Response Time Distribution of Flash Memory Accesses

Peter G. Harrison
Imperial College London, UK
pgh@doc.ic.ac.uk

Naresh M. Patel
NetApp, Inc., USA
naresh@netapp.com

Soraya Zertal
University of Versailles, FR
zertal@prism.uvsq.fr

ABSTRACT

Flash memory is becoming an increasingly important storage component among non-volatile storage devices. Its cost is decreasing dramatically, which makes it a serious competitor of disks and a candidate for being the storage device of the future. Consequently, there is an urgent need for models and tools to analyse its behaviour and evaluate its effects on a system's performance. We propose a fluid model with priority to investigate the response time characteristics of Flash memory accesses. This model can represent well the Flash access operations, respecting the erase/write/read relative priorities.

Keywords

Fluid model, Flash memory, Response time distribution, Preemptive mode.

1. INTRODUCTION

Storage devices based on Flash memory are becoming more and more prevalent in our daily life. This recent technology presents a panoply of devices, subject to continuous and intensive evolution – as commonly observed in the semiconductor manufacturing industry but especially prominent here in response to the market demand of mp3 players, mobile phones, digital cameras, solid state disks and other products.

However, the technology suffers from a lack of tools capable of analysing its behaviour and its effect on the whole system of which it forms a part. Moreover, there are no methodologies available to evaluate the performance it can deliver and the associated cost of its very specific characteristics. Such tools and methodologies must be representative of the real system, accurate on the scale of the semiconductor and quick to develop and use, in order to be in step with the wide variety of devices and their manufacturing speed. The mathematical modelling techniques we present address these requirements with respect to both their accuracy and representativeness. This requires both careful parameterisation and appropriate choice of modelling techniques to use.

Fluid models are well suited to representing the behaviour of Flash memory and its different access operations, namely erase, write and read, because they can take into account correlation between access streams as well as their relative priorities, and can handle different execution modes: preemptive or non-preemptive. We consider, as in [2], the three different sources of requests and a Flash component with an intermediate (fluid) buffer. Then, a Markov chain is used to describe the state of the access operations' sources, including their input and output rates. Although the tools supporting fluid models are not yet as well developed as those for discrete state Markov models [4], there is a wide range of applications which can be effectively modelled using them; Flash memory access is one such.

In the rest of the paper, the second section describes the Flash technology background – its physical characteristics, its access operations and its impact in the entreprise storage system domain. Section 3 is dedicated to the fluid model and how it is applied to represent our particular Flash memory system. We analyse both the no-priority and the high priority classes in terms of their respective response times, using busy period analysis. As a case study, we consider an online transaction processing (OLTP) system in section 4, which clarifies how the proposed, generic fluid model can be customised to represent a Flash storage system supporting this specific type of input. Numerical results are given in section 5. Finally, we conclude in section 6, suggesting further ideas for analysis that might be pursued and designs that may emerge in response to the predictions made efficiently by the models of this recent technology.

2. BACKGROUND ON FLASH MEMORY

Flash memory is becoming an increasingly important storage component among non-volatile storage devices because of its shock resistance, vibration tolerance, light weight and low energy consumption. It already has a wide range of applications, from its use in personal computers as a solid-state disk (SSD) [7] to critical environments such as satellite systems [14].

There are two types of Flash memory based on its manufacturing: NOR and NAND. The former has lower density and higher cost but provides fast random access and can be easily re-programmed, making it most suitable for storing codes. The second type has a large storage capacity and provides high performance for large read/write requests, making it more suitable for storing data [8]. In fact the den-

Valuetools '08 Athens, Greece

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384

sity is about 8 times more for NAND [11], at a cost that is 4 to 8 times cheaper than NOR. Further, sequential read and write accesses are faster for NAND, although NOR is significantly better for erasure times. However, erasures are typically pre-scheduled in NAND, essentially running in a garbage collector. Finally, NOR's corruption immunity is superior to NAND, partly because of bad blocks that exist from the time of manufacture.

All of these factors make NAND the technology of choice for many data centres and this is what we model here using fluid methods. Further, MLC_n (Multi Level Cell) technology multiplies the storage capacity of the Flash memory chip by having n-bit information per cell, where $n = 2$ or 4 in contemporary devices. We model essentially the basic SLC (Single Level Cell) technology with only one bit per cell, i.e. effectively MLC_1, but it is straightforward to adapt the techniques used to arbitrary n in the mapping from discrete storage requests onto volumes of fluid – see section 4.

2.1 NAND Flash memory access

A NAND Flash memory chip is composed of a fixed number of blocks, each of which is partitioned into a fixed number of pages. Every page consists of two areas: a data area for native (user) data and a spare area for data status (figure 1). A block is the erase operation's unit, whilst a page

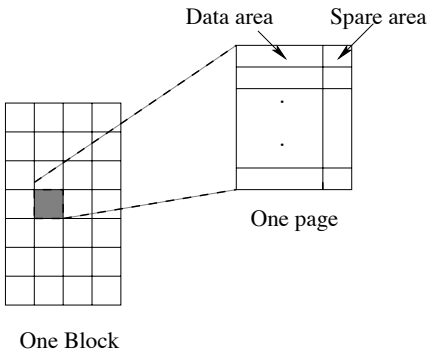

Figure 1: A NAND Flash block structure

is the read and write operation's unit. No 'in-place' updates are allowed on a NAND Flash and pages have to be written in order within a block. So, when data is modified, the new version must be written to an available page, then called the *live page*. The page containing the old version is considered a dead page and is invalidated. As time passes, the number of dead pages increases and the system reclaims them, in order to perform further write operations, by running a garbage collection process. This is possible only when these pages are erased. However, this erase/write unit mismatch generates additional copying of live pages from a block, when erasing it, to another one. Another serious constraint of NAND Flash technology is that the number of erase operations is limited to 10^5 [1] for SLC (Single Level Cell) and to 10^4 for MLC_2 (Multiple Level Cell) [13]. As any recycling of dead pages introduces block erasing, an even erase-count distribution over the Flash memory blocks cannot be achieved, which results in the "wear-levelling" problem. This has a significant negative impact on the longevity of the memory chip.

2.2 Flash management systems

Several file systems have been developed to manage data on Flash memories. JFFS (Journal Flash File System) is a log structured file system for the NOR Flash device [12]. Its second version (JFFS2) supports NAND devices with a sequential I/O interface and a more efficient garbage collection process, using a relatively reduced mounting time. YAFFS (Yet Another Flash File System) is the first file system designed specifically for NAND devices, considering data integrity as a priority [9], and its second version (YAFFS 2) accommodates a newer chip with larger pages. More recently, LogFS [10] supports snapshots and is more specific to large devices due to its reduced mounting time.

All of these Flash file systems have an FTL (Flash Translation Layer) composed essentially of two parts: an allocator process for the logical to physical space mapping and a cleaner process for the garbage collection. The mapping between the logical location and the physical location is performed using metadata on the pages' spare areas, mounted at the initialisation phase before any I/O operation. Garbage collection is performed in the background to make free space for write operations. The block size is an important design parameter which has a significant impact on the amount of metadata and so mounting time. The flash controller therefore has many design choices, which affect the output stream of storage requests it delivers to the Flash devices. Hence we consider a fairly general input stream using a Markov Modulated Poisson Process (MMPP) to model the input to a Flash memory array. Higher level models based on specific flash controller designs could then be developed by appropriately parameterising the base model we describe here.

2.3 Flash for enterprise storage

As more consumer devices incorporate Flash and new process technology is implemented, the cost of Flash is expected to decrease rapidly in the coming years. This trend may allow Flash to compete with high-speed enterprise disks in the future. Most enterprise storage systems already provide fast writes using Non-Volatile RAM but random read operations from disk incur long latencies (typically 5 to 10 milliseconds). On the other hand, sequential read operations from disk are very fast and so the randomness of the access pattern significantly impacts the observed performance. The key advantage of Flash is that the random reads and sequential reads have the same low latency per block. On the downside, Flash implementations have to deal with the complexities of the write constraints outlined above.

While 4KB page read operations from Flash have fairly low average latency (say 25 microseconds for access and 105 microseconds for channel transfer), the distribution of the latency may have a long tail because of contention with write and erase operations. Page write operations take about 105 microseconds on the channel and 200 to 700 microseconds for access (often an increasing function of erase count). Erasing a block (say 256 KB, or 64 pages) can take more than 1.5 milliseconds. These numbers vary considerably among the Flash device manufacturers, but often the latency concern stems from the possibility that small read operations have to wait behind relatively long write and erase operations. Hence we would like to model the distribution of latency for read operations (in particular) under various workloads. Fluid modelling techniques provide a promising way of improving our intuition about response time distributions for

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384

given Flash device characteristics.

3. FLUID MODEL

We use a fluid queue, with input defined by a four-phase continuous time semi-Markov chain (CTSMC), to model the behaviour of a Flash device. Such a queue is appropriate to model correlated streams of incoming data requests of different types, especially (but not exclusively) at moderate to high utilisation. In fact, in high utilisation scenarios, the erase operations and their related page copying can be generated frequently due to the high rate of native write/update operations. The four phases (or states), numbered 0 to 3, correspond to *off* (no input), *read requests*, *write requests* and *erases*. Class 1 fluid 'particles' (i.e. those arriving in phase 1) have priority over phases 2 and 3 to account for the priority actually given to reads, which are usually more critical in allowing a process to continue.

The queue is parameterised, for an n-state CTSMC so as to allow for model extensions with more than four phases, as follows:

- The CTSMC has probability transition matrix $P = (p_{ij} \mid 0 \leq i, j \leq n-1)$, defined at state transition instants;
- the state holding time in state i, given that the next state is j, has probability distribution $H_{ij}(t)$, for $0 \leq i, j \leq n-1$.
- the fluid arrival rate in phase i is the constant λ_i volume-units of fluid per unit time, for $0 \leq i \leq n-1$, so $\lambda_0 = 0$;
- the rate at which the server outputs fluid when its buffer is non-empty is the constant μ volume-units of fluid per unit time;
- the net input *rate matrix* $R = \mathrm{diag}(r_0, \ldots, r_{n-1})$, where $r_i = \lambda_i - \mu$ for $0 \leq i \leq n-1$, and the rate vector $\vec{r} = (r_0, \ldots, r_{n-1})$;
- the vector $e = (1, 1, \ldots, 1)$).

In fact we will require that all states other than the high-prority state 1 have exponential holding times with parameter that does not depend on the next state transited to; these are characterised by a *rate* γ_i for state $i = 0, 2, 3$, so that $H_{ij}(t) = 1 - e^{-\gamma_i t}$ for all states $j \neq i$. In this paper, we further assume that state 1 has exponential holding time, so we have a modulating continuous time Markov chain (CTMC), specified by a generator matrix, Q say[1].

The response time distribution (or, rather, its Laplace-Stieltjes transform (LST)) of each class in this fluid queue can be calculated using the following observations:

1. high priority fluid (arriving in phase 1) is processed as if the other classes had fluid arrival rates 0, which effectively constitute additional off-phases.
2. response time for a phase 1 particle is therefore the time taken to serve the phase-1 fluid in the buffer present at the arrival instant of that particle – that is, simply the volume of phase-1 fluid present divided by μ, given by the reduced model described in the previous point.
3. response time for particles of other lower priority phases is the sum of the time to serve all fluid already present on arrival – of volume F say – added to all additional fluid of class 1 that arrives during that time period. The amount of this additional fluid is the sum of the *busy periods* generated by each of these phase 1 arrivals, the number of which, when conditioned on the time period F/μ, is a Poisson random variable. Note that the fluid queue model for the period considered is that described in the first point since lower class fluid waits behind class 2 and so can be ignored here. However, F is the steady state fluid level in the original 4-phase fluid queue *with no priority classes*, because of the service rate being the same for all classes. The probability distribution of this fluid level is well known for any number of phases with any fluid input rates.

We therefore have to calculate:

- the steady state probability distribution of the fluid level F in the no-priority queue – a routine task;
- the busy period for a class 1 fluid arrival in the reduced model – this is determined in section 3.2 using results from [5];
- the Laplace transform of the response time distributions – this is done in section 3.4 in terms of conditional expectations on the number of high priority arrivals during the processing of the F units of fluid.

3.1 Non-priority fluid queue results

Here we restate relevant results for a single fluid queue with a Markov modulated arrival process of any number of phases. Consider a single, Markovian fluid queue, with generator matrix $Q = (q_{ij} \mid 1 \leq i, j \leq n)$ for the input process, which has equilibrium probabilities $\vec{\pi}$ (so that $\pi Q = \vec{0}$ and $\pi \vec{e}^{\mathrm{T}} = 1$).

Using the notation defined in the previous section, it can be shown that the vector density function of the fluid level at equilibrium (when this exists), denoted by $\vec{f}(x) = \frac{\partial \vec{F}}{\partial x}$ where $\vec{F}(x)$ is the corresponding probability distribution function, has Laplace transform:

$$\vec{f}^{*} = \vec{F}(0)R(R - Q/\theta)^{-1} \tag{1}$$

The constant $\vec{F}(0)$ is given by the boundary conditions that the fluid level in phases with fluid arrival rate greater than the service rate is positive with probability 1, and taking the limit $\theta \to 0$. For a Markovian on-off process with off-to-on rate a and on-to-off rate b, this gives

$$\vec{f}^{*}(\theta) = (0, F_2(0)) + \frac{\pi_1 \alpha}{\theta + \alpha}(1, (\lambda_1 - \mu)/\mu) \tag{2}$$

where $\alpha = \frac{r_1 b + r_2 a}{r_1 r_2}$ so that, noting $\vec{f}^{*}(0) = \vec{\pi}$, $F_2(0) = \frac{\alpha r_1}{a+b}$. Details can be found in [3], but note that this is not the only solution method; see for example [2].

For the high priority class's response time, we can consider the queue as if the other classes had zero arrival rate. To get the equilibrium probability density function of the volume of

[1]This assumption is not necessary, however, since it is only used to determine the equilibrium fluid level distribution, which can be derived fairly simply as a functional equation for an on-off process in which the off-period is exponential.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384

class 1 fluid in the queue, we simply solve the same problem with $\lambda_2 = \lambda_3 = 0$. Let this density function have Laplace transform denoted by $\vec{f}_1^{\,*}(\theta)$, that for the non-priority fluid level F being $\vec{f}^{*}(\theta)$, as above.

3.2 High priority class busy times

The model we solve here again has four phases, but three of these have zero fluid arrival rate. We therefore make a further simplification by aggregating the read and write phases, 2 and 3, into a new phase 2, which we assume to have exponential holding time. This is not as unreasonable as it may appear – in fact probably no less unreasonable than assuming read and write phases were exponential in the first place. Actually, the problem can be solved without this aggregation, but at greatly increased computational cost[2]. The mean holding time of the new phase 2, equal to the reciprocal of the parameter of its exponential distribution, is determined as the mean time elapsed between entering either state 2 or 3 and next entering either state 0 or 1. This is a routine, first passage time calculation, which yields a mean holding time of $h_{23} = \frac{\pi_2 h_2 + \pi_3 h_3}{\pi_2 + \pi_3}$ where

$$h_2 = \frac{q_{23} - q_{33}}{q_{22}q_{33} - q_{23}q_{32}} \quad \text{and} \quad h_3 = \frac{q_{32} - q_{22}}{q_{22}q_{33} - q_{23}q_{32}}$$

and the q_{ij} are the generators of the 4-phase process.

Consistent with [5], the set of off-states is denoted by $\mathcal{E} = \{0, 2\}$ and the set of all states by $\mathcal{S} = \{0, 1, 2\}$. The number of off-states is $n_e = |\mathcal{E}|$ and the total number of states is $n = |\mathcal{S}|$; thus in our example, $n_e = 2$ and $n = 3$.

We now define the following matrices:

- the $n \times n$ matrix $\Gamma = (\gamma_{ij})$, the generator matrix of the 3-phase Markov process, where we write $\gamma_i = -\gamma_{ii}$ for the positive total transition rate out of state $i \in \mathcal{S}$.
- the $1 \times n_e$ matrix (vector) $\mathbf{V} = (V_{2j}^*(\alpha))$, the j element of which is the required Laplace transform, with real parameter $\alpha \geq 0$, of the probability density function of the busy period – i.e. of the time elapsed between entering into the on-phase 1 when the queue is empty (hence, coming from an off-state) and the queue next becoming empty again in the off-phase j.
- the $n_e \times n$ matrix $\mathbf{U} = (u_{ij})$ by $u_{ij} = -\gamma_{ij}(1 - \delta_{ij})/r_i$ for $i \in \mathcal{E}$ and $j \in \mathcal{S}$.
- the $n_e \times n_e$ diagonal matrix $\mathbf{D} = (d_{ij})$ by $d_{ij} = [\beta - (\alpha + \gamma_i)/r_i]\delta_{ij}$ for $i, j \in \mathcal{E}$, where $\beta \in \mathbb{C}$.
- the $n \times n_e$ matrix $\mathbf{W} = (w_{ij})$ by $w_{1j} = V_{1j}^*(\alpha)$, $w_{ii} = 1$ for $i \in \mathcal{E}$ and $w_{ij} = 0$ otherwise, for $j \neq i \in \mathcal{E}$.

The $n_e \times n_e$ matrix $\mathbf{M} = \mathbf{D} - \mathbf{UW}$ has determinant, $\Delta \stackrel{\text{def}}{=} |\mathbf{M}|$ say, which is a polynomial of degree n_e in β and whose cofactors are polynomials of degree $n_e - 1$. Therefore each element in the inverse matrix $(\mathbf{D} - \mathbf{UW})^{-1}$ (assuming this exists, i.e. that the determinant $|\mathbf{M}|$ is non-zero) is a rational function of β which we may write in partial fractions as

$$\mathbf{L} = (\mathbf{D} - \mathbf{UW})^{-1} = \left(\sum_{s=1}^{n_e} \frac{a_{s;ij}(\alpha)}{\beta - b_s(\alpha)} \mid i, j \in \mathcal{E} \right)$$

[2] As we will see below, we would need to solve a cubic for a parameter β, rather than a quadratic, and then three simultaneous quadratic equations, rather than two.

assuming no degeneracies, so that $\{b_s(\alpha) \mid 1 \leq s \leq n_e\}$ are the distinct roots of the equation $\Delta = 0$ (in β) and the $a_{s;ij}(\alpha)$ are independent of β, defined by (dropping the parameter α when the meaning is clear):

$$a_{s;ij} = \left[\frac{c_{ji}}{\Delta_s} \right]_{\beta = b_s} \tag{3}$$

where $\mathbf{C} = (c_{ij} \mid i, j \in \mathcal{E})$ is the cofactor matrix of $\mathbf{M}$ and $\Delta_s = \Delta/(\beta - b_s) \equiv \prod_{1 \leq i \neq s \leq n_e} (\beta - b_i)$. Our special case of only a single on-phase [5] now gives a simplified result for the Laplace transform of the high priority class busy time distribution, $V^*(\theta)$:

THEOREM 1. *For $i \in \mathcal{F}, j \in \mathcal{E}$,*

$$V_{1j}^*(\alpha) = \sum_{k \in \mathcal{E}} \sum_{s=1}^{n_e} a_{s;kj} p_{1k} H_{1k}^*(\alpha - b_s r_1)$$

for general holding times in the high priority phase 1. In the Markovian case (when the holding time in phase 1 is exponential with parameter γ_1)

$$V_{1j}^*(\alpha) = \sum_{k \in \mathcal{E}} \sum_{s=1}^{n_e} \frac{a_{s;kj} q_{1k}}{\gamma_1 + \alpha - b_s r_1}$$

Theorem 1 has been implemented in Mathematica® and our numerical results are obtained using it, together with the analysis of response times given in the next section. To illustrate the theorem's use, we first need the matrix $\mathbf{M}$ to calculate the terms b_s and $a_{s;kj}$, for $j, k, s = 0, 2$.

The matrix $\mathbf{D} = \text{diag}(\beta + (\alpha + \gamma_0)/\mu, \beta + (\alpha + \gamma_2)/\mu)$, $u_{ij} = \gamma_{ij}/\mu$ $(i = 0, 2, j = 0, 1, 2)$ and

$$\mathbf{W} = \begin{pmatrix} 1 & 0 \\ V_{10}(\alpha) & V_{12}(\alpha) \\ 0 & 1 \end{pmatrix}$$

Hence

$$\begin{aligned} \mathbf{M} &= \mathbf{D} - \mathbf{UW} \\ &= \begin{pmatrix} \beta + \frac{\alpha + \gamma_0 - \gamma_{01} V_{10}^*(\alpha)}{\mu} & -\frac{\gamma_{02} + \gamma_{01} V_{12}^*(\alpha)}{\mu} \\ -\frac{\gamma_{20} + \gamma_{21} V_{10}^*(\alpha)}{\mu} & \beta + \frac{\alpha + \gamma_2 - \gamma_{21} V_{12}^*(\alpha)}{\mu} \end{pmatrix} \end{aligned}$$

$b_1(\alpha), b_2(\alpha)$ are the roots (assumed distinct) of the equation

$$\begin{aligned} \beta^2 + \frac{2\alpha + \gamma_0 + \gamma_2 - \gamma_{01} V_{10}^*(\alpha) - \gamma_{21} V_{12}^*(\alpha)}{\mu} \beta \\ + \frac{(\alpha + \gamma_0 - \gamma_{01} V_{10}^*(\alpha))(\alpha + \gamma_2 - \gamma_{21} V_{12}^*(\alpha))}{\mu^2} \\ - \frac{(\gamma_{02} + \gamma_{01} V_{12}^*(\alpha))(\gamma_{20} + \gamma_{21} V_{10}^*(\alpha))}{\mu^2} = 0 \end{aligned}$$

and $a_{1;ij}, a_{2;ij}$ are the i-j^{th} elements of the matrices

$$\mathbf{A}_1 = \frac{1}{b_1(\alpha) - b_2(\alpha)} \times \begin{pmatrix} b_1(\alpha) + \frac{\alpha + \gamma_2 - \gamma_{21} V_{12}^*(\alpha)}{\mu} & \frac{\gamma_{02} + \gamma_{01} V_{12}^*(\alpha)}{\mu} \\ \frac{\gamma_{20} + \gamma_{21} V_{10}^*(\alpha)}{\mu} & b_1(\alpha) + \frac{\alpha + \gamma_0 - \gamma_{01} V_{10}^*(\alpha)}{\mu} \end{pmatrix}$$

and

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384

$$\mathbf{A}_2 = -\frac{1}{b_1(\alpha)-b_2(\alpha)} \times$$
$$\begin{pmatrix} b_2(\alpha) + \frac{\alpha+\gamma_2-\gamma_{21}V_{12}^*(\alpha)}{\mu} & \frac{\gamma_{02}+\gamma_{01}V_{12}^*(\alpha)}{\mu} \\ \frac{\gamma_{20}+\gamma_{21}V_{10}^*(\alpha)}{\mu} & b_2(\alpha) + \frac{\alpha+\gamma_0-\gamma_{01}V_{10}^*(\alpha)}{\mu} \end{pmatrix}$$

Defining the vector $\mathbf{V}(\alpha) = (V_{10}^*(\alpha), V_{12}^*(\alpha))$, we finally have

$$\mathbf{V}(\alpha) = \mathbf{g}_1(\alpha) \cdot \mathbf{A}_1(\alpha) + \mathbf{g}_2(\alpha) \cdot \mathbf{A}_2(\alpha)$$

where the vectors $\mathbf{g}_s = (\gamma_{10}, \gamma_{12})/(\gamma_1 + \alpha - b_s(\alpha)r_1)$ for $s = 1, 2$.

3.3 Response time of high priority class

Let the queueing time of a high priority fluid particle – representing the start of a read – be Q_1. Then the LST of the distribution function of Q_1 is

$$Q_1^*(\theta) = \mathbb{E}[e^{-\theta Q_1}] = \mathbb{E}[e^{-\theta L_1/\mu}] = L_1^*(\theta/\mu) \qquad (4)$$

where L_1 is the volume of high priority fluid in the queue on arrival of the particle. This is simply the total volume of fluid present at equilibrium in the same queue when the arrival rates of fluid in all other phases than 1, that of the high priority class, are zero. $L_1^*(\theta)$ is therefore given by equation 1 for the four phase model, as $L_1^*(\theta) = f_1^*(\theta)/f_1^*(0)$.

3.4 Response time of low priority class

Let the queueing time of a low priority fluid particle – representing the start of a write or erase – be Q_2. Then the LST of the distribution function of Q_2 is

$$Q_2^*(\theta) = \mathbb{E}[\,\mathbb{E}[e^{-\theta(V_1+\ldots+V_N+L_2/\mu}|L_2]]$$

where L_2 is the volume of low priority fluid in the queue on arrival of the particle and N is the number of high priority (class 1) busy periods that punctuate the time period L_2/μ. The number of entries into a state of a Markov process over a given period depends on the sequence of states entered and the transition rates from those states to the state in question. For example, there may be no transitions from some states to the high priority case, so if the process spends a long time in such a state, there will be fewer entries into it. However, this is not the case we have and reads are almost uncorrelated with the phase in practice. Assuming that the transition rates from all (other) phases into the read phase, 1, are the same, i.e. that $\gamma_{01} = \gamma_{21}$ here, N is a Poisson random variable and we have:

$$\begin{aligned} Q_2^*(\theta) &= \mathbb{E}[e^{-\theta L_2/\mu}\,\mathbb{E}[\,\mathbb{E}[e^{-\theta V}]^N|L_2]] \\ &= \mathbb{E}[e^{-L_2(\theta+\gamma_{01}(1-V^*(\theta)))/\mu}] \\ &= L_2^*((\theta+\gamma_{01}(1-V^*(\theta)))/\mu) \\ &= f_2^*((\theta+\gamma_{01}(1-V^*(\theta)))/\mu)/f_2^*(0) \qquad (5) \end{aligned}$$

The expressions for $Q_1^*(\theta)$ and $Q_2^*(\theta)$ allow a direct comparison to be made between the latencies of the priority classes. It is straightforward to extract the moments of latency, by differentiation at $\theta = 0$, from which average performance and its variability can be compared. However, it is also possible to invert the Laplace transforms numerically, allowing information in the tails to be obtained and hence quantile-based benchmarks to be assessed.

4. EXAMPLE USE CASE AND MODEL PARAMETERISATION

For a concrete example, we consider an OLTP workload with read:write data ratio of 3:1. The random reads and random writes are both 8KB, which is the native database block size. However, we assume that the file system (such as Write Anywhere File Layout-WAFL [6]) converts all the random writes into sequential writes with negligible overhead. The Flash controller (and associated Flash translation layer) will take care of wear-levelling within Flash devices and WAFL will ensure that all Flash devices in the array are uniformly used. The OLTP application will generate traffic across the entire Flash array, but for our purposes here we consider a single Flash chip. Suppose this 32GB flash chip is servicing 1600 I/O's per second (so that the Flash access density of 50 I/O's per GB is an order of magnitude better than high-speed disk drives). We also assume that the chip handles at most a single command (read, write, or erase) at a time so that power requirements for the chip are minimal. In practice, Flash chips can pipeline several commands to different banks at the same time, though they will contend for the channel interface for data transfer. Given the read:write ratio of 3:1, and I/O size of 8KB, the Flash chip will see 2400 page read commands per second and 800 page write commands per second. In order to stay ahead of the write rate, the chip will have to execute 800/64=12.5 erase commands per second (assuming 64 pages per erase block).

A concept of volume, viz. *virtual bytes* ("vbytes"), helps transform real-world problems with multiple classes and fixed service rates into fluid models. Essentially, we take one of the classes and compute the bytes transferred in an operation and hence a volume service rate based on the average service time for that class. Suppose we have a large buffer for vbytes that get serviced at a fixed μ vbytes per μsec. For Flash, a 4KB read takes 105+25μsec, so that implies μ = 31.5 vbytes/μsec. The classes of interest here are as follows.

- to establish a reference point for the vbyte-measure of volume, when a 4KB read request is issued, we start a flow that puts 4096 vbytes into the buffer. Recalling that the buffer is drained at 31.5 vbytes/usec, this will take 130 μsecs to complete under no contention, as required;
- a 4KB write request takes 105+200μsec = 305μsecs. Although we are writing 4096 bytes of user data, we will need to put in 4(305/130) = 9610 virtual bytes on average to simulate the actual service time of 305μsecs;
- similarly, an erase takes 1500μsecs of actual time, so we have to feed the buffer a total of 4(1500/130) = 47,262 vbytes on average to simulate this service time.

In this way we can map between the amount of virtual bytes in the request buffer and the volume in the fluid queue model. The constant service – or *drain* – rate is simply a transfer rate in virtual bytes per μsec. During periods of no contention, a command starts a flow rate that is greater than the constant drain rate (31.5 vbytes/μsec) and the completion of the active period for that command-mode (read, write or erase) indicates its completion with the specified service time. Notice that the actual rate at which vbytes enter the buffer is arbitrary to some degree: it must exceed

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384

the drain rate and higher rates reflect increased burstiness, as discussed below.

For the modulating CTMC, we also have to estimate the mean state holding times, along with the transition probabilities between states. These parameters are necessary to construct the generator matrix and can only be obtained by monitoring workloads typical (in this use case) for Flash devices with OLTP workloads. In particular, greater burstiness can be represented by decreasing state holding times (increasing their out-transition rates) and correlation in sequences of command-mode types can be accounted for by the state transition probabilities.

5. NUMERICAL RESULTS

We used Mathematica® 5.2 to implemented the model described in section 3 and applied it to the use case of the previous section, as a preliminary illustration of its capability. The rates of fluid input in each command mode and drain rate, measured in vbytes per microsecond, were set as described in the previous section. The CTMC's instantaneous transition rates were set, somewhat arbitrarily, to:

$$\begin{pmatrix} -0.2 & 0.148 & 0.05 & 0.002 \\ 0.37 & -0.5 & 0.125 & 0.005 \\ 0.05 & 0.148 & -0.2 & 0.002 \\ 0.1 & 0.148 & 0.752 & -1.0 \end{pmatrix}$$

This choice reflects reasonable burstiness and plausible correlations in a typical OLTP sequence of database accesses. Moreover, it yields correctly the observed numbers of vbytes issued by each command type in one microsecond, *viz.* 9.83, 7.69, 0.59 for reads, writes and erases, respectively. These figures become $2400, 800, 12.5$ when scaled by the vbyte-factors $4096\times$ the single command relative access times. This is achieved by choosing instantaneous vbyte-arrival rates of $0, 43.0, 27.9, 220.5$ for each mode (rate 0 for idle mode) and using the fact that the CTMC with the above generators has equilibrium phase probabilities 0.4929, 0.2284, 0.276, 0.0027. The component-wise product of these vectors gives the observed average rates.

In this scenario, the flash unit is running at a moderate utilisation of 57% (average total arrival rate, 18.1, divided by the drain rate, 31.5) and so we can expect small fluid levels and access times at equilibrium. In fact, differentiating equation 2, we find that the mean fluid level is 23.6 and its standard deviation is 72.0 – suggesting fairly long tails. For the high priority reads, from equation 4, we obtain the queueing time probability density function shown in figure 2, with mean queueing time 0.39μsecs. For the low priority writes/erases, we get the density function shown in figure 3, with mean queueing time 1.98μsecs. This is broadly what we would expect at utilisations of around 50%.

To examine the effect of increased workload, we scaled all the arrival rates up by a factor 1.6, giving a new total average arrival rate of 28.97 vbytes/μsec and utilisation of 92%. The mean fluid level is now 853.3 and its standard deviation is 962.3. Mean queueing times grow to 24.8 and 102.1 μsecs for the high and low priority classes respectively, and the densities are shown in figures 4 and 5.

6. CONCLUSION AND FUTURE WORK

With some simplifiying assumptions, we have shown that fluid modelling techniques can provide a promising tool for the performance evaluation of NAND Flash memory systems. This approach has a considerable advantage over traditional queueing models in that it can account for both correlated input and priorities. Moreover, under heavy load, fluid models are well known to be accurate, often a stochastic limit of the analogous queue.

The preliminary numerical results are encouraging, showing the right qualitative behaviour, but clearly testing against first simulation and then experimental observations is crucial in the short term. We also intend to extend the fluid model developed here, which has a preemptive priority policy, to handle non-preemptive policies that better match current

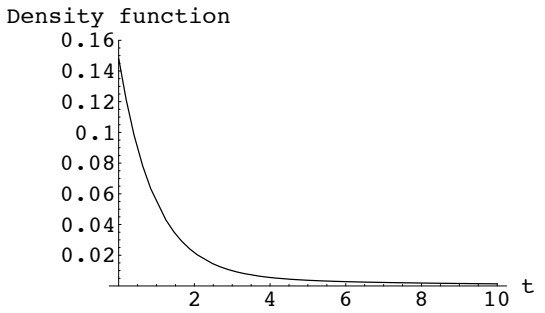

Figure 2: Queueing time density for high priority reads: moderate utilisation

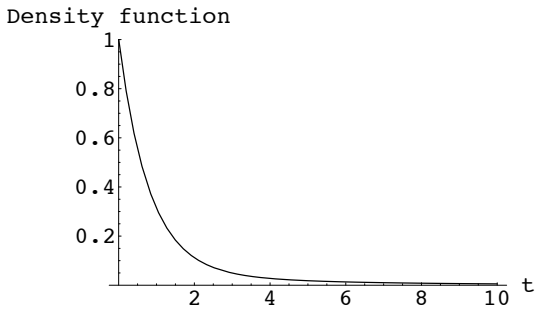

Figure 3: Queueing time density for low priority write/erases: moderate utilisation

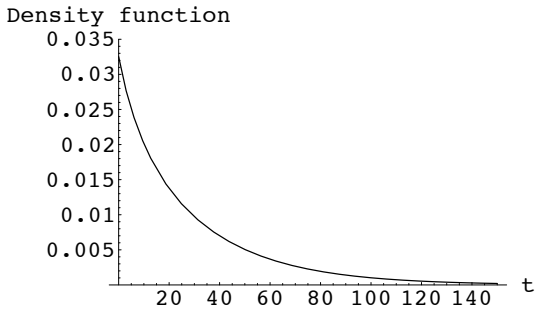

Figure 4: Queueing time density for high priority reads: high utilisation

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384

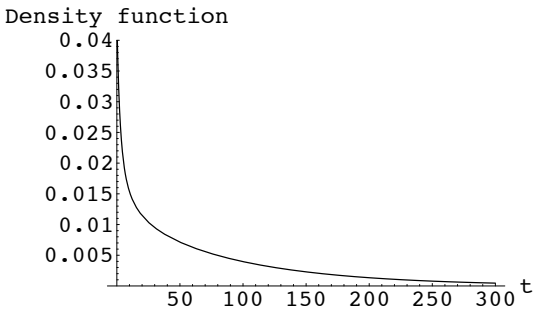

Figure 5: Queueing time density for low priority write/erases: high utilisation

Flash devices. Indeed a comparison of FCFS, premptive resume, and non-preemptive policies would show the impact of the long erase times. As Flash devices become denser, erase times are expected to increase and even if erases are done in the background, it is possible that a long sequence of writes will use up all available free space and force erases that could delay writes. This motivates the extension of our model to investgate these effects. On the hardware side, currently, Flash memory represents the lowest level in the hierarchy. The shared channel interface and flash dye are modelled as one resource, but in fact they are separate resources that are simultaneously held during command and data transfers. Another interesting extension would be to handle command queueing inside the Flash device so that more than one memory bank could be active at any one time. We also intend to consider a set of Flash devices, just like disk drives, and investigate data striping and parallel access policies, using analytical modelling and validation with respect to simulation.

7. REFERENCES

[1] Li-Pin Chang and Tei-Wei Kuo. Efficient management for large scale memory storage systems. In *Proc. ACM Symposium On Applied Computing*, 2004.

[2] D. Mitra D. Anick and M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bell Syst. Tech. Journal*, 8(61), 1982.

[3] Anthony G. Field and Peter G. Harrison. An approximate compositional approach to the analysis of fluid queue network. *Performance Evaluation*, 64, 2007. Proceedings of Performance 07.

[4] Marco Gribaudo and Miklos Telek. *Fluid Models in Performance*, volume 4486. SpringerLink, 2007.

[5] Peter G. Harrison. Busy periods in a fluid queue with multiple off- and drip-states. Technical report, Department of Computing, Imperial College London, 2008.

[6] Dave Hitz, James Lau, and Michael Malcolm. File system design for an NFS file server appliance. Technical report, Network Appliance Technical Report TR3002, 1995.

[7] Hyojun Kim and Seongjun Ahn. Bplru: A buffer management scheme for improving random writes in flash storage. In *Proc. 6th USENIX Conference on File and Storage Technologies*, 2008.

[8] Seung-Ho Lim and Kyu-Ho Park. An efficient NAND flash file system for flash memory storage. *IEEE Transactions On Computers*, 55(7), 2006.

[9] Aleph One Limited. Yaffs2 readme-linux v1.1, 2007. http://aleph1.co.uk.

[10] LogFS. Logfs home page. http://logfs.org/logfs/LogFS.

[11] Charles Manning. Introducing yaffs, the first NAND-specific flash file system, 2002. http://www.linuxdevices.com/articles.

[12] David Woodhouse. Journaling flash file system (jffs) and journaling flash filesystem 2 (jffs2). http://sources.redhat.com/jffs2/jffs2-html.

[13] Jen-Wei Hsieh Yuan-Hao Chang and Tei-Wei Kuo. Endurance enhancement of flash memory storage systems: An efficient static wear levelling design. In *Proc. DAC*, June 2007.

[14] Soraya Zertal. A reliability enhancing mechanism for a large flash embedded satellite storage system. In *Proc. IEEE International Conference on Systems (ICONS)*, 2008.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2008.4384
http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4384