# Predicting future recalls (revisited)

## Five years on, we look at how the automatic pre-fetching of offline files is faring

**Peter Edwards | Sr Systems & Data Administrator**

DMF Users Group| February 2017

# File access patterns, by name format

When users access offline files, there is frequently a pattern in the order of the implicit recalls presented to DMF.

- Filenames

| Example #1 | Example #2 | Example #3 | Example #4 | Example #5 |
|---|---|---|---|---|
| data.8 | data.20161230 | data.Wed | Data-nov.csv | datos.mié |
| data.10 | data.20161231 | data.Thu | Data-jan.csv | datos.jue |
| data.12 | data.20170101 | data.Fri | Data-mar.csv | datos.vie |
| data.14 | data.20170102 | data.Sat | Data-may.csv | datos.sáb |
| data.16 | data.20170103 | data.Sun | Data-jul.csv | datos.dom |
| ? | ? | ? | ? | ? |

What comes next? (Hint: for last column, locale is “es_ES”)

# File access patterns, by directory

- Directory contents

| Files in directory | Example #1 | Example #2 |
|---|---|---|
| apple.jpg | apple.jpg | grape.bmp |
| banana.jpg | banana.jpg | grape.gif |
| grape.bmp | grape.jpg | grape.jpg |
| grape.gif | ? | ? |
| grape.jpg | | |
| grape.png | | |
| lemon.jpg | | |

What comes next?

- Subdirectories

  Paths like foo/*/bar, with the variable part being the directory, are also supported.

# The algorithm – acquiring activity data

- dm_prefetch extracts in real time the messages in DMF's log relating to implicit recalls

- Get the file's fhandle and convert to a pathname using a database which is generated nightly from the output of dmscanfs (see refs)

- Split path into a directory name and a subdirectory+filename

  Eg: /datastore/asc/edw/dmf_test/dm_prefetch/data2011.nc could be split into /datastore/asc/edw/dmf_test and dm_prefetch/data2011.nc

The use of the directory name allows the processing of a subset of activity, so the actions of multiple users can be distinguished from each other.

# The algorithm – pattern matching

- There are multiple modules, a dozen or more, which each process a specific type of pattern match.

- These are in two general groups:
  - Determined by the presence of files in a directory
  - Determined by the format of the names of files, working with fixed substrings surrounding a variable one. Possibly in conjunction with the names of the subdirectories at the end of the pathname

- Examples:
  - dates: YYYYMMDD,  YYYYMM,  YYYY-MM-DD
  - numeric
  - month & day names: UPPER,  lower,  Mixed
  - wildcards: *something,  something*,  *
  - your own choices

# The algorithm – pattern matching (cont'd)

- Patterns related to substrings in the names of files understand the concept of *stride* (eg: every 3rd member of a sequence) and one or more *locales* where relevant (eg: month names in different European languages).

  Some types of mismatch have limits as to how far they can be extrapolated because of their circular nature (eg: a pattern of month names starting at April can be extrapolated as far as the next March, but not further).

  Non-English locales use ISO8859-15 aka Latin-9, not Unicode or UTF-8.

- For patterns relating to directory contents, the concepts of stride and fixed character positions in names don't apply. But as the only purpose of stride and positions is to predict the next filename(s), and we know the next files in a directory thanks to the equivalent of the "*ls*" command, that doesn't matter.

# The algorithm – choosing best fit

- Each module compares the newest filename against a recent history of others in the same directory, computing a *Degree of Confidence (DoC)*. Each then extrapolates their sequence into the future.

- The DoCs are enhanced slightly if the tapes required for the predicted recalls are currently mounted, or are about to be. (see refs)

- The pathnames predicted by the module with the highest resulting DoC are used to generate a *dmget* command which is run in the background, with the number of pathnames depending on the DoC.

# Is it worth it?

Inevitably, dm_prefetch recalls some files unnecessarily which is a waste, and measuring the cost/benefit is hard.

- An online file which is accessed leaves no trace in the DMF logs
- Neither does a file which is not accessed

Statistical methods were attempted, such as running the script on alternate days and looking at recall durations and other metrics to see if they showed a two day pattern.

This was not detectible, due to the normal variability of the system activity.

# Aha!

Years later, the penny dropped that there is a difference between a recalled file which is accessed shortly after, and one which isn't.

**The file's access time in the inode (atime)**

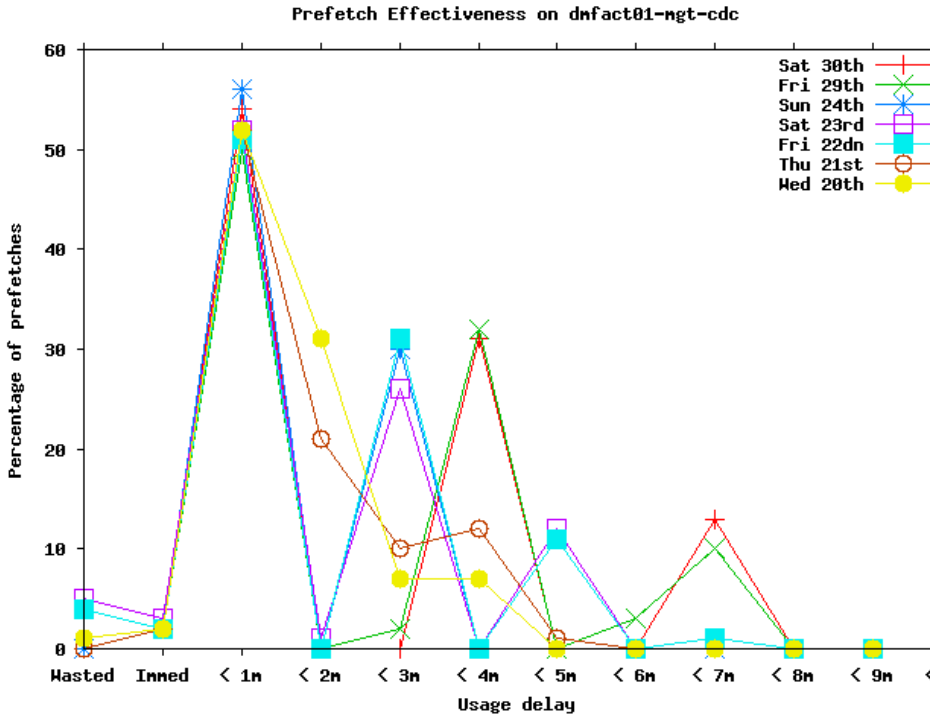
This allows us to see the effectiveness of prefetching.

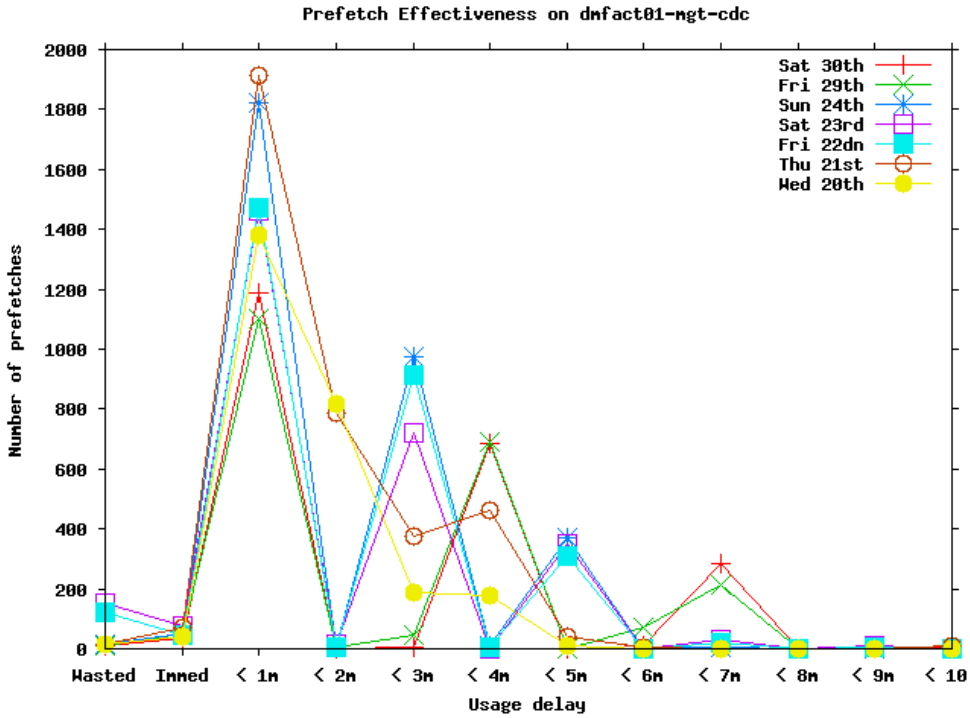
As expected, systems which rely heavily on implicit recalls, such as NFS and CIFS servers show the greatest benefit.

A more traditional system where users have direct access and have been educated for decades about the use of *dmget* benefits less.

# NFS fileserver

- One user, running 5 copies of a script tarring 5 million offline files, totalling 1TB

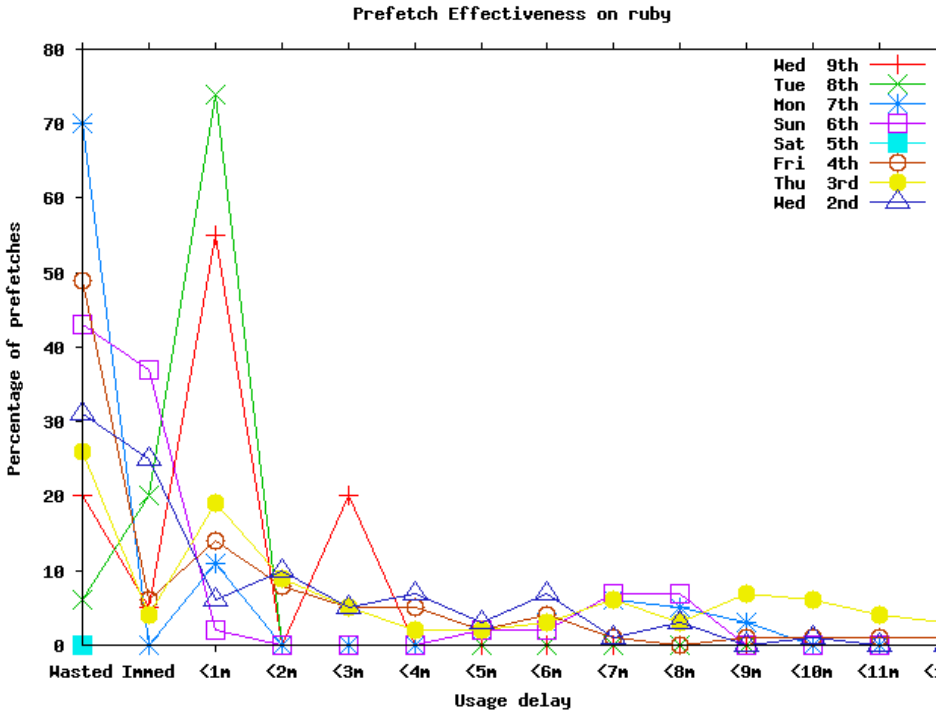


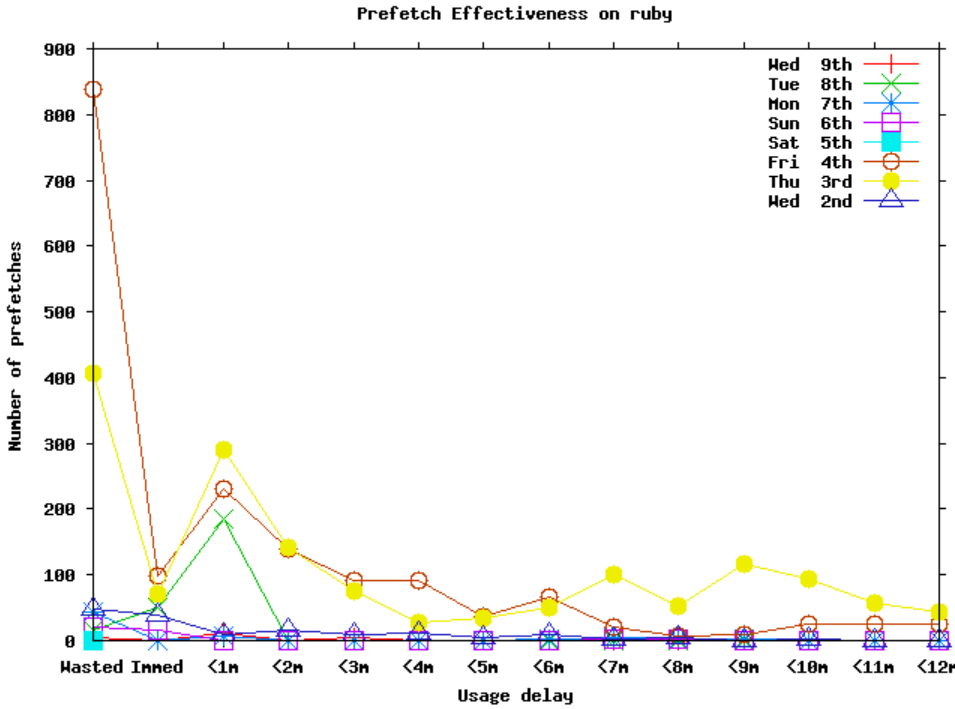




If those 1800 sub-minute prefetches each saved 30 secs, that's 3 hours for each script instance.

# General purpose system

- Dozens of concurrent *dmget*-aware users running batch and interactive work




On Friday, 50% or 420 recalls were wasted, but Tuesday & Wednesday were pretty good.

# References

- George Santayana: "Those who cannot remember the past are condemned to repeat it.“

- Detailed description:

  http://hpc.csiro.au/users/dmfug/Meeting_Nov2011/Presentations/predicting_future_recalls.pdf

- Other presentations

  http://hpc.csiro.au/users/dmfug/Meeting_Dec2012/Presentations/CSIRO_ASC_DMF_tools.pdf (slides 8, 9)

- Cover picture: Google Images

# Thank you

**CSIRO IMT Scientific Computing**
Peter Edwards
Sr Systems & Data Administrator

**t** +61 3 9545 2377
**e** peter.edwards@csiro.au
**w** https://wiki.csiro.au/display/ASC/Scientific+Computing+Homepage