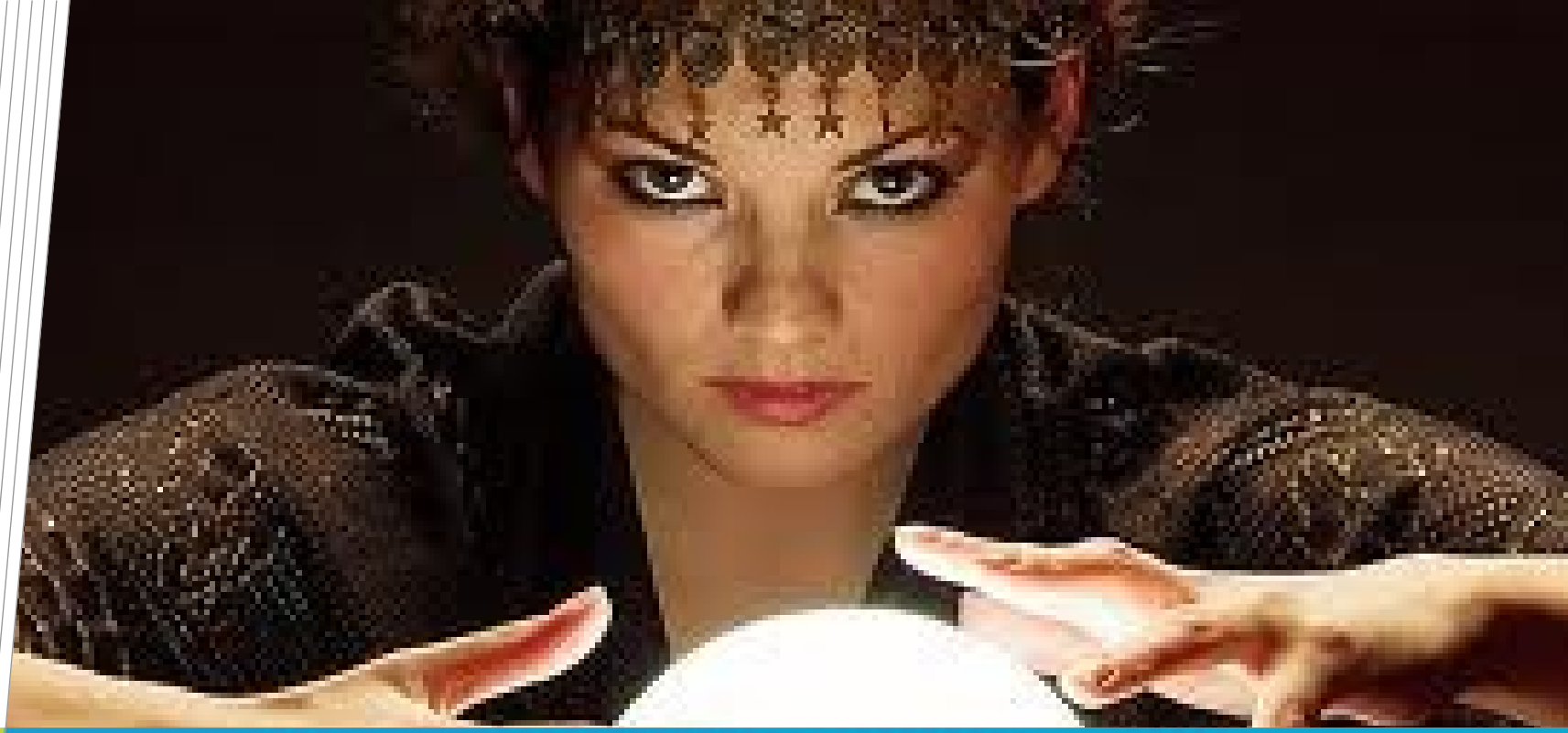
CSIRO



Predicting Future Recalls

Peter Edwards

CSIRO

Overview

A large number of DMF recalls of offline files due to implicit file accesses are not random, but follow structured patterns which make sense to the user. For example:

blah-JAN-blah	blah-050-blah	blah-Wed-blah	blah-20110130-blah
blah-FEB-blah	blah-070-blah	blah-Tue-blah	blah-20110131-blah
blah-MAR-blah	blah-090-blah	blah-Mon-blah	blah-20110201-blah
…	…	…	…

Another common sequence is that determined by the names of files in a directory, which is commonly the result of the use of the "*" wild card character somewhere in a path name.

We will identify such sequences and extrapolate them into the future, allowing us to recall offline files before they are required.

Mismatch types

The currently implemented mismatch types (MMT's) are:

- an ISO-format date of the form yyyy-mm-dd for years 1900-2200
- 8 digit integer, interpreted as a yyyymmdd date for years 1900-2200
- 6 digit integer, interpreted as a yyyymm date for years 1900-2200
- any other unsigned integer
- abbreviated English day & month names in three cases (lower, UPPER & Mixed)
- abbreviated non-English day & month names in the same cases (not particularly relevant to Australia)
- single character English alphabetic in both cases
- name expansions with leading or trailing “*” eg: “*blah” “blah*” “*”

No special delimiters are needed around the mismatches.

Daemon log messages

The perl script *dm_prefetch* runs continuously, tailing the DMF daemon's *dmdlog*, looking for implicit and explicit recalls.

A (trimmed-down) message for an explicit recall (eg: *dmget*) looks like:

```
Req=1618,request=recall,fhandle=0100018b3c26270001e8700020b2a032000,
fn=/datastore/qwerty,pri=0
```

and for an implicit recall (access via the kernel), it looks like:

```
Req=1531,request=krclrea,fhdl=0100018b3c26270001e87000061bd01f00120
```

Implicit recalls only

For the implicit recalls, the log message only contains the fhandle. To find the file's pathname, look it up from a sqlite database generated each night from a *dmscanfs* run.

This is indexed by fhandle, BFID and user-ID, though only the first of these is used here.

If the database lookup fails (eg: a recently created or renamed file), this recall is ignored.

If running in “check” mode (see later) and this recall was the result of a successful previous prediction, the details are recorded and no further action is taken.

Break up the pathname

Split the path into two parts, a directory- and file-name.

Eg: The file

/datastore/asc/edw192/dmf_test/dm_prefetch/data2011.nc

can be split on any slash (config option; default is 2nd last one).

We will refer to the part before the chosen slash as the "*directory name*" and that after as the "*filename*", slightly distorting the conventional meanings.

The filename is saved in a "*history buffer*" named after the directory name.

For explicit recalls, there is no further processing.

Reducing the Size of the Problem

Each new implicit recall is tested by all of the MMTs, most of which will fail to find their particular pattern.

To reduce the problem size, MMTs create their own temporary short-list from the relevant history buffer:

- The Mismatch_Suffix MMT makes a short-list of files in the same bottom-level directory whose names have the same ending (ie: files matching the “*blah” pattern).
- The Mismatch_Prefix MMT's short-list is all the files in the same bottom-level directory.
- The other MMTs (“Structured MMTs”) choose the subset of filenames which are of the same length as that of the new recall.

Mismatch identification

Structured MMTs compare the just-derived filename to the previous one in their short-list, looking for a mismatching sequence of characters. An example of Mismatch_Numeric's processing is:

```
C160/C160b_echam5_A2-ct-uf-m20210605.nc
C160/C160b_echam5_A2-ct-uf-m20210603.nc
                                    ^-- single digit mismatch,
                                        with a stride of -2
```

Some types lower the start position and/or raise the end position of the textual mismatch to get a result. So Mismatch_yyyymmdd processes the same filenames as:

```
C160/C160b_echam5_A2-ct-uf-m20210605.nc
C160/C160b_echam5_A2-ct-uf-m20210603.nc
                               ^^^^^^^^-- YYYYMMDD mismatch,
                                          with a stride of -2
```

Mismatch identification (cont'd)

Similarly, Mismatch_alpha sees:

```
C160b_echam5_A2-ct-uf-mlJun/input.nc
C160b_echam5_A2-ct-uf-mlJul/input.nc
                          ^----- single alpha mismatch,
                                 with a stride of -2
```

but for Mismatch_Month:

```
C160b_echam5_A2-ct-uf-mlJun/input.nc
C160b_echam5_A2-ct-uf-mlJul/input.nc
                        ^^^----- English month mismatch,
                                 with a stride of +1
```

For filename expansions, the concepts of “stride” and fixed character positions of mismatches don't apply. But as the only purpose of stride and positions is to predict the next filename, and we know the next files in a directory thanks to the “*ls*” command, it doesn't matter.

Examine history & extrapolate forwards

Each MMT which finds its own pattern then goes back through the previous files in its short-list, to determine how long this pattern has been going on for. The longer the run is, the higher the degree of confidence ("DoC") that any prediction(s) will be valid.

The MMTs also extrapolate forwards, to see how far the identified pattern could continue into the future; a longer potential run also increases the DoC but at a lesser rate. This provides a tie-breaking effect.

Some types of mismatch have limits as to how far they can be extrapolated (eg: alphabetics and months are constrained to the ranges "a" - "z" and January - December respectively).

Making Predictions

After all MMTs have finished, the one with the highest DoC is used to generate predictions. (MMTs have a defined and unique priority, which is used to break any ties due to identical DoC values.)

This winning MMT repeats its forward extrapolations, making predictions about pathnames of future recalls.

These may be used to generate *dmget* commands which are run in the background, or if in “check” mode just remembered to see if they are subsequently recalled.

(Doing both at the same time is not possible; think about it!)

How many to recall?

For each predicted file in turn, DoC is decremented by a small number if the tape containing its primary copy is already mounted on a drive, or if a previous prediction in the same batch is already causing that tape to be mounted.

Otherwise, a fresh mount will be needed, so DoC is decremented by a larger number to reflect the higher cost.

The filename to VSN mapping required can be done using an augmented version of the database mentioned above, and/or by calling *dmcatadm* via a setuid wrapper program.

(The database is better if you can afford it.)

Checking the files to be recalled

If:

- the predicted file exists
- the DoC is still positive
- the file is not too big
- the file hasn't been predicted before
- the file is in OFL or PAR state

then the file is recalled or remembered as appropriate.

This step is repeated for subsequent files, until the DoC is exhausted or a non-existent file is predicted.

Notes

- The possibility of a recall from a DCM is ignored. Use of COPAN MAID should “just work”.
- The script has been tested with OpenVault, but mostly with TMF which provides better identification of mounted tapes.
- The script does not yet attempt to move to the new dmdlog when it rolls over at midnight. Instead it terminates and should be restarted by cron a couple of minutes later. This also means that stale history buffers are cleaned up once a day.
- The month and day name MMTs can have multiple instances with different ISO8859-15 locales being specified; this has been tested with C, fr_FR and de_DE. UTF-8 is not supported due to its multi-byte nature.

Notes (cont'd)

- MMTs are implemented as perl modules (ie: object classes), allowing easy addition or subtraction of tests with single-line changes to the core code.
- The requirement of structured MMTs for the filenames to be of the same length allows for those of different lengths to tracked in the same history buffer without interference, thereby giving a degree of multithreading. Similarly for Mismatch_Suffix.
- Thanks to Google Images for the cover image.
- George Santayana: "Those who cannot remember the past are condemned to repeat it."

Does it work?

```
Recalls trapped: 1950 implicit and 8856 explicit
Implicit recalls skipped due to errors: 0
Number of dm_prefetch tape mounts: 0 new and 173 pre-existing
1497 predictions (94%) came true, with an average delay of 3962 secs
24 (2%) were pre-empted by users, and 65 (4%) may have been wasted
          Predictions 520      New Mounts 73   *suffix filename expansion
          Predictions 24       New Mounts 4    * or prefix* filename expansion
          Predictions 0        New Mounts 0    numeric (YYYYMMDD)
          Predictions 994      New Mounts 95   numeric (YYYYMM)
          Predictions 0        New Mounts 0    ISO date (YYYY-MM-DD)
          Predictions 48       New Mounts 1    numeric (arbitrary)
          Predictions 0        New Mounts 0    upper case MONTH (C)
          Predictions 0        New Mounts 0    lower case month (C)
          Predictions 0        New Mounts 0    camel case Month (C)
          Predictions 0        New Mounts 0    upper case DAY (C)
          Predictions 0        New Mounts 0    lower case day (C)
          Predictions 0        New Mounts 0    camel case Day (C)
          Predictions 0        New Mounts 0    single lower case alphabetic
          Predictions 0        New Mounts 0    single upper case ALPHABETIC
          Predictions 1586     New Mounts 173  TOTALS
Number of unfulfilled or in-progress predictions: 65
Number of history buffers: 175
```

Are you sure?

The effectiveness of dm_prefetch depends heavily on user work patterns, which makes it impossible to simulate.

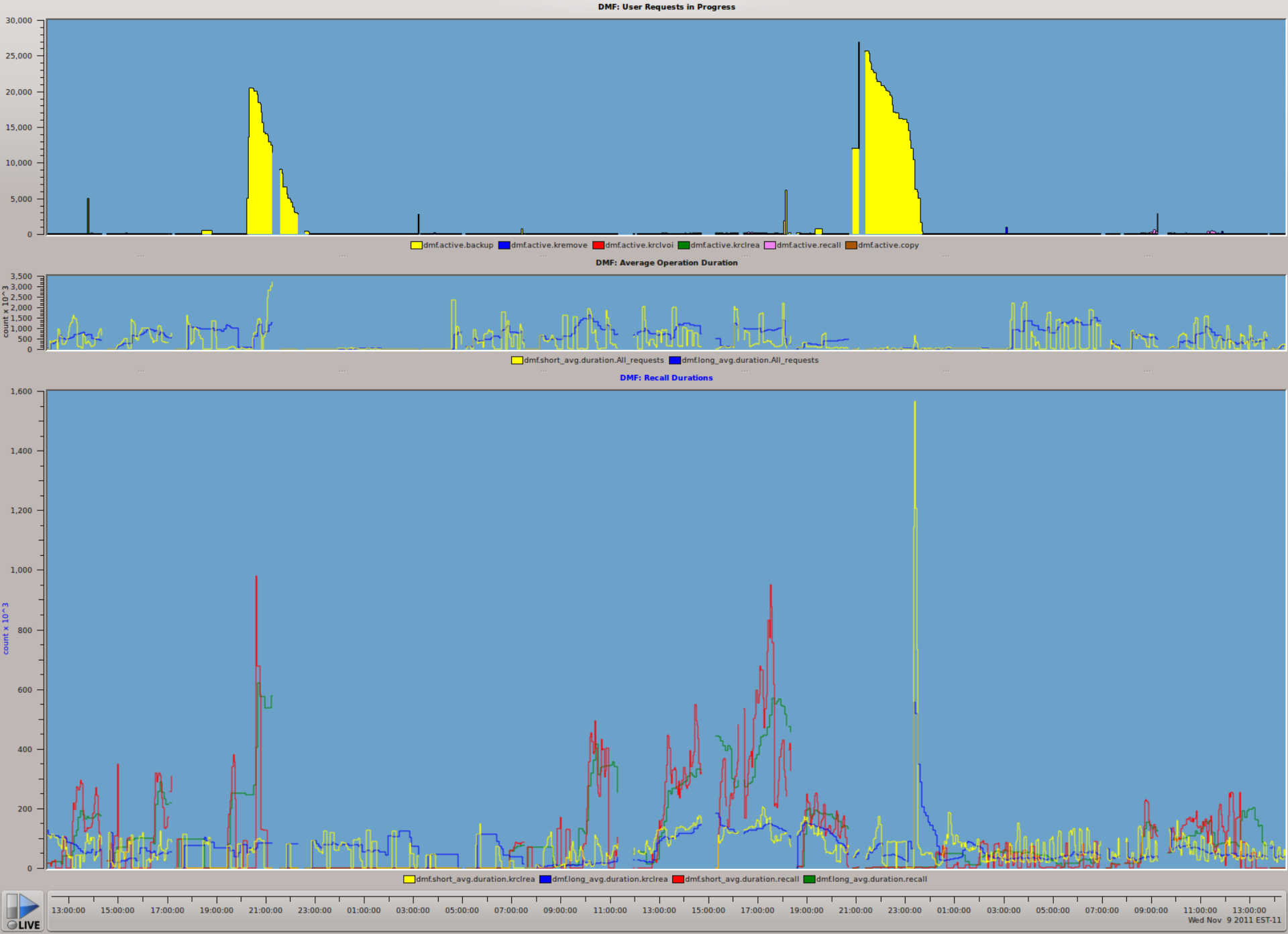
“Check mode” was fine for debugging but is not so useful for tuning.

We need an independent check to measure average recall times, for both recall types.

PCP can do this via the “dmf” PMDA (metrics such as *dmf.short_avg.duration.krclrea* and similar). This is currently broken, but can be kludged into life though it's a little unstable.

PCP Charts <2>

File Edit Record Options Help



Is it worth it?

Maybe...

If you have succeeded in training your users to do their own pre-fetching with *dmget*, then you probably don't need *dm_prefetch*.

But if you're tired of beating your head against that particular wall, or if your users access files via scp, Samba or Apache where *dmget* (or equivalent) is unavailable, then it may improve average access times.

If in your environment, users make significant use of structured naming systems and/or heavily use leading or trailing wild characters (or GUI equivalent), then it's more likely to be useful.

Advanced Scientific Computing

Peter Edwards
Systems Support Manager

Email: peter.edwards@csiro.au

www.csiro.au

Thank you

Contact Us
Phone: 1300 363 400 or +61 3 9545 2176
Email: enquiries@csiro.au Web: www.csiro.au

CSIRO