ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

ISSN2321-2152www.ijmece .com
Vol 6, Issue 3Aug 2018

Studying Execution Modes and Wear Leveling in Flash Memorie

G Anjaneyulu, Syed YousufUddin, Md Ismail

Abstract:
The impact of wear levelling on a Flash storage pack- age and its access operations' execution modes is in- vestigated. First, a simple, static logical to phys- ical mapping functions are proposed and their im- plied wear levelling is assessed for different address distributions, covering both unifrom access and hot spots, as well as the Flash chip utilisation within the whole package. Second, for the access execu- tion modes, different preemptive and non-preemptive priority schemes are considered with a range of IO arrival rates using Poisson, Erlang, Pareto and Geometric-based arrival processes. The analysis of the impact of the execution modes on the perfor- mance of the Flash memory is undertaken using a hardware simulator. The results obtained show clearly the good wear levelling obtained by the map- ping functions, even in presence of hot spots. In ad- dition, the effect of the chosen execution mode on the whole storage package for each IO workload type is clearly analysed and accurately quantified.

Keywords: Flash memory, Wear levelling, Priority, Preemption, Waiting time, IO performance, Chips utilisation.

Introduction

StoragedevicesbasedonFlashmemoryarebe-comingmoreand moreprevalentinour dailylife.Thisrecenttechnologypresentsapanoplyofde-vices, continually undergoing intensive evolution inresponse to market demand for MP3 players, mobilephones,digitalcamerasusingrawFlashdevicesandforlightweightlaptopcomputers,recently evendesktopcomputersusingFlash-baseddevicesinothertermsSolidStateDrives(SSD).Infact,their use is covering both consumer and enterprisestorage products replacing Hard Drive Disks (HDD),pushing them to archiving purpose [1].Since Flashtechnology is so widely used, its performance shouldbe precisely quantified and its impact on the wholesystem, in which it is embedded, assessed relative toIO profiles and its execution mode.However, thereare currently few studies providing such information,which cannot just be deduced from the behaviouralanalysis of other storage devices such as Hard DiskDrives(HDD)andmemories(SRAM,DRAM...etc)becausetheiraccessoperationsarecompletelydifferent [2].Some studies to adapt or/and proposeaccessalgorithmsforFlash-baseddeviceswereachieved [3, 4, 5] but they still restricted to specificapplications.

The main specific features of Flash memory mak-ing it so different from the other storage devices arethe limited erase cycles and the big disparity betweentheoperations access times. The firstone associatea fixed lifetime to the Flash chip since its manufac-turing,dependingon its typeand density.In orderto maximise this Flash longevity an even erase oper-ations distribution, thus a good wear levelling shouldbeguaranteed.Inthefirstpartofthispaperandprior to giving a quantitative characterisation of thesystemunder study,weconsidernexttheeffectoftheaddress mapping chosen to provide good wear level-ling and we propose a simple static map that guar-antees data availability, and hence maximum devicelongevity. The second feature deals with the fact thatservicetimesofthethreeFlashmemoryaccessoper-ations(read/write/erase)areconstantbutpresentasignificant disparitywhateverthechiptype.Eraseoperations,beingcostlyintime,introducelongde-laysforwaitingreadorwriteoperationsperformedafterthem.

Department of civil
g.anjaneyulu143@gmail.com, yousufsyed@gmail.com, mdismail786@gmail.com
ISL Engineering College.
International Airport Road, Bandlaguda, Chandrayangutta Hyderabad - 500005 Telangana, India.

Thesedelayssignificantlychangethedeliveredperformanceandmakeworkloadschedul-ingcrucial.Ontheotherhand,applicationsrequiregoodorganisationoftheirworkloadtoconsistentlyrealisefasteraccesswithouthavingtocustomiseineveryspecificcontextforeachaccessprofile. Whilsttrueforanystoragedevicetechnology,itisparticu-larlyimportantforFlashbecauseitsaccesstimeislargelylocation-independent,especiallyforreads.Itiscomplicatedandprobablyunnecessarytoincludethiskindofworkloadaccessoptimisationintheappli-cationlayersortoaddasoftwarelayertoscheduletheworkload generated byapplications according totheFlashoperation modes.Aneasyway tomeetgoodperformancewithoutanyschedulingofrequests,istochoosethemostappropriateexecutionmodefromthemostbasicprovidedones(priority,preemption)accordingtotheinputIOprofile.Inthesecondpartofthispaper,weconsidertheeffectoftheworkloadonFlashdeliveredperformanceusingdifferentIOin-terarrivaltimedistributions.Wefocusonthethreemainexecutionmodes:withoutpriorityamong thethreeaccessoperations(mode1),givingprioritytoreadsinanon-preemptivepolicy(mode2)andfinallygiving priority toreadswithpreemption(mode3).Intherestofthepaper,section2givesasuccinctpresentationofFlashtechnologybackground.Sec-tion3isdedicatedtothewearlevellingstudy.Itdescribestheusedtools,presentstheproposed map-pingfunctionsanddetailstheirvalidation.Section4isdedicatedtotheanalysisoftheIOprofilesandtheoperationsexecutionmodesimpactontheperfor-mance.Section4.3presents theobtained numericalresultsanddiscussiontheirsignificance.Finally,sec-tion 5 summarises our main conclusions and suggestsdirectionsforfuturework.

Background

Flashmemoryisconsideredamajor,non-volatilemass-storagecomponentduetoitsshockresistance,

vibration tolerance, light weight and low energy con-sumption; not to mention its high capacity.It is al-ready used in a wide range of applications and en-vironments,fromdailyentertainment,e.g.inMP3players, through personal computers, for web serversmachines[6]andcriticalsystemssuchassatellitesys-tems[7].

There are two types of Flash memory, labelled ac-cordingtoitsconstruction:NORandNAND.Theformerhaslowerdensityandhighercostbutprovidesfast random access and can be easily re-programmed,making it most suitable for storing code. Another ad-vantage of NOR is its lower susceptibility to corrup-tionthanNAND,partlybecauseofthebadblocksthatexistinthelatterfromthetimeofmanufac-ture.NANDFlash,ontheotherhand,hasaverylarge storage capacity and provides fast data accessfor large read/write requests, making it most suitablefor storing data [8]. This is the most widespread andthe one we consider.In fact the density is about 8timesmoreforNAND[9],atacostthatis4to8times cheaper than NOR. Although erases are signif-icantly faster on NOR, they can be pre-scheduled inNAND, essentially running in a garbage collector.Inaddition,MLC_n(MultiLevelCell)technologymul-tiplies the storage capacity of the Flash memory chipbyhaving *n*-bit information percell.MLC_2becomesaclassicaldevice,presentinmostmobilecomponentssuch as cameras and smart-phones.The first $MLC_3$was developed by Hynix Semiconductor in 2008, fol-lowed by Samsung in 2009 to produce initially mi-croSD cards and to support more competitive highdensity consumer electronics storage solutions in thenearfuture.

ANANDFlashmemorychipiscomposedofafixed number of blocks, each of which is partitionedinto a fixed number of pages.Every page consists oftwo areas:a data area for native (user) data and asparareafordatastatusinformation(figure1).

Ablockistheeraseoperation'sunitofstorage,whilst a page is the read and write operation's unit.No'in-place'updatesareallowedinNANDFlash.Whendataismodified, thenewversionmustbewrittentoanavailable page–called the*livepage*.Thepagecontainingtheoldversionisconsideredadeadpageandisinvalidated.Astimepasses,ficient garbage collection process, using a relativelyreduced mounting time.Recently, comes UBIFS [17]for Unsorted Block Images File System, designed byNokia for Flash-based devices as Solid State Drives(SSD).Itprovidesfastermountingtimeandgoodwear levelling comparing to the JFFS2 random one.The second class of file systems are designed to workunder any file system.We can cite YAFFS (Yet An-other Flash File System) which is the first file systemdesignedspecificallyforNANDdevices,considering the number of dead pages increases and the systemreclaimsthem,inordertoperformfurtherwriteoperations, by running a garbage collection process.However,theerase/write unitmismatchgeneratesadditionalcopyingofremaininglivepagesfromablock,whenerasingit,toanotherone.AnotherlimitationofoftheNAND Flash technology is that thenumberoferaseoperationsislimitedtoabout10^5 [10] forSLC (Single Level Cell) and to $10^4$for MLC_2(Multiple Level Cell) [11].As any recycling of deadpages introduces block erasing, an even erase-countdistributionovertheFlashmemoryblockscannotbeachieved,whichresultsinthe"wear-levelling"problem.

Thishasasignificantnegativeimpactonthelongevityofthememorychips.Muchlikethe'smallwriteproblem'intraditionalRAID5systems[12].

Many studies fromtheliterature were dedicatedto Flash memory, especially to associated file sys-tems and more recently but less significant to pro-vide formal Flash models, as in [13, 14].In fact,several file systems have been developed to managedata on Flash memories. We can separate them intotwo classes: Native Flash file systems and non-natinefile systems which can be used with any operatingsystems.The former class is used for raw

Flashmemories, directly integrated in embedded systemsas JFFS (Journal Flash File System) which is a log-structured file system for the NOR Flash device [15].Its second version (JFFS2) [16] supports NAND de-viceswithasequentialI/Ointerfaceandamoreef-dataintegrityasapriority[18].IttakesintoaccounttheFlashcon straintsandexploititsfeaturestomax-imisetheperformanceandtherobustness.Itssec-ondversion(YAFFS2)accommodatesanewerchipwith larger pages. More recently, LogFS [19] supportssnapshotsandismorespecifictolargedevicesduetoits reducedmountingtimeanditsefficientgarbagecollectionprocess[20].Finally,wecite[21,22]forhybridarchitectureshandling both FlashandRAM.AlloftheseFlashfilesystems have an FTL(FlashTranslationLayer)composedessentiallyoftwoparts :anallocatorprocessforthelogicaltophysicalspacemappingan dacleanerprocessforthegarbagecol-lection.Themappingbetweenthelogicallocationandthephysic aloneisperformedusingmetadatainthepages'spare areas, mountedat theinitialisationphasebeforeanyI/Ooperationtakesplace.Gar bagecollectionisperformedinthebackgroundtomake freespaceforwriteoperations.

Wearlevellinganalysis

The reliability aspect is capital in storage systems. InNAND Flash based systems, either with SLC or MLCtechnology,thereare4typesofreliabilityproblems:
Wearoutblocks,
Informationretentionloss,
Writedistrurbphenomenon,
Readdistrurbphenomenon.
The first class of problems is the most importantone as the chip blocks cannot be used anymore andthecontaineddataislostfortheuser.Toavoidthis situation, a good wear levelling should be guaranteedto exploit the longevity of the chip at its maximum.There are many algorithms to implement the wearlevelling.Mainly,theycanbesplitintotwocate-gories:*static*and*dynamic*algorithms.Theformerissimple touse,itscostisnegligibleandprovidesan average wear leveling quality for all IO profiles asitiscompletelyindependentfromtheapplicationsIOrequests accessing thestored data.Thesecondclass is a bit more complex to implement as it builtsstatistics first and keeps maintaing them over time,thenadaptsthewriterequestsdistributionusingthese statistics according to the blocks use indicator.Thisiscostlyinbothcomputingtimeandstoragespace butprovidesawearlevelling"alacarte"whichisconsideredoptimal.
In this work, we propose a very simple static map-ping functions and assess their impact on the deviceuse at different levels as well as on its real longevityduration.

In this section, we present the tools used to im-plementourmappingfunctionsandstudythere-sulted wear-levelling algorithm in subsection 3.1, wedescribe our proposition in subsection 3.2 and finallywevalidateitinsubsection3.3.

Architecture'sconfigurationandtools

In this study, we consider a specific architecture butthiscaneasily beextendedto alternate Flashcon-figurations by using their description files, availablein hardware libraries. In the present case, the targetarchitecture is a package composed of 16 chips of theK9KAG08U0MNAND-Flashof2GB[23],connectedby a single 40MBps channel. This Flash storage pack-age is seen as a single address space, a logical storagepool.
Wewrotea customised event driven simulatorin Cto represent the Flash storage system and its associ-ated modes of operation. In this section, we use onlythe mapping module where the proposed static wearlevelling algorithm is implemented.For all the per-formed simulations, we considered traces of 2M reqseach.

Proposition

Wegivebelowthestaticmappingfunctionsweproposetoimple mentthewearleveling:

$chipID = ladr \% nb\_chip$

$blocID = (chipID/nb\_chip) \% chip\_size$ $pageID = ladr/(chip\_size * nb\_chip)$

*ladr*is the logical address, *nb chip* is the number ofFlash chips in the package and *chip size* is the num-ber of pages per Flash chip. The % denotes the mod-ulooperationand(*chipID, blocID, pageID*)denotethe physical address of any page (address unit) in thepackage. We observed the wear levelling achieved atthree different levels: the chip, the block and the pagelevels and we considered three parameters: the num-berofaccessestoeveryFlashchipamongthe16in total, the number of erases of every block among the16 $\times$ 8 ones and the number of writes for every pageamongthe16 $\times 8 \times 64$ones.

Staticwearlevelingqualityvalida-tion

The validation of the efficiency of our mapping func-tions in providing a good wear levelling is performedtrough3phases:
Validation of a uniform utilisation of Flash chipswithinthepackage.
Figure 2 shows the mean chip utilisation (C_{use})and the package utilisation (P_{use}) as percentagesof time, against the arrival rate.Both are inde-pedentoftheexecution mode.Thisisrelatedto the constant service time, whether the serviceis delayed or not, interrupted or not.They arelinear, increasing with the arrival rate.The C_{use}represents the mean chip utilisation but can beconsidered as the chip utilisation because all thechips are almost equally used, as shown in fig-ure3,duetothegoodwearlevelling.
Validation of a good wear levelling at three hier-archicallevels:chip,blockandpage.
Figures4,5and 6represent showthegoodwear
focusing on the emulation of the hardware functionsand the second generates suitable IO traces for theconductedtests.
Weconsiderthearchitectureconfigurationde-scribed in 3.1, where data is manipulated using threecommands – read, write and erase – one at a time.The service times of these commands were estimatedas constants by measurement on

real chips as 130μs,305μs,1.5ms respectively,includingthebus trans-fer time.Therefore, our performance study focuseson the waiting time, unique parameter affecting theresponsetime.

We use our event driven simulator,representingthe Flash storage system and its associated modes ofoperation.Its execution module processes events asthey occur – for example the arrival of a request of agiven type or the completion of an access – by main-tainingastandard eventdiary.Inaddition totheFIFO execution mode, both priority and preemptionpoliciescanbeaccommodated.Thesimulatoren-surescorrectimplementationoftherelativeprioritiesbetween two classes: a high priority class composedof reads only and a low priority class composed ofwrites and erases, which have equal priority and areprocessed in order of arrival.In the case of the pre-emptiveprioritymode,awriteoreraseisinterruptedas soon as a read enters the system. The interruptedoperation being resumed as soon as there are againno reads outstanding, but subject to further interrup-tion. In non-preemptive mode, a write or erase, oncestarted,isallowed tofinish,any newread arrivalsbeingqueued.

For the investigation of the behaviour of a Flashstoragepackageservingvarious IOprofiles,devel-oped IO generator handles different probability dis-tributionsforboth logical adresses and IO interar-rivaltimes[24].

Experimentations

Simulations were run using different synthetic IO workloads generated using our generator. The re-quests' type is consistent with standardised OLTP criteria, e.g. a fixed read:write ratio of 3:1 with ar-rival rates ranging from 50req/s to 5000req/s, and each address trace had a total of 500,000 requests.Various interarrival times were considered using dif- ferent distributions: Poisson for the 'typical user' case2, Erlang for the multi-source case and Pareto for the heavy tailed case, in an attempt to be rep- resentative of read environments. The Erlang traces are chosen to see the effect of reduced variance in the interarrival times. In fact, every exponential stage in the Erlang-n random variable is n times as fast as the Poisson process. This gives a variance of 1/(nλ), lower than the variance of the interarrival time in the corresponding Poisson process with rate λ, by a fac- tor of n. Conversely, the Pareto traces were chosen to examine the effect of increased variance, again keep- ing the mean interarrival time the same at 1/λ. We used a Pareto random variable with range [1, [and probability distribution function F (x) = 1 x−α. Direct integration shows that this has nth moment α/(α n), which exists if and only if α > n; thus, 1/2 < λ < 1 for the variance to exist. To achieve this, we first scale a range of actual arrival rates, so as to satisfy the inequality, then generate the Pareto- based input traces, and then scale these back again by multiplying by the same scaling factor.

Resultsanddiscussion

The focus is placed on the queueing time for each ofthe threeoperation classes (read,write and erase)becauseitsvariabilityhasasignificanteffect onoverallFlashperformanceduetoitsfixedservic etime.

Queueing time is considered the main performancemetric for every type of operation because of the con-stantservice(oraccess)timeforallthreeaccesstyp es(read/write/erase).Thusqueueingtimeisapur eperformance metric, being entirely an overhead thatdependsonlyontheexecutionmode.Weinves ti-gated the three main modes:no priority among theoperations(mode1);non-preemptivepriorityto readoperations (mode 2); and finally preemptive read pri-ority(mode3).

Conclusion

In this paper, we have proposed simple static map- ping functions that ensure good wear levelling for uni- formly distributed accesses to the storage space, as well as for accesses with hot spots, even in quite ex- treme case with 1% of the storage space accessed 100 times more frequently than the other 99%. Such a static scheme avoids the complex implementation and the frequent statistics extraction and management routines called under dynamic mappings. Consider- ing the queueing time, we confirmed using the Pois- son distribution which provides a standard against which to assess other workload types, that (of course)
it increases as the arrival rate increases, more rapidly as the system approaches instability. We observed similar qualitative behaviour for the Erlang case but the queueing times are smaller, due to the lower vari- ance in the interarrival times, while the queueing times for the Pareto distribution, where the variance is larger, increase at certain arrival rates. We showed that the chips within the package are equally under- used even when the arrival rate is high, and similarly for the corresponding queueing time. This suggests that, rather than using the package as a sole unit, it is better to exploit the parallelism available among the Flash chips for improved usage of the hardware com- ponents and a reduced queueing time. This should
be achieved in the short term, taking into account the concurrent access management to chips, the re- quests' scheduling policies and the shared bus alloca- tion. Further, it should be extended to the Flash bi- dimensional vector configuration. In the longer term, we plan to extend our fluid model for the Flash pack- age storage system [14] to handle Flash chips operat- ing in parallel.

References

J. Gray, “Tape is dead,disk is tape, flash is disk, ram lo- cality is king,” 2007, pres. at the CIDR Gong Show, Asilomar,CA,USA.gineering Complex Computer Systems, 2007.

P. G. Harrison, N. M. Patel, and S. Zertal, “Response time distribution of flash memory accesses,” in Valuetools, 2008.

D. Woodhouse, “Jffs : The journalling flash file system,” in Ottawa Linux Symposium, 2001.

——, “Journaling flash file system (jffs) and journaling flash filesystem 2 (jffs2),” website, http://sources.redhat.com/jffs2/jffs2.

A. Schierl, G. Schellhorn, D. Haneberg, and W. Reif, Abstract Specification of the UBIFS File System for Flash Memory. Springer Berlin/Heidelb erg, 2009, vol. 5850/2009, pp. 190– 206.

A. O. Limited, “Yaffs2 readme-linux v1.1,” website, 2007, http://aleph1.co.uk.

LogFS, “Logfs home page,” website, http://logfs.org/logfs/LogFS.

J. Engel, D. Botle, and R. Mertens, “Garbage collection in logfs,” in Linyx conf., 2007.
Y. Park, S.-H. Lim, C. Lee, and K. H. Park, “Pffs: A scalable flash file system for the hybrid architecture of phase-change ram and nand flash,” in 2008 ACM Symposium onApplied Computing, 2008, pp. 1498–14 503.

Y. Park, S. Lim, C. Lee, and K. Park, “Pffs: a scalable flash memory file system for the hybrid architecture of phase-change ram and nand flash,” in ACM Symposium on Applied Computing, 2008.

Samsung, “K9kag08u0m nand-flash,” website, http://www.samsung.com/global/system/business/ semiconductor/product/.

R. Jain, The Art of Computer Systems Performance Analysis:Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley Interscience, 1991.