Profiling FlashFX® Disk Performance

Christopher Tacke, eMVP
Windows CE Product Manager, Applied Data Systems
Columbia, Maryland
April 23, 2002

Abstract

With embedded systems, just like most any information processing system, saving data is an important and integral function. With the constraints and expectations of real-time performance put on embedded systems, time requirements for data writing can be critical, and since persistent (flash) RAM is becoming such a commonplace data medium, it is important to determine the expected performance characteristics of the medium.

In this paper I profile the write performance of the Datalight FlashFX flash media manager (version 4.08) by creating and opening a single file on the flash disk and repeatedly adding a 256 byte block of data to that file until the disk was full. The elapsed time was recorded for the write operation on each block.

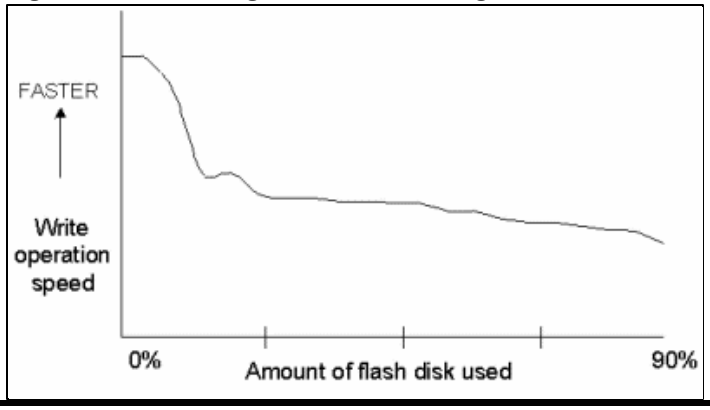
An analysis of the data shows that the execution time of write operations not requiring garbage collection does *not* seem to significantly increase as the disk becomes full. Analysis did, however, show that as the disk becomes more full, garbage collection has a doubly negative effect on performance by both increasing the time required to execute the garbage collection and increasing the frequency of garbage collection.

Introduction

Datalight's FlashFX Performance Characteristics[1] white paper provides a good introduction to how their FlashFX flash media manager's uses and is impacted by garbage collection.

Flash memory cannot simply be written to, but instead the target write area must first be erased, therefore write performance is directly affected by the erase performance. During normal write operations, garbage collections make space for the new data to be written and these garbage collections result in an alternating fast/slow performance curve for the flash disk.

Figure 1 – The Datalight Performance Degradation Curve



According to the Datalight white paper, flash disk performance degrades as the amount of disk used increases, with the highest degradation rate being when less than 25% of the flash disk is in use.

Figure 1 shows the performance degradation curve presented by Datalight.

While informative, the Datalight white paper does not quantify the degradations. The purpose of this paper is as an extension to the Datalight paper, providing actual collected data to quantify the performance degradation that flash disk users can expect.

Data Collection

Data collection was done using an Applied Data Systems' Graphics Master™ development system running Windows CE.NET. The Graphics Master™ used had 32MB of on-board flash media logically partitioned so that a 4MB flash disk was available through the FlashFX media manager to the test application.

A single-threaded application was written that created a single file on the flash disk and repeatedly appended 256 byte blocks of data to the file until the flash disk was full. With each write operation, the elapsed time was calculated and output through the device's debug port so that it could be collected and analyzed.

The application code, trimmed of error checking for clarity, can be seen in Listing 1

Listing 1 – Application Code

```
#define FILE_SIZE       256
#define TEST_FILE_NAME  T("\\FlashFX Disk\\Flash.Test")

int WINAPI WinMain( HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPTSTR    lpCmdLine,
                    int       nCmdShow)
{
  ULARGE_INTEGER freeSpace;
  ULARGE_INTEGER totalSpace;
  short          iterations;
  long           elapsed_time;
  HANDLE         hFile;
  BYTE           *outBuffer;
  DWORD          bytesWritten;

  // determine # of blocks to write
  GetDiskFreeSpaceEx(_T("\\FlashFX Disk"), &freeSpace, &totalSpace, NULL);

  DEBUGMSG(TRUE, (_T("%i FlashFX bytes free\r\n"), freeSpace));

  iterations = (short)(freeSpace.QuadPart / FILE_SIZE);

  DEBUGMSG(TRUE, (_T("Writing %i "), iterations));
  DEBUGMSG(TRUE, (_T("%i byte blocks\r\n"), FILE_SIZE));

  // create output file
  hFile = CreateFile(TEST_FILE_NAME, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                     FILE_ATTRIBUTE_NORMAL, 0);

  // create data block for writing
  outBuffer = (BYTE *)LocalAlloc(LPTR, FILE_SIZE);

  for(int i = 0 ; i < iterations ; i++)
  {
         // begin timing
         elapsed_time = GetTickCount();
```

```
            // write data block
            WriteFile(hFile, outBuffer, FILE_SIZE, &bytesWritten, NULL);

            // calculate ET
            elapsed_time = GetTickCount() - elapsed_time;

            // output ET
            DEBUGMSG(TRUE, (_T("%i ticks\r\n"), elapsed_time));
    }

    // close file
    CloseHandle(hFile);

    DEBUGMSG(TRUE, (_T("Cleaning up...\r\n")));

    // delete file and free memory allocation
    DeleteFile(_T("\\FlashFX Disk\\Flash.Test"));
    LocalFree(outBuffer);

    DEBUGMSG(TRUE, (_T("Testing complete.\r\n")));

    return 0;
}
```

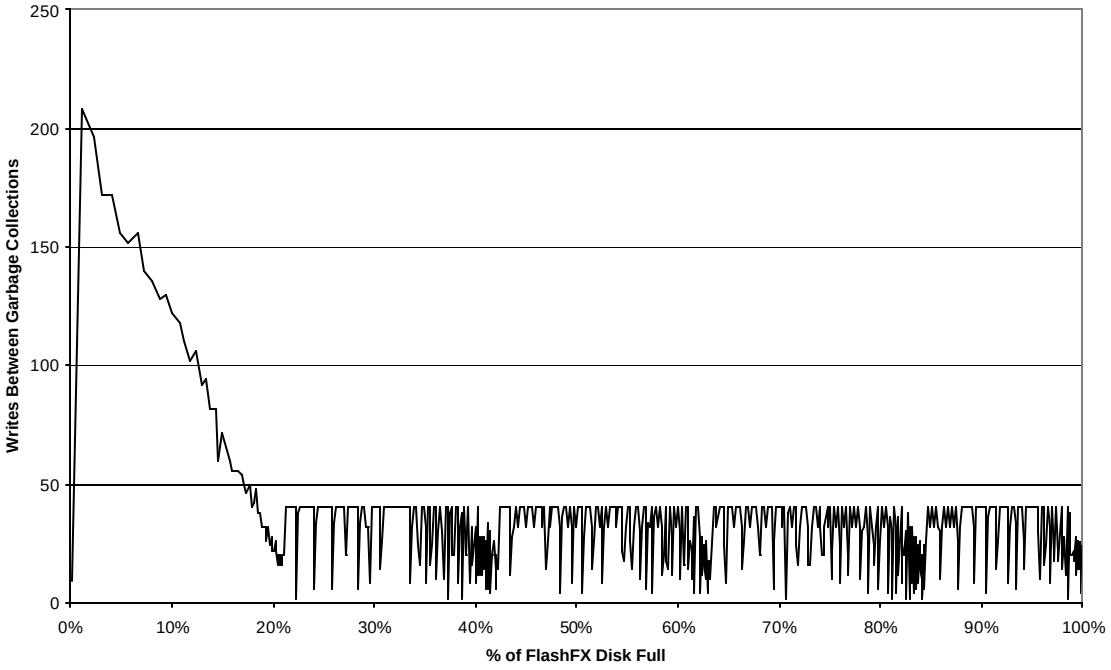
Results

In the resulting data set it was very easy to distinguish between data points in which garbage collection was performed and those in which it was not. The data split into two groups, one with 555 data points that had an average elapsed time of 1797 milliseconds (garbage collection occurred) and a second with 18,117 data points that had an average elapsed time of only 5.5 milliseconds (garbage collection did not occur).

The first data that I looked at was the periodicity of garbage collections, or the number of writes that the application performs between garbage collections. The periodicity started at approximately 200 writes between garbage collections when the flash disk was empty and decreased to approximately 25 writes between garbage collections when the flash disk was about 20% full. At that point the periodicity plateaued at a rate of 40 writes between garbage collections, with a significant number of data points ranging from as low as 3 to a maximum of 40.

Figure 2 shows the number of writes between garbage collections as a function of the percentage of filled flash disk space.

Figure 2 - Periodicity of Garbage Collection

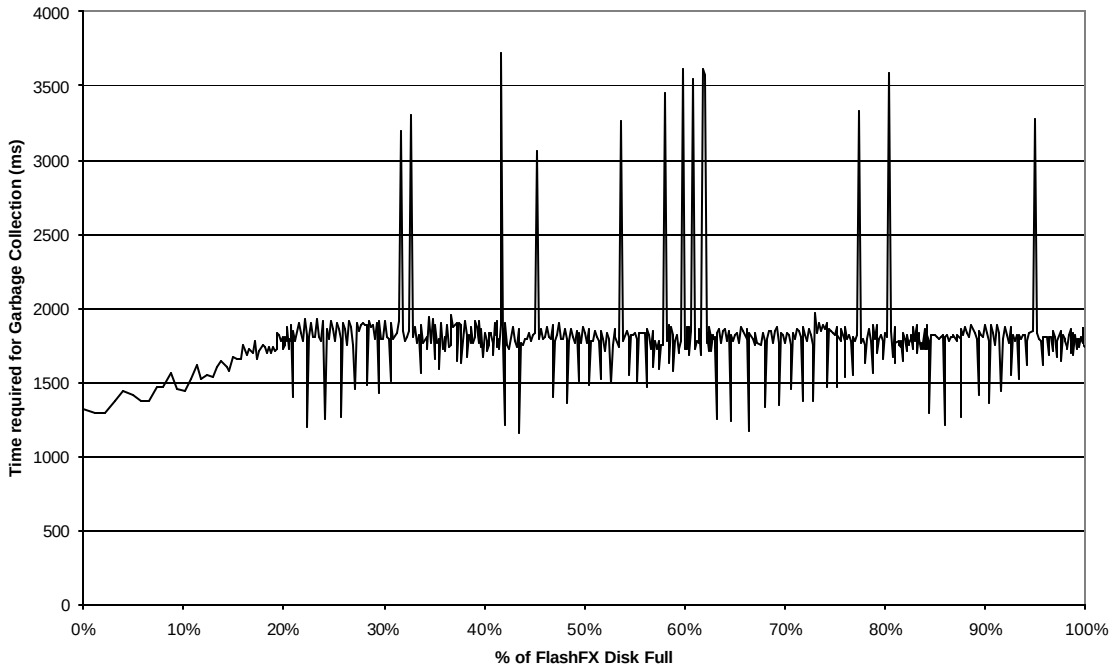


Next I looked at the elapsed time of the write functions that required garbage collection. The elapsed time started at approximately 1300 milliseconds when the flash disk was empty and gradually increased to approximately 1800 milliseconds when the flash disk was about 20% full.

At that point the elapsed time seemed to plateau at about 1800 milliseconds, but there were a significant number of outliers giving elapsed time values ranging from about 1200 milliseconds to over 3700 milliseconds.

Figure 3 shows the elapsed time of writes with garbage collections as a function of the percentage of filled flash disk space.

Figure 3 - Elapsed Times for Garbage Collection

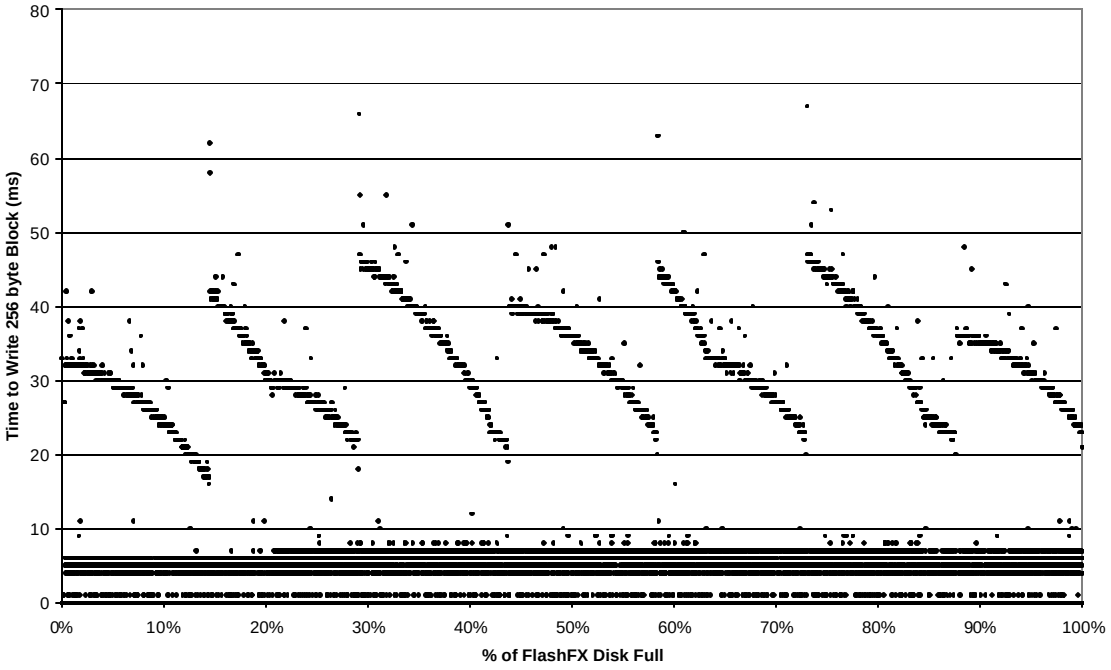


Since it was easy to distinguish the data points where garbage collection did not occur from those where it did, I extracted all non-garbage collection data points to get a profile of the actual write performance of the flash disk itself.

A majority (89.5%) of the data points determined to be non-garbage collection points had elapsed times less than 15 milliseconds with a moving average ranging from 4 to 7 milliseconds. The remaining 10% of the data points lie in "clusters" with elapsed times between 15 and 50 milliseconds.

Figure 4 shows the elapsed time of non-garbage collecting writes as a function of the percentage of filled flash disk space for the 256 byte blocks.

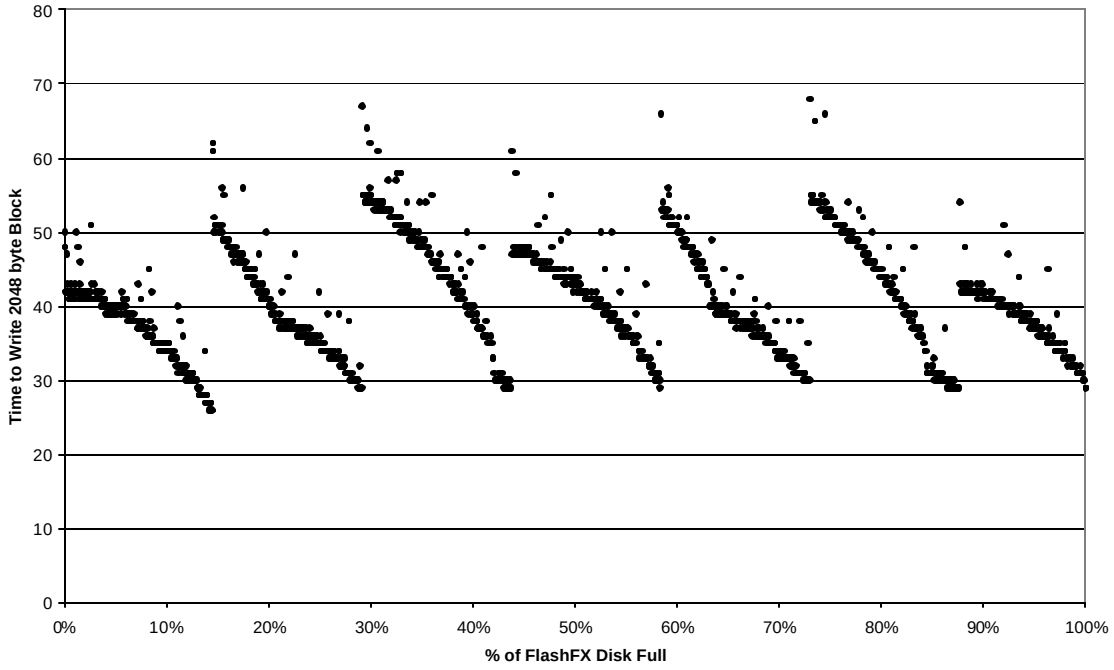
Figure 4 - Elapsed Time to write 256 bytes



Interestingly, running the same tests with block sizes of 2048 bytes instead of 256 bytes eliminates all of the very low end data points (probably due to the time required to physically write the data) but yields nearly identical data "clusters" as the 256 byte data.

Figure 5 shows the elapsed time of non-garbage collecting writes as a function of the percentage of filled flash disk space for the 2048 byte blocks.

Figure 5 - Elapsed Time to write 2048 bytes



Conclusion

When writing applications or devices that will be using flash media, it is extremely important to be aware of the performance implications of how data is written. While writing to the flash disk when it is less that 25% full can produce some fast values, the potential for long write latencies is high. Additionally, writing data to the flash disk has two negative performance effects. First, it increases the time required to do garbage collection and second it increases the frequency at which garbage collection occurs.

Once approximately 25% of the flash disk is full, performance of the flash disk seems to stabilize and additional data on the disk seems to have little or no effect on performance. Still, once 25% of the disk is in use, write latencies of over three (3) seconds are not unusual.

There is potential to mitigate this performance degradation to a small degree by calling for manual garbage collection at times when the flash disk is idle, but there are a few issues that may arise:

1. If the OS has mounted the flash disk, getting an application reference to the flash disk to make the garbage collection call may be difficult or impossible.
2. Garbage collection will still take the same amount of time and if the application needs to perform write functions on intervals less than the time required to do garbage collection, application performance will still be impacted.

3. Garbage collection may still need to be performed before a write even if it was called for manually, so write performance times still cannot be guaranteed.

While flash media is good for persistent storage, its write performance must be taken into account when designing an application or system. If guaranteed write times are required, such as when doing emergency data dumps during a power loss, then alternative data storage or power management techniques must be considered.

References

1. FlashFX® Performance Characteristics, Datalight, Inc. white paper, http://www.datalight.com/wp-flashfx-perform.htm