MMAP (Multi-core Method for Analysis Pipelines)

Documentation version 1.0.0, Sep 2020

SYNOPSIS

```
use MyMMAP qw(:DEFAULT);

#initilize
mnew($BASEPROJ,$NCPUS,$EFLOG,$WORKINGUSER,STARTTIME);

#retrieve data
readinDirectoryStructure($BASEPROJ,"e:data",1,QUIET);

#run
for(my $i = 0; $i < @NPROGS ; $i++) {
    startMMAP($NPROGS[$i]);
}

#reset before next set of tasks
resetMMAP();
```

BRIEF DESCRIPTION

Note: This documentation is accomplished with the help of the LMAP_S archive that contains additional modules that constitute dependencies of MMAP (i.e., *MyMMAP.pm*). One is the *MyISWU.pm* where the integrated software settings are defined, such as command-line default settings and naming conventions for specific algorithms (i.e., software with alternate settings). The second is *MyUtil.pm*, which provides generic settings of the implemented pipeline. Other module worth referencing is the *MyNotify.pm* (optional) that provides email notification capabilities upon termination of MMAP, whereas the email address is also useful to determine the ownership of MMAP/pipeline instances. The main *lmap-s.pl* application also exhibits how MMAP methods are employed (see "SYNOPSIS"). These are referenced throughout this documentation to help the reader understand few of the *MyMMAP.pm* functions/arguments. Hereafter, we refer to them by "LMAP_Slib/<module.pm>".

MyMMAP.pm module here described, enables and helps the developer to implement high-throughput pipelines for multi-core workstations. Hereby, it enables the scheduling of software tasks (with different data files) and their monitoring. With current functioning it provides systematic executions of different integrated software or algorithms.

For this monitoring *MyMMAP.pm* provides a user interface (UI) and interaction with 4 screens. During execution it shows the tasks executions in progress (screen 1), scheduled/planned and terminated (screen 2). Additionally, on exit, provides a process manager screen (screen 3) enabling the end user to terminate specific running tasks. These consists of the MMAP instance's own tasks (same ID) and other MMAP instances that may be found running in the same workstation. This can be done by choosing the task with system PID (each task with different ID) or by choosing a group of tasks with the same MMAP ID (MPID). In the latter case, it enables the termination of the group of tasks with associated MPID, whereas it is also possible to terminate tasks with differing MPID (i.e., from other MPID instances). Another informative screen (screen 4) is provided at the end of each MMAP instance, to show a resume of the terminated tasks. This screen information is also used for tasks status logging as indicated in the ("2) mnew()") method.

It provides two constructor methods, one for initializing monitoring UI colors ("1) cnew()") and another for the MMAP main settings ("2) mnew()"). Additionally, it contains 49 methods/functions, of which only 9 are exported and here documented. The remaining are internally differentiated from the exported ones, by the underscore character that is found as first character of the method name.

REQUIREMENTS

1. CPAN modules

Term::ANSIColor qw(:constants);
Sys::Info;
Sys::Info::Constants qw(:device_cpu);
threads;
threads::shared;
Thread::Semaphore;
sigtrap qw(handler _do_rexit INT QUIT TERM TSTP);
Term::ReadKey;
Cwd;
File::Spec::Functions qw(:ALL);
File::Copy;

2. UNIX programs

screen(1)
whereis(1)
ps(1)
tput(1)
printf(1)

CONSTANTS

MMAP provides constants that can help to quickly control few aspects, among which we highlight the following:

- COLORDFLT: change default behavior of color setting.
- SHOWMAX: the number of items/tasks shown in screen 2 - Task Status.
- DEBUGMODE: enable or disable debug mode (i.e., during implementation).
- VERBOSE: turn on/off MMAP UI monitoring information/screens.
- MMAPSTATUSFLNM: change default identity of tasks log file.

METHODS

Here, we present descriptions for the exported functions only.

---

1) cnew()

Constructor method to initialize the use of (ANSI) colors in MMAP screens. The returned table provides colors that can be employed for other purposes.

If not used, MMAP by default enables colors in "2) mnew()" with constant COLORDFLT.

Please see also the wantColor() function from LMAP_Slib/MyUtil.pm module. e.g., cnew(wantColor());

Arguments (1)

1. color on/off:

enable/disable the user of color (int/boolean);

Return (1)

1. table of colors:

hash table of default colors.

---

2) mnew()

Constructor to initialize MMAP method. Additionally, it performs detection of the screen version installed to enable better adaptation to the UI interface.

Arguments (5)

1. input directory:

a directory from where to derive necessary information (string);

it is designed to be the third level of a directory structure, whereas the 2nd level, is the project identification and the 1st level, the project location. e.g., BASEDIR/PROJID/LMAP_SPool

2. number of CPUs:

the definition of the maximum number of cores utilized - one per task (int); either given by user, or else, by automatic estimation;

3. final tasks status logging:

flag the tasks logging process on/off (int/boolean);

4. instance owner:

an optional user identification (e.g., email address - see also LMAP_Slib/MyNotify.pm module) (string);

5. instance start time:

the pipeline start data/time; if not given, one is set by default (string).

Return (0)

Void.

---

3) getNCPUS()

Get the number of CPUs, threshold.

Arguments (0)

Void.

Return (1)

1. the number of CPUS (int):

current number of CPUs (int); this might have been modified by user interation i.e., increasing/decreasing the number of tasks.

---

4) resetMMAP()

Reset MMAP data struture of scheduled files, after the previous software tasks scheduling.

Arguments (0)

Void.

Return (0)

Void.

---

5) setMMAP()

Set or define MMAP scheduled files.

Arguments (1)

1. files table:

list of new files to schedule (string).

Return (0)

Void.

---

6) getMMAP()

Get the MMAP scheduled files.

Arguments (0)

Void.

Return (1)

1. files table:

list of new files to schedule (string).

---

7) getMMAPID()

Get the created MMAP ID - MPID for the current instance.

Arguments (0)

Void.

Return (1)

1. MPID:

2 letter and 4 digits code. e.g., UP1234.

---

8) startMMAP()

Main MMAP method. It conducts the scheduling and initialization of monitoring procedures.

Arguments (1)

1. software name:

provide the software name/identification/algorithm to use for planned files execution.

Return (0)

Void.

---

9) finalize()

After MMAP method terminates, clean the threads list. Kill running, join joinable threads and reset the terminal settings.

Arguments (0)

Void.

Return (0)

Void.

---

10) unmarkFT()

Unmark processed data files.

Arguments (1)

1. files to unmark:

list of files to unmark from control table (string).

Return (0)

Void.

---

11) readinDirectoryStructure()

Perform a directory scan (BFS-based), to collect all available input files.

A control tag helps to identify the files at the start (s), end (e), or with full (f) text locations.

Arguments (4)

1. target directory:

from where to make the file sweep (string);

2. file control tag:

composed of two parts (string), the match indicator and the textual identification; three match indicators are possible: s (start), e (end), f (full). E.g., 'e:txt' will collect all text files (files with txt extension).

3. recursive:

indicate if the file sweep is recursive or not (int/boolean). If not, it only searches the 1st level directory.

4. verbose:

(optional) flag to print to STDERR several messages information (int/boolen).

Return (1)

1. scheduled files:

list of files for tasks scheduling (string).

CITATION

In case this work is useful, please (also) cite both: LMAP (2016) and LMAP_S (2019)

SEE ALSO

2. 2019 LMAP_S Homepage: https://lmap-s.sourceforge.io

Article doi: 10.1186/s12859-019-3292-5

1. 2016 LMAP Homepage: http://lmapaml.sourceforge.net

Article doi: 10.1186/s12859-016-1204-5

AUTHOR

For any comments/suggestions/questions please contact:

Written by Emanuel Maldonado <emaldonado@ciimar.up.pt>.

LICENSE