FastCache V1.1

Philip D'Ath

1 License Agreement

Any reference to FastCache refers to all the FastCache's, irrespective of the processor that the particular version was designed for. FastCache is an unregistered trademark of Philip D'Ath.

1. COPYRIGHT: FastCache and the related documentation are copyright. You may not use or modify the programs or documentation or any copy except as expressly provided in this agreement
2. LICENSE You have the non-exclusive right to use any enclosed program. You may load the program into your computer's temporary memory (RAM). You may physically transfer the program from one computer to another. You may not decompile, disassemble, reverse engineer, modify or translate the program. After a trial period of 3 months you must register your copy of FastCache for each machine that FastCache is run on. All other rights and uses not specifically granted in this license relating to items created by the author of FastCache are reserved by the author of FastCache. Where clarification is required of the above paragraph, the author of FastCache shall decide the clarification. Where such clarification is deemed illegal, the latter sentence will be withdrawn from this license.
3. BACKUP-UP AND TRANSFER: You may make any number of physical backups of the program and documentation provided the backups remain in your immediate possession. You must reproduce and include the copyright notice on any backup copy. You may not transfer or license the product to another party.
4. TERMS: This license is effective until terminated. You may terminate it by destroying the program and documentation and all copies thereof. This license will also terminate if you fail to comply with any term or condition of this Agreement. You agree upon such termination you agree to destroy all copies of the program and documentation. FastCache is distributed under the concept of Shareware.
5. PROGRAM DISCLAIMER: THE PROGRAMS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF ANY PROGRAM IS ASSUMED BY YOU. Should the program prove defective, you assume the entire cost of all necessary servicing, repair, or correction. Further, the author of FastCache does not warrant, guarantee, or make any representation regarding the use of, or the results of the use of, the program in terms of corrections, accuracy, reliability, currentness, or otherwise; and you rely on the program and results solely at your own risk.
6. LIMITATION OF LIABILITY: THE AUTHOR OF FastCache ACCEPTS NO LIABILITY FOR ANY DAMAGE THAT MAY RESULT FROM USE OF THIS PRODUCT AND SHALL NOT IN ANY EVENT INCLUDE DAMAGES FOR LOSS OF USE OR LOSS OF ANTICIPATED PROFITS OR BENEFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL

COSTS, EXPENSES OR DAMAGES, INCLUDING WITHOUT LIMITATION ANY DATA OR INFORMATION WHICH MAY BE LOST OR RENDERED INACCURATE, EVEN IF THE AUTHOR OF FastCache HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. MISCELLANEOUS: This agreement represents the entire understanding regarding the programs and related documentation, and supersedes any prior purchase order, communications or representations. This agreement may only be modified by a written amendment signed by the author of FastCache. If any provision of the agreement shall be deemed unlawful, void, or for any reason, unenforceable it shall be deemed separate from, and shall in no way affect the validity and enforceability of the remaining provisions of the agreement.

   This license agreement shall be governed by the laws of the New Zealand.

8. The program License Agreement shall inure to the benefit of the author of FastCache.

   You should not be intimidated by this agreement. It is expected that common sense should prevail. This agreement is modeled after the copyright that comes with the C compiler, DICE (by Matthew Dillon), and I acknowledge this.

2 Introduction

2.1 History

In 1992 I came across a program called SmartDisk. This program was a hard drive cache. After using this program for a short time I was very impressed by the performance increase it could give.

During that year I learnt about cache's at the University of Waikato, and several caching algorithms. In particular, I learned about direct map, set associative and fully associative caches, and the merits of each. A direct map cache is the easiest to implement, has a low overhead, but can waste large chunks of the cache's memory.

A set associate cache (which is what SmartDisk uses) is a little smarter with its memory usage. Essentially, a set associate cache is a whole lot of little direct map caches combined. Obviously, a set associate cache is a little more tricky to implement since several direct map caches have to be managed. The most complex of the three caching algorithms is a fully associate cache. This has the greatest processing overhead of the three, but results in much better cache memory usage.

After having learned about these three particular caching algorithms, and remembering what SmartDisk was like, and that it used a set associate cache I decided to write a hard drive cache that implemented the more difficult fully associate cache algorithm. And I'm pleased to say, it came out better that I expected.

Additionally, a second algorithm has to be used to determine what action should occur when a cache miss occurs. I chose to implement an LRU (Least Recently Used) algorithm, as this is also one of the best available.

2.2 Features:

Below is some of the many features I have built into FastCache.

- Fully associate cache (one of the best algorithms)
- LRU cache replacement policy (one of the best)
- Can handle multiple drives
- Can handle removable media

- All cache settings are determined at run time
- Optional write retention
- Does not require large continuous chunks of memory
- Uses a hashing system to locate date (one of the best)
- Performs both forward and reverse prefetching
- Will utilize the blitter to move data, if possible

I have spared little in writing FastCache. It uses some of the best algorithms (and more often than not, most complex algorithms) for nearly everything.

3 Requirements

FastCache V1.1 requires KS2.0 and above. FastCache no longer supports KS1.3. FastCache should ONLY be used on devices that support seeking, like hard drives.

I recommend using FastCache V1.1 with 1Mb of memory or more, although it works very happily with 512Kb (the default parameters are set up for 512Kb). FastCache will run with smaller caches, but you will be strangling it. With 512K FastCache performs nicely. However, the more memory you give to FastCache, the better it goes. Theoretically, there should be a point where if more memory is added, performance decreases. However, because I have used a hash searching system, I expect this limit would be very high (>10Mb for a 68030). As yet, I have been unable to find this limit.

There are two version of FastCache supplied in this archive. "FastCache.68000" is designed to run on a 68000, so it's suitable for all Amigas. "FastCache.68020" is designed for a 68020 or better, and makes use of some of its extra instructions and addressing modes.

3.1 Warnings

I must give you some warnings as well. Because FastCache uses RAM to store the most often used data, "bad" things could happen if you have poorly behaved programs that trash memory.

If you use write retention MAKE SURE YOU WAIT FOR THE CACHE TO BECOME EMPTY BEFORE RESETTING YOUR MACHINE. Otherwise, if data is waiting to be written out, and you perform a reset before the cache is flushed, then the data will be lost.

You should use FastCache with care if your hard drive has errors on it. FastCache will warn you if an error occurs, but it may be too late if you have write retention enabled. The worst problem occurs when a write request is made. If write retention is on FastCache *may* retend the data in the cache (then again, it might write it straight to disk). Then at some later stage it may write it to the disk. If an error occurs FastCache can not report it to the program that made the request, because it would have already been acknowledged. If this occurs FastCache will give you an error (it pops up a little requester), but there is little that can be done.

4 Changes

- Found a bug where if two or more copies of FastCache V1.0 were running, and one was quit then the other(s) could crash. This was a semantic error. Once a library vector has been patched it is very difficult to remove the patch (and can be impossible it something else has patched it in the mean time). FastCache V1.1 can not be quit, only suspended. Once suspended it will free most memory, and wait till it has been activated again.
- Found a bug in the write back monitor process. This bug causes the cache to be flushed far more frequently than it was supposed to be. The writing speed of the cache is now faster with write retention enabled.
- Removed the option to have a chip memory buffer. I put it this option in just in case some old driver from KS1.3 still required it. As KS1.3 is no longer supported, neither is this option. This removed a whole level of complexity from the cache.
- Added an ARexx port to control FastCache, and to allow statistical information to be obtained about the caches performance.

5 Installation

FastCache can be run from either WorkBench or the CLI. I recommend placing FastCache in your WBStartup drawer.

FastCache requires the library "rexxhost.library". Copy this file from the "libs" drawer in the archive to your own "LIBS:" directory.

The first time FastCache is started it becomes active. If you run it again with the same parameters it will suspend, and free most memory. If started again with the same parameters it will allocate the required memory again, and become active.

You must run a separate copy of FastCache for every physical device yu wish to cache. Note that one physical hard drive might contain several partitions (e.g. DH0:, DH1:, etc), but it is one physical device. FastCache will cache everything on the physical device.

Before continuing, you must find out the device name and unit number of the device you wish to cache. Some examples are:

- GVP Controllers:
  'DEVICE=gvpscsi.device', 'UNIT=0' or 'UNIT=1'
- Commodore Controllers:
  'DEVICE=scsi.device', 'UNIT=6'

Other unit numbers may also be used (valid unit numbers range from 0 to 7).

Once you have this information, select whether you want a WorkBench start (recommended) or a CLI start.

5.1 WorkBench

When started from WorkBench FastCache gets its parameters from the ToolTypes field of its icon. Copy the appropriate FastCache for our machines processor to your WBStartup drawer. Click once on the icon, then using the right mouse button select "Icons/Information" from the WorkBench menu. You will see two parameters, 'DEVICE=' and 'UNIT='. Enter in the device and

unit number you have already obtained, and save the new values. Now simply double click on FastCache, and the cache will start running.

5.2 CLI

Edit your '`User-Startup`' file, and add a line like:

'`FastCache.680x0 -DEVICE=gvpscsi.device -UNIT=0`'

Make sure you substitute in the appropriate device name and unit number.

5.3 Continuing On

FastCache should now be installed. You may find that reading the "Parameters" section useful if you want to fine tune the cache for your system, or if you want to change the default cache size (which is 512kb).

6 Parameters

The available parameters and their descriptions are:

Note that if you change a parameter **after** FastCache was been started then you will need to reboot to make the parameter take effect.

6.0.1 DEVICE

Format: DEVICE=<devicename>

Default: gvpscsi.device

Example: DEVICE=gvpscsi.device

Description:

This specifies the name of the device that is to be cached.

See also: UNIT

6.0.2 UNIT

Format: UNIT=<unit number>

Default: 0

Example: UNIT=0

Description:

This parameter specifies which unit number to cache. Each drive will have a unit number. The first drive is usually unit 0. Note that a Hard Drive may have many partitions on it. This unit number does not refer to those partitions, but to the drive itself. All partitions on a selected drive will be cached. **Warning:** There is no element in a device structure to determine which unit is being talked to. Hence this parameter may be unreliable on some systems. It has been tested on a GVP SCSI interface with no problems. If you have multiple drives, use FastCache with care to begin with.

See Also: NO_UNIT_CHECK

6.0.3 NO_UNIT_CHECK

Format: NO_UNIT_CHECK

Default: Off (units are checked)

Example: NO_UNIT_CHECK

Description:

This parameter is designed for those Hard Drive cards that the unit selection above does not work on. This option tells FastCache not to try and check IO requests for the unit they are destined for. **Warning:** ONLY USE IT IF YOU HAVE ONE HARD DRIVE

See Also: UNIT

6.0.4 NUM_LINES

Format: NUM_LINES=<number of lines to use>

Default: 256

Example: NUM_LINES=256

Description:

This tells FastCache how many lines it should allocate for the cache. I wont give you the technical description, but in a multitasking environment several programs could be wanting data from several different places on the disk. To enable effective management of the cache (especially with regard to prefetching) each of these requests are assigned a line. Several lines can also be combined to form a longer line as well for programs which make many requests for data in a similar place. This parameter can be any number greater than or equal to 1. If you want a bigger cache this should be one of the first parameters that you increase.

NOTE: To calculate the size of the cache multiply the number of lines by the line size by the block size (512).

See Also: LINE_SIZE

6.0.5 LINE_SIZE

Format: LINE_SIZE=<size of each line in blocks>

Default: 4

Example: LINE_SIZE=4

Description:

This is the line size for the cache. I wont give a technical description, but this essentially controls how much pre-fetching is done. For a device with a slow seek time use a small line size. For a fast seeking device a bigger line size can help. Note, if the data on your drive is not contiguous then use a small line size (like 4, or even 2). This parameter MUST be a power of two (e.g. 2, 4, 8) and must be greater than or equal to 1. If you supply a parameter which is not a power of two, it will be rounded to a power of two. Note that both reverse and forward prefetching is done once the direction of the read/write operation is established (e.g. Try using Aquarium and search backwards for something).

I recommend line sizes no larger than 8. To get big caches try increasing the number of lines.

NOTE: To calculate the size of the cache multiply the number of lines by the line size by the block size (512).

See Also: NUM_LINES

6.0.6 WRITE_RETENTION_TIME

Format: WRITE_RETENTION_TIME=<maximum time (in seconds) to keep changed data>

Default: 300

Example: WRITE_RETENTION_TIME=60

Description:

This controls the MAXIMUM amount of time (in seconds) that data which has been changed can be held in the cache. To disable write retention set this parameter to 0.

Warning: If you want to reset your computer you must wait the smaller of WRITE_RETENTION_T or QUIET_WRITE_TIME (in the case that no IO is occurring, plus a few more seconds) **BEFORE** you can safely reset your computer. Note that if you try to suspend FastCache it may take WRITE_RETENTION_TIME before FastCache responds.

See Also: QUIET_WRITE_TIME

6.0.7 QUIET_WRITE_TIME

Format: QUIET_WRITE_TIME=<time to flush cache in if drive quiet>

Default: 1

Example: QUIET_WRITE_TIME=1

Description:

This parameter only takes effect when write retention is enabled and there is data in the cache that has been changed. If this is the case, then FastCache must at some time write the changed data back to the disk. During disk activity there often comes time when no IO occurs because a program is working something out. This is an ideal time to output changed data, since no other IO request will be impeded. This parameter monitors the time (in seconds) since the last IO request made by the relevant filing system, and if there has been no IO requests for the specified amount of time the changed data is written back to the disk in the hope that there will be no other disk IO requests during that time.

NOTE: Filing systems, like the FastFileSystem, often continue to present IO requests after you finish an activity for a further 1-2 seconds. Hence, a QUIET_WRITE_TIME of 1, will appear to wait three seconds before writing out a request during a quiet spell. This is normal and correct behavior.

Warning: If you want to reset your computer you must wait the smaller of WRITE_RETENTION_T or QUIET_WRITE_TIME (in the case that no IO is occurring, plus a few more seconds) BEFORE you can safely reset your computer.

6.0.8 DONOTWAIT

Format: DONOTAIT

Default: -

Example: DONOTWAIT

Description:

This is actually a WorkBench parameter, and tells WorkBench not to wait for the program to finish. This is recommended for WorkBench starts.

6.0.9 TOOLPRI

Format: TOOLPRI=<tool startup priority>

Default: 0

Example: TOOLPRI=1

Description:

This is actually a WorkBench parameter, and tells WorkBench the startup priority of this tool, when it is placed in the WBStartup drawer.

6.0.10 ?

Format: ?

Default: -

Example: ?

Description

This is a CLI only parameter, and displays help on using FastCache.

7 ARexx

FastCache V1.1 has an ARexx port that enables you to control the cache, as well as get statiscal information about how it is performing. The name of the ARexx port that is created consists of "FastCache.", followed by the device name, followed by a ".", followed by the unit number.

For example, for the device '`gvpscsi.device`', unit 0, the ARexx port name would be '`FastCache.gvpscsi.device.0`'. For the '`scsi.device`', unit 6, the ARexx port name would be '`FastCache.scsi.device.6`'.

7.0.1 ACTIVATE

Format: ACTIVATE

Example: Address 'FastCache.trackdisk.device.0'
ACTIVATE

Description:
This tells FastCache to start caching data. This command is most useful after doing a SUSPEND or ALTERNATE. When FastCache first starts up, it is in this mode. If the cache was previously suspended when this command is received, then memory will be allocated for the cache again.

See also: SUSPEND, ALTERNATE

7.0.2 SUSPEND

Format: SUSPEND

Example: Address 'FastCache.trackdisk.device.0'
SUSPEND

Description:
This tells FastCache to stop caching data. This command is most useful after doing an ACTIVATE or ALTERNATE, or when you are about to do something that may cause memory corruption. This command will also flush the cache, and free up a more of the cache memory.

See also: ACTIVATE, ALTERNATE, FLUSH

7.0.3 ALTERNATE

Format: ALTERNATE

Example: Address 'FastCache.trackdisk.device.0'
ALTERNATE

Description:
If FastCache was suspended, if will become active. If FastCache was active it will become suspended. If the cache is suspended then all data will be flushed first.

See also: ACTIVATE, SUSPEND, FLUSH

7.0.4 FLUSH

Format: FLUSH

Example: Address 'FastCache.trackdisk.device.0'
FLUSH

Description:
Any data that has been written to the cache, but is still waiting to be written out to the device will be forced out with this command. Note that the command may return before the data has actually been flushed (it only registers a request to flush the data). To make certain that the data has been flushed, execute this command twice (the second call will be delayed until the first has been processed).

See also: SUSPEND

7.0.5 STATS

Format: STATS

Example: Address 'FastCache.trackdisk.device.0'
STATS statistics=result
parse var statistics ReadHits ReadMisses WriteHits WriteMisses
TotalReads=ReadHits+ReadMisses
TotalWrites=WriteHits+WriteMisses

Description:
This command asks FastCache how many read hits, read misses, write hits and write misses have occurred since the last time the STATS command was received. Some

explanation is required for how the miss figures are arrived at. To make the numbers have any meaning they are updated in an atomic operation by FastCache (otherwise they would constantly be out of sync with each other). A request is often broken into smaller operations by FastCache. Initially it may be that none of the request is in the cache, and so a miss will occur, which is usually followed by a prefetch. Now when the next part of the request is processed FastCache will find the data in the cache although it wasn't there originally, due to the fact that the data was prefeteched by the time that FastCache wanted the data. The effect of this is that, even though you may know that FastCache can't possibly have all the data, it will often still report a partial hit, which is due to the pre-fetching of the data, before a section of the request required it.

See also: SUSPEND

8 FastCacheStats

FastCacheStats is a program written in C that uses the FastCache ARexx port to display cache statistics.

When started from CLI it takes one parameter in the ToolTypes field of its icon, '`PORT`'. This parameter should be set equal to the name of the name of the ARexx port that FastCache is using (refer to the ARexx section). For example, for gvpscsi.device, unit 0:

```
PORT=FastCache.gvpscsi.device.0
```

When run from CLI, a parameter of '`-PORT`' should be specified, and set equal to the ARexx port that FastCache is using. For example, for gvpscsi.device, unit 0:

```
FastCacheStats -PORT=FastCache.gvpscsi.device.0
```

When started, 18 numeric panels will be displayed and all are reset back to 0. Please refer to the ARexx command STATS for more of an explanation of how some of the numbers are generated.

1. "Total Read Hits" gives the number of bytes that have been supplied from the cache during read requests.
2. "Total Read Misses" gives the number of bytes that were not supplied from the cache (had to be obtained from the drive) during read requests.
3. "Total Read Samples" says how many times a sample by FastCacheStats resulted in information. FastCacheStats queries FastCache every second to see if there has been any activity. If a read request has been presented, then it is counted as a sample. This is done because some of the statistics are not defined when no activity has occurred.
4. "Total Write Hits" gives the number of bytes that were written to the cache, and did not have to be written to disk immediately.
5. "Total Write Misses" gives the number of bytes that had to be written back to disk immediately.
6. "Total Write Samples" says how many times a sample by FastCacheStats resulted in information. FastCacheStats queries FastCache every second to see if there has been any activity. If a write request has been presented, then it is counted as a sample. This is done because some of the statistics are not defined when no activity has occurred.
7. "Instantaneous Read Hits" gives the number of bytes that were supplied to a read request during the last sample taken by FastCacheStats. "Instantaneous Read Misses" gives the number of bytes that had to be read from disk during the last sample.

8. "Instantaneous Read Misses" is the number of bytes that had to be fetched from disk for a read request during the last sample.
9. "Instantaneous Read Hit Ratio" is the ratio of instantaneous read hits to the total instantaneous read operations. The ideal value is 1.000, and the worst is 0.000. Note that if no operations occur, then this value will be held until one does occur, and a new sample takes place.
10. "Average Read Hits" is the average number of bytes supplied from the cache to meet a read request per sample.
11. "Average Read Misses" is the average number of bytes that had to be fetched from disk to meet a read request per sample.
12. "Average Read Hit Ratio" is the ratio of total read hits to total reads. The ideal value is 1.000, and the worst is 0.000.
13. "Instantaneous Write Hits" is the number of bytes that were supplied from cache for a read request during the last sample.
14. "Instantaneous Write Misses" is the number of bytes were unable to be stored directly in cache during a write request during the last sample.
15. "Instantaneous Write Hit Ratio" is the ratio of instantaneous write hits to the total instantaneous write operations. The ideal value is 1.000, and the worst is 0.000. Note that if no operations occur, then this value will be held until one does occur, and a new sample takes place.
16. "Average Write Hits" is the average number of bytes supplied from the cache to meet a write request per sample.
17. "Average Write Misses" is the average number of bytes that were not able to be stored directly in cache during a write request per sample.
18. "Average Write Hit Ratio" is the ratio of total write hits to total writes. The ideal value is 1.000, and the worst is 0.000.

9 Registration

FastCache is released under the concept of Shareware. If after a 3 month trial period you still wish to use FastCache I would appreciate $20 in either NZ, Australian or American dollars. The Australian and US prices (both being $20) may seem artificially higher, but this covers the cost of changing the funds to $NZ. By registering you may not necessarily gain anything. However, if I get enough support I will improve (yes, I can improve it further) FastCache. If this is the case, I will attempt to notify all registered users.

Please send this completed form to:

```
Philip D'Ath
9 Elmwood Crescent,
Pukete,
Hamilton,
NEW ZEALAND.


Name:     _________________________________________________

Address:  _________________________________________________
          _________________________________________________
          _________________________________________________
          _________________________________________________

Hard Drive Controller:   _________________________________

Funds Enclosed:        $ ___________

The most  important thing to add/change to/in FastCache is:
          _____________________________________________________
          _____________________________________________________
          _____________________________________________________


Comments:
          _____________________________________________________
          _____________________________________________________
          _____________________________________________________
          _____________________________________________________
```

10 Index

(Index is nonexistent)

Table of Contents