Profiling pMatlab and MatlabMPI Applications Using the MATLAB 7 Profiler

Hahn Kim, Albert Reuther
{hgk, reuther}@ll.mit.edu
MIT Lincoln Laboratory, Lexington, MA 02420

1 Introduction

The most common reason to write parallel programs is to improve performance. Consequently, the ability to measure the runtime performance of a parallel program is of primary importance. One of the most common ways to measure performance of serial programs is to use a profiler. MATLAB© contains a built-in profiler, which can also be used to profile parallel MATLAB programs written with pMatlab and MatlabMPI. The profiler can be run from the MATLAB command prompt or a GUI launched from the MATLAB "Start" menu. This paper describes how to use profiler's command line interface with pMatlab and MatlabMPI, since the GUI simply interfaces with the command line interface. This document assumes the user is using MATLAB 7.

We assume that the reader is already familiar with pMatlab or MatlabMPI. For more information about pMatlab or MatlabMPI, see [1][2][3][4][5]. For more details on the profiler, please refer to the help file available from MATLAB by running "`help profile`" at the MATLAB command prompt.

2 Using the Profiler

2.1 Profiler Results

The MATLAB profiler records the following information for each function/script:

- Execution time
- Number of calls
- Parent/child functions
- Code line hit count
- Code line execution time

2.2 Turning the Profiler On/Off

The profiler is controlled using the `profile` command. To start the profiler, simply insert "`profile on`" at the beginning of the section of code you wish to profile:

```
% Start profiling
profile on;
```

To stop recording profile information, insert "`profile off`" at the end of the section of the code you wish to profile:

```
% Stop profiling
profile off;
```

To resume profiling after the profiler has been stopped without clearing previously recorded information, insert "`profile resume`" where you wish to resume profiling. This allows you to profile non-contiguous sections of code:

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government

```
% Resume profiling
profile resume;
```

Due to the SPMD nature pMatlab and MatlabMPI, each rank is an independent MATLAB process. Consequently, the `profile` commands are run by each rank just as in a regular MATLAB process. Each rank profiles itself and only itself, independent of all other ranks in the pMatlab or MatlabMPI job.

2.3 Saving Profiler Results

In a serial MATLAB program, profile results can be returned to the user's workspace so that they can be viewed. However, extra effort is required to view profile data for a pMatlab and MatlabMPI job since profile data generated by multiple MATLAB processes. Fortunately, the profiler returns the profile results as a variable, allowing remote pMatlab and MatlabMPI processes to save their profile data to disk.

The profiler can generate two data structures that contain the current status of the profiler and the profiler results using the "`profile('status')`" and "`profile('info')`" commands, respectively:

```
% Get current status of profiler
profile_status = profile('status');

% Get profile results
profile_info   = profile('info');
```

These data structures can be saved to a file, specified by the `filename` parameter, using MATLAB's MAT-file format.

```
% Save profile_status and profile_info to a .mat file
% so it can be viewed later by loading the file and using
% profsave.
save(filename, 'profile_status', 'profile_info');
```

The profile information saved in the `profile_info` variable is used to generate a report, using the `profsave` command. The report is saved to disk as a set of HTML files in a directory specified with the `dirname` parameter. The report summarizes the profile results described in Section 2.1:

```
% Generate the HTML report for local rank
profsave(profile_info, dirname);
```

Since all ranks will save `profile_status` and `profile_info` and the HTML report to disk, each rank must use unique values for `filename` and `dirname`. The simplest way to accomplish this is to use the rank of each MATLAB process in the file and directory names:

```
% Create a unique filename
profile_profname = ['profile_' num2str(my_rank)];

% Save the profile_status and profile_info to a .mat file
% so it can be viewed later by loading the file and using
% profsave.
save(profile_profname, 'profile_status', 'profile_info');

% Generate HTML report for the local rank
profsave(profile_info, profile_profname);
```

The code presented in this section will generate the following:

1. `profile_<rank>.mat` – Contains `profile_status` and `profile_info`.
2. `profile_<rank>` – Directory containing the HTML files for the profile report.
3. `profile_<rank>/file0.html`, `profile_<rank>/file1.html`, `profile_<rank>/file2.html`, `profile_<rank>/file3.html`… – Files containing the profile report. Each file contains profile information for a single function in the program. `file0.html` represents the top level function; this is the file the user should open first.

3 Viewing Profile Results

To view these resulting files, open the `profile_<rank>/file0.html` file in your favorite web browser for each rank:

- `…/profile_0/file0.html`
- `…/profile_1/file0.html`
- `…/profile_2/file0.html`
- `…/profile_3/file0.html`
- Etc.

Figure 1 shows the results of profiling the `pBlurimage.m` example program contained in the examples directory in pMatlab. Note a separate browser window is opened for the profile report generated by each rank.

4 Template Code

Figure 2 contains template code that summarizes the code presented in this document. These templates can be used with to profile any MatlabMPI or pMatlab program.

5 Determining MPI_Recv Wait Times

Profiling the amount of time that a pMatlab or MatlabMPI process spends waiting to receive messages can expose inefficiencies in the program; time spent waiting for messages could potentially be better spent on computation. In the `profile_<rank>/file0.html` file, select the link for the `MPI_Recv` function and see how much time has been spent in the receive wait loop, including the `MatMPI_Sleep` function.

6 MATLAB Settings

The MATLAB profiler requires Java, which means every process in a pMatlab or MatlabMPI job must have Java enabled. Consequently, MATLAB can not be launched using the `-nojvm` flag on either the user's computer or on remote machines. In fact, Mathworks no longer support MATLAB running in `nojvm` mode. When launching MATLAB on your computer, do not use the `-nojvm` flag; unless you launch directly from your operating system's command prompt, it is unlikely that this will happen. To ensure that remote MATLAB processes launch with Java enabled, make sure that the `machine_db_settings.matlab_command` setting in `MatMPI_Comm_settings.m` does not contain `–nojvm`; it is okay to ignore the `DON'T CHANGE` comment.

Note that the `profsave` command automatically attempts to launch a web browser to display the HTML report. If MATLAB is configured correctly on the user's desktop machine, then `profsave` will launch a web browser. However, remote MATLAB processes will not be able to launch a web browser since they are not running the MATLAB GUI. Instead, the following warning message will be displayed in the `MatMPI/*.out` files for each remote rank:

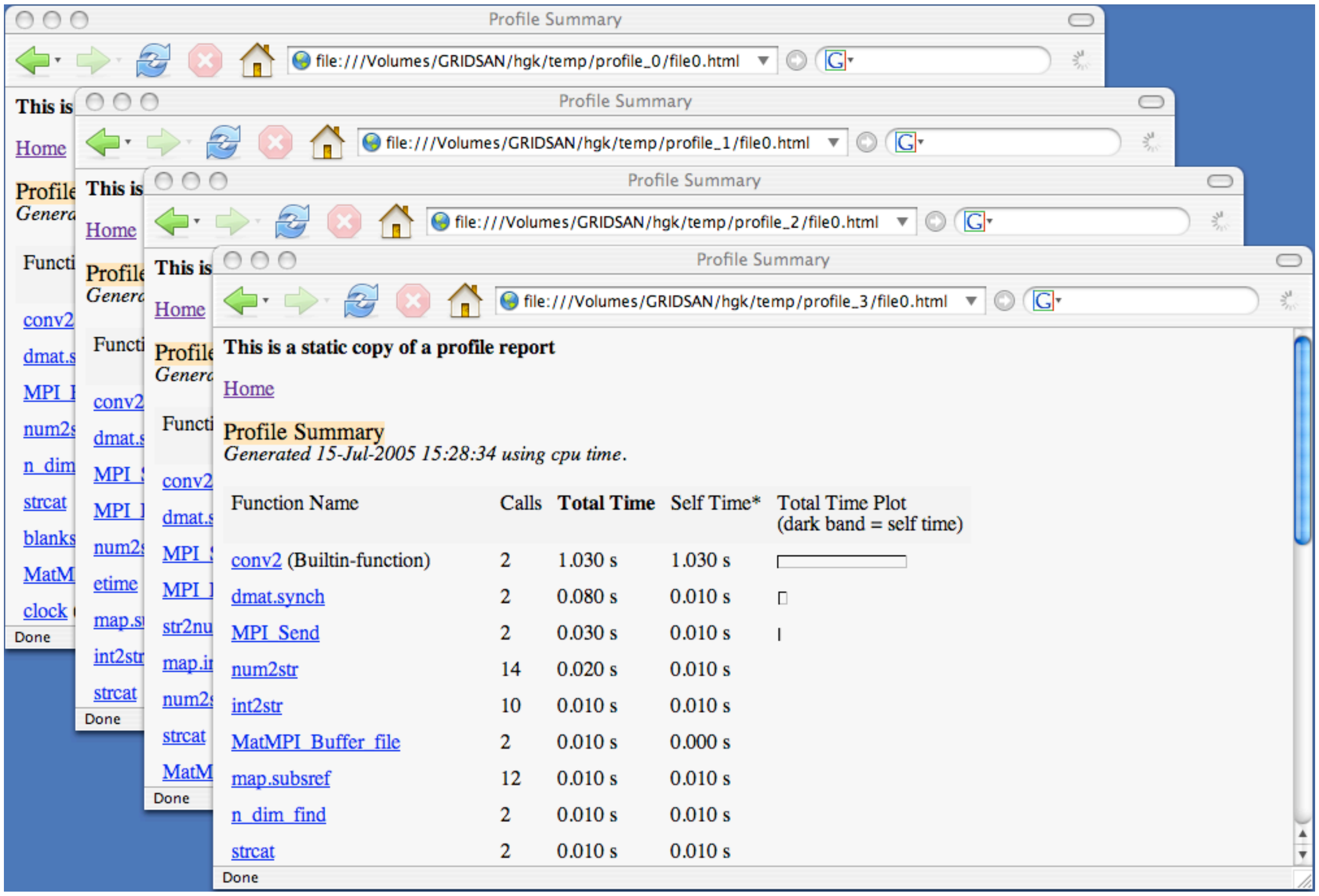
```
    which: no netscape in (/usr/local/bin:/bin:/usr/bin)

    To learn how to configure your Web browser type 'help docopt'
```

This is not an error but, unfortunately, there is no way to disable launching the web browser in the `profsave` command. Simply ignore this message.

7 Conclusion

Using the MATLAB profile suite in MatlabMPI applications will make performance improvements easier to determine, implement, and test.



Profile Summary
file:///Volumes/GRIDSAN/hgk/temp/profile_0/file0.html
file:///Volumes/GRIDSAN/hgk/temp/profile_1/file0.html
file:///Volumes/GRIDSAN/hgk/temp/profile_2/file0.html
file:///Volumes/GRIDSAN/hgk/temp/profile_3/file0.html
Done

This is a static copy of a profile report

Home

Profile Summary
Generated 15-Jul-2005 15:28:34 using cpu time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
conv2 (Builtin-function)	2	1.030 s	1.030 s	
dmat.synch	2	0.080 s	0.010 s	
MPI_Send	2	0.030 s	0.010 s	
num2str	14	0.020 s	0.010 s	
int2str	10	0.010 s	0.010 s	
MatMPI_Buffer_file	2	0.010 s	0.000 s	
map.subsref	12	0.010 s	0.010 s	
n_dim_find	2	0.010 s	0.010 s	
strcat	2	0.010 s	0.010 s	

Figure 1 – Profile reports generated by each pMatlab or MatlabMPI rank should be opened in a separate web browser window.

MatlabMPI

```
% Initialize MatlabMPI
MPI_Init;

% Get local rank
my_rank = MPI_Comm_rank;

% Start the profiler
profile on;

% ...
% CODE TO PROFILE
% ...

% Stop the profiler
profile off;

% ...
% CODE NOT TO PROFILE
% ...

% Resume profiling
profile resume;

% ...
% MORE CODE TO PROFILE
% ...

profile off;

% Get current status of profiler
profile_status = profile('status');

% Get profile results
profile_info  = profile('info');

% Create a unique filename
profile_profname = ['profile_' ...
                    num2str(my_rank)];

% Save profile_status and profile_info to
% a .mat file so it can be viewed later
% by loading the file and using
% profsave.
save(profile_profname, ...
     'profile_status', ...
     'profile_info');

% Generate HTML report for the local rank
profsave(profile_info, ...
         profile_profname);

% Finalize MatlabMPI
MPI_Finalize;
```

pMatlab

```
% Initialize pMatlab
pMatlab_Init;

% Get local rank
my_rank = pMATLAB.my_rank;

% Start the profiler
profile on;

% ...
% CODE TO PROFILE
% ...

% Stop the profiler
profile off;

% ...
% CODE NOT TO PROFILE
% ...

% Resume profiling
profile resume;

% ...
% MORE CODE TO PROFILE
% ...

profile off;

% Get current status of profiler
profile_status = profile('status');

% Get profile results
profile_info = profile('info');

% Create a unique filename
profile_profname = ['profile_' ...
                    num2str(my_rank)];

% Save profile_status and profile_info to
% a .mat file so it can be viewed later
% by loading the file and using
% profsave.
save(profile_profname, ...
     'profile_status', ...
     'profile_info');

% Generate HTML report for the local rank
profsave(profile_info, ...
         profile_profname);

% Finalize pMatlab
pMatlab_Finalize;
```

Figure 2 – Templates for adding profiler code to MatlabMPI and pMatlab programs.

8 References

[1] J. Kepner. “Parallel Programing with MatlabMPI.” 5th High Performance Embedded Computing workshop (HPEC 2001), September 25-27, 2001, MIT Lincoln Laboratory, Lexington, MA

[2] J. Kepner and N. Travinin. “Parallel Matlab: The Next Generation.” 7th High Performance Embedded Computing workshop (HPEC 2003), September 23-25, 2003, MIT Lincoln Laboratory, Lexington, MA.

[3] J. Kepner, A. Reuther, H. Kim. “Parallel Programming in Matlab Tutorial.” MIT Lincoln Laboratory.

[4] H. Kim, J. Mullen. “Introduction to Parallel Programming and pMatlab.” MIT Lincoln Laboratory.

[5] H. Kim, N. Travinin. “pMatlab v0.7 Function Reference.” MIT Lincoln Laboratory.