FuSnap: Fuzzy Control of Logical Volume Snapshot Replication for Disk Arrays

Guillermo Navarro, *Member, IEEE*, Milos Manic, *Senior Member, IEEE*

Abstract - **This manuscript presents FuSnap, a fuzzy logic based controller that monitors and controls the snapshot process of a logical storage volume in a disk array. As disks do not linearly respond to the arrival rate of user accesses, FuSnap makes use of fuzzy logic as the means to achieve better control of their response time. The goal of the FuSnap controller is to reduce the response time caused by the copy-on-writes that occur during the snapping of a storage logical volume. The FuSnap controller, based on the response time of user accesses, makes the decision on whether to proceed with a copy-on-write or a redirect-on-write when a source logical volume is being copied to a snapshot logical volume. The benefits of FuSnap approach are twofold. Firstly, significant reductions in response time of user requests are obtained with the FuSnap approach over the traditional Copy-on-Write snap approach. Secondly, these reductions in response time make the point-in-time copy of data a process less disruptive for database users. FuSnap was verified with two setups using HPUX workstations, one setup with 8 and the other with 32 disks.**

Index Terms— **Disk Arrays, Embedded Systems, Fuzzy Control, Fuzzy Logic.**

I. Introduction

SNAPSHOT of logical volumes is an area of research of high interest for storage companies that aim at improving the availability of the data while at the same time providing data replication [1], [6]. Logical volume snapshot is a feature that translates into easier backup management, faster recovery, and reduced exposure to data loss [1], [2]. The snapshot feature is typically provided by storage companies like IBM (Tivoli Storage Manager), HP (Business Copy), EMC (SnapView), NetApp (SnapDrive), and Hitachi (Copy-on-Write Snapshot) [3]-[8]. Furthermore, logical volume snapshots make data mining of the data stored in storage devices easier by enabling users to take "snapshots" of the data at certain points in time.

By using the snapshot feature, users can create a point-in-time copy of a logical volume or LU (Logical Unit). From the user's standpoint, the snapshot feature creates an instant copy of the original logical volume. This gives users the means to preserve a point-in-time copy (the *snapshot*) of the data in a source logical volume. If the data in the source gets corrupted or lost, the user can go back to the snapshot and recover the data from that point in time. The original volume with the data to be replicated will be referred to as the *source volume*, or just the *source*, for short. The copy of the original volume will be referred to as the *snapshot volume*, or the *snapshot*, for short.

Improvements in the management of snapshot replication have been proposed in [9]-[13]. Performance improvement in terms of data transfer have been shown in [14] by Guangjun. Shah proposed a Logical Volume Manager 2 (LVM2) scheme that is an optimization of LVM that improved the read performance of the snapshot volume by 40% in [15]. Variations of the basic snapshot algorithm such incremental or iterative snapshots have been proposed before in [16]-[18]. Brinkman et al. proposed a scheme for snapshot in cluster environments [19].

Fuzzy control has been used with Proportional and Integral (PI) control before. Perry et al. in [20] proposed the design of a PI fuzzy logic controller for dc-dc conversion. Precup et al. in [21] presented a merge of fuzzy and iterative learning control for fuzzy control systems focused on Takagi-Sugeno PI fuzzy controllers. Luo et al. proposed in [22] a fuzzy-PI-based control strategy for static synchronous compensator used in electric power distribution systems. Sun et al. in [23] make use of fuzzy logic reasoning to optimize the gains of PI controller as part of the fuzzy logic based control for flywheel energy storage equipment. Fuzzy control can also be applied to other type of controllers such as proportional and derivative (PD) and proportional, integral and derivative (PID). In [24] Su et al. present the design of a two-input-single-output PD fuzzy controller for nonlinear systems. Wang et al. in [25] propose the design of a control scheme for static var compensators using fuzzy PID for the close loop section. Mostefai et al. in [26] presents a fuzzy observer-based control strategy for the compensation of nonlinear friction in a robot joint structure. Precup et al. in [27] propose a new fuzzy control solution employing 2 degrees of freedom PI fuzzy control for a class or servo systems.

This manuscript presents a FuSnap, a fuzzy PI control algorithm that drastically improves the response time of the user requests (reads or writes) during the snapshot process. The organization of this paper is as follows: Section II presents the copy-on-write and redirect-on-write snapshot

Manuscript received January 13, 2009. Accepted for publication June 17, 2010.

Guillermo Navarro is with Hewlett Packard, Boise, ID 83714 USA (e-mail: guillermo.navarro@ hp.com).
Milos Manic is with the University of Idaho, Idaho Falls, ID 83402 USA (e-mail: misko@ieee.org).

techniques. Section III presents a model for the snapshot and the modified process. Section IV presents the fuzzy control algorithm. Section V presents the experimental results. Section VI presents the conclusions.

II. Background of Point-in-Time Copy Technologies

The FuSnap controller improves the response time during the snapshot process by providing an intelligent way of combining two snapshot technologies: 1) Copy-on-Write (CoW) and 2) Redirect-on-Write (RoW). These two snapshot technologies will be described in two following subsections. The classification of snapshot techniques will be based mostly on the classification provided by Simitci in [1] and Xiao in [2].

A. Copy-on-Write (CoW)

Source logical volumes are divided into D_{Bv} data blocks, where B_v is the total number of data blocks composing the source volume. Right after the snapshot volume is created, the pointers to data blocks on each volume (source and snapshot), point to the source volume (these pointers to data blocks are in some papers also referred to as metadata [15]). This is illustrated in Fig. 1(a). If the user reads a block of data that has not been written to since the creation of the snapshot volume, the data will be read from the source volume. On the other hand, if the user reads a data block that has been written to since the creation of the snapshot, the data will be read from the snapshot volume. For the purposes of clarity, the first user write to a data block after the snapshot volume has been created will be referred to as the *first user write*.

If a first user write occurs to one of the data blocks in the source volume, for example D_j, then this block of data must be copied to the snapshot volume before that first user write occurs so that the original point-in-time data block D_j is preserved. Once the first user write occurs, the D_j data block in the source volume is modified so it is now referred to as the *updated* D_j' data block. This snapshot technology is called *copy-on-write* because every first user write to the source volume causes the disk array to copy the original data block from the source to the snapshot volume before proceeding with the user write. The copy of a data block to the snapshot volume before the first user write can occur adds an extra delay to that first user write, as it has to wait for the copy. The extra delay is called the copy-on-write *penalty*. When a data block from the source volume has been copied to the snapshot volume then the original data block is said to have been *snapped*.

After the copy-on-write is accomplished, the pointers to the respective data blocks must be updated (metadata must be updated). Fig 1(b) now shows the source volume with the updated D_j' block and the snapshot volume with the original D_j block. The snapshot volume data block pointers have to point to the original data blocks to maintain access to the point-in-time data. Therefore, the snapshot volume data block pointer to the original D_j block now points to the snapshot volume because that is where the original D_j block is preserved now. If the user accesses the snapshot volume, the user will be able to read the original D_j data block. If the user accesses the source volume, the user will read the newly updated D_j' data block. Fig 1(b) illustrates the space efficiency advantage of the snapshot solution. The space used on the snapshot volume is used only if there are new first writes to the source volume. Hence, subsequent writes to the same data block will not cause a copy-on-write.

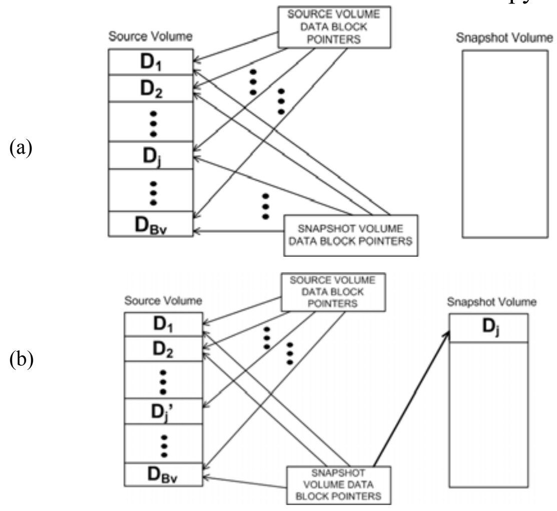

(a)
Source Volume
SOURCE VOLUME DATA BLOCK POINTERS
Snapshot Volume
D_1
D_2
D_j
D_{Bv}
SNAPSHOT VOLUME DATA BLOCK POINTERS
(b)
Source Volume
SOURCE VOLUME DATA BLOCK POINTERS
Snapshot Volume
D_j
D_1
D_2
D_j'
D_{Bv}
SNAPSHOT VOLUME DATA BLOCK POINTERS

Fig. 1. Snapshot a) right after creation and b) after copy-on-write

B. Redirect-on-Write (RoW)

In case of RoW, the new user writes to the source volume are redirected to another volume, set aside for the snapshot [2]. This redirection avoids the copy-on-write penalty since the writes proceed without the need of a copy-on-write of the original data to the snapshot volume. But in this case, the original volume still contains the original point-in-time data, while the snapshot volume contains the updated block, which is the reversal of the copy-on-write scenario.

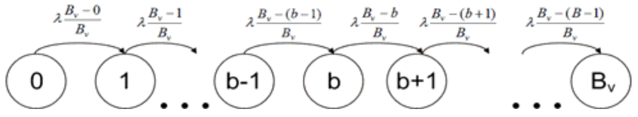
III. Modeling of the Copy-on-Write Snapshot

A. Markov Chain Model of the Probability of a Snap

The purpose of modeling the traditional copy-on-write snapshot was to understand how the probability of a snap changes as the snapshot takes place under a constant arrival rate on-line transaction processing (OLTP) workload. The probability of a snap is one of the state variables of the process to be controlled (the snapshot process). Also, the change of the probability of a snap as a function of time has implications on the stability of the FuSnap controller as it will be explained at the end of section IV. To understand how the probability of a snap changes the equations that estimate the probability are derived. The snapshot process can be modeled as a process characterized by the binomial distribution. A Markov Chain (MC) with a finite customer population [28] was the starting point. In this section, the term *snap* will be used as a synonym for copy-on-write. The snapshot process can be modeled by a MC with a finite customer population under the four considerations: 1) the number of data blocks to snap is finite, so the more data blocks are snapped, the lower the probability to snap more is; 2) the write workload applied to the source volume is random, like OLTP workloads; 3) the size (in KB) of the user writes is the same for all writes to the source volume; and 4) the writes do not cross the data blocks boundaries, that is, a write will only modify the data within one data block. These assumptions are in line with the accesses to databases, like Oracle for example [29]. The process can be understood intuitively by explaining how the snapping occurs. At the beginning, right after a snapshot volume has been created, the snapshot volume is empty, as there are no snapped data blocks yet. After the creation of the snapshot, write requests from a user come at a constant rate *λ* into the source volume. Since no data blocks have been snapped, the writes will cause a snap to occur. In more mathematical terms, the probability is one that a write will cause a snap right after the snapshot volume is created. As more data blocks are snapped, the probability of a user write causing a snap will decrease. The sum of the snapped data blocks for a volume will be denoted by *b*. Recall that B_v is the total number of blocks that make up the source volume. The probability of a write causing a snap then is:

$$P_{snap} = \frac{B_v - b}{B_v} \tag{1}$$

This formula corresponds to the intuitive expectation. If no data blocks have been snapped, then $b = 0$ and the probability of a user write causing a snap is 1. If all of the data blocks have been snapped, then $b = B_v$, and the probability of a write causing a snap is zero, which means no more snaps will occur. The MC that models those probabilities is shown in Fig. 2.



$\lambda\frac{B_v-0}{B_v}$
$\lambda\frac{B_v-1}{B_v}$
$\lambda\frac{B_v-(b-1)}{B_v}$
$\lambda\frac{B_v-b}{B_v}$
$\lambda\frac{B_v-(b+1)}{B_v}$
$\lambda\frac{B_v-(B-1)}{B_v}$
0
1
b-1
b
b+1
B_v

Fig. 2. Markov chain of copy-on-write Snapshot.

To derive the equation for the transient analysis of the MC, differential equations were obtained assuming equilibrium in terms of the input and output flow from each state [28]. The differential equation for the probability of being in the state P_0 at time *t* is:

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) \quad \text{with} \quad P_0(0) = 1 \tag{2}$$

The solution of (2) is:

$$P_0(t) = e^{-\lambda t} \tag{3}$$

The differential equation for the probability of $P_1(t)$ is:

$$\frac{dP_1(t)}{dt} = \lambda P_0(t) - \lambda\left(\frac{B_v - 1}{B_v}\right)P_1(t) \tag{4}$$

The solution of (4) is:

$$P_1(t) = B_v(1 - e^{-\frac{\lambda t}{B_v}})e^{-\frac{\lambda t}{B_v}(B_v - 1)} \tag{5}$$

The differential equation for the probability of $P_2(t)$ is:

$$\frac{dP_2(t)}{dt} = \lambda\left(\frac{B_v - 1}{B_v}\right)P_1(t) - \lambda\left(\frac{B_v - 2}{B_v}\right)P_2(t) \tag{6}$$

The solution of (6) is:

$$P_2(t) = \frac{(B_v - 1)B_v}{2}(1 - e^{-\frac{\lambda t}{B_v}})^2 e^{-\frac{\lambda t}{B_v}(B_v - 2)} \tag{7}$$

By induction, the probability of being in state *b* is:

$$P_b(t) = \frac{B_v!}{b!(B_v - b)!}(1 - e^{-\frac{\lambda t}{B_v}})^b e^{-\frac{\lambda t}{B_v}(B_v - b)} \tag{10}$$

The factorial term in equation (10) is a binomial coefficient, so the equation now becomes:

$$P_b(t) = \binom{B_v}{b}(1 - e^{-\frac{\lambda t}{B_v}})^b e^{-\frac{\lambda t}{B_v}(B_v - b)} \tag{11}$$

Equation (11) can be interpreted as the probability of having *b* blocks snapped at time *t*. This equation shows that the snapshot process for a constant write arrival rate *λ* is governed by a binomial distribution.

B. *Practical Snapshot probability equation*

Equation (11) has the form of a binomial probability mass function (p.m.f.):

$$p_n(k) = \binom{n}{k} p^k q^{(n-k)} \tag{12}$$

where the equivalent terms are:

$$n = B_v, \quad k = b, \quad q = e^{-\frac{\lambda t}{B_v}}, \quad p = 1 - q$$

The problem with (11) is that for practical uses, the number of blocks B_v that make up a volume is a large number. For example, a 64GB source volume will be made up of B_v = 64GB/128KB = 524,288 blocks. Obtaining the factorial of such big numbers can render the use of (11) impractical. That is why the authors propose the use of the equivalent terms *p* and *q* of the binomial p.m.f:

$$q = e^{-\frac{\lambda t}{B_v}} \tag{13}$$

$$p = 1 - e^{-\frac{\lambda t}{B_v}} \tag{14}$$

It is interesting to consider the behavior of (13) and (14) at *t*=0 and as *t*→∞. At *t*=0, or at the beginning of the snapshot process, the probability of causing a snap is one, as it has been established by (11). It can be observed that (13) has a value of one at *t*=0 and (14) has a value of zero. As time goes by and the user writes keep arriving at a *λ* rate into the source volume, the value of (13) goes to zero. The snapshot probability equation $p_{snap}(t, \lambda, Bv)$ is then:

$$p_{snap}(t, \lambda, B_v) = e^{-\frac{\lambda t}{B_v}} \tag{15}$$

The probability of not causing a snap would be described by

(14) and it could be now taken as the probability of not having a snapshot:

$$\overline{p_{snap}}(t,\lambda,B_v) = 1 - e^{-\frac{\lambda t}{B_v}} \qquad (16)$$

Equation (15) and (16) can be used to determine how the disk array will recover the response time and throughput that it had before the snapshot process started. These equations explain why user requests may experience high response times at the start of a snapshot when the disk array is subjected to a constant arrival OLTP workload.

C. *Model of the CoW process*

The model of the copy-on-write process is based on the response time delivered by disk drives under an OLTP workload. The two most important measures of the OLTP workload imposed on the disk array are the arrival rate in IOs per seconds (IO/s) and the response time in milliseconds [ms]. Assuming the write cache memory is in write-through mode, the response time that disk drives deliver under certain IO/s arrival rate is the key feature that will determine the response time of the user accesses (reads or writes). Fig. 3 shows the response time of a drive under an increasing arrival rate for a 15,000 revolutions per minute (RPM) disk.

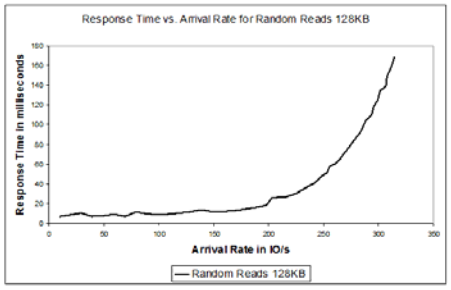


Fig. 3. Response time vs. Arrival Rate for 128KB

The response time of an access (read or write), t_{acc}, from a disk is a function of the arrival rate on the disk, λ_d:

$$t_{acc} = f(\lambda_d) \qquad (17)$$

The response time introduced by the copy-on-write process, T_{cow}, is caused by the delay of a read of the data block, T_r, from the disk where the source data block is located plus the delay of the write of that data block, T_w, to the disks where the snapshot data block will be located. This can be expressed in this equation:

$$T_{cow}(\lambda_d) = T_r(\lambda_d) + T_w(\lambda_d) \qquad (18)$$

The capital "*T*" letters indicate the response time is for large block transfers. The data blocks copied during the copy-on-write process are large in size compared to the user writes. For example, data blocks can be 128KB in size whereas user writes can be 8KB in size.

A flow of user writes is received by a disk array. Some of the user writes, according to the p_{snap} probability, will cause a snap and therefore those user writes will have to wait for the copy-on-write before being carried out (copy-on-write penalty). And some of the other writes, according to the 1-p_{snap} probability, will proceed directly to be carried out. The arrival rate of the user writes, λ_w, along with the p_{snap} probability, determines the arrival rate all disks in the disk array will receive, λ_D. Fig. 4 illustrates this process.

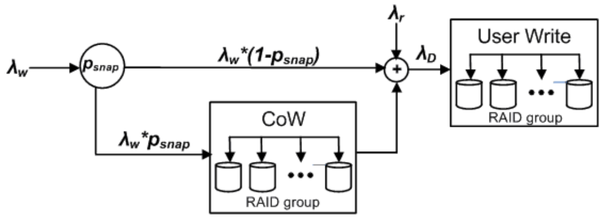


Fig. 4. User writes arrival rate and arrival rate caused by snaps

The copy-on-write process causes extra disk accesses on the disk array. If a write to a data block causes a snap that triggers a copy-on-write then a data block (for example, 128KB in size), has to be read from a disk and it has to be written on some other disks depending on the RAID level used by the snapshot volume. For example, if RAID1 is used on the snapshot volume, then a copy-on-write will generate one read of a data block from a disk and two writes to different disks. Therefore, three more accesses on disks in the disk array were generated in the background. The accesses generated by the copy-on-write that depend on the RAID level of the snapshot volume are defined by the α_{RL} factor. For RAID1 the α_{RL} =2, which is the number of writes needed for each data write.

The total extra arrival rate on the disk array generated by the copy-on-writes, λ_{cow}, is:

$$\lambda_{cow} = p_{snap} * \lambda_w * (1 + \alpha_{RL}) \qquad (19)$$

The total arrival rate on the disk array, λ_D, including user reads, is:

$$\lambda_D = \lambda_r + \alpha_{RL} * \lambda_w + \lambda_{cow} \qquad (20)$$

For the sake of simplicity, it was assumed that the arrival rate is balanced across all the disks in a disk array, N_d, and the arrival rate on each disk is

$$\lambda_d = \lambda_D / N_d \qquad (21)$$

The snapshot process occurs while users are accessing a disk array. If a user write causes a snap to occur, the user write has to wait for the snap to take place before proceeding with the user write (the copy-on-write penalty). Therefore, besides the normal response time for a user write, t_w, the response time is increased by the copy-on-write delay. In other words, the final response time the user write experiences with a copy-on-write, t_{cow}, is the sum of the two as shown in the next equation:

$$t_{cow}(\lambda_d) = t_w(\lambda_d) + T_{cow}(\lambda_d) \qquad (22)$$

The average time for the user writes is:

$$\overline{t_w}(\lambda_d) = p_{snap} * t_{cow}(\lambda_d) + (1 - p_{snap}) * t_w(\lambda_d) \qquad (23)$$

This can be more simply expressed by:

$$\overline{t_w}(\lambda_d) = t_w(\lambda_d) + p_{snap} * T_{cow}(\lambda_d) \quad (24)$$

D. Model of the proposed CoW-RoW process

This manuscript presents a snapshot process that reduces the response time during the snapping of the source volume. The presented snapshot process is a combination of the CoW and RoW processes, facilitated by the fuzzy controller.

The snapshot process is modified by introducing a control input parameter named *snap throttle factor* u_{th}. This actuating variable (control input), represents the percentage of copy-on-write that will be allowed out of the all the snaps generated by user writes. The other snaps will generate a redirect-on-write. The modified CoW-RoW process is illustrated in Fig. 5

The modified CoW-RoW process now redirects a fraction of the copy-on-writes to redirects-on-write. The reduction in the number of copy-on-writes reduces the arrival on the disks which in turn reduces their response time. The reduction in the disks' response time in turn reduces the response time experienced by user accesses. The extra arrival rate on the drives is now:

$$\lambda_{row-cow} = \lambda_w * [u_{th} * p_{snap} * (1+\alpha_{RL}) + (1-u_{th}) * \alpha_{RL}] (25)$$

And the total arrival rate on the disk array, λ_D, including user reads, is:

$$\lambda_D = \lambda_r + \alpha_{RL} * \lambda_w + \lambda_{row-cow} \quad (26)$$

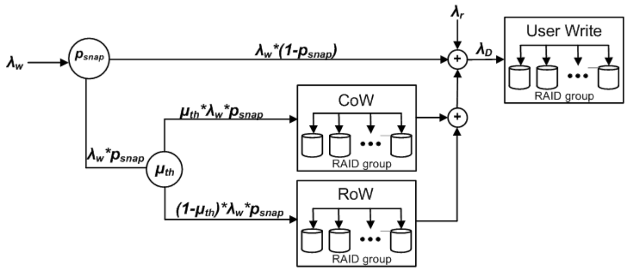


Fig. 5. Modified CoW-RoW process

The user writes now will experience smaller response time since the delay introduced by the redirect-on-writes, t_{row} is significantly lower than t_{cow}. The average response time experienced by user writes with the modified CoW-RoW process is expressed in the following equation, where to make the equation more readable the dependency on λ_d is assumed for the delays t_w, t_{cow} and t_{row}:

$$\overline{t_w} = (1-p_{snap})t_w + p_{snap}[\mu_{th}t_{cow} + (1-\mu_{th})t_{row}] \quad (27)$$

One possible simplification can be made if for practical purposes is assumed that the redirects-on-write are the same as user writes, since the user write is redirected to the snapshot volume instead of the source volume but with no other extra step in the process. This further entails that $t_{row} \approx t_w$, and (22) can be simplified as:

$$\overline{t_w}(\lambda_d) = t_w(\lambda_d) + p_{snap} * \mu_{th} * T_{cow}(\lambda_d) \quad (28)$$

This equation clearly shows why the response time is better with the CoW-RoW process if the snap throttle factor, u_{th}, is less than 1. This is one fundamental part of the process. The determination of the input control u_{th} and the control of the snapshot process with the fuzzy control are explained in the next section.

IV. Snapshot fuzzy control

A. Purpose and Rationale of FuSnap

The FuSnap controller can be considered as dynamic and optimal Takagi-Sugeno fuzzy-logic based controller. The block diagram of the FuSnap snapshot fuzzy controller is illustrated in Fig. 6. The purpose is to minimize the average response time of user accesses t_w, and t_r during a snapshot process by controlling the dynamics of the snapshot process.

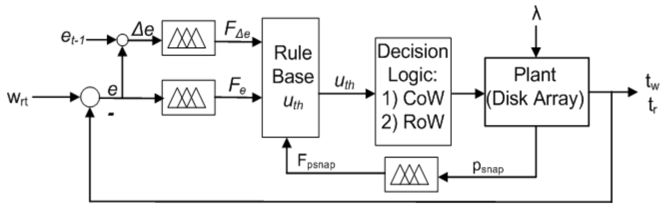


Fig. 6. Snapshot fuzzy controller.

Modeling hard disk drives has been an area of research for a long time. While some authors proposed analytical models for hard disk drives, like Shriver et. al. in [31] and Triantafillou et al. in [32], or fuzzy logic approach for disk scheduling policy by Abu et al. [33], other authors claimed that the data driven modeling needs to be used because the disk drives cannot be analytically modeled [30]. Regardless of the school of thought, the modeling and simulation difficulties arise due to the seek scheduling policies or the internal mechanical complexities. Determining an analytical model of a disk array becomes even more difficult when a customer uses a mixture of different disk drives with different mechanical features or disk drives from different manufacturers. It is for these reasons that the authors propose the use of a fuzzy control for the regulation of the snapshot process that takes into account the disk drive characteristics.

B. High level modeling of FuSnap

The controlled system has two inputs: the arrival rate of writes, λ_w, and the arrival rate of reads, λ_r. The total arrival rate, λ, is the sum of the input parameters of the controlled system (disk array):

$$\lambda = \lambda_r + \lambda_w \quad (29)$$

The outputs of the system to be controlled (disk array) are the average response times experienced by the user accesses (reads or writes), t_r, and t_w:

$$y(t_i) = [y_1 \quad y_2] = [t_w \quad t_r] \quad (30)$$

The state variables required for the FuSnap controller are 1) the probability of snapped blocks in the volume, p_{snap}, which is a value in the [0,1] range; and 2) the numbers of copy-on-writes per time unit, in other words, the arrival rate of copy-on-writes in the disk array, λ_{cow}.

$$x(t_i) = [x_1 \quad x_2] = [p_{snap} \quad \lambda_{row-cow}] \qquad (31)$$

The control input variable is the snap throttle factor, u_{th}

$$u(t_i) = [u_1] = [u_{th}] \qquad (32)$$

The FuSnap controller also requires a reference variable - the *reference response time w_{rt}*. The reference response time represents the *maximum acceptable response* time during the snapshot process. The maximum response time used in this paper was 30ms. The 30ms value comes from the Oracle performance tuning guide [28] as a response time value that gives a good indication of an overly active I/O system.

In order to control the outputs, they have to be periodically monitored every T_m seconds. The decision on how often to monitor can be based on the maximum acceptable response time and the performance of the disk array controller. The sampling of the outputs is performed at intervals of time T_m. Each sample is denoted by (t_i), where i is the *i-th* sample of the output that occurred at a time t_i, as in:

$$t_i = T_m * i \quad \text{where} \quad i = 0,1,2,\ldots \qquad (33)$$

The equation for the first state variable, p_{snap}, when the source volume is under an OLTP workload with constant arrival rate for user writes (15) for FuSnap controller becomes:

$$\frac{dp_{snap}}{dt_i} = -\frac{\lambda_w}{B_v} e^{-\frac{\lambda_w t_i}{B_v}} \qquad (34)$$

The equation for the second state variable, $\lambda_{row-cow}$, assuming also a constant arrival rate for user writes is:

$$\lambda(t_i)_{row-cow} = \alpha_{RL} * \lambda_w * u_{th}(p_{snap}) \qquad (35)$$

The equations for the outputs are based on the arrival rate the disks are being imposed. Equation (28) can be used for the first output of the controller, the user write response time:

$$t_w(t_i) = t_w(\lambda_d) + p_{snap} * \mu_{th} * T_{cow}(\lambda_d) \qquad (36)$$

For the other output, the average response time for reads, t_r, the equation (17) becomes:

$$t_r(t_i) = t_r(\lambda_d) \qquad (37)$$

Equation (37) expresses the response time for user reads based on the arrival rates on the disks.

C. Decision Logic

If a user write causes a snap, then FuSnap makes a decision about the three possible choices to execute: 1) perform a copy-on-write at the time when the user write is being served; 2) defer the copy-on-write operation by executing a redirect-on-write; 3) perform a copy-on-write of the target data block if a redirect-on-write already took place for that data block. The way the fuzzy controller throttles the snapshot process is by controlling the percentage of copy-on-writes that are caused by user writes (option 1), versus the percentage of user writes with deferred copy-on-write (option 2). This percentage is the output of the snapshot fuzzy controller and is named *snap throttle factor u_{th}*. For example, if $u_{th} = 0.4$, this means that only 40% of the user writes that cause a snap will also generate a copy-on-write. The other 60% of the user writes that are causing a snap will generate a redirect-on-write.

D. Estimation and fuzzification of the probability of a snap

The probability of a snap is used as part of the determination of the snap throttle factor. The $f_{snap}(t_i)$, in addition to being an indication of the percentage of blocks snapped at a time t_i, also denotes the probability of further snaps. For example, if 90% of the blocks in a volume have been snapped, the probability of user accesses causing further snaps is only 10% (assuming a random user access over the volume). The probability of a snap at time t_i is:

$$p_{snap}(t_i) = 1 - f_{snap}(t_i) \qquad (38)$$

The probability of a snap $p_{snap}(t_i)$, the error $e(t_i)$, and the change in error $\Delta e(t_i)$, are the three variables used by the fuzzy controller to compute the snap throttle factor, $u_{th}(t_i)$. In order to be used by FuSnap, these three variables need to be first fuzzified as shown in [34]. The fuzzification of p_{snap} is done in very straightforward fashion. If the probability of snap is below or equal to 0.5, it is mapped to the *Low Probability* (LP) fuzzy descriptor. If the probability of a snap is greater than 0.5, it is mapped to the the *High Probability* (HP) fuzzy descriptor. The membership function of probability of a snap is therefore defined by:

$$\mu_{psnap}(p_{snap}) = \begin{cases} 0 & if \;\; p_{snap} \le 0.5 \\ 1 & if \;\; p_{snap} > 0.5 \end{cases} \qquad (39)$$

The final fuzzification of the p_{snap} value is denoted by $F_{psnap}(\mu_{snap})$, and is defined as:

$$F_{psnap}(\mu_{snap}) = \begin{cases} LP & if \;\; \mu_{snap} = 0 \\ HP & if \;\; \mu_{snap} = 1 \end{cases} \qquad (40)$$

E. Control Error computation and fuzzification

The output $y(t_i)$ is compared with the reference response time w_{rt} to compute the control error, e:

$$e(t_i) = y(t_i) - w_{rt} \qquad (41)$$

The change in the control error, Δe, is also computed:

$$\Delta e(t_i) = e(t_i) - e(t_{i-1}) \qquad (42)$$

The final goal in the fuzzification of the control error e and change in the control error Δe is to map them to one of three fuzzy descriptors, *Zero* (ZE), *Positive Error* (PE), and *Negative Error* (NE), respectively. These fuzzy descriptors apply to both the control error e and change in control error Δe. The purpose of these fuzzy descriptors is obvious – they indicate when the control error is close to zero, or in case where the error does exist, whether the control error is positive or negative. This fuzzification is first performed via three triangular membership functions, μ^{ZE}, μ^{NE} and μ^{PE}, based on the reference response time w_{rt}. The membership functions are described using a dummy variable error, ε, since these membership functions are the same for both e and Δe:

$$\mu_e^{ZE}(\varepsilon, w_{rt}) = \begin{cases} 1 - \dfrac{2\varepsilon}{w_{rt}} & if\ \varepsilon > 0 \\ 1 & if\ \varepsilon = 0 \\ 1 + \dfrac{2\varepsilon}{w_{rt}} & if\ \varepsilon < 0 \end{cases} \tag{43}$$

$$\mu_e^{PE}(\varepsilon, w_{rt}) = \begin{cases} 1 & if\ \varepsilon \geq w_{rt} \\ \dfrac{4\varepsilon}{3w_{rt}} - \dfrac{1}{3} & if\ \varepsilon\ in\ (\dfrac{1}{4}w_{rt}, w_{rt}) \\ 0 & if\ \varepsilon \leq \dfrac{1}{4}w_{rt} \end{cases} \tag{44}$$

$$\mu_e^{NE}(\varepsilon, w_{rt}) = \begin{cases} 0 & if\ \varepsilon \geq -\dfrac{1}{4}w_{rt} \\ -\dfrac{4\varepsilon}{3w_{rt}} - \dfrac{1}{3} & if\ \varepsilon\ in\ (-\dfrac{1}{4}w_{rt}, -w_{rt}) \\ 1 & if\ \varepsilon \leq -w_{rt} \end{cases} \tag{45}$$

The membership functions (43), (44) and (45) here shown are for the control error e (if ε = e), and for the change in control error Δe (if ε = Δe). The graphical representation of the membership functions is shown in Fig. 7.

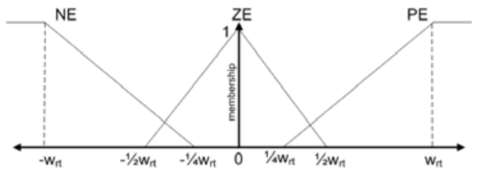


Fig. 7. Membership functions for e and Δe.

To finish the fuzzification, the control error e and the change in control error Δe are mapped into one of the fuzzy descriptors (NE, ZE, or PE). This is accomplished by comparing the values obtained for the three membership functions (43), (44), and (45). Depending on which of the three has the maximum value the fuzzy value of the error F_e, and the fuzzy value of the change in error $F_{\Delta e}$, are mapped into one of the fuzzy descriptors NE, ZE or PE:

$$F_e = \max(\mu_e^{NE}, \mu_e^{ZE}, \mu_e^{PE}) \tag{46}$$

$$F_{\Delta e} = \max(\mu_{\Delta e}^{NE}, \mu_{\Delta e}^{ZE}, \mu_{\Delta e}^{PE}) \tag{47}$$

For example, if the output $y(t_I)$ is 45ms, then using (41) the error e is 15ms. The membership values, obtained by using (43), (44) and (45), are μ^{ZE}=0, μ^{NE} =0, and μ^{PE}=1. It is clear that the maximum value corresponds to μ^{PE}. Using (40), the fuzzy value of the error F_e will be mapped to Positive Error, PE. This same procedure is used for the change in error to map it into one of the fuzzy descriptors, NE, ZE or PE.

F. Rule Base to obtain u_{th}

The rule base can now be built based on the following heuristic criteria. First criterion is: if the user response time is high, then the control error, e, is fuzzy positive error, PE, and the controller needs to reduce the number of copy-on-writes occurring. Therefore, the snap throttle factor u_{th} is reduced. Second criterion is: if the user response time is low, then the controller can increase the number of copy-on-writes occurring. Therefore, the snap throttle factor u_{th} is increased. The probability of more copy-on-writes and the change in error are also taken into account.

The next step once the three fuzzified input variables e, Δe, and p_{snap}, are estimated, is the evaluation of the fuzzy rules. The output of the fuzzy rules is the *change in snap throttle factor $\Delta u_{th}(t_i)$*. This value will denote the change in the snap throttle factor for the current iteration. The rule base is in Table 1. The rules are of the form:

$$\begin{aligned} &if\ p_{snap} \in F_{snap}\ and\ e \in F_e\ and\ \Delta e \in F_{\Delta e} \\ &then\ u_{th}(t_i) = u_{th}(t_{i-1}) + \Delta u_{th}(t_i) \end{aligned} \tag{48}$$

where $\Delta u_{th}(t_i)$ can be in the [-1,1] range. Based on the chosen rule, an equation (48) is computed for the FuSnap controller. The snap throttle factor u_{th} value is in the [0.05,1] range. The value 0.05 as the minimum for u_{th} was based on empirical observations of actual snapshot processes. This value allows some copy-on-writes to proceed and make a little progress with the snapshot. The initial values when a snapshot volume is created are $u_{th}(0)$ =0.05 and $e(0)$ = 0

.TABLE 1
RULE BASE FOR SNAPSHOT FUZZY CONTROLLER

	Rule Input Variables			Rule Output
	p_{snap}	e	Δe	Δu_{th}
R_1	HP	PE	PE	-0.2
R_2	HP	PE	NE	-0.1
R_3	HP	ZE	PE	-0.1
R_4	HP	ZE	PE	-0.1
R_5	HP	NE	ZE	+0.05
R_6	HP	NE	NE	+0.05
R_7	LP	PE	PE	-0.05
R_8	LP	PE	ZE	-0.05
R_9	LP	ZE	PE	-0.05
R_{10}	LP	NE	PE	+0.05
R_{11}	LP	NE	NE	+0.05

G. Stability of the Fuzzy Controller

The fuzzy system presented here is globally asymptotically stable based on the fact that it meets the condition for the state variables, which according to [34] shows that state variables converge to a reference vector as time goes to infinite. In the case of the FuSnap controller, it is clear that the probability of a snap, p_{snap} and therefore the $\lambda_{row\text{-}cow}$ arrival rate (25) converges to zero as user writes access more source volume data blocks as time goes by.

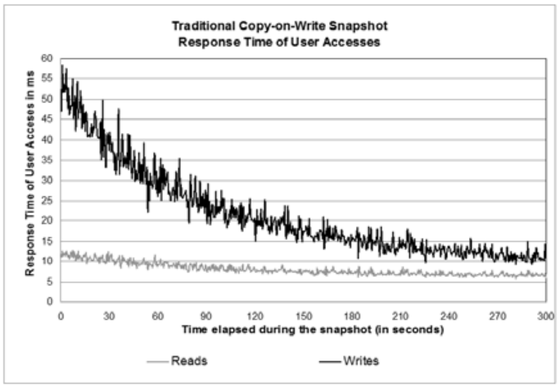
$$\lim_{t \to \infty} x(t) \to 0 \tag{49}$$

V. Experimental results

A. Results on a small setup with 8 disks

The FuSnap controller was tested with a setup that consisted of an HP 7640 Itanium workstation with 64GB of memory and with HPUX 11.23 installed. An MC534C fibre channel disk enclosure was filled with eight BF072255B2C disks. The traditional copy-on-write and FuSnap were implemented in C language and compiled with HP cc. The implementation was executed as a parent process in the user space and not as a part of the kernel. The parent process performed the following functions: 1) spawned user requests at a constant rate using the fork() Unix function; 2) kept track of the data blocks written, snapped and or with a redirect-on-write. The data block table was in shared memory so it could be updated by the spawned user requests; 3) monitored the response time of the user requests; 4) implemented the FuSnap control logic. Using this setup a comparison was run with an 8KB workload, 50% reads at 500 IO/s. The source volume was a RAID1 4GB in size using data blocks of 128KB laid out in an evenly fashion over all the 8 disks.

(a)



(b)

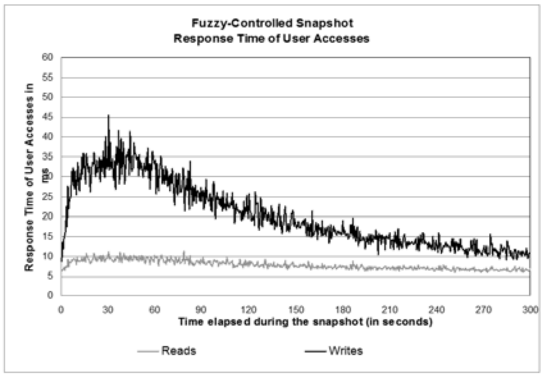


Fig. 8. Comparison of Response Time at 500 IO/s for small setup

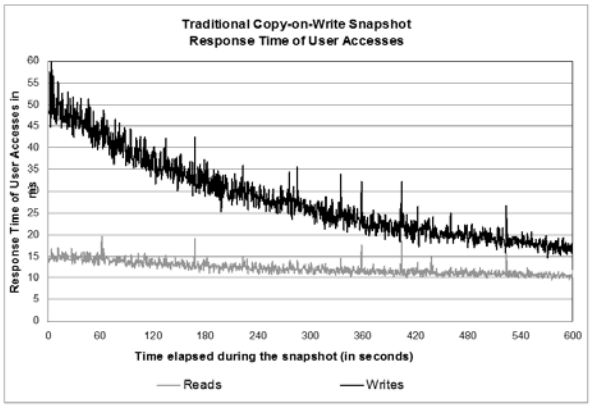
The results in Fig. 8 show the traditional copy-on-write (a) implementation delivering initial response times for user writes (black line) in the 60ms range with some tops out in the 80 to 90ms range. For user reads (gray line) the traditional copy-on-write delivered a response time in the 15 ms range. The FuSnap controller implementation shown at Fig. 8, part (b), proved superior since it could keep the initial response time for user writes (black line) in the low 40ms range. For the user reads (gray line), the response time delivered by FuSnap was in the 10ms or less range.

B. Results on a setup with 32 disks

The FuSnap controller was also tested with a setup that consisted of an HP 7640 Itanium workstation with 64GB of memory and with RH Linux 2.6.18 installed. Four M6412A fiber channel disk enclosures were filled with twelve BF146DA47C disks. The traditional copy-on-write and FuSnap were implemented in C language and compiled with gcc. The implementation details were the same as the used in the previous setup with eight disks. Using this setup a comparison was run with an 8KB workload, 50% reads at 1,000 IO/s. The source volume was a RAID1 16GB in size using data blocks of 128KB laid out in an evenly fashion over all the 32 disks.

The results in Fig. 9 show the traditional copy-on-write (a) implementation delivering initial response times for the user writes (black line) in the 50-60ms. For user reads (gray line), the traditional copy-on-write delivered an initial response time in the 15 ms range. The FuSnap controller implementation shown at Fig. 9, part (b), proved superior since it could keep the initial response time for user writes (black line) under the 40ms range. For user reads (gray line), the response time delivered by FuSnap was in the 12ms or less range.

(a)



(b)

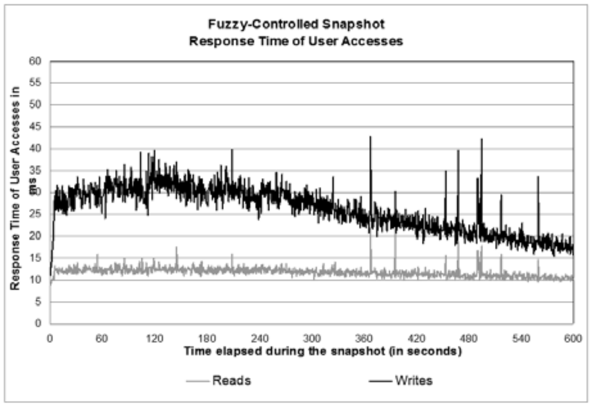


Fig. 9. Comparison of Response Time at 10,00 IO/s for 32 disk setup

VI. Conclusions

The greatest benefit FuSnap delivers is to avoid the high response time peak at the beginning of a snapshot process as predicted by the equations (15) and (16) developed for the traditional copy-on-write snapshot. These equations can provide a guide for the snapshot behavior even for different disks speeds and disk arrays if the snapshot process is the traditional copy-on-write. The improvements in response time FuSnap delivers show how computationally intelligent techniques, namely fuzzy logic, 1) can be applied to the data backup management for disk arrays; 2) can outperform traditional techniques like copy-on-write; 3) can be used to control the nonlinear response of disks. The FuSnap controller proves that it can provide two benefits: 1) help in ensuring quality-of-sevice (QoS) where a database needs constant access and 2) make the backup of data a less disruptive process for the users of a database.

REFERENCES

[1] H. Simitci, “Storage Network Performance Analysis”, *Wiley Publishing Inc.*, 2003.

[2] W. Xiao, Y. Liu, Q. Yang, J. Ren, and C Xie, “Implementation and Performance Evaluation of Two Snapshot Methods on iSCSI Target Storages,” In *Proc. NASA/IEEE Conference on Mass Storage Systems and Technologies*, May, 2006.

[3] C. Brooks, P. MacFarlane, N. Pott, M. Trcka, E. Tomaz, “IBM Tivoli Storage Management Concepts”, IBM Redbooks, June 2006.

[4] Hewlett-Packard; “HP StoreageWorks Business Copy EVA QuickSpecs”, DA-11616, ver. 20, Feb. 26, 2008.

[5] EMC Corporation; “EMC CLARiiON SnapView Snapshots and Snap Sessions Knowledgebook: A Detailed Review ”, Whitepaper, Apr. 2008.

[6] Network Appliance, Inc., “Technical Overview of NetApp SnapDrive”, Technical Report TR- 3197, April 2007.

[7] C. Bertrand, “Examining Hitachi Copy-on-Write Snapshot Software Capabilities for Hitachi Thunder 9500™ V Series Storage Systems”, Hitachi Data Systems White Paper, Aug. 2004. #

[8] B. Dufrasne; A. Indryana; C. Schoessler; B. Youngs, “IBM System Storage DS4000 Series and Storage Manager 10.30”, IBM Redbooks, March 2009.

[9] A. Azagury; M. E. Factor; J. Satran; W. Micka, “Point-in-Time Copy: Yesterday, Today and Tomorrow”, In Proc. *NASA and IEEE Mass Storage Systems (MSS)*, pp. 259-270, April 2002.

[10] Elnikety, S.; Pedone, F.; Zwaenepoel, W., “Database replication using generalized snapshot isolation”, *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems, pp. 73-84, Oct. 2005*.

[11] Shrira, L.; Xu, H., “SNAP: efficient snapshots for back-in-time execution”, *Proceedings of the 21st IEEE Conference on Digital Object Identifier*, pp. 434-445, Apr. 2005.

[12] Merchant, A.; Kun-Lung Wu; Yu, P.S.; Ming-Syan Chen, “Performance Analysis of Dynamic Finite Versioning Schemes: Storage vs. Obsolence”, *IEEE Trans. on Knowledge and Data Engineering*, 1996.

[13] A. Brinkmann; S. Effert; M. Heidebuer; M. Vodisek, “Realizing Multilevel Snapshots in Dynamically Changing Virtualized Storage Environments,” *IEEE ICN/ICONS/MCL*, 2006.

[14] Xie Guangjun; Lu Qi; Feng Wang; Gang Wang; XiaoGuang Liu.; Jing Liu, “ESnapII – A Writable Dependent Snapshot System with Shared Cache”, *Ninth ACIS Int. Conf. on Software Eng., Artficial Intelligence, Networking, and Parallel/Distributed Computin*, Aug. 2008.

[15] Bhavana Shah, ”Disk Performance of Copy-On-Write Snapshot Logical Volumes”, *master degree thesis, The University Of British Columbia*, 2006.

[16] Liu Zhenjun; Xu Lu; Feng Shuo; Yin Yang, “The Design and Implementation of an Iterative Snapshot System*”, Jisuanji Gongcheng Yu Yingyong*, vol. 42, no.14, pp. 11-15, May 2006.

[17] Xu Guangping; Wang Gang; Liu Jing, “Design of repetitious points incremental snapshots based on same snapshot volume”, *Computer engineering and applications*, January 2005.

[18] L. Zhong, W. Gang, L. Jing, “A Technology of Implementing Sequential Points Snapshot in the Storage Subsystem”, *Computer engineering and applications*, March 2004.

[19] A. Brinkmann, S. Effert, "Snapshots and Continous Data Replication in Cluster Storage Environments", SNAPI ’07, 4th International Workshop on Storage Network Architecture and Parellel I/Os, 2007.

[20] A.G. Perry, Guang Feng, Yan-Fei Liu, P.C. Sen, "A Design Method for PI-like Fuzzy Logic Controllers for DC–DC Converter", IEEE Trans. on Industrial Electronics, vol. 54, no. 5, pp. 2688-2696, Oct. 2007..

[21] R.-E. Precup, S. Preitl, J. K. Tar, M. L. Tomescu, M. Takacs, P. Korondi, P. Baranyi, "Fuzzy Control System Performance Enhancement by Iterative Learning Control", IEEE Trans. on Industrial Electronics, vol. 55, no. 9, pp. 3461-3475, Sept 2008.

[22] A. Luo, C. Tang, Z. Shuai, J. Tang, X. Y. Xu, D. Chen, "Fuzzy-PI-Based Direct-Output-Voltage Control Strategy for the STATCOM Used in Utility Distribution Syst," IEEE Trans. on Industrial Electronics, vol. 56, no. 7, pp. 2401-2411, July 2009.

[23] Xiang-Dong Sun; Kang-Hoon Koh; Byung-Gyu Yu; Matsui, M., “Fuzzy-Logic-Based *V/f* Control of an Induction Motor for a DC Grid Power-Leveling System Using Flywheel Energy Storage Equipment,” IEEE Trans. on Industrial Electronics, vol. 56, no. 8, pp. 3161-3168, Aug. 2009.

[24] J.-P. Su, T.-E. Lee, K.-W. Yu, "A Combined Hard and Soft Variable-Structure Control Scheme for a Class of Nonlinear Syst," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 9, pp. 3305-3313, Sept 2009.

[25] Juanjuan Wang, Chuang Fu, Yao Zhang, "SVC Control System Based on Instantaneous Reactive Power Theory and Fuzzy PID," IEEE Trans. on Industrial Electronics, vol. 55, no. 4, pp. 1658-1665, April 2008.

[26] L. Mostefai, M. Denai, S. Oh, and Y. Hori, "Optimal control design for robust fuzzy friction compensation in a robot joint," IEEE Trans. Ind. Electron., vol. 56, pp. 3832-3839, Oct. 2009.

[27] R.-E. Precup, S. Preitl, I. J. Rudas, M. L. Tomescu, and J. K. Tar, "Design and experiments for a class of fuzzy controlled servo systems." IEEE/ASME Trans. Mechatronics, vol. 13, pp. 22-35, Feb. 2008.

[28] Kleinrock, L., “Queueing Systems: Volume I: Theory”, *John Wiley & Sons, Inc.*, 1975.

[29] Chan, I., “Oracle Database Performance Tuning Guide 11g Release 1 (11.1)”, Part Number B28274-02, *Oracle*, July 2008.

[30] C. Ruemmler, J. Wilkes, “An Introduction to disk drive modeling”, *IEEE Computer* 27(3):17-29, March 1994.

[31] E. Shriver, A. Merchant, J. Wilkes, “An analytic behavior model for disk drives with readahead caches and request reordering”, ACM SIGMETRICS Performance Evaluation Review, Vol. 26, Issue 1, pp. 182-191, June 1998.

[32] P. Triantafillou, S. Christodoulakis, C. Georgiadis, “A Comprehensive Analytical Performance Model for Disk Devices under Random Workloads”, IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 1, Jan/Feb. 2002.

[33] M. S. Abu Talip, A. H. Abdalla, A. Asif, “Fuzzy Logic Based Algorithm for Disk Scheduling Policy”, *IEEE International Conference of Soft Computing and Pattern Recognition*, Dec. 2009

[34] K. Michels, F. Klawonn, R. Kruse, A. Nürnberger, “Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers”, *Springer-Verlag*, 2006.

Guillermo Navarro (M’01): Guillermo Navarro, IEEE Member, received his M.S. in Computer Science from the University of Houston Clear Lake in 1997. Currently, he is pursuing a PhD degree from the University of Idaho. He has worked for Hewlett Packard since 1986. He has been a Software Engineer writing embedded software for LaserJet printers and disk arrays. He is currently working for the EVA Storage Lab in Boise, Idaho as a Storage Performance Engineer. He has published several papers in IEEE conferences about disk array performability and data rebuild algorithms using fuzzy logic and neural networks. He has coauthored four patents for disk array algorithms.

Milos Manic, PhD (S'95-M'05-SM'06): Dr. Milos Manic, IEEE Senior Member, has been leading Computer Science Program at Idaho Falls and is a Director of Modern Heuristics Group. He received his Ph.D. degree in Computer Science from University of Idaho, Computer Science Dept. He received his M.S. and Dipl.Ing. in Electrical Engineering and Computer Science from the University of Nis, Faculty of Electronic Engineering, Serbia. He has over 20 years of academic and industrial experience, including an appointment at the ECE Dept. and Neuroscience program at University of Idaho Moscow. As university collaborator or principal investigator he lead number of research grants with the Idaho National Laboratory, NSF, EPSCoR, Dept. of Air Force, and Hewlett-Packard, in the area of data mining and computational intelligence applications in process control, network security and infrastructure protection. Dr. Manic has published over hundred refereed articles in international journals, books, and conferences.