Evaluation of the parallel performance of the TRIGRS v2.1 model for rainfall-induced landslides

Massimiliano Alvioli[1], Daniele Spiga[2], and Rex L. Baum[3]

[1]**CNR IRPI, via Madonna Alta 126, 06128, Perugia, Italy**
[2]**INFN, Sezione di Perugia,via A. Pascoli 1, 06123, Perugia, Italy**
[3]**US Geological Survey, P.O. Box 25046, Mail Stop 966, Denver, CO 80225-0046, USA**

Corresponding author:
Massimiliano Alvioli[1]

Email address: massimiliano.alvioli@irpi.cnr.it

ABSTRACT

The widespread availability of high resolution digital elevation models (DEM) opens the possibility of applying physically based models of landslide initiation to large areas. With increasing size of the study area and resolution of the DEM, the required computing time for each run of such models increases proportionally to the number of grid cells in the study area. The aim of this work is to present a new parallel implementation of TRIGRS (Alvioli and Baum (2016)), an open-source FORTRAN program designed for modeling the timing and distribution of shallow, rainfall-induced landslides, and to discuss its parallel performance. We investigated the parallel performance by evaluating running time, speedup and efficiency of the code on a commonly available multi-core machine, on a high-end multi-node machine and on a cloud computing environment, showing the advantages and limitations of each case and discussing the possible weak points of using a general-purpose cloud environment.

Keywords: TRIGRS, Shallow landslides, MPI, Slope stability

INTRODUCTION

The TRIGRS model allows modeling of the spatial distribution of landslides, obtained by computing transient pore-pressure changes, and attendant changes in the factor of safety due to rainfall infiltration, using a simple infinite-slope description on a cell-by-cell basis. Time dependence is implemented in the model by time-dependent rainfall infiltration, resulting from storms that have durations ranging from hours to a few days. The computing time of each run may vary widely, and it is mostly related to the size of the study area, the number and frequency of the input rainfall time series, and the number of required output maps, which is user-defined.

The motivation for having a parallel, and substantially faster, code is generally found in the need to perform multiple simulations, for example to calibrate parameters, or to increase the spatial extent of simulations, or both (Alvioli et al. (2014); Raia et al. (2014); Mergili et al. (2014)). In the specific case of TRIGRS, we have shown that the simulation of a large test area is possible with great detail in many respects: the high resolution DEM, the dense rainfall pattern in time, the number of output maps. Using the new version of the code, we were able to reduce run time from about one day to

about an hour for a 1340-km^2 test area Baum et al. (2016). This allowed us to obtain a detailed time series of model outputs and match them with the field observations, providing a robust understanding of landslide phenomena in the area (Alvioli and Baum (2016)).

METHODS

The previous version of the code (Baum et al. (2008)) was improved with new features (Iverson (2000); Baum et al. (2010)) and parallelized within the Message Passing Interface (MPI) framework (MPI Forum (2012)). In this work we illustrate the performance gain obtained with the parallel version of the code, with respect to the serial version. We stress that the improvements made to the serial version of the code were mostly aimed at making the parallel implementation practical, and do not contribute appreciably to the performance gain of the new version of the code. Thus, users may assume that the latest version of the serial program has the same performance of the previous version, as far as running time is concerned.

In this work, we show the performance gain with respect to the serial code on commercial hardware (CNR IRPI, Perugia), on a high-performance multi-node machine (Galileo cluster, CINECA, Bologna) and on the OpenStack cloud environment (CERN, Geneva). The performance of the parallel code was assessed both measuring overall running time T_{Tot} and using standard quantities such as speedup

$$S(N_p) = \frac{T_{Tot}(N_p = 1)}{T_{Tot}(N_p)} . \tag{1}$$

the ratio of serial ($T_{Tot}(N = 1)$) to parallel ($T_{Tot}(N)$) running times as a function of the number of processes N_p, and efficiency

$$E(N_p) = \frac{S(N_p)}{N_p} = \frac{T_{Tot}(N_p = 1)}{N_p \, T_{Tot}(N_p)} . \tag{2}$$

For a better assessment of the parallel performance, we have split the overall running time into three terms: the time required for data reading and scattering to all the computing processes, T_R, the time required for core computing and gathering of the partial results from each computing process, T_C, and time required from writing the results on disk, T_W; this is summarized in Fig. 1. The

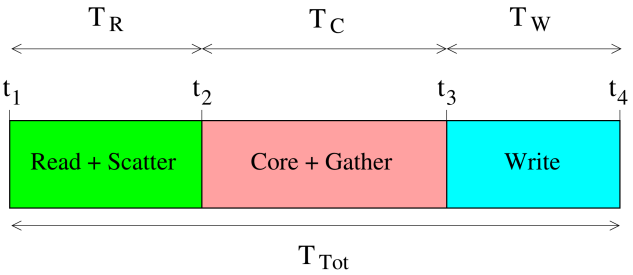

Figure 1. The overall running time T_{Tot} decomposed into three intervals, as described in the text.

other quantities we have considered, speedup S and efficiency E, have been calculated according

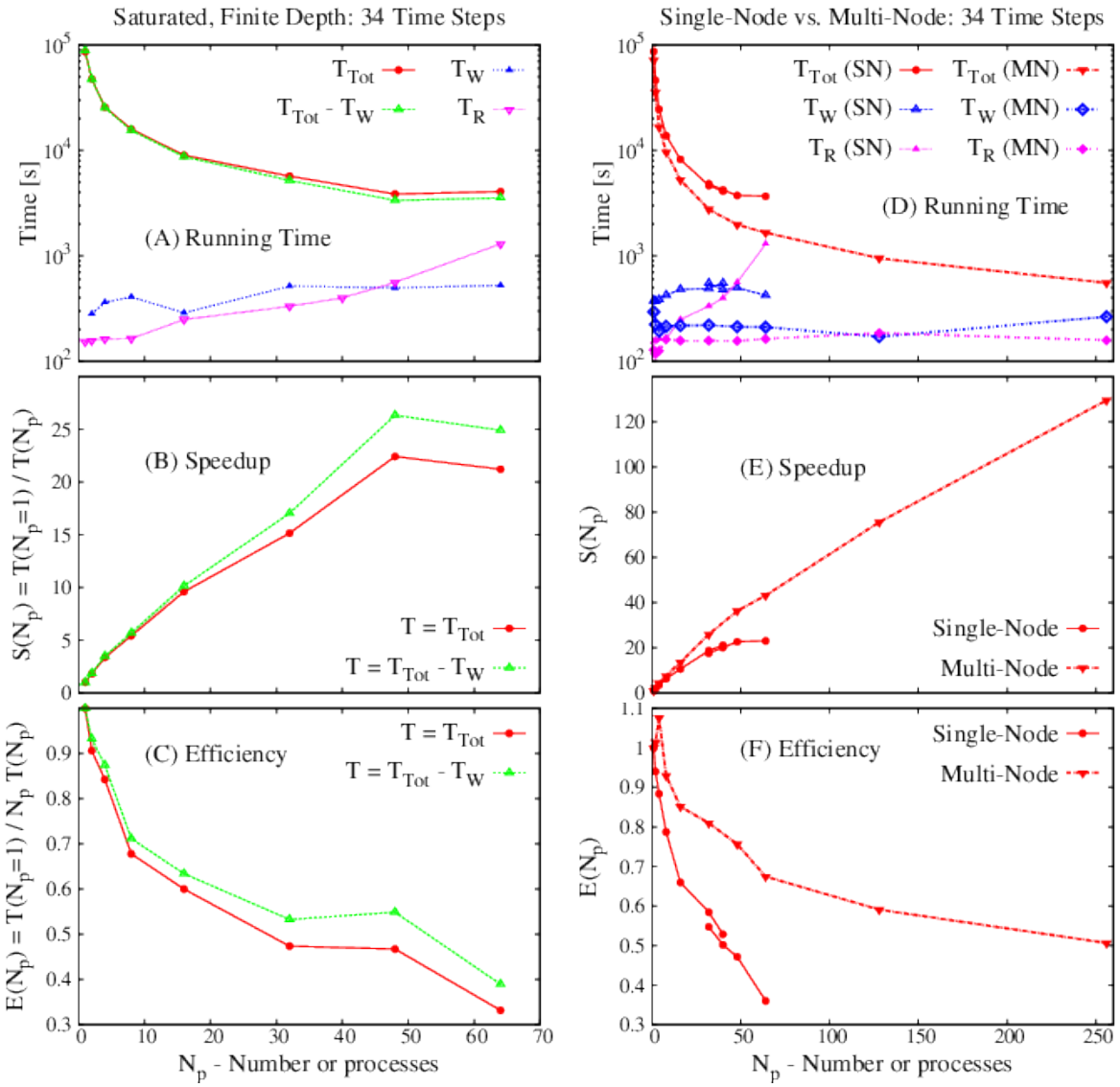

Figure 2. Results for the parallel performance of TRIGRS v2.1, as a function of the number of processes in the MPI pool, N_p. A: total (T_{Tot}), read + data scattering + compute + data gathering ($T_{Tot} - T_W$), write (T_W) and read + data scattering (T_R) running times. B: parallel speedup with respect to T_{Tot} and $T_{Tot} - T_W$ running times. C: parallel efficiency with respect to the same partial running times as in B. The left column refers to execution on a single-node machine equipped with 64 computing cores, while the right column shows results of execution on the Galileo cluster at CINECA; we considered $N_p \leq 256$; the results of the left column are also shown for comparison. The 34 time-steps referred to in the Figures are the number of time intervals of the storm considered in the simulation.

Eqs. (1) and (2), respectively, considering separately the overall running time T_{Tot} and replacing T_{Tot} with the partial running time $T = T_{Tot} - T_W$. The corresponding results are shown in Fig. 2. When run on a single-mode, high-end multi-core machine, the code offers performance gain up

to about $N_p = 50$, for our case study. The major limitation for a further performance gain are the operations summarized with T_R and T_W, the data read and scatter operations presenting the most severe drawback when N_p increases over 50. When run on a multi-node, high-performance machine, the limitations due to read-write and data scatter-gather operations are largely removed and we do not see performance degradation within the $0 < N_p \leq 256$ range. We conclude that the parallel performance is very satisfactory, since the code provides sufficient performance improvement on a single node machine with a number of computing cores which is well beyond the typical number of cores available on common desktop machines. A further performance increase can be obtained running the code on a large machine equipped with high-performance input-output devices and high-speed interconnection network such as the Galileo cluster.

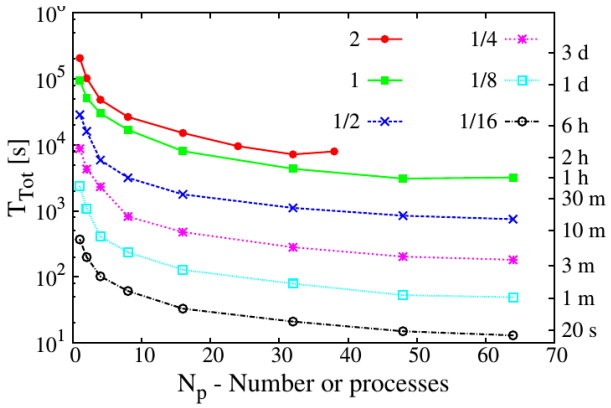

Figure 3. Dependence of the total running time on the size of the study area. "1" refers to the problem size considered in Fig. 2 (with a total of 13,410,000 valid cells), "2" refers to a problem with double size in terms of valide cells, "1/2" to one-half the size, and so on.

We have investigated the dependence of the parallel performance upon the problem size, which, in turn, is practically linearly proportional to the number of grid cells. We show in Fig. 3 the total computing time required for running the code with the study area used in Fig. 2, and also with study areas of twice, one-half, one-fourth, one-eighth and one-sixteenth of the original size. We find, as expected, that performance degradation when running on our single-node machine occurs for smaller N_p in the larger problem with respect to the original one, and we do not reach the point at which performance starts getting poorer for the smaller-size cases.

Subsequently, we have investigated the parallel performance of the code on a typical cloud environment setup. In particular, we have used virtual machines with sixteen computing cores each, using the OpenStack cloud environment installed at CERN. Results for the total and read-scatter running times are shown in Fig. 4, compared to the ones obtained on the single-node and the multi-node machines considered earlier. We find that the performance gain is comparable to the single-node case when the code is run within one node. We find severe limitations mostly due to the read-scatter fraction when the code is run across multiple nodes in OpenStack: the performance appears to be degraded and depending on the network status in each particular run, as expected.

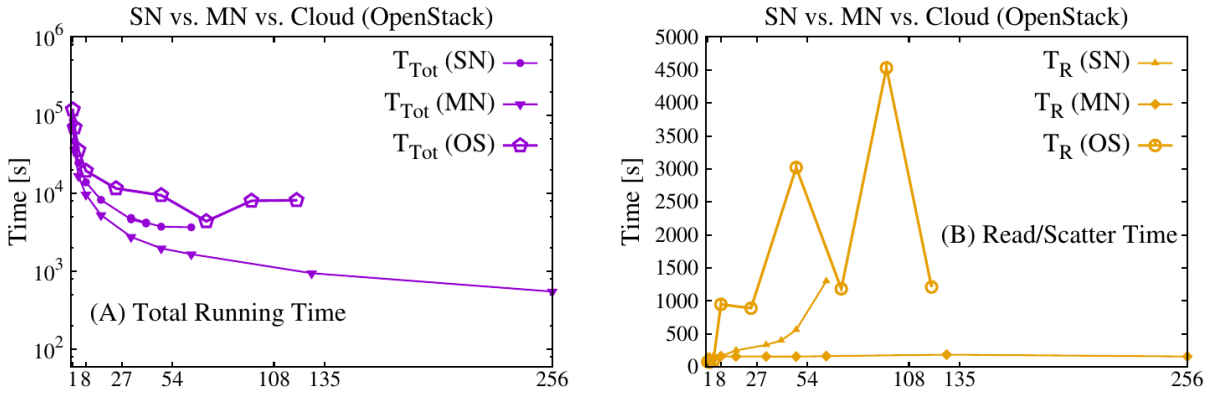

Figure 4. Total (A, T_{Tot}) and read-scatter (B, T_R) running times obtained on OpenStack cloud environment compared to the ones obtained on the single-node and multi-node machines considered in Fig. 2.

CONCLUSIONS

Users of the code can run the new parallel version using any kind of multi-core machine equipped with a modern FORTRAN compiler and MPI libraries. We have optimized both the serial and parallel part of the code in a way that aims at improving performance without requiring extra expertise from the users while keeping a simple, easy-to-maintain code. In particular, we did not make use of native MPI I/O (Prabhat and Koziol (2014)), which will require major code design revision, extra system requirements and user expertise; we leave the MPI I/O implementation for a future release of the code. For the same reasons, we did not consider GPU programming.

Performance results show a nice balance between the point at which file input and output dominates parallel speedup and the great performance improvement when using a high-performance machine. We show that the performance on a general-purpose cloud environment might be limited by network communications between the computing nodes, which should be tuned for this specific purpose. Most users would find the possibility of running the code on demand on MPI-optimized cloud resources more appealing than on massively multi-core machines. Our performance assessment, though, shows that a straightforward use of a cloud environment might not be rewarding, and extra tuning of the cloud software for out MPI-enabled code mighj be necessary, which is probably true for many MPI applications.

In conclusion, availability of a parallel version of TRIGRS enables simulation of landslide initiation from storm events affecting large heterogeneous regions. This greatly aids analysis of the effects of rainfall patterns and terrain complexity on landslide initiation.

SOFTWARE AVAILABILITY

The software is available for download at `http://geomorphology.irpi.cnr.it/tools/trigrs` and `https://github.com/usgs/landslides-trigrs`. The datasets used in this work and in Alvioli and Baum (2016) are available in Ref. Baum et al. (2016).

ACKNOWLEDGEMENTS

M. Alvioli acknowledges the CINECA award under the ISCRA initiative, for the availability of high performance computing resources and support. M. Alvioli thanks CNR for a "Short Mobility Grant", 2014, during which part of this work was completed.

REFERENCES

Alvioli, M. and Baum, R. (2016). Parallelization of the {TRIGRS} model for rainfall-induced landslides using the message passing interface. *Environmental Modelling & Software*, 81:122 – 135.

Alvioli, M., Guzzetti, F., and Rossi, M. (2014). Scaling properties of rainfall-induced landslides predicted by a physically based model. *Geomorphology*, 213:38–47.

Baum, R. L., Godt, J. W., and Savage, W. Z. (2010). Estimating the timing and location of shallow rainfall-induced landslides using a model for transient, unsaturated infiltration. *Journal of Geophysical Research: Earth Surface*, 115(F3). F03013.

Baum, R. L., Jones, E. S., Alvioli, M., Kean, J. W., and Godt, J. W. (2016). Map and model input and output data covering n 40.0^o – 40.375^o and w 105.25^o – 105.625^o in the northern Colorado Front Range for analysis of debris flow initiation resulting from the storm of September 9 – 13, 2013. `http://www.mpi-forum.org/` (Sep. 2015).

Baum, R. L., Savage, W. Z., and Godt, J. W. (2008). *TRIGRS- A Fortran Program for Transient Rainfall Infiltration and Grid-Based Regional Slope-Stability Analysis, Version 2. 0*. Number 2008-1159, 75 p. in Open-File Report.

Iverson, R. M. (2000). Landslide triggering by rain infiltration. *Water Resources Research*, 36(7):1897–1910.

Mergili, M., Marchesini, I., Alvioli, M., Metz, M., Schneider-Muntau, B., Rossi, M., and Guzzetti, F. (2014). A strategy for GIS-based 3-D slope stability modelling over large areas. *Geoscientific Model Development*, 7:2969–2982.

MPI Forum (2012). Message Passing Interface (MPI) Forum Home Page. `http://www.mpi-forum.org/` (Sep. 2015).

Prabhat and Koziol, Q. (2014). *High Performance parallel I/O*. Chapman and Hall/CRC, Boca Raton, USA.

Raia, S., Alvioli, M., Rossi, M., Baum, R. L., Godt, J. W., and Guzzetti, F. (2014). Improving predictive power of physically based rainfall-induced shallow landslide models: a probabilistic approach. *Geoscientific Model Development*, 7(2):495–514.