International Journal of Computational Science and Information Technology (IJCSITY)
Vol. 10, No. 1, February 2022

HIGH PERFORMANCE COMPUTING ON THE RASPBERRY PI

Erin M. Hodgess

Western Governors University, Salt Lake City, UT, USA

ABSTRACT

We considered building high performance tools on the Raspberry Pi 4. We implemented OpenMP and OpenCoarrays Fortran in conjunction with the statistical language R. We found that the OpenCoarrays is more effective when working with vectors, while OpenMP is better in the arena with large matrices in a geostatistics application. These results can be very useful for researchers with limited access to high performance tools or limited funding.

KEYWORDS

OpenMP, Coarrays Fortran, R.

1. INTRODUCTION

We have been working with spatial and spatio-temporal data for several years. Existing functions either took an exhorbitant amount of time to process, or would not complete at all. We also have worked with high performance computing tools, both on supercomputers and with gaming laptops. We worked with the Stampede I and Bridges supercomputers [1]. We were able to upgrade ordinary kriging processes which took 2 hours and speed them up to 6 minutes on the Stampede supercomputer and 2.5 minutes on the Bridges supercomputer.

We then worked with gaming laptops with NVIDIA graphics cards. Initially, we used the Portland Group (PGI) compiler with access the GPU processors for speed-up potential. We were able to both reduce processing times as well as bringing in larger data sets [2].

When giving a presentation on these spatio-temporal enhancements, a member of the audience asked if there was any way that these processes could be made available to users without gaming laptops. We did not have a suitable answer at that time, but that was really the motivation for this project.

We experimented with a Raspberry Pi single board computer to see if either OpenMP or OpenCoarrays Fortran would be available, and both are. We used the latest operating system for the Raspberry Pi to implement high performance tools for ease of processing. We installed R, OpenMP, and OpenCoarrays Fortran on the 64 bit version of the Raspberry Pi OS. We compared code using R functions, along with Fortran subroutines with both OpenMP [3] components and OpenCoarrays [4] components.

In our investigations, our method was the single program, multiple data (SPMD) structure, as discussed in Pacheco [5]. We used different data in each processor to produce our final results.

DOI : 10.5121/ijcsity.2022.10101

We then wrote code for two applications: estimation of the area under the curve for π, as discussed in Mattson *et al* [6] and an ordinary kriging method for spatio-temporal data as found in Hodgess and Mhoon [7]. We tied the OpenMP and OpenCoarrays programs into R [8] as subroutines in order to generate our results.

2. DEFINITION OF THE PROBLEM: I. RIEMANN SUMS

We produced an estimation of π by using Riemann sums under a curve. Our equation will be:

$$\int_0^1 \frac{4}{1+x^2}\, dx = \pi \tag{2.1}$$

We discussed 3 methods for solution. We first produced an R function using vectorization. Next, we used the version found in Mattson *et al* for OpenMP and created a Fortran subroutine that is called from R. Finally, we modified the OpenMP version into an OpenCoarrays subroutine, also called from R.

We ran a simulation study on each method. We ran 10000 replications on each method, and recorded the run time (in seconds) for each replication. Here are the results:

We saw that using vectorized R code is the clear winner for this case. But we did see an interesting result between OpenMP and OpenCoarrays. The OpenCoarrays process was much more efficient than the OpenMP process.

Table 1: Run Times

Version	Mean Run Time	SE Run Time
Vector R	0.003	0.000
OpenMP	1.763	0.008
Coarrays	0.419	0.002

3. DEFINITION OF THE PROBLEM: II. ORDINARY KRIGING OF SPATIO-TEMPORAL DATA

Using the structure developed in Hodgess and Mhoon [7], we take a spatio-temporal vector:

$$\mathbf{z} = z(s_1, t_1), z(s_2, t_1), \ldots, z(s_n, t_n) \tag{3.1}$$

where s_i and t_i are locations and times of the response variable from a Gaussian random field.

Ordinary kriging for spatio-temporal data is essentially a generalized least squares model. It is more complicated, as we must take both space and time into account, which is what produces the massive matrices. We make the assumption of first order stationarity:

$$E[\mathbf{z}] = \mathbf{u} \tag{3.2}$$

We also assume second order stationarity,

$$C_{st} = \text{Cov}(\mathbf{z}(s, t))(\mathbf{z}(w, v)) \tag{3.3}$$

The spatial distance is h and the temporal distance is u, such that

$|s - w| = h,$
$|t - v| = u$

for all values of s, w, t and v.

For spatio-temporal data, there are several authorized models of covariance structure, including separable, metric, and sum-metric. For our process, we select the sum-metric, as it reflects real world models. Our final covariance model is:

$$C_{sm} = C_s(h) + C_t(u) + C_{joint}(\mathrm{p}(h^2 + ku^2)), \quad (3.4)$$

where $C_s(h)$ is the covariance matrix for the spatial direction, $C_t(u)$ is the covariance matrix for the temporal direction, and C_{joint} is the joint effect component, with k as the parameter for anistropy.

The overall sample mean, as described in Rossiter [9] is seen to be

$$\hat{\mu} = \left(\mathbf{1}'(\mathbf{C}_{sm})^{-1}\mathbf{1}\right)^{-1} \cdot \left(\mathbf{1}'(\mathbf{C}_{sm})^{-1}\mathbf{z}\right). \quad (3.5)$$

For a one point forecast, we obtain:

$$\hat{\mathbf{z}} = \hat{\mu} + c_0\mathbf{C}_{sm}^{-1}(\mathbf{z} - \hat{\mu}\mathbf{1}). \quad (3.6)$$

We can create multiple point forecasts, both in space and in time. Our steps are:

1. Fit the semivariogram for the spatial, temporal, and joint effects using authorized models via an automatic procedue in R.
2. Fit the sample space-time variograms.
3. Fit the space-time models based on the results of Steps 1 and 2.
4. Calculate the estimates of the sill, nugget, range, and anistropy from existingR functions.
5. Now we calculate the overall mean and covariance matrix.
6. Finally, we calculate the predicted values.

We look at the Irish Wind data, as seen in Pebesma [10], and which originally appeared in Haslett and Raftery [11]. The data are daily wind speeds at 12 stations around Ireland, and they run from January 1961 - December 1978. We begin with 3 months, 1 year and 2 years. The output is a spatio-temporal grid with interpolated values.

We run the existing R functions, as well as the new methods with OpenMP and OpenCoarrays.

We see that all 3 methods are successful in the 3 month duration, and the new methods pull away dramatically in the 1 year segment. Finally, for 2 years of spatio-temporal data, only the OpenMP method is able run to completion. However, these results indicate that complex models can run on a single board computer.

Table 2: Spatio-Temporal Run Times in Minutes

Duration	R	OpenMP	OpenCoarrays
3 months	0.20	0.22	0.22
1 year	8.59	4.27	4.48
2 years	-	35.03	-

4. TECHNICAL NOTES

The actual Raspberry Pi cost $75, while the 128 GB micro disk card cost $19. We use the headless setup for the Pi, and run it on an old MacBook laptop.

All of the software is free. The Raspberry Pi operating system has the gcc/gfortran and OpenCoarrays already loaded. We also download OpenMP and OpenMPI [12] for some of our processes. We use the pdbDMAT [13] package to aid with the large matrices.

5. CONCLUSION

We have demonstrated that the Raspberry Pi can be a viable tool for high performance computing on a budget. For less than $100, we can produce efficient results in integration and spatio-temporal analysis. These tools can be easily accessed via a laptop.

We hope that these studies may be useful for researchers without access to supercomputers or even gaming laptops. The Raspberry Pi can be an effective tool for many extensive models.

REFERENCES

[1] E. Hodgess, "Using r with high performance tools on a windows laptop," *submitted to 3rd Conference of the Arabian Journal of Geosciences*, 2020.

[2] E. Hodgess and K. Mhoon, "Advances in ordinary kriging using the stampede and bridges supercomputers," *Journal of Geological Resource and Engineering*, vol. 6, pp. 14 – 18, 2018.

[3] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel programming in OpenMP*. Morgan Kaufmann, 2001.

[4] A. Fanfarillo, T. Burnus, V. Cardellini, S. Filippone, D. Nagle, and D. Rouson, "Opencoarrays: open-source transport layers supporting coarray fortran compilers," in *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models*, p. 4, ACM, 2014.

[5] P. S. Pacheco, *Parallel Programming with MPI*. Morgan Kaufmann, 1996.

[6] T. G. Mattson, Y. He, and A. E. Koniges, *The OpenMP common core: making OpenMP simple again*. The MIT Press, 2019.

[7] E. M. Hodgess and K. Mhoon, "Ordinary kriging for spatio-temporal data sets using a windows laptop," in *XIV Congreso de Informatica y Geociencias (GEOINFO 2019)*, p. 4, Cuban Geology Society, 2019.

[8] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[9] D. G. Rossiter, "Geostatistics and open source statistical computing." http://www.css.cornell.edu/faculty/dgr2/teach/index.html# university-of-twente-nl, 2015. Accessed: 2022-01-15.

[10] E. J. Pebesma, "Multivariable geostatistics in s: the gstat package," *Computers and Geosciences*, vol. 30, no. 7, pp. 683–691, 2004.

[11] J. Haslett and A. E. Raftery, "Space-time modelling with long-memory dependence: Assessing ireland's wind power resource," *Applied Statistics*, vol. 38, no. 1, pp. 1 – 50, 1989.

[12] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall,

"Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, (Budapest, Hungary), pp. 97–104, September 2004.

[13] D. Schmidt, W.-C. Chen, G. Ostrouchov, and P. Patel, "pbddmat: Programming with big data – distributed matrix algebra computation," 2012. R Package.

APPENDIX

Here are the steps to prepare the Raspberry Pi for our processing.

OpenMPI
sudo apt-get install --reinstall openmpi-bin libopenmpi-dev

OpenMP
sudo apt-get install libomp-dev

Stuff for R:

sudo apt-get install libreadline-dev sudo apt-get install
xorg-dev sudo apt-get install liblzma-dev
sudo apt-get install libpcre++-dev sudo apt-get install libcurl4-
openssl-dev sudo apt-get install libpango1.0-dev
sudo apt install default-jdk Check version

java --version

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-arm64
export PATH=$PATH:$JAVA_HOME/bin
sudo apt install texlive-latex-extra
sudo apt install emacs

Note: both gfortran and opencoarrays are on the 64 bit OS

R

tar xf R-4.1.2.tar.gz
./configure make sudo
make
install For s2

installation:

sudo apt install libssl-dev

For geospatial libraries:

sudo apt-get install binutils libproj-dev gdal-bin
sudo apt install udunits-bin libudunits2-dev

For BLAS
sudo apt install libblas3 liblapack3 liblapack-dev libblas-dev

For pbd:
remotes : : install_github("RBigData/pbdSLAP")
remotes : : install_github("RBigData/pbdBASE")
remotes : : install_github("RBigData/pbdDMAT")