Comparing the performance of the NaN-toolbox on Matlab and on Octave with OpenMP enabled.

Alois Schloegl,

e-mail: alois.schloegl@ist.ac.at

Institute for Science and Technology Austria

27 Jun 2011

Introduction

The aim of the is work is testing whether the use of OpenMP [1] can be beneficial to the NaN-toolbox for statistics and machine learning [2], which relies on some core functions that are implemented in C and can be used through the mex-interface. Moreover, in the past Octave did not perform as well as Matlab [3]. It is the aim to investigate which platform provides the best performance for resource intensive computations.

Method

Hardware: 12 Core Processore (2 x 6-core AMD Opteron)

Operating System: Debian Squeeze AMD64

Software:

> Octave 3.4.1 compiled from sources with the option “openmp” enabled.
>
> Matlab 7.11 2010b

Two algorithms from the NaN-toolbox [1] were used for performance testing. SUMSKIPNAN_MEX sums the columns of a matrix, and COVM_MEX computes the covariance matrix. The algorithms were modified for the use with OpenMP. Matlab supports parallelization, however, in order to be able of using multiple cores, Octave must be compiled with the flag –enable-openmp.

The timing was tested with the internal clock (tic,toc) and cputime functions of Matlab and Octave. Moreover, top was used to check whether some other processes took significant CPU time; no signficant CPU usage from other processes were observed. The script for testing the performance is shown below.

```
%% Generate Test data
y = randn(1e7,32);


flag=1;
N=10;
t1=repmat(N,2);
t2=repmat(N,2);
for k=1:N;
     tic;t=cputime();
     [s,n]=sumskipnan_mex(y,1);    % sum of columns
     t1(k,1)=cputime()-t; t1(k,2)=toc;

     tic;t=cputime();
     [c,n]=covm_mex(y,[],flag);    % covariance matrix
     t2(k,1)=cputime()-t; t2(k,2)=toc;
end;
exp(-diff(log(mean(t1))))
```

Results

Table 1: Performance of Octave-3.4.1 with OpenMP on scicomp01left

Octave run	sumskipnan cputime [s]	sumskipnan real time [s]	covm cputime [s]	covm real time [s]
1	3.532	0.300	102.250	8.540
2	3.548	0.299	101.680	8.496
3	3.488	0.292	101.510	8.476
4	3.496	0.293	100.780	8.416
5	3.496	0.292	100.830	8.420
6	3.544	0.296	101.760	8.493
7	3.500	0.294	100.690	8.406
8	3.540	0.297	101.750	8.494
9	3.516	0.293	101.020	8.436
10	3.520	0.298	100.430	8.385
mean	**3.518**	**0.295**	**101.270**	**8.456**
s.d.	0.022	0.003	0.596	0.050

Table 2: Performance of Matlab 7.11 on scicomp01left

Matlab run	sumskipnan cputime [s]	sumskipnan real time [s]	covm cputime [s]	covm real time [s]
1	4.440	0.384	132.830	11.101
2	4.430	0.382	131.340	10.962
3	4.450	0.381	132.130	11.028
4	4.430	0.381	131.820	11.012
5	4.440	0.381	132.120	11.033
6	4.440	0.382	132.740	11.081
7	4.450	0.382	132.420	11.060
8	4.420	0.381	131.080	10.939
9	4.430	0.381	131.830	11.007
10	4.440	0.381	131.660	10.999
mean	**4.437**	**0.382**	**131.997**	**11.022**
s.d.	0.009	0.001	0.568	0.050

Table 3: Speedup factor by using OpenMP. The speed up factor was computed as the ratio betweeen the average time needed and the average CPU time.

Machine	Software	Test	CPUTIME	Actual time [s]	Speed up
scicomp01left	Octave	SUMSKIPNAN	3.518	0.295	11.912
scicomp01left	Octave	COVM	101.270	8.456	11.976
scicomp01left	Matlab	SUMSKIPNAN	4.437	0.382	11.624
scicomp01left	Matlab	COVM	131.997	11.022	11.976

Table 1 and Table 2 show the results for the N=10 repetitions,its mean and standard deviation for Octave 3.4.1 and Matlab 7.11, resp. The average results are summarized in Table 3. An speed-up factor (realtime over cputime) of almost 12 is shown for all results. For the same test, Octave was about 30 % faster than Matlab.

Discussion

Observing CPU load using “top” showed that the 12 Processor cores were used. The speedup of a factor of almost 12 could be achieved for both algorithms (sumskipnan and covm) and both platforms (Octave and Matlab). This shows that the use of OpenMP within Octave can be a significant advantage in shared memory systems.

Moreover, we can see that Octave is about 30% faster than Matlab in the present test. This improvement has been observed only when “openmp” was enabled within Octave. Without openmp, the improvement has not be observed (results are not shown here).

The present test routines (summing of a matrix and computing the covariance matrix) are important functions for statistics, machine learning and signal processing. They are used to train statistical classifiers, like LDA, QDA and RDA methods, and compute multivariate covariance functions.

References

[1] The OpenMP® API specification for parallel programming http://openmp.org/wp/

[2] The NaN-toolbox v2.0: A statistics and machine learning toolbox for Octave and Matlab® for data with and w/o MISSING VALUES encoded as NaN's.
Revision 8325 from the octave-forge repository (http://octave.sourceforge.net)

[3] Alois Schloegl, BioSig - An application of Octave, 2006
available online at http://arxiv.org/abs/cs/0603001