we reconstruct data in FS nodes. As described in Section III-B, on average, each FS node reconstructs $\frac{k}{n-1}$ number of erasure coding objects. Since each object still has $d-1$ blocks stored in FS nodes, to reduce data transfer, a FS node with a block for an object is chosen for reconstructing the object. A FS node retrieves $d-2$ blocks from other FS nodes and an encoded block from an ES node, and then reconstructs the object using the d blocks. To start with, blocks from ES nodes are transferred to destination FS nodes. Specifically, a FS node first waits for powering up ES nodes and then receives data from them. It is idle for T_{act} seconds and then spends $\frac{64\times k}{min(A_{EC},(n-1))\times b}$ seconds in accepting $\frac{k}{n-1}$ data blocks from ES nodes. The corresponding energy consumption is $P_{idle} \times T_{act} + P_{tran} \times \frac{64\times k}{min(A_{EC},(n-1))\times b} J$. Next, a FS node retrieves the $d-2$ blocks from other FS nodes to reconstruct an object. That is $T_{tran} = \frac{64\times(d-2)}{b}$ seconds. As discussed in Section III-B, data blocks are transferred following their object order so that data transfer and decoding can be launched simultaneously. After receiving all blocks for an erasure coding object, a FS node starts reconstructing the object while receiving blocks for another. Suppose the decoding time of an object is T_{comp}. There are two scenarios:
  (a) When $T_{tran} \geq T_{comp}$, data transfer and decoding overlap for $T_{comp} \times (\frac{k}{n-1} - 1)$ seconds when decoding all but the last set. A FS node spends $T_{tran} + (T_{tran} - T_{comp}) \times (\frac{k}{n-1} - 1)$ seconds exclusively in data transfer and T_{comp} seconds exclusively in decoding. Thus, the energy consumed by a FS node in this period is:
  $$\begin{aligned} & E_{tran} + E_{t+c} + E_{comp} \\ &= P_{tran} \times (T_{tran} + (T_{tran} - T_{comp}) \times (\frac{k}{n-1} - 1)) \\ &+ P_{t+c} \times T_{comp} \times (\frac{k}{n-1} - 1) + P_{comp} \times T_{comp} \end{aligned} \quad (8)$$
  (b) When $T_{tran} < T_{comp}$, data transfer and decoding overlap for $T_{tran} \times (\frac{k}{n-1} - 1)$ seconds when transferring all but the first set. A FS node spends $(T_{comp} - T_{tran}) \times (\frac{k}{n-1} - 1) + T_{comp}$ seconds exclusively in decoding and T_{tran} seconds exclusively in data transfer. The energy consumed in this period is:
  $$\begin{aligned} & E_{tran} + E_{t+c} + E_{comp} \\ &= P_{tran} \times T_{tran} \\ &+ P_{t+c} \times T_{tran} \times (\frac{k}{n-1} - 1) \\ &+ P_{comp} \times ((T_{comp} - T_{tran}) \times (\frac{k}{n-1} - 1) + T_{comp}) \end{aligned} \quad (9)$$

After data reconstruction, FS nodes send the reconstructed k blocks to a WS node. That is $\frac{64\times k}{b}$ seconds in data transfer and the corresponding energy cost of a FS node is: $P_{tran} \times \frac{64 \times k}{b}$.

Putting them together, we have:

$$E(FS)_{EC} = \begin{cases} \text{if } T_{tran} \geq T_{comp} \\ (n-1) \times (P_{idle} \times T_{act} + P_{tran} \\ \times (\frac{64 \times k}{b} + \frac{64 \times k}{min(A_{EC}, (n-1)) \times b} \\ +(T_{tran} - T_{comp}) \times \frac{k}{n-1} + T_{comp}) \\ +P_{t+c} \times T_{comp} \times \frac{k-n+1}{n-1} \\ +P_{comp} \times T_{comp}) \\ \\ \text{if } T_{tran} < T_{comp} \\ (n-1) \times (P_{idle} \times T_{act} + P_{tran} \\ \times (\frac{64 \times k}{b} + \frac{64 \times k}{min(A_{EC}, (n-1)) \times b} \\ +T_{tran}) + P_{t+c} \times T_{tran} \times \frac{k-n+1}{n-1} + \\ P_{comp} \times ((T_{comp} - T_{tran}) \times \frac{k}{n-1} \\ +T_{tran})) \end{cases} \tag{10}$$

C. Energy spent in WS

The energy consumed by the newly activated WS node is the same for the two systems. That is:

$$\begin{aligned} E(WS)_R &= E(WS)_{EC} \\ &= E_{act} + P_{tran} \times \frac{64 \times k}{b} \end{aligned} \tag{11}$$

D. Energy Consumption of the Two Approaches

Based on the design of the two systems (see Section III), we anticipate that $A_{EC} \ll A_R$, i.e., a smaller number of ES nodes need to be active in the erasure-coding-based system. Thus, the erasure-coding-based system consumes less energy in ES (see Equations (4) and (5)). However, unlike replication-based system in which data can be used directly, erasure coding requires extra computation in FS nodes. Therefore, erasure-coding-based system consumes more energy in FS (see Equations (7) and (10)).

V. Simulation

In this section, we compare the energy consumption of failure recovery in replication-based vs. erasure-coding-based Hadoop systems. We first profile our local testbed, a cluster called $Bugeater2$, to get realistic system parameters for energy models developed in Section IV. Then, we simulate a FS node failure and recovery in clusters of varied sizes and estimate the energy consumption involved in the process.

In $Bugeater2$ cluster, a node has two AMD Opteron(tm) Processors 248 (2.2GHz, 64bit), 4GB Memory and one 80G-B SATA disk. The speed of the switch that connects these nodes is $1Gbps$. We use a Server Tech CWG-CDU power distribution unit (PDU) to measure the energy consumption. The energy consumed to boot up a machine is 24650 Joule, that is, $E_{act} = 24650J$, while the energy consumed to shut down a machine is very small and can be ignored, hence, we set $E_{inact} = 0J$. When a machine is idle, it consumes 70 Joule per second, i.e., $P_{idle} = 70J/s$. A node consumes 90 Joule per second when busy in decoding, that is, $P_{comp} = 90J/s$. Based on our measurements, the extra energy consumed in data transfer is negligible. Thus, we set $P_{tran} = P_{idle} = 70J/s$ and $P_{t+c} = P_{comp} = 90J/s$. The time spent in activating a machine in $Bugeater2$ is about 30 seconds, i.e. $T_{act} = 30s$. Moreover, although the ethernet is supposed to be $1Gbps$, i.e., $128MB/s$, but in practice, we can achieve a transmission rate of only $35MB/s$. Thus, we set $b = 35MB/s$.

We simulate clusters of varied sizes. As mentioned, a cluster is divided into three sets: FS, ES, and WS. The FS has n number of nodes, where n ranges from 4 to 96. The number of nodes in ES depends on the adopted redundancy mechanism and is proportional to the size of FS. If replication approach is used, we have $(r-1) \times n$ number of nodes in ES, where r is the replication factor. While if erasure coding approach is used, $e \times \lceil \frac{n}{d} \rceil$ number of nodes are included in ES. There are 2 backup nodes in WS. $n \times 20GB$ of data are assumed to be stored in FS nodes, i.e., an average of $20GB$ per node. The data block size is set at $64MB$. Therefore, about $20GB/64MB = 320$ blocks are stored in a node (i.e., $k = 320$).

To compare the failure recovery in replication-based vs. erasure-coding-based system, we consider two situations: 1) same fault tolerance level and 2) same storage space consumption. In both situations, we always use Hadoop's default replication factor $r = 3$ for the replication approach. Based on the performance comparison [8] of different erasure coding libraries, we employ $Jerasure$ erasure coding in the erasure-coding-based system.

A. Results for Same Fault Tolerance Level

We first consider the situation where replication-based and erasure-coding-based systems have the same fault tolerance level, that is, when $e = r - 1 = 2$.

In order to compute the total energy consumption spent in erasure-coding-based system, we still need to know the amount of time spent in decoding and reconstructing an erasure coding object, i.e., T_{comp}. According to the erasure coding algorithm, when the block size is fixed, the object reconstruction time depends on the object size and the encoding rate, i.e., d and $\frac{d}{d+e}$. Table I presents the measured computation time on a $Bugeater2$ node as the object size d changes.

To estimate energy consumed during the failure recovery, i.e., E_R vs. E_{EC} (see Equations (4), (7), and (11) vs. Equations (5), (10), and (11)), the number of ES nodes to be activated, i.e., A_R and A_{EC}, should be derived. Given a cluster, we simulate the recovery of a random node failure 10 times and use the averages as the values for A_R and A_{EC}. Simulation results show that fewer numbers of ES nodes need to be activated in erasure-coding-based Hadoop clusters. More importantly, to recover the failure in such systems, the energy consumed could be much less. Figure 1 shows the estimated energy consumption of failure recovery

TABLE I
OBJECT RECONSTRUCTION TIME WHEN $e = 2$

object size (d)	reconstruction time (sec)
3	0.514
4	0.756
5	0.874
6	1.086
8	1.754
10	2.889
12	3.98
16	6.442
20	10.598
24	16.363

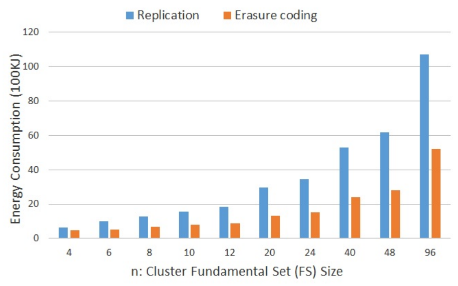

Fig. 1. Energy consumption of failure recovery (same fault tolerance level)

in the two systems as cluster size increases[3]. As we can see, the energy consumption is always less in the erasure-coding-based system and the energy saving increases with the cluster size.

B. Results for Same Storage Space Consumption

We also compare replication-based and erasure-coding-based clusters when the sizes of the two are the same. That is, the two clusters have the same number of nodes and consume the same amount of storage space. By letting $e = (r - 1) \times d$, there are always $r \times n + 2$ number of nodes in both systems. Due to space limitation, here we only display the estimated energy consumption in Figure 2(detailed simulation and analytical results can be found in [12]). It again shows that the erasure-coding-based approach is a better choice than the replication-based method and the gain in energy saving increases with the cluster size.

VI. Conclusion and Future Work

In this paper, we investigate energy efficient failure recovery in Hadoop clusters, where either replication or erasure coding is adopted as the data redundancy mechanism. To evaluate the energy efficiency of failure recovery in the two systems, we construct energy models, simulate node failure recovery in clusters of varied sizes, and analyze the energy consumed. Simulation and analytical data show that the failure recovery in the erasure-coding-based Hadoop system often outperforms the replication based counterpart. On average, the former requires 60% of the energy as that of the later and the energy saving increases with the cluster size.

[3]In Figures 1 and 2, for a given cluster we use the lowest energy consumption E_{EC} of the erasure-coding-based system that is achieved with an optimal setting of d.

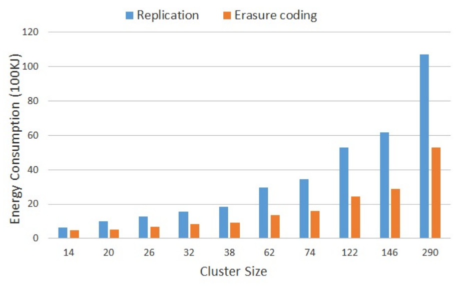

Fig. 2. Energy consumption of failure recovery (same storage space consumption)

This paper simulates and analyzes the energy consumption under different scenarios based on realistic data profiled from our testbed cluster. In the future, we plan to validate our results by building the Hadoop systems (e.g., in Green Server Farm at University of Illinois at Urbana-Champaign) and collecting the real energy consumption data for failure recovery.

References

[1] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, December 2007.

[2] Emerson Network Power. *Energy Logic: Reducing Data Center Energy Consumption by Creating Savings that Cascade Across Systems*, Nov 2007.

[3] Xiaobo Fan, Wolf dietrich Weber, and Luiz Andr Barroso. Power provisioning for a warehouse-sized computer. In *In Proceedings of ISCA*, 2007.

[4] Jacob Leverich and Christos Kozyrakis. On the energy (in)efficiency of hadoop clusters. *SIGOPS Oper. Syst. Rev.*, 44:61–65, March 2010.

[5] John Markoff. Data centers's power use less than was expected, July 2011.

[6] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: eliminating server idle power. In *ASPLOS'09*, pages 205–216, 2009.

[7] Bianca Schroeder and Garth A. Gibson. Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you? In *Proceedings of the 5th USENIX conference on File and Storage Technologies*, FAST '07, Berkeley, CA, USA, 2007. USENIX Association.

[8] Catherine D. Schuman and James S. Plank. A performance comparison of open-source erasure coding libraries for storage applications. Technical report, University of Tennessee, 2008.

[9] Predrag T. Tošić. Microsoft online services division research and development,http://newsroom.unl.edu/announce/stories/11480, Februray 2013.

[10] U.S. Department of Energy. *Quick Start Guide to Increase Data Center Energy Efficiency*, June 2010.

[11] Hakim Weatherspoon and John Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 328–338, London, UK, UK, 2002. Springer-Verlag.

[12] Weiyue Xu. Energy-efficient failure recovery in hadoop cluster, M.S. Thesis, University of Nebraska-Lincoln, May 2013.