COMP 696: Advanced Parallel Computing
Note : Profling Parallel Code

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Due: 09/14/15
Posted: 09/14/15
Updated: 09/14/15

Table of Contents

Profling Results using TAU on tuckoo

- Use X11 forwarding will allow you to vizualize data or run graphical applications.
- log onto tuckoo using X11 *SSH*
  `http://www-rohan.sdsu.edu/faculty/mthomas/courses/f15/comp696/topics/tools/comp696-ssh-xterm.pdf`
- see `http://www-rohan.sdsu.edu/faculty/mthomas/courses/f15/comp696/topics/tools/comp696-ssh-xterm.pdf`

Profling Results using TAU on tuckoo

- Use X11 forwarding will allow you to vizualize data or run graphical applications.
- log onto tuckoo using X11 *SSH*
  `http://www-rohan.sdsu.edu/faculty/mthomas/courses/f15/comp696/topics/tools/comp696-ssh-xterm.pdf`
- see `http://www-rohan.sdsu.edu/faculty/mthomas/courses/f15/comp696/topics/tools/comp696-ssh-xterm.pdf`

"Wave" Generator Using Matrix-Matrix Multiplication of Airy Disk Function

- For this assignment you will use *matrix-matrix* multiplication (not the Hadamard product, see above) to calculate a matrix product using the matrices defined by the Airy disc/Fraunhaufer Diffraction pattern described above:

  $$F(x,y) = f * cos^2(r) * e^{(-gr^2)}$$

  where (x, y) are cartesian coordinates, $r = \sqrt{x^2 + y^2}$, and f and g are constants that control the shape of the waves.
- This particular form allows us to factor the function into a matrix multiplication equation of the form:

  $$F(x,y) = A * B$$

  where:

  $$A = f * cos^2\left(\sqrt{x^2 + y^2}\right) \quad \text{and} \quad B = e^{-g\left(x^2+y^2\right)}$$

- Notes:
  1. (x, y) are the grid cartesian coordinates, and f and g are constants that control the shape of the waves.
  2. The results will differ from those obtained using Hadamard matrix multiplication.

Matlab code: Mat-Mult to generate Waves

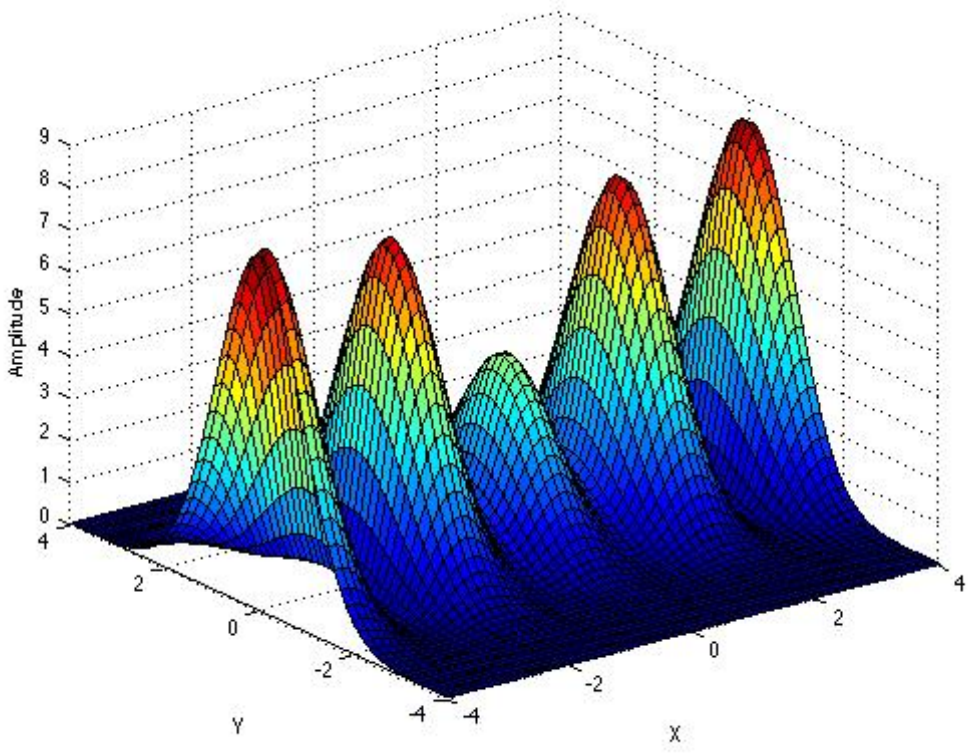
```
%
% Wave Function Test Case: Matlab code
%   Calculates modified bessel function
%       using mat-mat multiplication:  C=A.*B
%
%   By Mary Thomas (March, 2014)
%       updated: March, 2015
%       Created: March, 2014
%
%  scaling factors affect max amplitude
%  number of wavelengths
%
%
clear all;
ni=64;
fprintf('init wave function test case\n');
%%
% f: max amplitude
f=1.0;
f=0.5;
%f=.25;
%f=.1;

% g: inverse wavelength
g=.1;
g=0.25;
%g=.5;
%g=.75;
%g=1.;

% s:  number of wavelengths in hat ~ s+1
s=1;
%s=2;
%s=3;
s=4;
%s=6;
```

```
x = linspace(-s , s ,ni);
y = linspace(-s , s ,ni);

fprintf('init x,y\n');

%% serial matrix-matrix multiplication
%
C=zeros(ni,ni);
A=zeros(ni,ni);
B=zeros(ni,ni);

for i=1:ni
    for j=1:ni
        r(i,j)=sqrt(x(i)^2 + y(j)^2)*pi*f;  %convert to ra

        xpts(i,j) = x(i); % these are for plotting
        ypts(i,j) = y(j);

        A(i,j) = f*(cos(r(i,j)))^2;

        B(i,j) =  exp(-g*r(i,j)^2);
    end
end

fprintf('calc A,B ok\n');
%%
% using r(i,k) produces waves
for i=1:ni
    for j=1:ni
        for k=1:ni;
            C(i,j) = C(i,j) + A(i,k) * B(k,j);
        end
    end
end
fprintf('calc C ok\n');
```

Matlab code: Mat-Mult to generate Waves

```
%% plot 3D surface node mesh -- waves
figure;
surf(xpts,ypts,C);
title(['Bessel Waves -- Calc C(i,j)  Ni=',num2str(ni),', S=',num2str(s),',  F=',num2str(f),',  G=',num2str(g)]);
xlabel('X');ylabel('Y');zlabel('Amplitude');
grid on;

%% plot 3D surface node mesh -- waves
D=A*B;
figure;
surf(xpts,ypts,D);
%title(['Bessel Waves -- Calc D=A*B,  Ni=',num2str(ni),', S=',num2str(s),',  F=',num2str(f),',  G=',num2str(g)]);
xlabel('X');ylabel('Y');zlabel('Amplitude');
grid on;
```

"Wave" Function: Generated by Calculation of Airy Disk Function



"Wave" Generator Using Matlab code to calculate Matrix-Matrix Multiplication of Airy Disk Function

Visualizing "Wave" Generator: Airy Disk Function on tuckoo

- Modify makefile for profiling tool(s); make wave-dyn
- Run MPI based wave-dyn.c on tuckoo
- generate output files
- run profiling tool

GNU Project PROFiler (Gprof) Utility

The gprof utility is included in most Unix systems. It is used to profile program execution at the procedure level and profiles procedures according to their call graphs. gprof displays the following information:

- The parent of each procedure.
- An index number for each procedure.
- The percentage of CPU time taken by that procedure and all procedures it calls (the calling tree).
- A breakdown of time used by the procedure and its descendents.
- The number of times the procedure was called.
- The direct descendents of each procedure.

Code Example: run job from command line

```
[mthomas@tuckoo]$ cat makefile
```

```
=====================================
MAKE FILE
=====================================
MPICC = mpicc
CC    = gcc

wave-dyn: wave-dyn.c
$(MPICC) -pg -o wave-dyn wave-dyn.c


=====================================
RUN FROM COMMAND LINE
=====================================
[mthomas@tuckoo]$ mpirun -np 9 ./wave-dyn-mpt 900 s c 0.5 0.25 4
  ...

=====================================
PROFILING: using -p option in make
=====================================
 [mthomas@tuckoo]$ gprof wave-dyn   gmon.out
Flat profile:
Each sample counts as 0.01 seconds.
 cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 ..
```

Call graph generated by Gprof

```
 granularity: each sample hit covers 2 byte(s) for 0.83% of 1.21 seconds
index % time    self  children    called     name
                1.21    0.00       3/3           Fox [2]
[1]    100.0    1.21    0.00       3         Local_matrix_multiply [1]
-----------------------------------------------
                0.00    1.21       1/1           main [3]
[2]    100.0    0.00    1.21       1         Fox [2]
                1.21    0.00       3/3           Local_matrix_multiply [1]
                0.00    0.00       1/1           Set_to_zero [8]
                0.00    0.00       1/5           Local_matrix_allocate [4]
-----------------------------------------------
                                                <spontaneous>
[3]    100.0    0.00    1.21                  main [3]
                0.00    1.21       1/1           Fox [2]
                0.00    0.00       4/5           Local_matrix_allocate [4]
                0.00    0.00       2/2           Calc_matrix [5]
                0.00    0.00       1/1           Setup_grid [9]
                0.00    0.00       1/1           Build_matrix_type [6]
                0.00    0.00       1/1           Print_matrix [7]
-----------------------------------------------
                0.00    0.00       1/5           Fox [2]
                0.00    0.00       4/5           main [3]
[4]      0.0    0.00    0.00       5         Local_matrix_allocate [4]
-----------------------------------------------
                0.00    0.00       2/2           main [3]
[5]      0.0    0.00    0.00       2         Calc_matrix [5]
-----------------------------------------------
                0.00    0.00       1/1           main [3]
[6]      0.0    0.00    0.00       1         Build_matrix_type [6]
-----------------------------------------------
                0.00    0.00       1/1           main [3]
[7]      0.0    0.00    0.00       1         Print_matrix [7]
-----------------------------------------------
                0.00    0.00       1/1           Fox [2]
[8]      0.0    0.00    0.00       1         Set_to_zero [8]
-----------------------------------------------
                0.00    0.00       1/1           main [3]
[9]      0.0    0.00    0.00       1         Setup_grid [9]
```

Limitations of Gprof

- Slows down code performance so turn off when not needed
- Primarily designed for serial code, so does not handle parallel very well.

Tuning and Analysis Utilities (TAU)

- TAU Home: `https://www.cs.uoregon.edu/research/tau/home.php`
- ParaProf Manual: `https://www.cs.uoregon.edu/research/tau/docs/paraprof/`
- TAU program examples: `http://acts.nersc.gov/tau/`
- `http://www.nccs.nasa.gov/images/TAU-brownbag.pdf`
- `https://portal.tacc.utexas.edu/c/document_library/get_file?uuid=fc609b77-b727-4bff-81a4-d30caa4013d4&groupId=13601`
- and many more

Setting up TAU on tuckoo: ENV

```
####
# configuration for TAU performance toolkit
### tau install dir:
export TAUROOT=/usr/local/tau/
export TAU_MAKEFILE="/usr/local/tau/x86_64/lib/Makefile.tau-r
export TAU_TEST_MAKEFILE=$TAU_MAKEFILE
export TAU_OPTIONS="-optVerbose -optCompInst"
export LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PAT
#export TAU_PROFILE=1
#export TAU_TRACE=1
#export TAU_SAMPLING=1
```

Setting up TAU on tuckoo: Makefile

Call graph generated by Gprof

```
[mthomas@tuckoo wave-mpt]$ cat Makefile
include $(TAU_TEST_MAKEFILE)

CXX = $(TAU_CXX)
CC = tau_cc.sh

CFLAGS          = $(TAU_INCLUDE) $(TAU_DEFS) $(TAU_MPI_INCLUDE)
LIBS            = $(TAU_MPI_LIBS) $(TAU_LIBS) $(LEXTRA1) $(EXTRALIBS) -lm
LDFLAGS         = $(USER_OPT)  $(TAU_LDFLAGS)

TARGET          = wave-dyn-mpt

EXTRAOBJS       =

RM = /bin/rm -rf

##################################################
all: $(TARGET)

install:  $(TARGET)

OBJS = wave-dyn-mpt.o

$(TARGET): $(OBJS)
$(CC) $(LDFLAGS) $(OBJS) -o $@ $(LIBS)

# Compilation rule
.c.o:
$(CC) $(CFLAGS)  -c $< -o $@

clean:
$(RM) $(OBJS) $(TARGET) $(OBJS:.o=.inst.c) \
profile.* tautrace.* events.* *.elg
##################################################
```

Running Job generates profile files

```
[mthomas@tuckoo wave-mpt]$ mpirun -np 9 ./wave-dyn-mpt 900 s z 0.5 0.25 4
[mthomas@tuckoo wave-mpt]$ ls -al
drwxrwxr-x 2 mthomas mthomas    4096 Sep 15 05:35 .
drwxrwxr-x 6 mthomas mthomas    4096 Sep 11 13:39 ..
-rwx------ 1 mthomas mthomas     577 Sep 15 04:37 batch.wave-dyn
-rw-r--r-- 1 mthomas mthomas     781 Sep 15 05:34 Makefile
-rw-rw-r-- 1 mthomas mthomas    7211 Sep 15 05:35 profile.0.0.0
-rw-rw-r-- 1 mthomas mthomas    2521 Sep 15 05:35 profile.1.0.0
-rw-rw-r-- 1 mthomas mthomas    2592 Sep 15 05:35 profile.2.0.0
-rw-rw-r-- 1 mthomas mthomas    2588 Sep 15 05:35 profile.3.0.0
-rw-rw-r-- 1 mthomas mthomas    2589 Sep 15 05:35 profile.4.0.0
-rw-rw-r-- 1 mthomas mthomas    2593 Sep 15 05:35 profile.5.0.0
-rw-rw-r-- 1 mthomas mthomas    2589 Sep 15 05:35 profile.6.0.0
-rw-rw-r-- 1 mthomas mthomas    2588 Sep 15 05:35 profile.7.0.0
-rw-rw-r-- 1 mthomas mthomas    2592 Sep 15 05:35 profile.8.0.0
-rwxrwxr-x 1 mthomas mthomas 2194506 Sep 15 05:35 wave-dyn-mpt
-rw-rw-r-- 1 mthomas mthomas   37415 Sep 15 01:02 wave-dyn-mpt.c
```

Running Job generates profile files

Commands used to analyze TAU profiles include:

- *pprof* :
- *ParaProf* :
  - sorts and displays profile data generated by TAU.
  - Execute *pprof* in the directory where profile files are located.
  - interactive graphic visualization.
  - requires X11 xterm for display onto your local monitor.
  - `https://www.cs.uoregon.edu/research/tau/docs/paraprof/`

sss

Using TAU: looking at profile.* files

```
[mthomas@tuckoo wave_mpt]$ pprof
Reading Profile files in profile.*
NODE 0;CONTEXT 0;THREAD 0:
```

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name
100.0	7	5,366	1	1	5366946	.TAU application
99.9	0.276	5,359	1	14	5359473	main
58.1	0.061	3,118	1	11	3118707	Fox
57.1	3,065	3,065	3	0	1021896	Local_matrix_multiply
20.8	935	1,117	2	110203	558523	Calc_matrix
20.2	1,082	1,082	1	0	1082195	MPI_Init()
2.6	137	137	100001	0	1	Calc_coeffs [THROTTLED]
0.7	38	38	4	0	9590	MPI_Bcast()
0.7	36	36	1	0	36234	MPI_Finalize()
0.6	34	34	4800	0	7	MPI_Send()
0.3	13	13	3	0	4571	MPI_Sendrecv_replace()
0.2	9	9	5400	0	2	MPI_Cart_rank()
0.1	0.02	4	1	7	4885	Setup_grid
0.1	4	4	1	0	4296	MPI_Cart_create()
0.0	0.929	0.929	1	0	929	Set_to_zero
0.0	0.565	0.565	2	0	282	MPI_Cart_sub()
0.0	0.043	0.043	5	0	9	Local_matrix_allocate
0.0	0.018	0.041	1	6	41	Build_matrix_type
0.0	0.014	0.014	1	0	14	MPI_Type_contiguous()
0.0	0.005	0.005	1	0	5	MPI_Type_struct()
0.0	0.003	0.003	1	0	3	MPI_Type_commit()
0.0	0.002	0.002	1	0	2	MPI_Cart_coords()
0.0	0.002	0.002	5	0	0	MPI_Comm_rank()
0.0	0.002	0.002	2	0	1	MPI_Comm_size()
0.0	0.001	0.001	3	0	0	MPI_Address()

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 0

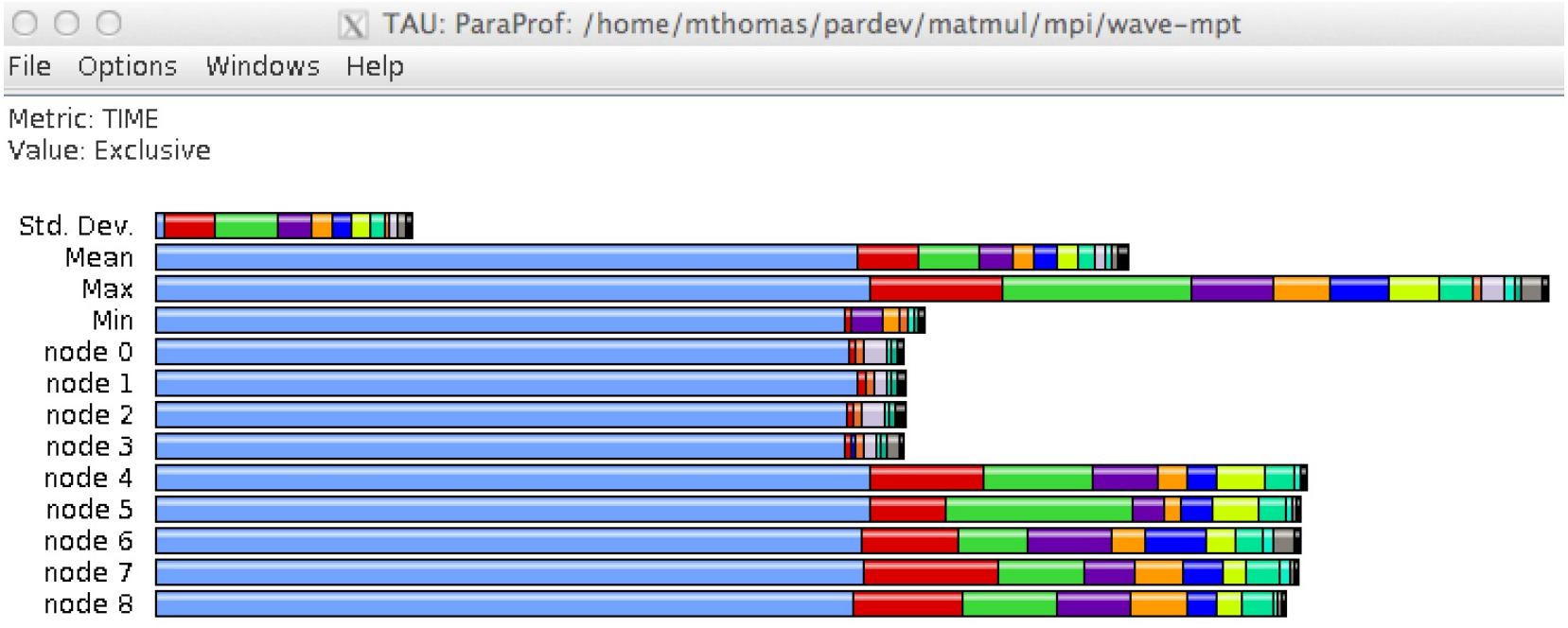
NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.	Event Name

Using TAU: looking at profile.* files

```
[mthomas@tuckoo wave_mpt]$ pprof
FUNCTION SUMMARY (mean):
```

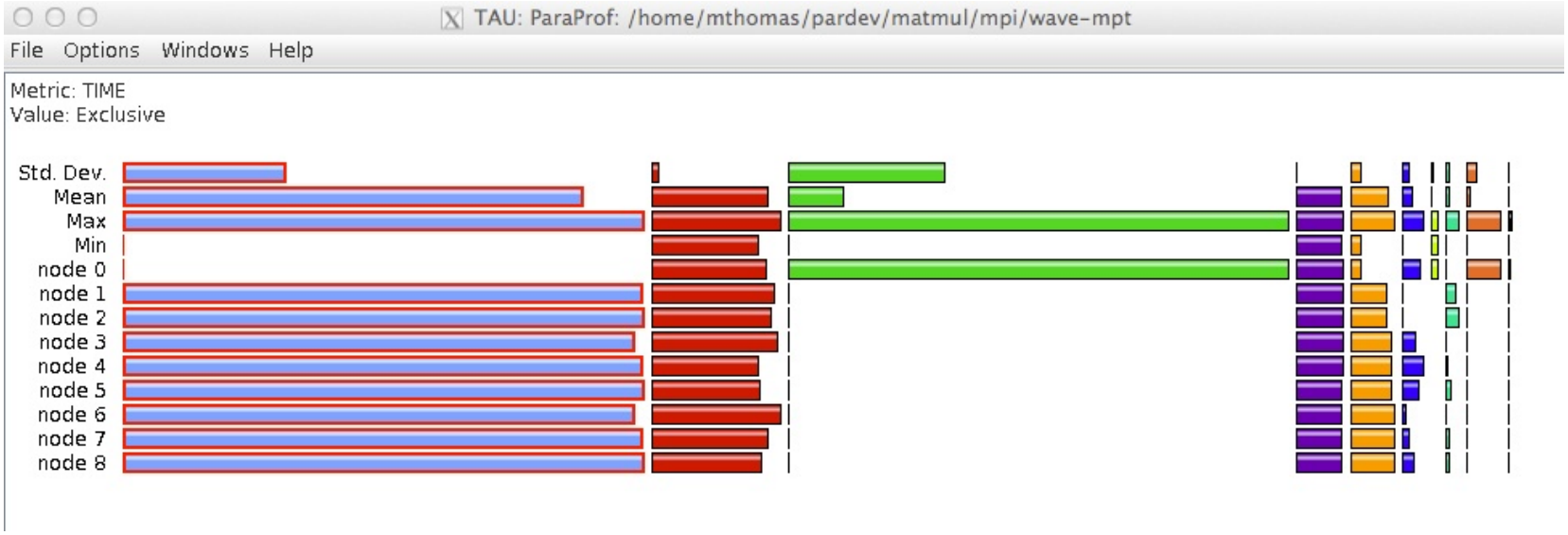
%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name
100.0	11	5,355	1	1	5355709	.TAU application
99.8	0.201	5,344	1	14	5344680	main
59.5	0.0574	3,185	1	11	3185130	Fox
50.3	2,695	2,695	3	0	898500	Local_matrix_multiply
19.8	1,062	1,062	1	0	1062000	MPI_Init()
19.3	104	1,034	2	12779.9	517138	Calc_matrix
17.0	909	909	533.333	0	1706	MPI_Recv()
6.1	326	326	3	0	108759	MPI_Sendrecv_replace()
3.1	165	165	4	0	41325	MPI_Bcast()
1.0	55	55	1	0	55635	MPI_Finalize()
0.3	15	15	11111.2	0	1	Calc_coeffs [THROTTLED]
0.1	0.0191	4	1	7	4409	Setup_grid
0.1	3	3	533.333	0	7	MPI_Send()
0.1	3	3	1	0	3792	MPI_Cart_create()
0.0	1	1	600	0	2	MPI_Cart_rank()
0.0	0.943	0.943	1	0	943	Set_to_zero
0.0	0.594	0.594	2	0	297	MPI_Cart_sub()
0.0	0.0424	0.0424	5	0	8	Local_matrix_allocate
0.0	0.0141	0.036	1	6	36	Build_matrix_type
0.0	0.013	0.013	1	0	13	MPI_Type_contiguous()
0.0	0.00489	0.00489	1	0	5	MPI_Type_struct()
0.0	0.00267	0.00267	5	0	1	MPI_Comm_rank()
0.0	0.00233	0.00233	1	0	2	MPI_Type_commit()
0.0	0.00189	0.00189	1	0	2	MPI_Cart_coords()
0.0	0.00167	0.00167	3	0	1	MPI_Address()
0.0	0.00111	0.00111	2	0	1	MPI_Comm_size()

TAU: Using ParaProf to visualize profile files.



ParaProf: Histogram of Time spent in routines

TAU: Using ParaProf to visualize profile files.



ParaProf: Expanded histogram of Time spent in routines

TAU: Using ParaProf to visualize profile files.

TAU: Using ParaProf to visualize profile files.