Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 12 (2012) 326 – 331

Procedia Computer Science

Complex Adaptive Systems, Publication 2
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2012- Washington D.C.

"An Integer Programming Power Optimization in Storage Systems"

"Muhittin Yilmaz, Pratyush Valluri, Sasikanth Pagadrai"

"Texas A&M University-Kingsville (TAMUK), Kingsville, TX, USA"

Abstract

This paper presents a linear integer programming framework for effective power management in storage systems. A sample memory system with different data banks is considered for optimal energy consumption during data operations by manipulating the data among banks. The memory bank four-level power state schemes, namely, active, stand-by, nap, and power down states, are included for superior power management of the storage system by formulating a linear integer optimization framework that includes plausible data manipulations, energy consumption levels, data migration, and compression options. The numerical results illustrate the efficiency of the proposed framework in terms of power management of storage systems with respect to available approaches with two-level power state operations.

"Keywords: Storage optimization; integer programming; power optimization"

1. Introduction

Technological developments on electronic data systems have attracted renewed interest on optimal storage system speed, capacity, or power consumption formulations. Due to popularity, large storage and limited power capabilities of mobile electronic devices, the power consumption concerns have resulted in associated optimization frameworks. Furthermore, intensive code execution and associated power consumption have also yielded power management techniques for more efficient compiler operations. Due to a large fraction of overall energy consumption in storage systems [1, 2], energy optimization techniques including integer linear programming (ILP) with realistic constraints are used for multiple banked memories [1], loop optimizations of multiple bank memories [2] and for parallelizing applications in on-chip multiprocessors for faster code execution [3]. A single memory unit is typically divided into several banks [2] for superior power management in which only the required portion of the memory unit is maintained at active state while the rest of the unused memory regions are forced to power saving modes. However, the number of different power operation levels for memory banks in storage systems indicates a potential to improve the power efficiency of the memory unit in spite of anticipated higher formulation complexity for adaptive power management schemes, in addition to available storage system improvements.

Storage system operations have benefitted from several approaches. A self-organized distributed storage system has utilized a swarm-inspired algorithm, i.e., an ant colony optimization, in order to locate stored data in a large scale storage network [4] by exploiting the swarm intelligence method strength on complex systems, its adaptability in varying environments and its robustness against failure while noticing a need for constant optimization due to its probability to fail. An optimization framework which uses the addresses and the transmission of the information as a base was proposed for network oriented storage optimization [5] for search engines and multimedia applications. The uncertainty concerns about the size of temporary data storages in a large scale

 doi:10.1016/j.procs.2012.09.079

distributed storage system during stream processing applications for both incoming and intermediate results yielded an approach using the retention value functions which are helpful in placing as well as retaining of data of highest value for the corresponding optimization schemes [6]. A security-aware cache replacement algorithm in storage systems has presented an optimization scheme to ensure highest bandwidth under various desired security levels [7].

This paper improves storage system power consumption by extending earlier research results [1] on an ILP based energy optimization for banked memories in a dynamic random access memory (DRAM). The earlier approach using two power levels for memory banks is enhanced by defining four power levels with associated new cost and constraint functions in a new ILP formulation that is a subset of linear programming while ensuring some or all of optimization variables as integers, also called as mixed integer programming in which integer and non-integer decision variables coexist. The objective and cost functions of the ILP are linear and a global optimal solution is guaranteed with low computational complexity, if a solution exists, in many different applications including superior memory operations [8-13]. The paper consists of the following; Section I introduces storage optimization perspectives, Section II presents the storage system power management principles, Section III describes the ILP power optimization framework, Section IV covers the numerical results, and Section V concludes.

2. Storage System Power Management

Multi-level storage systems exhibit different operation characteristics under different data exchange schemes and provide an opportunity to effectively manage the energy consumption by exploiting the operation characteristics. Among many different storage systems, a DRAM with multi-banks, as shown in Fig. 1 for an n-bank memory module, can be considered for typical optimal power management approaches. A single DRAM module is divided into banks of equal sizes. Different operation levels and execution sequence steps are utilized to formulate a power optimization scheme under system constraints for various operation levels. The power management concept can be extended to storage system clusters including different technology, power consumption characteristics, and speed.

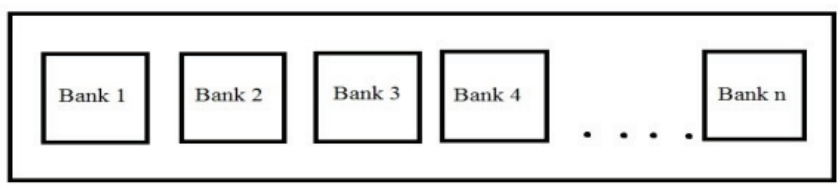

Figure 1: A memory module with its n banks.

The storage system power management framework development procedure includes the following steps:

- Gather the total number of execution steps, ‘S’ in Table 1, and the usage of different memory modules, ‘M’ in Table 1, and data inside those modules.
- Analyze the memory usage for every memory step and study the execution pattern for constraint development.
- Define system-level, user-defined, a-priori known parameters, as shown in the first column of Table 1, user-defined and a-priori known energy consumption variables among different power state transitions, as shown in the third column of Table 1, and binary (0-1) variables, as shown in Table 2, for subsequent cost and constraint functions.
- Develop the associated linear constraints by using the system-level variables for memory module operations. The constraints should closely reflect the memory module operations under different operation conditions and different changes in states of the memory banks.
- Develop memory bank power states for different operational properties. There are four states defined in this study: a) Active state implies memory block access, i.e., read or write, operations; b) Stand-by state is the first low power state with highest energy consumption among stand-by, nap, and power-down states; c) Nap state is the second low power state with medium level energy consumption; and d) Power-down state is the lowest energy consumption state. The operational overhead for moving from low power states to the active state increases, with stand-by and power-down yielding lowest and highest overhead amounts, respectively.
- Define a linear cost function in terms of power consumption.
- Solve the constrained power optimization problem by using the linear integer programming approach.

Table 1: The System-level Parameter and Energy Consumption Variable Descriptions

Variable	System Level Parameters	Variable	Energy Consumptions
N	Total number of banks in each memory module	AE	By an active and accessed memory bank
S	Total number of steps during execution	NE	For the bank which is Non accessed and active
M	Total number of memory blocks	DE	By deactivation of a memory block
$R_{m,s}$	Indicates whether the block 'm' is being accessed at 's' (0-1 variable)	ME	By migrating a block
$Size_{block}$	Size of an individual memory block	CompE	For compressing a data block
$Size_{bank}$	Size of an individual bank	DecompE	For decompressing a data block
RP	Performance overhead of activating a bank	SBE	In standby state
DP	Overhead occurred for moving a bank from power down from active	NAE	In Nap state
NP	Overhead for moving a bank from power down to nap	SAE	In moving a block from standby to active state
SP	Overhead occurred from moving a bank from power down to standby	NACE	In moving a block from nap to active
PSA	Overhead for moving a bank from standby to active	PDACE	In moving a block from power down to active
PNA	Overhead for moving a bank from nap to active	PDNAE	In moving a block from power down to nap
PNASB	Overhead for moving a bank from nap to standby	PDSBE	In moving a block from power down to standby
PSBNA	Overhead for moving a bank from standby to nap	ACSBE	In moving a block from active to standby
PACSB	Overhead for moving a bank from active to standby	ACNAE	In moving a block from active to nap
PACNA	Overhead for moving a bank from active to nap	SBNAE	In moving a block from standby to nap
O	Overhead Occurred	SBPDE	In moving a block from standby to power down
O_{max}	Maximum overhead possible	NASBE	In moving a block from nap to standby
CP	Compression Overhead	NAPDE	In moving a block from nap to power down
DEP	Decompression Overhead	PDE	By a bank in power down state
PSBPD	Overhead for moving a bank from standby to power down		
PNAPD	Overhead for moving a bank from nap to power down		

Table 2: The Storage System Power Management Binary (0-1) Variables

Variable	Description	Variable	Description
$L_{m,s,n}$	if 'm' is in 'n' at 's'	$SA_{n,s}$	If the bank 'n' is moved from standby to active at 's'
$CA_{n,s}$	If the bank 'n' is activated at 's'	$NAC_{n,s}$	If a bank 'n' is moved from nap to active at 's'
$PA_{n,s}$	If bank 'n' is previously activated in 's'	$NASB_{n,s}$	If a bank 'n' is moved from nap to active at 's'
$A_{n,s}$	If bank 'n' is active at 's'	$SBNA_{n,s}$	If a bank 'n' is moved from standby to nap at 's'
$X_{n,s}$	If the bank 'n' is activated from power down at 's'	$ACSB_{n,s}$	If a bank 'n' is moved from active to standby at 's'
$Y_{n,s}$	If the bank 'n' is sent to power down from active at 's'	$ACNA_{n,s}$	If a bank 'n' is moved from active to nap at 's'
$Z_{m,s}$	If block 'm' is migrated at 's'	$SBPD_{n,s}$	If a bank 'n' is moved from standby to power down at 's'
$V_{m,s}$	If the block 'm' is compressed at 's'	$NAPD_{n,s}$	If a bank 'n' is moved from nap to power down at 's'
$W_{m,s}$	If the block 'm' is decompressed at 's'	$PD_{n,s}$	If a bank 'n' is in power down mode at 's'
$PDNA_{n,s}$	If the bank 'n' is moved from power down to nap at 's'	$SB_{n,s}$	If a bank 'n' is in standby state at 's'
$PDSB_{n,s}$	if the bank 'n' is moved from power down to standby at 's'	$NA_{n,s}$	If a bank 'n' is in nap state at 's'

3. Linear Integer Programming Power Management Framework

The optimization framework includes a linear cost function and linear constraints. The parameters and variables in Tables 1-2 are used to derive the associated constraints under different operation power state changes and overhead values associated with data migration and compression/decompression operations. The first 25 constraint functions are given in Table 3, in which the C1-C13 constraints involve power state changes. The C1 constraint denotes the condition for a bank activated at step 's' and not active at '(s-1)', while X denotes the activation of the bank from power down mode and A denotes the active state of the bank. The C2 constraint denotes a bank going to the power down mode and which is active at '(s-1)' and not active at 's', while Y denotes the bank 'n' moving from active to power down at step 's'. The C3 constraint denotes a bank n moved to active from stand by

state at step‘s’. The C4 constraint denotes a bank moved to the active state from nap state at step‘s’. The C5 constraint denotes a bank ‘n’ moved to the standby state from nap state at step ‘s’. The C6 constraint denotes a bank ‘n’ moved from standby to nap state at step‘s’. The C7 constraint denotes a bank ‘n’ moved from power down to nap state at step ‘s’. The C8 constraint denotes a bank ‘n’ moved from power down to standby state at step ‘s’. The C9 constraint denotes a bank ‘n’ moved from power down to active state at step ‘s’. The C10 constraint denotes a bank ‘n’ moved from active to standby state at step ‘s’. The C11 constraint denotes a bank ‘n’ moved from active state to nap state at step ‘s’. The C12 constraint denotes a bank ‘n’ moved from standby to power down at step ‘s’. The C13 constraint denotes a bank ‘n’ moved from nap state to power down at step ‘s’. The C14 constraint denotes a required condition to satisfy if a particular block migrates from one bank to another and includes a block ‘m’ migrated at ‘s’ from one bank to another, where Z represents the migration of a block ‘m’ at step‘s’. The C15 constraint denotes the condition that a data ‘m’ block can only be in one bank at a given time. The C16 constraint represents the condition that the size of the blocks should not exceed the size of the bank at any given step‘s’ during the execution, i.e., a relationship between the sizes of the memory blocks and banks. The C17 constraint denotes the condition that if a bank ‘n’ is being accessed, then it is in active state, where R denotes if the memory block ‘m’ is being accessed at step ‘s’. The C18 and C19 constraints indicate the operation preferences of an activated bank remaining active from k steps if there no subsequent access to it, while k-step choices may yield different execution as well as power management patterns. The C20 and C21 constraints denote the condition that a bank ‘n’ is active if there is a current or previous activation. The C22 constraint ensures the condition for compression as variable V tells if a block ‘m’ is compressed at step ‘s’. The C23 constraint shows the condition for decompression of a data block as the W variable represents decompression of a block ‘m’ at step ‘s’. The overhead concerns for each operation during the optimization formulation are integrated in the C24 expression. The system limitations impose that the overhead is tracked at each step to avoid a violation of maximum possible overhead. The overheads associated with the storage system operation are expressed in Eqns. 1a-14a in Table 4. Then, the C24 expression contains the total overhead ‘O’, while the C25 constraint ensures that the total overhead is not larger than the maximum overhead possible.

Table 3: The Storage System Power Management Linear Integer Programming Constraints

Eqn. No	Expression	Eqn. No	Expression
C1	$X_{n,s} \geq A_{n,s} - A_{n,(s-1),} \forall_{n,s}$	C13	$NAPD_{n,s} \geq Y_{n,s} - NA_{n,(s-1),} \forall_{n,s}$
C2	$Y_{n,s} \geq A_{n,(s-1)} - A_{n,s,} \forall_{n,s}$	C14	$Z_{m,s} \geq L_{m,s,n} - L_{m,(s-1),n,} \forall_{m,n,s}$
C3	$SA_{n,s} \geq A_{n,s} - SB_{n,(s-1),} \forall_{n,s}$	C15	$\sum_{i=1}^{N} L_{m,s,i} = 1, \forall_{m,s,n}$
C4	$NAC_{n,s} \geq A_{n,s} - NA_{n,(s-1),} \forall_{n,s}$	C16	$(Size_{block}) * (\sum_{i=1}^{M} L_{i,s,n}) \leq (\text{Size}_{\text{bank}}), \forall_{m,s,n}$
C5	$NASB_{n,s} \geq SB_{n,s} - NA_{n,(s-1),} \forall_{n,s}$	C17	$A_{n,s} \geq R_{m,s} * L_{m,s,n,} \forall_{m,n,s}$
C6	$SBNA_{n,s} \geq NA_{n,s} - SB_{n,(s-1),} \forall_{n,s}$	C18	$PA_{n,s} \geq CA_{n,t,} \forall_{n,s,t}$
C7	$PDNA_{n,s} \geq NA_{n,s} - Y_{n,(s-1),} \forall_{n,s}$	C19	$s - k \leq t \leq s$
C8	$PDSB_{n,s} \geq SB_{n,s} - Y_{n,(s-1),} \forall_{n,s}$	C20	$A_{n,s} \geq CA_{n,s,} \forall_{n,s}$
C9	$X_{n,s} \geq A_{n,s} - Y_{n,(s-1),} \forall_{n,s}$	C21	$A_{n,s} \geq PA_{n,s,} \forall_{n,s}$
C10	$ACSB_{n,s} \geq SB_{n,s} - A_{n,(s-1),} \forall_{n,s}$	C22	$V_{m,s} \geq C_{m,s,n} + L_{m,(s-1),n} - 1, \forall_{m,s,n}$
C11	$ACNA_{n,s} \geq NA_{n,s} - A_{n,(s-1),} \forall_{n,s}$	C23	$W_{m,s} \geq L_{m,s,n} + C_{m,(s-1),n} - 1, \forall_{m,s,n}$
C12	$SBPD_{n,s} \geq Y_{n,s} - SB_{n,(s-1),} \forall_{n,s}$	C25	$O \leq O_{\max}$
C24	$O = \sum_{i=1}^{N} \sum_{j=1}^{S} (1a + 2a + 3a + .. + 12a) + \sum_{i=1}^{M} \sum_{j=1}^{S} (13a + 14a), \forall_{m,s,n}$		

The optimization framework cost, i.e., objective, function considers energy consumptions due to memory bank operations, data migration and compression/decompression operations. Using the 1b-17b Eqns. in Table 4, the total energy consumption for different states and state changes can be expressed as $B = \sum_{i=1}^{N}\sum_{j=1}^{S} 1b + 2b + 3b... + 17b, \forall_{n,s}$.

The energy consumption due to data migration is given as $Emig = \sum_{i=1}^{M}\sum_{j=1}^{S} (Z_{i,j} * ME), \forall_{m,s}$. The memory block data compression/decompression operations also result in energy consumption that is stated as $Ecomp = \sum_{i=1}^{M}\sum_{J\setminus j=1}^{S} ((Vi,j * CompE) + (Wi,j * DecompE)), \forall_{m,s}$. Consequently, the storage system power optimization problem can now be written as

$$\begin{array}{l} Minimize\ (B + Emig + Ecomp) \qquad (1) \\ subject\ to \\ Constraints\ C1\text{-}C25 \end{array}$$

Table 4: The Expressions for Overhead Occurrences during Different Operations on the Storage Device (For equations 1a-12a: i=n, j=s, for all n, s and for 13a and 14a: i=m, j=s, for all m, s), and for Energy Consumptions in Different Cases (i=n, j=s, for all n, s)

Eq. No	Equation	Overhead occurrence by	Eq. No	Equation	Energy consumed by
1a	$PDNA_{i,j} * NP$	moving a bank 'n' from power down to nap	1b	$CA_{,j} * AE$	an active and accessed bank
2a	$PDSB_{i,j} * SP$	moving a bank 'n' from power down to standby	2b	$(A_{i,j} - CA_{i,j}) * NE$	an active and not accessed bank
3a	$SA_{i,j} * PSA$	moving a bank 'n' from standby to active	3b	$SB_{i,j} * SBE$	the standby state by a bank
4a	$NAC_{i,j} * PNA$	moving a bank 'n' from nap to active	4b	$NA_{i,j} * NAE$	the nap state by a bank
5a	$NASB_{i,j} * PNASB$	moving a bank 'n' from nap to standby	5b	$Y_{i,j} * DE$	moving a bank from active to power down
6a	$SBNA_{i,j} * PSBNA$	moving a bank 'n' from standby to nap	6b	$SA_{i,j} * SAE$	moving a bank from standby to active
7a	$X_{i,j} * RP$	activating a bank	7b	$NAC_{i,j} * NACE$	a bank to move from nap to active
8a	$Y_{i,j} * DP$	a bank 'n' to go to power down mode from active	8b	$PDAC_{i,j} * PDACE$	a bank to move from power down to active
9a	$ACSB_{i,j} * PACSB$	a bank 'n' from moving from active to standby	9b	$PDNA_{i,j} * PDNAE$	a bank to move from power down to nap
10a	$ACNA_{i,j} * PACNA$	a bank 'n' from moving from active to nap	10b	$PDSB_{i,j} * PDSBE$	a bank to move from power down to standby
11a	$SBPD_{i,j} * PSBPD$	moving a bank from standby to power down	11b	$ACSB_{i,j} * ACSBE$	a bank to move from Active to standby
12a	$NAPD_{i,j} * PNAPD$	moving a bank from nap to power down	12b	$ACNA_{i,j} * ACNAE$	a bank to move from active to nap
13a	$V_{i,j} * CP$	compressing a memory block 'm'	13b	$SBNA_{i,j} * SBNAE$	a bank to move from standby to nap state
14a	$W_{i,j} * DEP$	decompressing a memory block 'm'	14b	$SBPD_{i,j} * SBPDE$	a bank to move from standby to power down
			15b	$NASB_{i,j} * NASBE$	a bank to move from nap to standby
			16b	$NAPD_{i,j} * NAPDE$	a bank to move from nap to power down
			17b	$PD_{i,j} * PDE$	a bank in power down mode

The 1a-14a expressions in Table 4 indicate various overhead terms during storage operations, resulting in a compact expression of C24 for the total overhead in Table 3. The 1b-17b expressions in Table 4 define energy consumptions for different memory bank operations, resulting in the total energy consumption B in the optimization cost function.

4. Numerical Simulations

The numerical simulations for two-level power states [1], i.e., active and power down, and for four-level power states, i.e., active, stand by, nap, and power down, in this study are obtained by implementing the storage system power management optimization framework in Eqn.1. The constraints and the objective functions were defined for two-level and four-level power state conditions. The binary integer programming frameworks for both two-level and proposed four-level power states were implemented in Matlab [14] by assuming a particular execution pattern, various power state changes, and a-priori known, same overhead and power consumption levels, and by verifying size and location conditions.

Based on the binary integer programming results for the same system parameters and assumptions, the proposed four-level power state framework outperformed the two-level power state framework [1] in reduced energy consumption by 12.7%, indicating an effective optimization tool for storage system power management.

5. Conclusions

An integer linear programming framework for storage system power optimization is successfully presented for effective data operations. The simulation results have indicated superior power management by defining higher number of power states for each memory bank. Although the results have been developed for a DRAM memory module with a number of banks, the same proposed framework can easily be extended to storage systems that include new and old electronic components for low- and high-demand operations with optimal power management.

The proposed power management system is currently being extended to hybrid and distributed practical storage systems with a number of different memory modules.

References

1. O. Ozturk, M. Kandemir, “Integer Linear Programming Based Energy Optimization for Banked DRAMs”, GLSVLSI-2005, April 17-19, 2005, Chicago, Illinois, USA.
2. M. Kandemir et al., “Influence of Loop Optimizations on Energy Consumption of Multi-bank Memory Systems”, Compiler Construction, April 2002.
3. I. Kadayif, M. Kandemir, U.Sezer, “An Integer Linear Programming Based Approach for Parallelizing Applications in On- Chip Multiprocessors”, DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.
4. H. Muhleisen, T. Walther, R. Tolksdorf, "Data Location Optimization for a Self-organized Distributed Storage System", 2011 World Congress on Nature and Biologically Inspired Computing (NaBIC), pp.176-182, October 19-21, 2011.
5. X. Zhang; X-N Zhao; , "Storage System Optimization in Data-Intensive Environment," 2011 International Conference on Computer and Management (CAMAN), pp.1-6, May 19-21, 2011.
6. K. Hildrum, F. Douglis, J. L. Wolf, P. S. Yu, L. Fleischer, A. Katta, “Storage Optimization for a Large Scale Distributed Stream Processing Systems,” ACM Transactions on Storage , February 2008, New York, USA.
7. M. Nijim, X. Qin, M. Yilmaz, “CaPaS: An Optimal Security-Aware Cache Replacement Algorithm for Cluster Storage Systems,” International Journal of High Performance Systems Architecture (IJHPSA), 3, 4, February 2012, pp. 216-232 (17).
8. W. Ma, G. Agrawal, "An Integer Programming Framework for Optimizing Shared Memory Use on GPUs," 2010 International Conference on High Performance Computing (HiPC), pp.1-10, December 19-22, 2010.
9. F. Li, J. Huang, A. Lippman, “A Linear Integer Programming Approach to Analyze P2P Media Streaming”, Information Sciences and Systems, March 19-21, 2008, Princeton, New Jersey, USA.
10. S. A. Yazhini, D. S. HarishRam, "High Level Synthesis of Data Flow Graphs using Integer Linear Programming for Switching Power Reduction", 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN-2011), pp.475-479, July 21-22, 2011.
11. B. Gou, “Generalized Integer Linear Programming Formulation for Optimal PMU Placement”, IEEE Transactions on Power Systems, August 2008.
12. E. Senn, D. Monnereau, A. Rossi, N. Julien, “Using Integer Linear Programming in Test-Bench Generation for Evaluating Communication Processors,” 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, August 27-29, 2009, Patras.
13. A. Allam, J. Ramanujam, G. Baumgartner, P. Sadayappan, “Memory Minimization for Tensor Contractions using Integer Linear Programming”, Parallel and Distributed Processing Symposium , April 25-29, 2006, Rhodes Island, USA.
14. MATLAB. http://www.mathworks.com/help/toolbox/optim/ug/brnox9y.html