2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Improving Energy Efficiency of Database Clusters through Prefetching and Caching

Yi Zhou*, Shubbhi Taneja‡*, Mohammed Alghamdi†, Xiao Qin*
*Department of Computer Science and Software Engineering
Auburn University, Auburn, Alabama 36849-5347
Email: {yzz0074, shubbhi, xqin}@auburn.edu ‡corresponding author:shubbhi@auburn.edu
†Computer Science Department
Al-Baha University, Al-Baha City, Kingdom of Saudi Arabia

***Abstract*—The goal of this study is to optimize energy efficiency of database clusters through prefetching and caching strategies. We design a workload-skewness scheme to collectively manage a set of hot and cold nodes in a database cluster system. The prefetching mechanism fetches popular data tables to the hot nodes while keeping unpopular data in cold nodes. We leverage a power management module to aggressively turn cold nodes in the low-power mode to conserve energy consumption. We construct a prefetching model and an energy-saving model to govern the power management module in database clusters. The energy-efficient prefetching and caching mechanism is conducive to cutting back the number of power-state transitions, thereby offering high energy efficiency. We systematically evaluate energy conservation technique in the process of managing, fetching, and storing data on clusters supporting database applications. Our experimental results show that our prefetching/caching solution significantly improves energy efficiency of the existing *PostgreSQL* system.**

***Index Terms*—Energy efficiency, prefetching, energy conservation**

I. Introduction

Improving energy efficiency of parallel database systems becomes indispensable for building data centers supporting transnational database applications [1]. Traditional energy-saving techniques for clusters are inadequate for parallel database systems running on clusters. To address this problem, we focus on reducing energy consumption cost of large-scale database systems, which are comprised of multiple nodes or severs. We show how to develop a parallel database system called *GreenDB* - an energy-effective system running on clusters. At the heart of *GreenDB* is a prefetching and caching mechanism, which fetches hot data from passive nodes into active nodes. *GreenDB* offers energy savings by turning a large number of passive nodes into the low-power mode, which keeping a small number of active nodes to serve queries. The power management of *GreenDB* is governed by two mathematical models, namely, a prefetching model and an energy consumption model. We show how to build these two models and how to integrate the models into the prefetching and caching mechanism.

The following four motivations make energy-efficient prefetching and caching in GreenDB desirable and achievable.

1) Database clusters are cost-effective platforms for parallel database systems.
2) A pressing demand of reducing energy cost of large database systems.

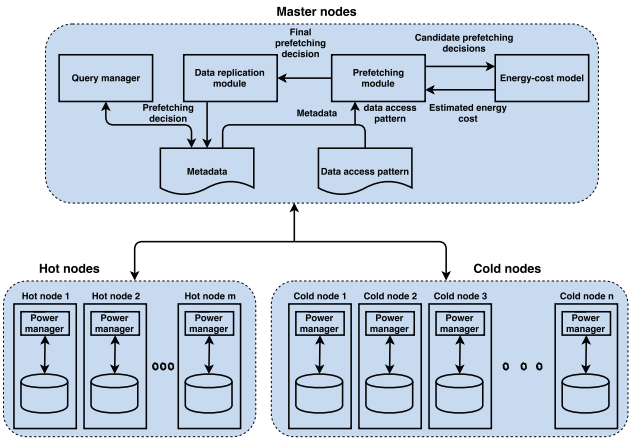


Fig. 1. The architecture of the GreenDB system.

3) Energy-saving techniques designed for single-server database systems are inadequate for DBMS on clusters.

II. System Design

We design the software architecture of *GreenDB* (see Fig. 1, which manages a master node and a group of hot and cold database nodes (or nodes for short). The overarching goal is to keep the cold nodes in the low-power state for a long time period, while making the hot nodes respond queries accessing popular data tables.

A. Query Manager

The query manager handles queries issued by database clients. Sophisticated queries may be partitioned by the query manager to form multiple sub-queries. The query manager relies on the metadata manager to determine the location of data to be accessed by queries. If database tables are residing in a hot node, the queries can be immediately answered. Otherwise, the corresponding cold node should be waked up to fetch and duplicate the tables to a hot node.

978-1-5386-5815-4/18/$31.00 ©2018 IEEE
DOI 10.1109/CCGRID.2018.00065

IEEE computer society

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:08:06 UTC from IEEE Xplore. Restrictions apply.

TABLE I Notation for the Description of the Prefetching Module.

Notation	Desription
L	Current lookahead. $q \in L$ is a reference in the lookahead
table(q)	Talbe accessed in reference $q \in L$
cold node(q)	Cold node in which *table*(q) is residing
Q	Subset of the lookahead Q; for any q in Q, table(q) is active, i.e., $\forall$ q $\in Q$: table(q) is active
H	A set of tables present in the hot node
$E_{saving}(\tau)$	Energy saving contributed by prefetching table τ
Q^+	For any q in Q^+, we have node(q) $\in Q$, $E_{saving}(\tau) > 0$,q$\notin H$, and$\exists\ q \in Q$: *table(q)* = τ
H^+	The set of tables with the highest energy savings in $Q^+ \cup H$

B. Prefetching Module

The prefetching module has two salient features. First, the module keeps track of the access patterns of records in tables to pinpoint popular data. Second, the module creates replicated copies of popular tables to be placed on the most appropriate hot nodes. The decisions of making replicated tables largely depends on potential energy savings which are predicted by our energy-cost model. The existing synchronous-based prefetching strategy (or *SyncPrefetch* for short) in PostgreSQL is expensive. Unlike *SyncPrefetch*, the prefetching scheme in GreenDB takes the full advantage of asynchronous replications to cut energy cost of fetching hot data from cold nodes.

We propose the prefetching algorithm, which aims to prefetch the most energy-efficient tables into the hot node.

Algorithm 1 The Prefetching Module: $Prefetching()$

Input:
a query q
GreenDB with m **cold nodes**
1: **if** $table(q)$ is present in hot node **then**
2: update L;
3: query q is serviced by hot node with $table(q)$;
4: **else**
5: **if** $cold\ node(q)$ is in standby state **then** power up $cold\ node(q)$;
6: **end if**
7: Fetch $table(q)$ to serve query q;
8: **end if**
9: Estimate the energy savings of queries $Q \in L$;
10: Update the energy savings of tables in the hot node;
11: Fetch tables in $Q^+ \cap H^+$
12: **if** disks of the hot node is full **then** Evicting the blocks in H–H^+ with the lowest energy savings as necessary;
13: **end if**

The prefetching algorithm (see Algorithm 1) outlined above initializes the query and the number of cold nodes to q and m, respectively . Next, if the $table(q)$ is resident in the hot node, it serves the query by hot node, otherwise it will power up the cold node in which the $table(q)$ is located and fetch the $table(q)$ into the hot node from the cold node(see Lines 1-7). Finally, we estimate the energy-efficiency of the tables in the lookahead table, fetch most energy-efficiency tables of whole GreenDB into the hot node if they are not kept in the hot node (see Lines 9-11). Unpopular tables are evicted from the hot nodes to release space for newly fetched ones (see Line 12). The above prefetching algorithm will not be triggered, if prefetching tables has detrimental impacts on the energy efficiency of the system.

C. Energy-cost Model

To facilitate the replacement strategy in GreenDB, we delve into an energy-cost model projecting energy savings offered by keeping candidate data tables into the hot nodes while shutting down the cold nodes. The energy-cost model provides guideline for GreenDB to (1) evict unpopular data tables from the hot nodes and (2) fetch popular tables from the cold nodes to improve energy efficiency.

D. Power Manager

The power manager in GreenDB dynamically and automatically changes the power mode of cold nodes to converse energy. For instance, if a cold node has been sitting idle for a given threshold, the power manager immediately transition the node into the low-power mode. Such a threshold can be automatically adjusted in GreenDB depending on workload conditions. For another example, prior to prefetching tables from a cold node to a hot node, the power manager wakes up the cold node.

III. Experimental Results

A. Data Table Size

This set of experiments is focused on evaluating the impacts of data table size (data size for short) on the energy savings offered by GreenDB after replicated tables are created on hot nodes. We set the number of cold nodes to four; we set the hit rate of the hot nodes to 25% (see Fig. 2(a)), 50% (see Fig. 2(b)), and 75% (see Fig. 2(c)), respectively. We compare our GreenDB with the two existing schemes - *PRE-BUD* [2] and *DPM* [3]. For comparison purpose, we also evaluate the database system where no energy-saving scheme is deployed; we refer to this case as *Non-Energy Saving* or *Non-E* for short.

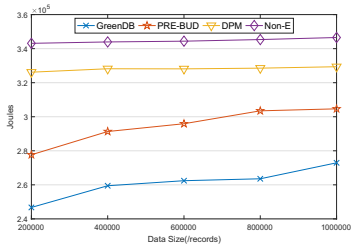
Fig. 2 reveals that the data size imposes noticeable impact on the energy efficiency of GreenDB and PRE-BUD strategy when the hit rate is lower than 75%. Unlike GreenDB and PRE-BUD, the energy consumption of DPM and Non-E are insensitive to data size. The data table size has very little impact on energy efficiency on DPM and Non-E, because power management in DPM and Non-E has nothing to do with creating replicated data tables from cold to hot nodes.

B. The Number of Nodes

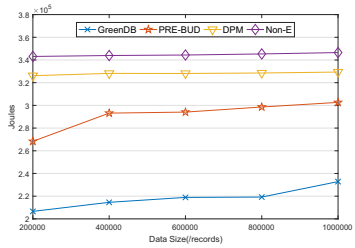
Now we evaluate the impact of the number of nodes on GreenDB. The number of hot node is fixed at one; the number of cold nodes is set to 4, 8, and 12, respectively. The hit rate is configured at the level of 85%. Fig. 3 shows the energy consumption of GreenDB, PRE-BUD, DPM, and Non-E under three data table sizes (i.e., 200K, 400K, and 800K records).

Not surprisingly, we discover from Fig. 3 that when the number of cold nodes goes up, energy savings offered by GreenDB becomes more pronounced. This energy-efficiency trend is expected, because an increasing number of cold nodes are kept in the low-power mode to reduce energy consumption of the database system. After the hot node fetches popular tables from the cold nodes, lightly loaded cold data nodes are more likely to

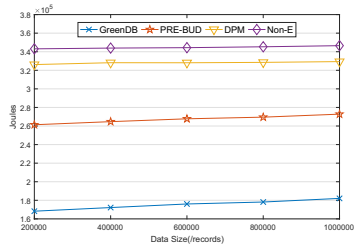
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:08:06 UTC from IEEE Xplore. Restrictions apply.



(a) Total energy consumption of DBMS when the hit rate is 25%.

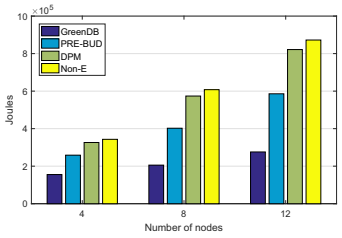


(b) Total energy consumption of DBMS when the hit rate is 50%.

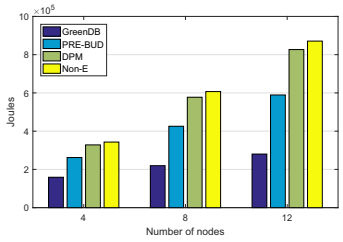


(c) Total energy consumption of DBMS when the hit rate is 75%.

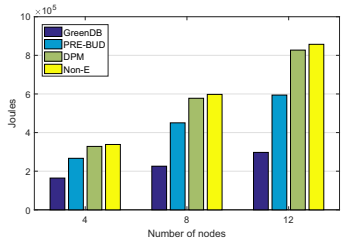
Fig. 2. Total energy consumption of database clusters while data size is varied for three different values of the hit rate: (a) 25%, (b) 50%, (c) 75%.



(a) Total energy consumption of database clusters while data size is fixed at 200K records



(b) Total energy consumption of database clusters while data size is fixed at 400K records



(c) Total energy consumption of database clusters while data size is fixed at 800K records

Fig. 3. Total energy consumption of database clusters while the number of nodes is varied. Data size is fixed at (a) 200K, 400K, 800K records.

be switched into the power-saving mode to conserve energy by increase idle periods that are longer than the break even time.

C. Hit Rates

Hit rate measures the probability that a query can be serviced by hot nodes rather than cold nodes. We configure the data table size at $200K$, $400K$ and $600K$ records. The number of cold nodes is set to four. Fig. 4 shows the total energy consumption as a function of hit rate values.

We observe from Fig. 4 that a high hit rate improves GreenDB's energy efficiency. For instance, when the hit rate is set to 80% and 20%, GreenDB reduces energy consumption of Non-E by 41.0% and 15.1%, respectively (see Fig. 4(c)). This trend is expected, because increasing hit rate enlarges the time period in which cold nodes are kept in the low-power state. A low hit rate leads to a strong likelihood that cold nodes are frequently waked up to serve requests. These experimental results are consistent with those plotted in Fig. 2. We argue that it is unrealistic to achieve a hit rate of 100% in dynamic workload conditions, a 100% hit rate may be accomplished if query access patterns are given a priori.

D. Inter-arrival Delays

Now we study the impact that query inter-arrival rates on the energy savings of GreenDB. Fig. 5 shows the total energy consumption of the database system as a function of the inter-arrival delay. Again, we set the number of cold nodes as four; the data size is fixed at $1,000,000$ records. The hit rate is set to 55%, 65%, 75%, and 85%, respectively.

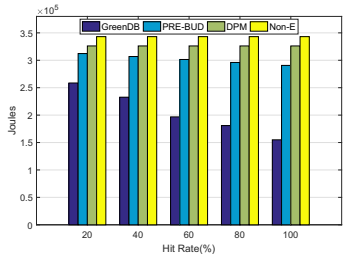
Fig. 5 demonstrated that when the query inter-arrival delay is less than 10 seconds, GreenDB and PRE-BUD are unable to yield any energy savings. Such a low energy efficiency is attributed by the lack of large idle windows that keep cold nodes in the energy-saving state. Unfortunately, in these scenarios, GreenDB and PRE-BUD end up consuming more energy than the traditional DPM due to the prefetching overhead. Both GreenDB and PRE-BUD start delivering energy savings when the inter-arrival delay increase. Intuitively, DPM relies on large idle times between consecutive queries to achieve energy savings; GreenDB, on the other hand, proactively creates large idle windows for cold nodes by redirecting queries to the hot node in the database system.

E. Number of Disks per Cold Node

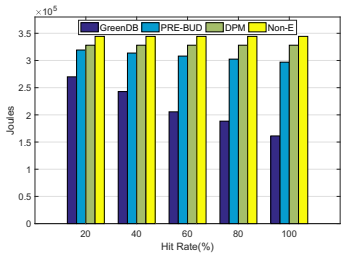
Now we evaluate the capacity impact of cold nodes. In particular, we investigate whether or not increasing the number of disks in each cold node can optimize the energy efficiency of database systems.

We discover from Fig. 6 that increasing the number of disks in each node marginally improves the energy efficiency of GreenDB. The results indicate that adding two extra disks merely increases the energy savings by approximately 2.4% and 2.9% when hit rate is set to 75% and 90%, respectively. The number of disks per node has little impact on GreenDB's energy efficiency, because each disk's power is fairly small compared with that of the entire node. This experiment suggests that with respect to energy saving, a node-level power manager (e.g., GreenDB) is more efficient than a disk-level one (e.g., PRE-BUD). The results show that

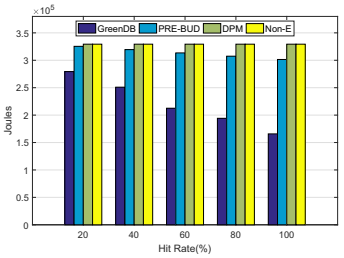
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:08:06 UTC from IEEE Xplore. Restrictions apply.



(a) Total energy consumption when the data size is fixed at 200K records

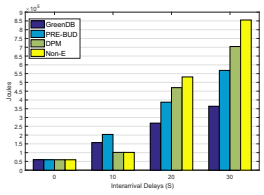


(b) Total energy consumption when the data size is fixed at 400K records

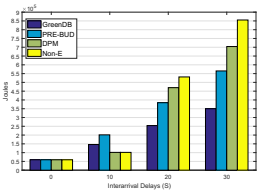


(c) Total energy consumption when the data size is fixed at 600K records

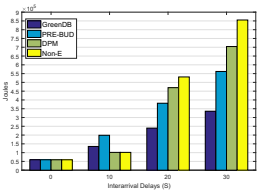
Fig. 4. Total energy consumption for different hit rate values where the data size is fixed at: (a) 200K records, (b) 400K records, (c) 600K records.



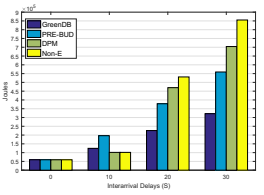
(a) Total energy consumption when the hit rate is : 55%.



(b) Total energy consumption when the hit rate is : 65%.

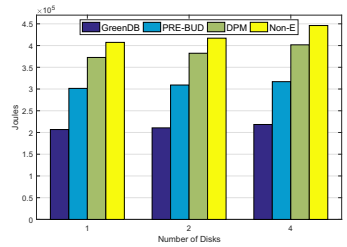


(c) Total energy consumption when the hit rate is : 75%.

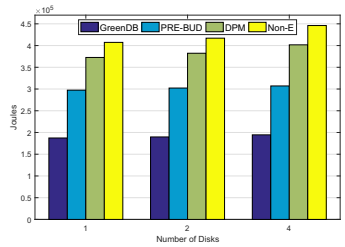


(d) Total energy consumption when the hit rate is : 85%.

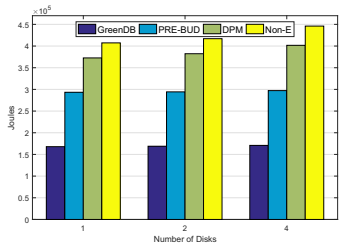
Fig. 5. Total energy consumption for different delay values where the hit rate is : (a) 55%, (b) 65%, (c) 75%, and (d) 85%.



(a) Total energy consumption of database clusters when hit rate is fixed at: 60%



(b) Total energy consumption of database clusters when hit rate is fixed at: 75%



(c) Total energy consumption of database clusters when hit rate is fixed at: 90%

Fig. 6. Total energy consumption of database clusters while the number of data disks of cold node is varied. Hit rate is fixed at: (a) 60%, (b) 75%, (c) 90%

GreenDB is superior to PRE-BUD in terms of energy efficiency.

IV. Conclusions

In this study, we proposed an energy-efficient prefetching/caching mechanism to fetch and cache popular data tables into hot nodes. We built a prefetching model in conjunction with an energy saving model, which offers guidelines to GreenDB to conserve energy. GreenDB exhibits two salients features. First, GreenDB aggressively keeps cold nodes in the low-power mode in long time periods. Second, GreenDB lowers energy-saving overhead by reducing the number of power-state transitions.

We conducted extensive experiments to quantitatively evaluate the energy efficiency and performance of GreenDB in terms of fetching and managing data tables in database clusters. We compared GreenDB's prefetching strategy with an existing one implemented in Postgresql. Experimental results show that GreenDB conserves the energy consumption of the existing scheme by up to 98.4%. The findings suggest that one may optimize the energy efficiency of GreenDB by configuring table size, hit rates, number of nodes, number of disks, and inter-arrival delays.

References

[1] P. Mahadevan, S. Banerjee, P. Sharma, A. Shah, and P. Ranganathan, "On energy efficiency for enterprise and data center networks," *IEEE Communications Magazine*, vol. 49, no. 8, 2011.

[2] A. Manzanares, X. Qin, X. Ruan, and S. Yin, "Pre-bud: Prefetching for energy-efficient parallel i/o systems with buffer disks," *ACM Transactions on Storage (TOS)*, vol. 7, no. 1, p. 3, 2011.

[3] J. Kim, M. Ruggiero, and D. Atienza, "Free cooling-aware dynamic power management for green datacenters," in *High Performance Computing and Simulation (HPCS), 2012 International Conference on*. IEEE, 2012, pp. 140–146.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:08:06 UTC from IEEE Xplore. Restrictions apply.