Parallelization in Multiple Imputation

Sven Nekula, Joshua Simon and Eva Wolf
Otto Friedrich University, Bamberg

What is Parallelization?

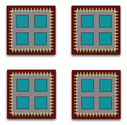

Figure 1: 4 processors with 4 cores each

Parallelization is a technique to fasten time-consuming computations. It uses several cores on a **CPU** parallely and splits up the computational work on them. Afterwords, the results are merged. This can reduce the runtime significantly. Parallelization is nevertheless limited to a certain kind of tasks: The computations to be parallelized must be independent from each other, as information cannot be cross-accessed over the cores while the process is running.

Theory

In **parallel programming**, the multiple cores of a computational system want to be used best to decrease computation time. Gene **Amdahl** was the first one to desribe the boundaries of that project: Every parallel process also requires additional workload, so called "data management housekeeping". The speed up through parallel processing will tend to 0 at a certain amount of processing units involved, as this **overhead** workload exceeds the capacity of the computational unit which it is assigned to.

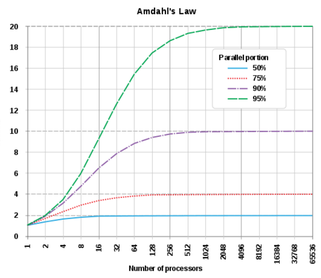

Figure 2: Amdahls Law

Multiple imputation is a method to complete a dataset with missing information. It relies on the estimation of the missing values through different methods. What is common to them is that we use not a single imputation run, but several. The results of all imputation runs are then merged and lead to realistic uncertainty of the estimators given the missing data. It is prone to be parallelized, as the several imputation runs can be processed independend of each other on different cores and easily merged afterwards.

Methodology

As the **dataset** we used a simple data generator of normally distributed random variables. The data set created contains 10 variables, of which some depend on each other. The sample size is n=10000. Parallelizationfavors complex data sets, so the amount of variables and their interdependency should not be to small.

Time measurement was done with the *system.time* function, which returns 3 values: User CPU-, User System- and Elapsed time. User CPU is the time needed for the current task such as an execution in R. System CPU describes the time needed by the operating system to organize that task such as opening folders or asking for the System time. Elapsed time is the wall clock time that passed while the function and background processes were running.

The **method** of the **mice** algorithm was set on defaut, pmm.

The **speed up** value is calculated by the serial time (runtime without parallelization) divided by the runtime of the current parallelization implementation [**cite:chapple2016mastering**]. Values below one show that the parallelized run took longer than the serial run.

Results

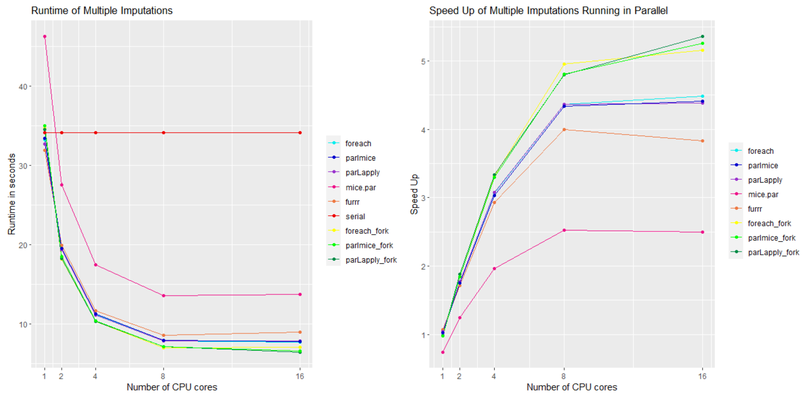

Fig. 1: Runtime and Speedup from 1 up to 8 cores

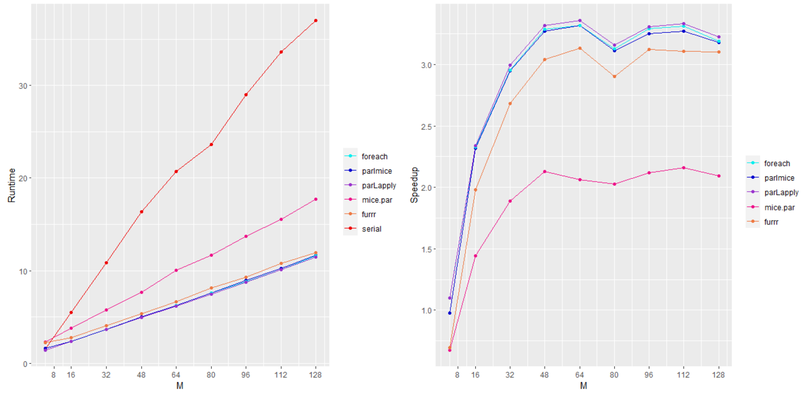

Fig. 2: Runtime and Speedup from 1 up to 128 Imputation runs

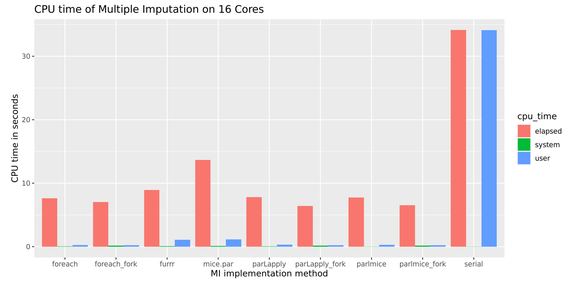

Fig. 3: Time components for the different implementations on 16 cores

Comparison of Implementations

foreach::foreach usability ★★★★★ runtime ★★★★★

mice::parlmice usability ★★★★★

micemd::mice.par usability ★★★★★

parallel::parLapply usability ★★★★★

furrr::future_map usability ★★★★★

future.apply::future_lapply usability ★★★★★

PSOCK vs. *FORK*

Application

The opposite case to paralellized processes is the intended time-intensive, serial processing of Big Data with **disk framing**. In that setting we have data that exceeds the **RAM capacity** of the system, so that the data needs to a split up into several smaller chunks. These are then processed in portions that fit the available memory. Nevertheless, also for **disk framing**, parallelization is a useful tool to decrease computation time. The system chooses as many chunks of data as they fit into the RAM and then processes them parallely, so that use of the computational power of the system can be made, while the memory capacity limits the speed by the amount of data which is processed at once.

References