DOI:10.11918/201906111

星载固态存储系统自适应闪存转换层设计

张伟东[1,2],董振兴[1],朱 岩[1],安军社[1]

(1. 中国科学院复杂航天系统电子信息技术重点实验室(中国科学院国家空间科学中心),
北京 100190; 2. 中国科学院大学,北京 100190)

摘 要: 传统星载存储系统闪存转换层(Flash Translation Layer, FTL)算法采用页级FTL映射方案和固定分区的文件管理策略,存在主机占用率高、系统响应时间长以及没有充分考虑FLASH磨损均衡等问题.为此,对传统星载存储系统方案和星载固态存储系统的工作原理深入分析研究,结合实际型号任务需求,提出了一种数据驱动的自适应超级块闪存转换层算法(Data-driven Adaptive Superblock FTL, DASFTL).DASFTL算法采用自适应超级块的分级地址映射方案,其中超级块映射表(Superblock Mapping Table, SMT)作为一级映射,页地址映射表(Page Mapping Table, PMT)作为二级映射,以提高系统的响应速度;将超级块作为FLASH地址管理的最小单元,以减少存储系统对主机的依赖;引入动态块回收权重作为超级块分组和目标回收块选择的标准,以均衡FLASH芯片内各物理块的磨损程度,延长其使用寿命.搭建硬件测试平台对DASFTL算法进行验证,实验结果表明,提出的数据驱动的自适应超级块闪存转换层算法相比于传统星载FTL算法在主机占用率和系统响应速度上分别有51.7%、46.1%的提升.长时间工作下FLASH芯片内部各物理块的擦除次数更加均衡,有效避免部分物理块过早磨损,提升FLASH芯片使用寿命.

关键词: 星载固态存储器;NAND FLASH;闪存转换层;存储系统
中图分类号: TP333 **文献标志码:** A **文章编号:** 0367-6234(2020)05-0075-07

Adaptive flash translation layer design for spaceborne solid state storage systems

ZHANG Weidong[1,2], DONG Zhenxing[1], ZHU Yan[1], AN Junshe[1]

(1. Key Laboratory of Electronics and Information Technology for Space Systems
(National Space Sciences Center, Chinese Academy of Sciences), Beijing 100190, China;
2. University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The traditional flash translation layer (FTL) algorithm of spaceborne storage system adopts page-level FTL mapping scheme and fixed partition file management strategy, which has many problems, such as high host occupancy, long system response time, and no consideration of FLASH wear balance. Therefore, a data-driven Adaptive Superblock FTL (DASFTL) algorithm was proposed based on the analysis of the working principle of traditional spaceborne storage system and spaceborne solid state storage system. DASFTL adopts the hierarchical address mapping scheme of adaptive superblock, in which the super block mapping table is the first level mapping and the page mapping table is the second level mapping for improving response time. The superblock was taken as the smallest unit of FLASH address management to reduce the dependence of storage system on host. Dynamic block recovery weight was introduced as the criterion of superblock grouping and target recovery block selection to balance the wear degree of the physical blocks in FLASH chip and prolong its service life. In the end, a hardware test platform was built to verify the DASFTL algorithm. Results show that compared with the traditional spaceborne FTL algorithm, the host occupancy rate and the system response time of the DASFTL algorithm were increased by 51.7% and 46.1%, respectively. The balanced wear performance of the FLASH chip was significantly improved, which can prolong the service life of the FLASH chip.

Keywords: spaceborne solid state storage; NAND FLASH; flash translation layer; storage system

随着国内航天事业的高速发展,卫星搭载的有效载荷种类和数量日益增多,对星上数据存储管理提出了更高要求.星载固态存储系统作为卫星的数据中心,承载着有效载荷数据的存储与发送[1].NAND FLASH因其存储密度高、非易失等特点已成为当前星上数据的主要存储介质.

固态存储系统一般由主机、闪存转换层及

收稿日期: 2019-06-17
基金项目: 中国科学院空间科学先导卫星专项(XDA15320100)
作者简介: 张伟东(1994—),男,硕士研究生;
朱 岩(1973—),男,研究员,博士生导师
通信作者: 朱 岩,zhuyan@nssc.ac.cn

FLASH 芯片组成. 闪存转换层是位于主机和 FLASH 芯片间的中间软件层,目前针对 FTL 的研究大多集中在商用 FLASH 存储系统中,对于航天环境下固态存储算法的研究成果较少. 基于商用 FLASH 系统设计的 FLT 算法管理复杂、处理器性能要求高、接口速率存在瓶颈,不适用于航天任务[2];传统星载 FTL 算法[3-5]大多采用页级地址映射方式和固定分区的文件管理模式,其优点是实现简单、可靠性高,但存在主机占用率高、系统响应时间长以及没有充分考虑 FLASH 芯片的磨损均衡等问题. 因此,基于卫星型号任务实际应用,提出一种适用于星载固态存储系统数据驱动的自适应超级块 FTL(Data-driven adaptive superblock FTL ,DASFTL)算法,以解决传统星载存储系统存在的主机占用率高、系统响应时间慢以及 FLASH 芯片磨损不均衡等问题.

1 星载固态存储系统概述

星载固态存储系统采用 NAND FLASH 作为主存储器介质,现场可编程逻辑阵列(Field Programmable Gate Array, FPGA)作为 FLASH 控制器,并配备中央处理器(Central Processing Unit, CPU)单元负责地址管理和系统维护. 一种典型星载固态存储系统结构见图 1.

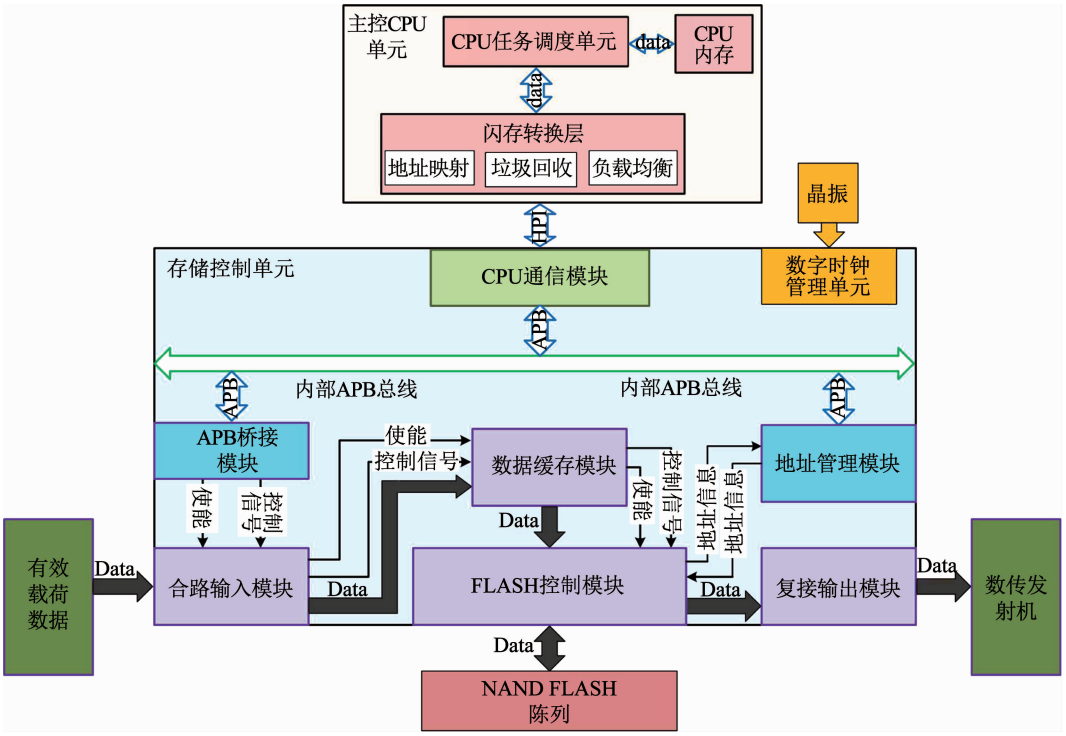


图 1 典型星载固态存储系统结构框图

Fig. 1 Structural diagram of typical spaceborne solid state storage system

卫星有效载荷产生的数据首先进入合路输入模块,数据按格式打包后送入存储控制单元,同时存储控制单元向主控 CPU 单元请求写数据页地址. FLASH 控制模块收到源包数据后,进行 RS 编码(Reed-solomon codes),并送入随机存取存储器(Random Access Memory, RAM)中进行缓存. 当 RAM 缓存的数据达到 FLASH 一页大小时,将缓存数据写入主控 CPU 单元预分配的页地址. 回放数据时,主控 CPU 单元向存储控制单元发送回放块地址和回放指令,存储控制单元在收到回放指令和地址后,将数据从存储区读出并解码,最后将数据加载至复接输出单元[6].

闪存转换层作为中间软件层位于主控 CPU 单元中,完成地址映射、垃圾回收和负载均衡等功能[7]. 地址映射将主控 CPU 单元中应用的逻辑地址转换为 FLASH 芯片中的物理地址. 垃圾回收为当收到数据写入请求时,FTL 寻找一个空页用于写入数据,若不存在空页或空余页余量不足,将会触发垃圾回收并选择一个完整的块作为回收对象. 垃圾回收通过擦除过时的数据块来释放存储空间. 负载均衡是在 FLASH 芯片的使用过程中,使其内部块的磨损程度尽可能均衡.

页级 FTL[8]、块级 FTL[9]和页块混合级 FTL[10]是目前 FTL 设计中三种主要映射方案. 页级 FTL 以页为单位将一个逻辑地址映射到一个物理地址,其优点为转换效率高,但存储在 RAM 中的地址映射表较大. 块级 FTL 以块为单位进行逻辑地址到物理地址的映射,其基本思想是逻辑块中的逻辑页偏移

量与物理块中物理页的偏移量相同. 相比之下，块级 FTL 的映射表要比页级 FTL 映射表小得多，但块级 FTL 的转换效率较低. 页块混合级 FTL 使用块级映射技术来获得相应的物理块地址，使用页级映射技术来定位可用的物理页，页块混合级 FTL 在较长时间的极限工况下表现出较好的地址转换效率[11].

2　数据驱动的自适应超级块 FTL 算法(DASFTL)

2.1　地址映射

Jung 等最先提出基于超级块的闪存转换层算法[12]，其比块级 FTL 算法更进一步，它将多个连续的逻辑块组合成一个超级块. 例如，超级块的大小为 4，那么逻辑块编号为 0，1，2，3 的 4 个逻辑块构成了超级块 0. 虽然超级块闪存转换层算法能在块级 FTL 算法上进一步减少 RAM 的占用空间，但其只是由几个连续的逻辑块组合而成，无法根据实际系统中数据的运行特点进行调整. 因此，提出了一种数据驱动的自适应超级块 FTL 算法(DASFTL)，其地址映射机制见图 2.

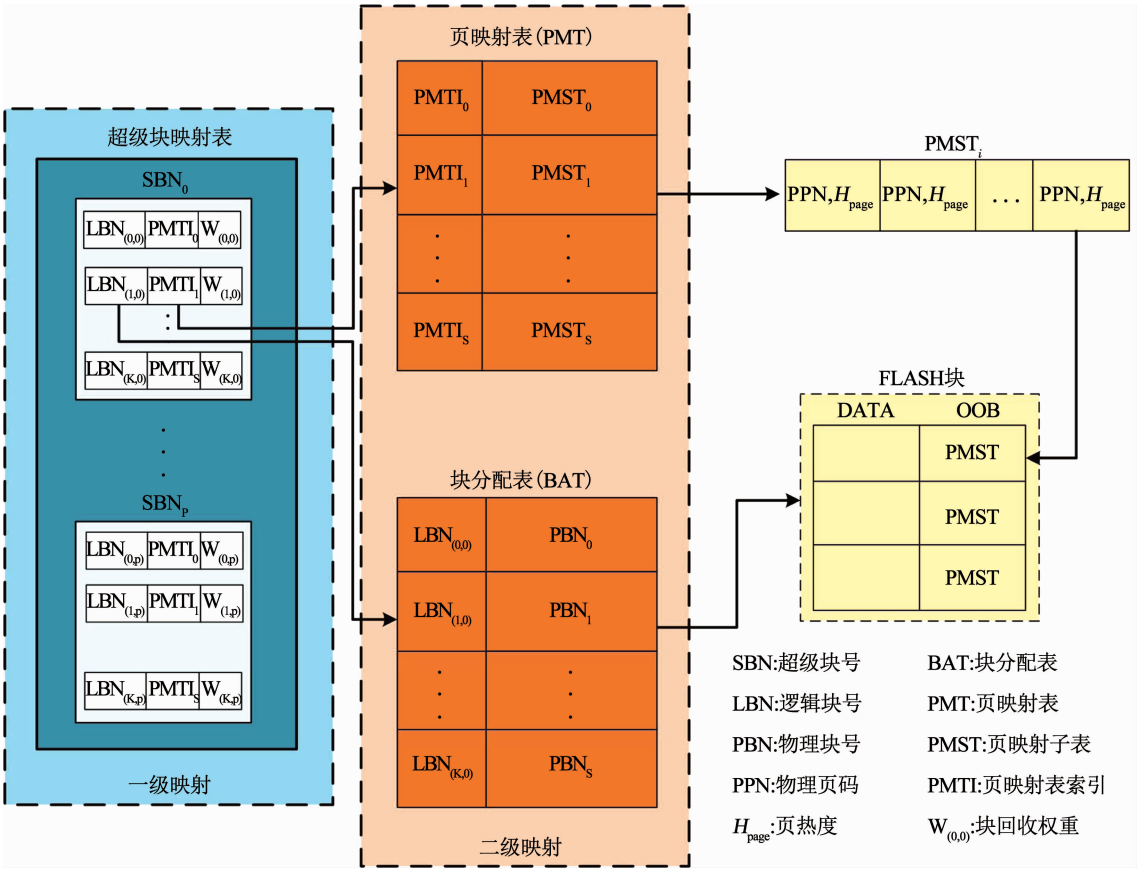


图 2　DASFTL 地址映射机制

Fig. 2　Address mapping mechanism of DASFTL

超级块映射表 SMT 和页地址映射表 PMT 组成两级映射，其中超级块映射表 SMT 为一级映射，页地址映射表为二级映射. 超级块映射表 SMT 由超级块号 SBN 进行索引，每个超级映射表项由逻辑块号 LBN、页映射表 PMT 和逻辑块权重 W_{Recy} 组成. 与 Superblock FTL 不同，在 DASFTL 中，超级块不是由几个连续的逻辑块简单组合而成，而是根据系统中数据运行的特点动态组合而成. 每个超级块中所拥有的逻辑块数为 K，K 的取值范围从 0 到存储系统中所有有效物理块的个数. 这 K 个逻辑块可以是连续的逻辑块也可以是不连续的，并且不同超级块的 K 值可能不同. 具体哪些逻辑块组合成为一个超级块由块回收权重 W_{Recy} 决定. 在超级块内部采用页级映射，超级块中每一个映射表项都可以映射至其对应的实际物理页上. PMT 被分为 S 个页映射子表(PMST_1、PMST_2、…、PMST_s)，由页映射表索引 PMTI 进行索引. 页映射表索引的内容为物理页码，页映射子表的内容由物理页码 PPN 和页热度 H_{page} 两部分组成，其中 S 的值可由 OOB 区的大小和逻辑块数计算得到. 假设每个物理块包含 α 个物理页，每个 OOB 区可以存储 $m(0<m\leqslant\alpha)$ 个页映射条目，则每个超级块所含有的页地址映射项为 $m\times\alpha$ 个，由此可

得 $S=(m\times\alpha)\div K$. 逻辑块号 LBN 和页地址映射表偏移量 PMToffset 的计算公式为:

$$\mathrm{LBN}=\mathrm{LPN}\div\alpha, \tag{1}$$

$$\mathrm{PMToffset}=\alpha\times\mathrm{SBO}+\mathrm{LPN}\%\alpha. \tag{2}$$

式中:LBN 为逻辑块号,LPN 为逻辑页号,α 为每物理块中的物理页数,SBO 为超级块偏移量,PMToffset 为页地址映射表偏移量.

每个 FLASH 块中含有的物理页数 α 是固定的,因此逻辑块号可以由逻辑页号 LPN 除以物理页数计算得到. 之后查找超级块映射表可以得到其所在的超级块号 SBN 和此超级块偏移量 SBO,利用式(1)~(2)就可以计算得到该逻辑页号在所对应的页地址映射表偏移量 PMToffset. 根据得到的 PMToffset 查找页地址映射表 PMT 就可以得到相应的物理页地址. 在 DASFTL 算法中,每个页映射子表 PMST 都将在 FLASH 写操作过程中,写入到最新分配的物理页 OOB 区. 在逻辑地址转换物理地址的过程中,可以通过读取 PMTI 所指示物理页的 OOB 区来直接获得 PMST. 这种分级映射方式通过维护两个短的映射链表来实现地址转换,避免了维护一个很长的地址链表所造成的搜索时间长和 RAM 占用率高等缺点,从而提高系统的响应速度并减少主机的占用,适用于 RAM 空间有限和实时性强的嵌入式系统.

2.2 读写地址管理

星载固态存储系统主要工作在连续读/写模式下. 传统星载存储系统大都采用页级 FTL 映射方案,其中每次读/写操作都需向主机请求地址,当存储系统连续写入数据时会造成存储系统大量占用主机资源,并且随着读写请求的不断累积系统响应时间将会变长. 为此,DASFTL 算法将超级块作为读/写操作的最小单位,当存储控制单元接收到数据并向主控 CPU 单元产生写请求时,主控 CPU 单元会一次性将块回收权重 W_{Recy} 最小的超级块地址分配给存储控制单元,若一次写请求将整个超级块地址都分配之后仍未完成则向主控 CPU 单元请求新的超级块,若还有部分地址未分配则该超级块的地址分配信息存储在存储控制单元中待下次写请求继续使用.

对给定超级块的第一个写请求是由该超级块的第一个空闲逻辑块来提供服务. 当一个物理块被选中分配给该逻辑块之后,物理块中的页将按顺序执行写操作. 当其中一个物理页被分配之后,首先从该页的 OOB 区读出其相对应的页映射子表,将页映射子表中的 PPN 更新为将写入的新页面的 PPN. 然后将新的 PMST 写入到 OOB 区域、数据写入到 DATA 区域. 写操作响应 T_{wr} 时间为

$$T_{\mathrm{wr}}=T_{\mathrm{wrask}}+T_{\mathrm{rdoob}}+T_{\mathrm{wrpg}}. \tag{3}$$

式中:T_{wrask} 为发出写请求到主机返回写地址的系统响应时间,T_{rdoob} 为读取 OOB 中旧 PMST 的时间,T_{wrpg} 为写入一页数据和更新 OOB 区 PMST 的时间.

对于读操作,主控 CPU 单元会给定起始逻辑页 LPN,然后 LPN 被翻译成逻辑块号 LBN 和超级块偏移量 SBO. 通过 LBN 和 SBO 可以计算得到此逻辑页所对应的页映射子表 PMST 及其所在的物理块号,之后读取 PMST 可以得到其所对应的物理页码. 读操作响应时间 T_{rd} 为

$$T_{\mathrm{rd}}=T_{\mathrm{rdask}}+T_{\mathrm{rdoob}}+T_{\mathrm{rdpg}}. \tag{4}$$

式中:T_{rdask} 为发出读请求到主机返回读地址的系统响应时间,T_{rdoob} 为读取 OOB 中旧 PMST 的时间,T_{rdpg} 为读取一页数据的时间.

2.3 垃圾回收

当存储系统没有足够的空闲块来响应写请求时,将会触发垃圾回收. 卫星处于地面接收站的范围内时,会将存储系统内的数据下传至地面,随后数据就会被标记为“冷数据”. 当触发垃圾回收机制时将优先回收冷数据块. 星载存储系统中,对于冷数据的回收只执行擦除操作. 传统星载存储系统大都基于固定分区的文件管理策略[5],分区内部采用顺序回收机制. 由于不同分区的数据热度和输入速率不同,会导致某些物理块的擦写次数不断攀升,直到该块的擦写次数到达上限变为无效块. 为此,DASFTL 算法不采用分区的文件管理策略,而将所有文件进行动态管理,其中超级块作为擦除的最小单位. 在超级块内部,同一分区的逻辑地址是连续的但是其对应的物理地址不一定是连续的.

在 DASFTL 中,为兼顾存储系统的性能和使用寿命,引入数据页热度 H_{page} 和物理块擦除次数 N_{earse} 来作为目标擦除块的选择因素. H_{page} 由式(5)计算得到. 其中,f 为一段时间内回放次数,$f_{k+1}-f_k$ 为第 k 次及第 $k+1$ 次触发垃圾回收时该页的回放次数. 页回放次数越多则该页的热度越大.

$$H_{\mathrm{page}}=\begin{cases}f_{k+1}-f_k, & 0<(f_{k+1}-f_k);\\ 0, & 0>(f_{k+1}-f_k).\end{cases} \tag{5}$$

将页热度 H_{page} 代入式(6)可得块平均热度 $\overline{H}_{\mathrm{block}}$. 其中,$\alpha$ 为每个物理块内所含页数.

$$\overline{H}_{\mathrm{block}}=\frac{\sum_{i=1}^{\alpha}H_{\mathrm{page}}(i)}{\alpha}. \tag{6}$$

将 $\overline{H}_{\mathrm{block}}$ 和 N_{earse} 代入式(7)可得块回收权重 W_{Recy}. 其中,P 为权值系数,取值范围为[0,1];N_{thre} 为块擦除次数阈值;$H_{\max}$ 为目标块最大热度. 当 P 为 0 时,块回收权重只考虑当前块的平均热度. 随着 P

不断增大，块平均热度对块回收权重的影响越来越低，块擦除次数对块回收权重的影响越来越大. 当 P 为 1 时，块回收权重只与块擦除次数有关. 存储系统在工作中，权值系数 P 可以随着系统当前的工作状况进行变化. 为平衡存储系统的读写性能和 FLASH 的使用寿命，缺省设置 P 为 0.5.

$$W_{\text{Recy}}=\left(\frac{P(N_{\text{thre}}-N_{\text{earse}})}{N_{\text{thre}}}+\frac{(1-P)(H_{\max}-\overline{H}_{\text{block}})}{H_{\max}}\right)\times 10. \tag{7}$$

在 DASFTL 算法中，块回收权重 W_{Recy} 将作为超级块组成单位的度量单位. 存储系统初次使用时，每个逻辑块的初始块回收权重都为 0. 随着存储系统的运行，超级块映射表 SMT 会计算并保存每个逻辑块的块回收权重. 块回收权重 W_{Recy} 相同的逻辑块将被组合成为一个超级块，即使这些逻辑块所对应的物理块地址是离散的. 图 3 为 DASFTL 算法垃圾回收流程图. 当超级块映射表 SMT 中没有足够的空闲块来响应写操作时将触发垃圾回收. 与响应写操作相同，主控 CPU 单元在响应垃圾回收操作时，将块回收权重最大的整个超级块下发给存储控制单元. 存储控制单元会按照顺序依次回收超级块内的逻辑块. 在完成一次回收之后超级块映射表 SMT 将更新块回收权重和新的映射索引.

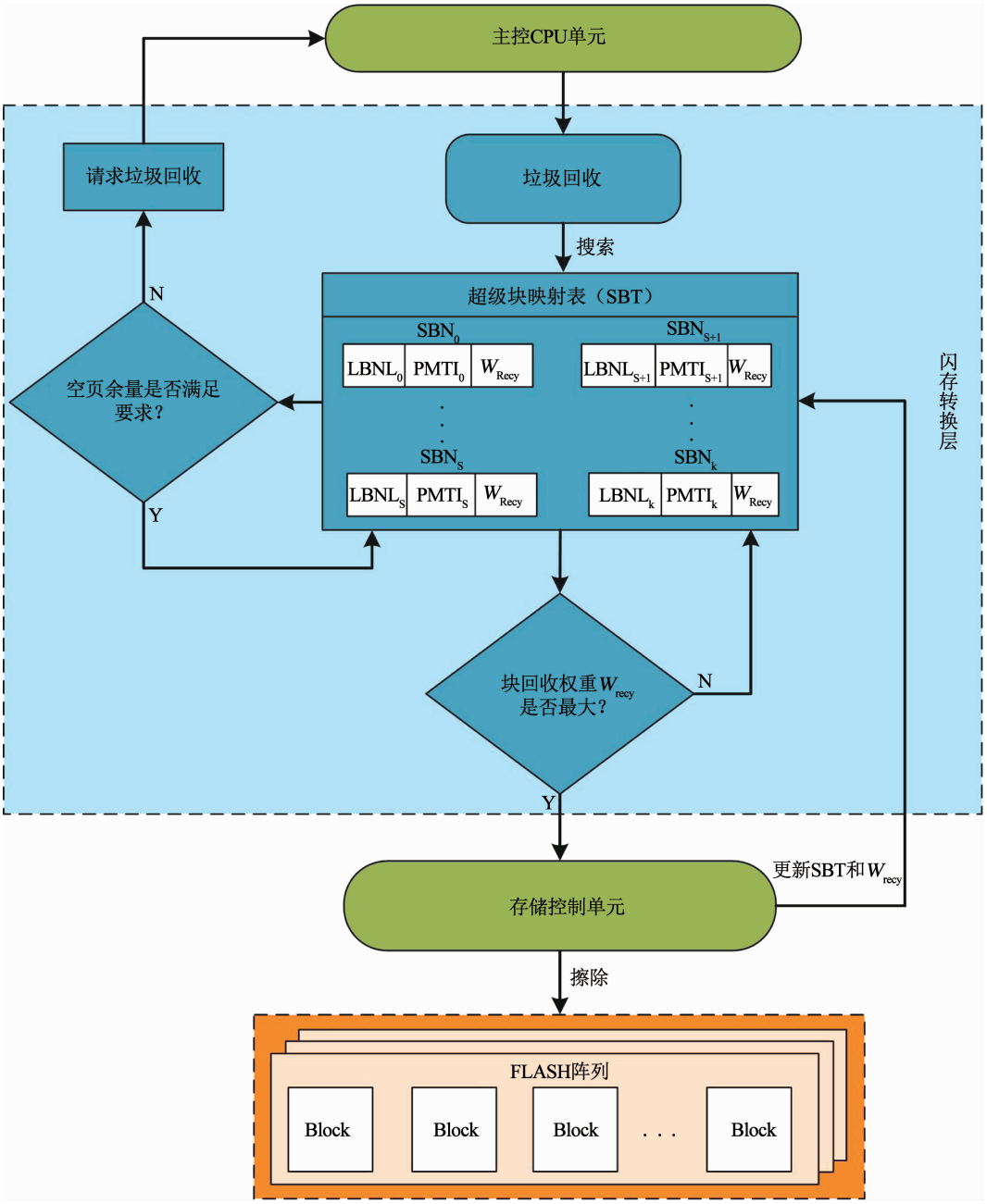


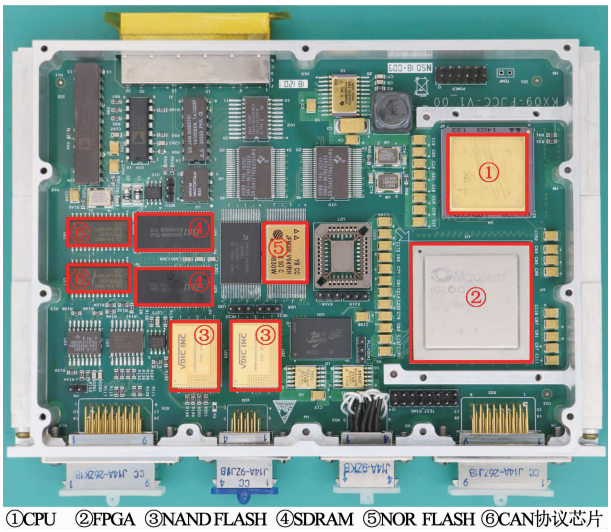
图 3　DASFTL 垃圾回收流程图

Fig. 3　Flow chart of DASFTL garbage collection

3 实验结果与分析

3.1 实验平台搭建

为评估本文提出的 DASFTL 算法,设计并搭建了星载固态存储系统硬件测试平台,见图 4.



①CPU ②FPGA ③NAND FLASH ④SDRAM ⑤NOR FLASH ⑥CAN协议芯片

图 4 星载固态存储系统测试平台

Fig. 4 Test platform for spaceborne solid state storage system

测试平台选用龙芯 1E 作为主控 CPU,龙芯 1E 是以 WH1770 为核心的高性能应用处理器. 软件工作在内嵌的 VxWorks 操作系统上,用于存储系统的任务管理与调度、中断与异常管理以及闪存转换层算法. CPU 内存采用三片 ISSI 公司 IS45S163 型支持 EDAC 的 SDRAM,容量为 3 Gb. CPU 程序存储器选用复旦微电子生产的 JFM29LV64 型号 NOR FLASH,容量为 64 Mb. FPGA 选用 Microsemi 公司 M2GL150T FPGA 芯片,完成大容量固态存储管理、数据接收、复接及部分逻辑粘合功能. 大容量固态存储器采用两片珠海欧比特公司 VDNF64G08 型 NAND FLASH,单片容量 64 Gb,总容量为 128 Gb. FLASH 芯片参数和内部编程时间见表 1.

3.2 实验结果与分析

基于测试平台,对 DASFTL 算法与传统星载 FTL 算法在主机占用率、读/写响应时间和 FLASH 块擦除次数进行测试. 在主机占用率测试中,共进行 300 000 次 I/O 请求,其中 0 ~ 100 000 次存储系统只进行写数据操作;之后对存储区数据进行全擦除,将存储系统设置在写数据与读数据同时工作模式下,进行 100 000 次 I/O 请求实验;最后重置存储区,工作模式设置为最大工况,存储系统同时进行写、读、擦操作. 记录每次 I/O 操作时主机占用率,为排除系统其它部分占用主机资源对实验造成干扰,屏蔽系统内其他功能并以每 25 000 次 I/O 为一组计算主机占用率的平均值,实验结果见图 5.

表 1 FLASH 芯片参数及编程时间

Tab. 1 Parameters and programming time of FLASH

Parameters	总块数	块内页数	页大小/KB	块擦除时间/μs	页编程时间/μs	页读取时间/μs	OOB 区读取时间/μs
Values	32 768	128	4	2 000	265	88	28

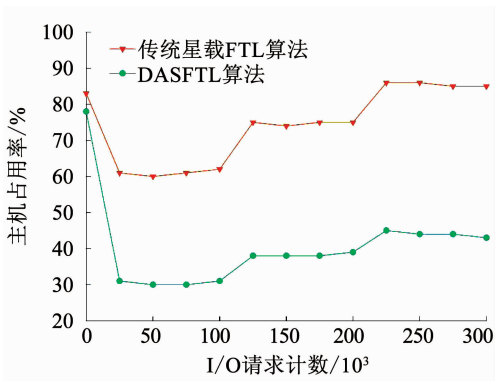


图 5 传统星载 FTL 算法与 DASFTL 算法主机占用率

Fig. 5 Host occupancy rate of traditional spaceborne FTL and DASFTL algorithms

系统上电后,主控 CPU 单元会遍历整个存储区并建立索引表,此时两者主机占用率都在 80% 左右. 当进行写操作时,DASFTL 主机占用率迅速下降并维持在 31.2% 左右,而传统星载 FTL 算法维持在 60.5% 左右. 当同时进行读写操作时,DASFTL 算法的主机占用率维持在 39.7% 左右,传统星载 FTL 算法上升至 75.0% 左右. 最大工况下,DASFTL 算法占用率稳定在 44.8% 左右,传统星载 FTL 算法则上升至 86.1%. 综上,采用超级块地址映射机制的 DASFTL 算法相比于传统星载 FTL 算法在主机占用率上平均降低 51.7%.

分别独立进行 100 000 次读、写操作实验,通过龙芯 CPU 记录每次收到读/写请求至返回读/写地址的系统响应时间(T_{wrask}/T_{rdask}),通过式(3) ~ (4),计算得每次读、写操作的响应时间 T_{rd} 和 T_{wr}. 对 T_{rd} 和 T_{wr} 进行累和求均值,测试结果见图 6.

传统星载 FTL 算法读/写操作平均响应时间分别为 206 μs 和 408 μs,其中系统响应时间 T_{rdask} 为 90 μs、T_{wrask} 为 115 μs. DASFTL 算法读/写操作平均响应时间分别为 159 μs 和 344 μs,其中系统响应时间 T_{rdask} 为 43 μs、T_{wrask} 为 51 μs,相比于传统星载 FTL 算法读/写操作的系统响应时间下降了 46.1%.

模拟卫星工作模式,设置三路输入数据分别为文件 1、文件 2 和文件 3. 文件 1 数据输入速率为 100 Mbps,文件 2 输入速率为 150 Mbps,文件 3 输入速率为 50 Mbps. 实验剔除 75 块初始无效块,实际有效物理块为 32 693 块. 设置实验运行时间为当存储控制单元共执行 326 930 次擦除操作时停止. 实验结果见图 7.

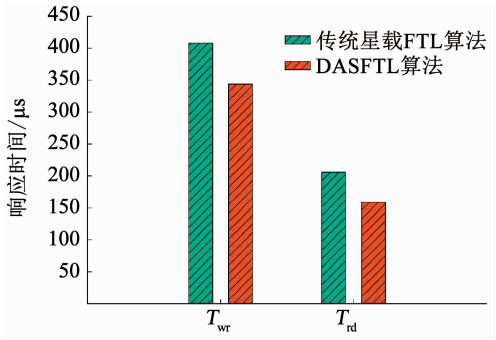


图 6　传统星载 FTL 算法与 DASFTL 算法读/写响应时间

Fig. 6　Read/Write response time of traditional spaceborne FTL and DASFTL algorithms

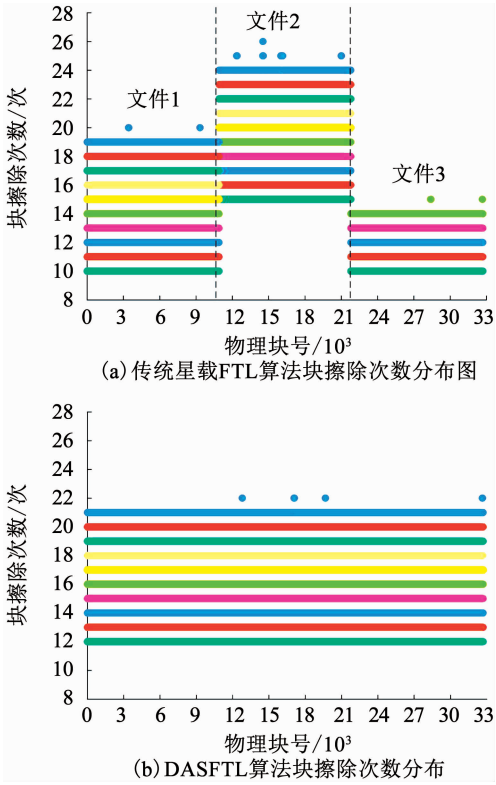


图 7　传统星载 FTL 算法与 DASFTL 算法物理块擦除次数分布

Fig. 7　Distribution of physical block erasure times of traditional spaceborne FTL and DASFTL algorithms

基于固定分区的传统星载 FTL 算法因各文件输入速率的差异，造成不同物理块的擦除次数呈现显著差异，文件 2 分区的物理块擦除次数高于文件 1 和文件 3 分区. DASFTL 算法的擦除块选择策略只与当前块的热度和擦除次数相关，相同实验条件下，DASFTL 算法的各物理块擦除次数相比于传统星载 FTL 算法更加均衡，有效避免了部分物理块过早磨损.

综上，DASFTL 算法相比于传统星载 FTL 算法在主机占用率、系统响应时间以及 FLASH 磨损均衡等方面均有明显的性能提升. DASFTL 算法的实际测试结果与预期结果吻合，证明了该算法的可行性和实用性.

4　结　论

针对传统星载闪存转换层算法存在的问题，提出了一种数据驱动的自适应超级块 FTL 算法，采用了基于超级块的地址映射策略，并引入动态块回收权重作为目标回收块选择的标准. 实验结果表明，DASFTL 算法相比于传统星载 FTL 算法在主机资源占用率由 81.4% 下降至 44.8%、系统读/写响应时间由 206 us/408 us 提升至 159 us/344 us，响应时间提升 46.1%. FLASH 芯片磨损均衡性能方面 DASFTL 算法在长时间工作下，各物理块擦除次数更加均衡，可有效避免部分物理块的过早磨损，提升 FLASH 芯片的使用寿命. 但 DASFTL 算法仍存在代码实现复杂、后期维护困难等问题.

参考文献

[1] 李姗，宋琪，朱岩，等. 星载大容量固态存储器快速可靠启动算法设计[J]. 哈尔滨工业大学学报，2015，47(10)：100
LI Shan，SONG Qi，ZHU Yan，et al. Design of quick initialization algorithm for spaces borne solid state recorder[J]. Journal of Harbin Institute of Technology，2015，47(10)：100. DOI：10.11918/j.issn.0367－6234.2015.10.022

[2] YANG Mingchang，CHANG Yuanhao，KUO Teiwei，et al. Reducing data migration overheads of flash wear leveling in a progressive way [J]. IEEE Transactions on Very Large Scale Integration Systems，2016，24(5)：1808. DOI：10.1109/tvlsi.2015.2495252

[3] 许志宏. 面向星载一体化综合电子系统的固态存储技术研究[D]. 北京：中国科学院大学，2017
XU Zhihong. Research on solid-state storage technology for integrated spaceborne electronic systems[D]. Beijing：University of Chinese Academy of Sciences，2017

[4] 宋琪. 星载固态存储管理技术的应用研究[D]. 北京：中国科学院大学，2015
SONG Qi. Applied research of space-borne solid state storage management technology [D]. Beijing：University of Chinese Academy of Sciences，2015

[5] 董振兴. 星载固态存储文件化管理方案应用研究[D]. 北京：中国科学院大学，2017
DONG Zhenxing. Applied research on filing management scheme of spaceborne solid state storage[D]. Beijing：University of Chinese Academy of Sciences，2017

[6] 董振兴，朱岩，许志宏，等. 星载存储器吞吐率瓶颈与高速并行缓存机制[J]. 哈尔滨工业大学学报，2017，49(11)：52
DONG Zhenxing，ZHU Yan，XU Zhihong，et al. Bottleneck analysis of spaceborne memory throughput and high-speed parallel caching mechanism design[J]. Journal of Harbin Institute of Technology，2017，49(11)：52. DOI：10.11918/j.issn.0367－6234.201611144

[7] JI S，SHIN D. An efficient garbage collection for flash memory-based virtual memory systems [J]. IEEE Transactions on Consumer Electronics，2010，56(4)：2355. DOI：10.1109/TCE.2010.5681112

[8] MA Dongzhe，FENG Jianhua，LI Guoliang. LazyFTL：A page-level flash translation layer optimized for NAND flash memory [C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. Athens，NJ：ACM，2011：12

[9] GUAN Yong，WANG Guohui，WANG Yi，et al. BLog：Block-level log-block management for NAND flash memory storage systems[J]. ACM SIGPLAN Notices，2013，48(5)：111. DOI：10.1145/2499369.2465560

[10] CHOUDHURI S，GIVARGIS T. Real-time access guarantees for NAND flash using partial block cleaning [C]//Proceedings of Software Technologies for Embedded and Ubiquitous Systems，6th IFIP WG 10.2 International Workshop. Anacarpi，IT：Springer-Verlag，2008. DOI：10.1007/978－3－540－87785－1_13

[11] QIN Zhiwei，WANG Yi，LIU Duo，et al. Real-time flash translation layer for NAND flash memory storage systems [C]// Proceedings of 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium. Beijing：IEEE，2012：1. DOI：10.1109/RTAS.2012.27

[12] JUNG D，KANG J U，JO H，et al. Superblock FTL：A superblock-based flash translation layer with a hybrid address translation scheme [J]. ACM Transactions on Embedded Computing Systems，2010，9(4)：1. DOI：10.1145/1721695.1721706

（编辑　苗秀芝）