GreenEDFS: Saving Energy in the Clouds Using an Efficient Distributed File System

Debessay Fesehaye
Department of Computer Science
UIUC, 201 N Goodwin Ave
Urbana, IL 61801-2302, USA
dkassa2@uiuc.edu

Klara Nahrstedt
Department of Computer Science
UIUC, 201 N Goodwin Ave
Urbana, IL 61801-2302, USA
klara@cs.uiuc.edu

Abstract

This is just a short abstract describing a simple extension of our previous work on efficient distributed file system (EDFS) for cloud computing. The extension, GreenEDFS, aims at making EDFS more energy efficient.

The GreenEDFS scheme uses EDFS mechanism to constantly monitor the cloud resources. It then moves computation, communication and storage loads from less loaded (under utilized) areas of the cloud to other less loaded areas of the cloud by ensuring full resource utilization and avoiding unnecessary over-utilization (buffers, delays and losses). GreenEDFS moves load (computation, communication, storage) from one area of the cloud to the other every greening periods. These periods can be user defined or obtained and triggered using some clever ways, for instance when the load in a given cloud node gets below a certain threshold or when there are some user-defined number of under-utilized resources. The movement of traffic in the cloud (greening) can be done both at a local cloud rack level or globally by going all the way up in the cloud structure the same way EDFS performs the resource allocation.

The scheme dynamically expands and contracts the cloud size based on the demands and continuous monitoring provided by EDFS. This can result in a clever significant energy saving mechanism without compromising performance. We are working to demonstrate the energy saving of our scheme using simulation and real testbeds.

I. How GreenEDFS Works

By adding a few features to EDFS and ensuring full utilization of the resources, some energy savings can be made as follows:

- Denote $K_i/d_i$ and $\Lambda_i$ to be the resource capacity and total current load at block server (BS) $i$ with $d_i$ being a control interval. Each rate monitor (RM) $i$ associated with each BS $i$ monitors the surplus value
  \[ S_i = \frac{K_i}{d_i} - \Lambda_i \]
  along with the other EDFS things it does (such as computing the resource rate allocations for both down and up links).
- If $S_i < 0$ or if using a tolerance value $t_i$ and $S_i < t_i$, then, the RM $i$ turns a surplus flag on along with the other information such as rate EDFS sends to the resource allocator (RA) which is associated with the name node server (NNS) of BS $i$. The RM also sends its $S_i$ value to its NNS.
- At this point the NNS can employ any of the following approaches.
  1) **Using the EDFS rate**: Here the NNS records the $S_i$ flag values of the local and higher level racks. It then moves loads at an EDFS rate from a BS $k$ with $S_k$ flag on and with higher EDFS rate to the BS with the flag on and lower EDFS rate until the $S_i$ of the recipient BS $i$ is 0 or $t_i$.
  2) **Using the surplus value $S_i$**: Here the NNS records the positive $S_i$ values both at a local rack and higher rack levels the same way EDFS calculates the rates. The NNS then moves load from the BS with positive and higher surplus value to a BS with positive and lower surplus value until the surplus value $S_i$ of the recipient BS $i$ is 0 or $t_i$. 

3) The NNS can as well apply any user specified policy to decide which BS server sends data to which one. For instance if the BS in one area are using less energy bill than the BS in another area then load can be adaptively moved from the nodes with higher energy bill to the nodes with lower energy bill.

- In all cases significant energy savings can be made by adaptively moving load from some nodes to other nodes eventually scaling down and up the cloud size without affecting performance.