Accelerating MRI data analysis by using Matlab toolboxes and Linux cluster

Mingyi Li[1], Erik Beall[1], and Mark Lowe[1]
[1]*Cleveland Clinic, Cleveland, Ohio, United States*

Introduction:

MRI Neuroimaging research generally needs to handle a large amount of data. Besides demand of high storage, great CPU power is also required to rapidly process data and generate results. Starting several years ago, the trend of boosting CPU performance has shifted from increasing clock speed of a single CPU to integrating multiple cores into one CPU. In order to fully utilize multi-core CPUs, parallelization is a must. Neuroimaging data intrinsically is easy to be paralleled because of possible data independence on the voxel, volume and subject levels during the analysis process. However, the practical skill of using programming language, parallel library and runtime environment could be beyond the expertise of many researchers.

One research project in our lab is to investigate a new measure similar to intrinsic connectivity distribution[1] on brain fMRI data. To obtain the measure, time series correlation of each voxel to all the other voxels on the brain gray matter needs to be calculated, followed by Gaussian curve fitting of the distribution histogram. We developed a data analysis pipeline in Matlab[2]. It took about four and half hours on average to generate the results for one dataset and five to six days for datasets of thirty subjects. Shorter turn-around time was desired. As a solution, we used two levels of parallelization to cut the total running time. Matlab Parallel Computing Toolbox (MPCT) was used to accelerate the computation on the single subject level. Then Matlab Compiler Toolbox (MCT) was used to convert the parallelized Matlab code to executable binary code. Many copies of executable code could be run simultaneously on our Linux cluster and the parallelization on multiple subjects' level was achieved. The total running time was thus reduced to several hours after the two levels of parallelization. Such parallelization methodology is relatively easy to be implemented and is also applicable to a wide range of MRI data analysis. In this abstract, we will focus on the parallelization of data analyze pipeline. The detailed description and neurophysiologic findings of the new intrinsic connectivity measure will be presented separately.

Methods:

RS-fMRI scans were acquired in 6 low-motion subjects using a bite-bar at 3 Tesla Siemens scanner. Data was corrected for volumetric motion, physiologic noise and spatially filtered to 4mm FWHM. A T1 MPRAGE was also acquired and segmented to provide gray matter (GM) mask. For each GM voxel, correlation of time series to all other GM voxels were computed and distribution histogram was fitted to a Gaussian distribution. The area between the fitted Gaussian curve and actual distribution was calculated on the tail of the curves.

Since the correlation calculation and curve fitting is independent for each GM voxel, using "parfor" to substitute "for" loop and telling Matlab to use multiple CPUs would evoke the parallel functionality in MPCT. MCT provides two different targets: compiling the Matlab script into a standalone application is straight forward; compiling Matlab function to shared library provides a more flexible calling interface although it requires some C/C++ programming skill to write a wrapper. Executables generated from both case can be run on computers or clusters without Matlab installation. But the free Matlab Compiler Runtime (MCR) environment must be installed.

For benchmarking purpose, we used the same hardware platform and software environment to run the three different methods on the 6 datasets. The computer is a DELL Precision T5500 workstation with dual Intel Xeon E5-2630 CPUs and 32GB of RAM. The software environment is Matlab R2012a on CentOS Linux 6.4.

Results:

The running time for the subject dataset is shown in the table. The unit is in second. On average, 5.67 times of speedup is gained when 12 CPU cores are used in MPCT. In our case, the speed of MCT compiled code run in MCR is very close to code run in Matlab.

method\subject	1	2	3	4	5	6	Ave±Std
1 CPU	16886	16027	12131	21174	14395	17752	16394±3074
12 CPUs MPCT	3057	2812	2106	3998	2474	3177	2889±651
12 CPUs MPCT+MCT	3114	2772	2064	3972	2473	3203	2933±660

Conclusion:

MRI data analysis can gain great boost in speed by using MPCT. If adequate computer power like cluster is available, MCT can be used to generate binary code and data analysis for multiple subjects can be run simultaneously.

Discussion:

In Matlab R2012a, MPCT can support up to 12 CPUs. If a higher number of CPU is supported in the future, the computation time will become even shorter when paralleled. The limitation of this study is the data analysis pipeline must be written in Matlab.

Reference:

1. Sheinost et al., NeuroImage, 2012
2. www.mathworks.com