A Parallel Python Application for Seismic-volcanic Event Location

Guillermo Cornejo-Suárez, Leonador Van der Laat, Esteban Meneses

***Abstract*—Seismic-volcanic signals are harder to locate than seismic-tectonic signals. The difficulty lies in the emergent waveform of seismic-volcanic signals. The alternative methods requires an exhaustive search over the volcano's volume in order to find the most likely epicenter. We present the algorithmic analysis of an accepted method to locate seismic-volcanic signals and the design of a parallel application that implements it.**

We demonstrated that the implementation scales linearly with the number of events and the number of nodes. From the experimental data, we developed a model to predict execution time with an accuracy of 90%. We discuss widely about model derivation and assumptions. In special, the practical advantages of using simple experimental models instead of complex ones, if the simpler one has high accuracy. In that case the the ignored factors are represented by the error model.

I. Introduction

In this work, we present the design and analysis of a parallel Python application for locating the source of seismic-volcanic signals. Seismic activity in volcanoes isn't limited to classical tectonic activity, it includes signals radiated from the motion of internal fluids and gases. These signals are very different from tectonic ones, which makes then impossible to study with the standard methods [1].

Tectonic signals are easy to *pick* because its waveform has an abrupt beginning and the event last a few seconds (top part of Figure 1). In the other hand, the volcanic signal waveform is *emergent*, it ramps up slowly from background noise and can last days. The classic method for seismic event epicenter location depends on clearly identifying the event's beginning, therefore, it cannot be used on volcanic signals [1], [2].

The workaround method involves extracting features from the signal at seismometers and then simulating how seismic propagation affects those features. This is calculated for every point in a regular grid over the volcano's volume. The grid point that produces the smaller error is chosen as the most likely event epicenter [1], [3], [4], [5], [6].

Naturally, this method is sensitive to grid resolution. The finer it becomes the more precise it gets, but also increases computing time. Therefore, researchers must trade-off grid resolution for computing time. There are two obvious strategies to diminish this problem, not mutually excluding: an smarter search algorithm and parallelism.

The first one uses an optimization technique like hill climbing or random walk. Nevertheless, it's sill possible to increase grid resolution until computing time turns unacceptable. The second, in its simplest form, uses brute force to evaluate every grid point, but divides the work in many processors, effectively dividing computing time by the number of processors. We will show that even this naive approach works if the number of processors scales proportionally to the number of grid points.

In this work, we build a parallel implementation of the method proposed by [1]. First, we analyze the algorithm looking for data interdependence and parallelism opportunities (section III), we conclude that all tasks could be executed in parallel using a single reduce instruction at the end (section IV). Second, we present a parallel implementation using the Message Passing Interface Standard (MPI) and demonstrate it scales linearly, as predicted by theory (section V). Finally, we present a simple experimental model to estimate execution time.

II. The Method

In classic seismology, the tectonic signal source is modelled as an explosion with orientation and the ground effect as a linear system. Therefore, the signal perceived by the seismometer is modelled by:

$$m(\mathbf{x}, t) = r(t) * g(\mathbf{x}, t; \mathbf{x_s}) * s(\mathbf{x_s}, t) \tag{1}$$

Where:

- $s(\mathbf{x_s}, t)$ is the source signal with epicenter at $\mathbf{x_s}$.
- $g(\mathbf{x}, t; \mathbf{x_s})$ is the linear system that models how the source signal is perceived at $\mathbf{x}$.
- $r(t)$ is the instrument response.
- $m(\mathbf{x}, t)$ is the ground motion at station location $\mathbf{x}$.

The standard method for seismic-tectonic event location [7] considers only time of arrival to every seismic station, essentially reducing the waveform to:

$$m(\mathbf{x}, t) = u(\mathbf{x}, \mathbf{x_s}) * s(\mathbf{x_s}, t) \tag{2}$$

Where $u(\mathbf{x}, \mathbf{x_s})$ is the delay function and models how fast the seismic waves propagates. As mentioned, in general, it's not possible to clearly identify the beginning of seismic-volcanic signals. As figure 1 shows, the tectonic signal starts abruptly, while the volcanic signal ramps up slowly from background noise. A simple analogy is to compare the beginning of a thunder with the beginning of a teapot whistle. The seismic-tectonic signal would be the thunder and the seismic-volcanic one would be the teapot whistle.

Thus, another waveform characteristic different from time of arrival must be used. Seismic waves, as any other mechanical

Guillermo Cornejo-Suárez and Esteban Meneses are with the Colaboratorio Nacional de Computacin Avanzada, del Centro Nacional de Alta Tecnologa and with the Tecnolgico de Costa Rica.

Leonardo Van der Laat was with the Observatorio Vulcanolgico y Sismolgico de Costa Rica

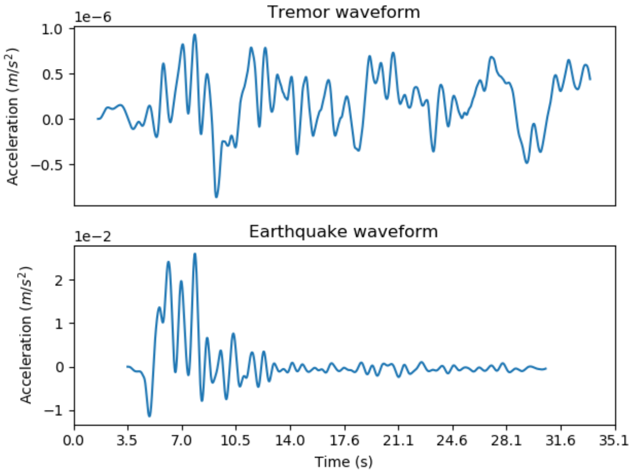


Figure 1: Waveform difference between a seismic-volcanic signal and a seismic-tectonic one. First, notice that the tremor amplitude is in the same order of magnitude as the background noise (first 3.5 s of the signal), also it seems similar to noise and last for a long time. The seismic-tectonic signal is around 5 orders of magnitude bigger than background noise, has a clear beginning and last only for a few seconds.

wave that expands in a dissipate medium, losses amplitude as a function of distance. [1] suggest to use Equation (3) to model amplitude decay as function of distance:

$$A(r) = A_0 \frac{e^{-Br}}{r} \tag{3}$$

with

$$B = \frac{\pi f}{Q\beta}$$

Where:

- $A(r)$ is the wave amplitude at a distance r from the source.
- A_0 is the source amplitude.
- f is the source signal's fundamental frequency.
- Q is the quality factors, it models how much energy is transformed into heat every cycle.
- β is the wave velocity at frequency f.

The factor $1/r$ corresponds to the geometric attenuation (or geometrical spreading): as a wave expands, its energy must fill a bigger volume, thence, the energy per unit of volume decreases. The factor e^{-Br} is the anelastic attenuation: how fast energy is transformed into heat. Notice that time doesn't appear explicitly in equation (3), as wanted. Also, the full waveform is reduced to a single value, obtained by the maximum peak or the root mean square (RMS) value.

Before reducing the seismometer signal to a single amplitude value, it must be preprocessed:

1) Deconvolve the instrument response, the $r(t)$ from Equation (1).
2) Convolve with the site effect factor.
3) Band-filter around f, from Equation (3).

Sometimes the shallow layers below the seismometer amplify the seismic signal. The site effect factor is a transfer function that models that amplification [8], [9], [10].

Now, for every grid point (x, y, z) and for a given source amplitude A_0, calculate the distance to every seismometer r and substitude the values in equation (3). Estimate the total error with equation (4), where i iterates over the stations and the superscript *obs* means observed (recorded) signal at every station:

$$Err = \sqrt{\frac{\sum(A_i - A_i^{obs})^2}{\sum(A_i^{obs})^2}} \tag{4}$$

The amplitude value A_0 is also unknown, so it's part of the search space, that means, equations (3) and (4) must be evaluated in a range of amplitudes values. Finally, the pair of source position and amplitude value that produces the minimum error is chosen as the most probable epicenter for the event.

III. Algorithm analysis

Succinctly, the method consist on finding the tuple (x, y, z, A_0) that minimizes the function $Err(x, y, z, A_0, A^{obs})$, as defined in equations (3) and (4). The brute force approach evaluates Err for every possible combination of parameters within the physic and geographic limits. Then, at the end, applies a `min()` reduce operation. The following pseudo-code displays a plain implementation of the algorithm:

Listing 1: A plain implementation of the source location algorithm

```
err is a map of tuple to float

for x in x_range:
    for y in y_range:
        for z in z_range:
            for A_0 in A_range:
                error = 0
                obs   = 0
                for s in station_list:
                    r := s.distance(x, y, z)
                    A := A_0 * attenuation(r)
                    error += power(A - s.
                        amplitude, 2)
                    obs   += power(s.amplitude
                        , 2)
                err[(x, y, z, A_0)] = sqrt(
                    error / obs)
loc := min(err)
```

Every evaluation of Err is completely independent, thus, the algorithm is *embarrassingly parallel*. Also, every event is independent, providing another source of parallelism for locating all the events in a catalog. The challenge is to distribute every evaluation of Err among the cores and nodes of a supercomputer considering the modern many-core architectures and non-uniform memory access.

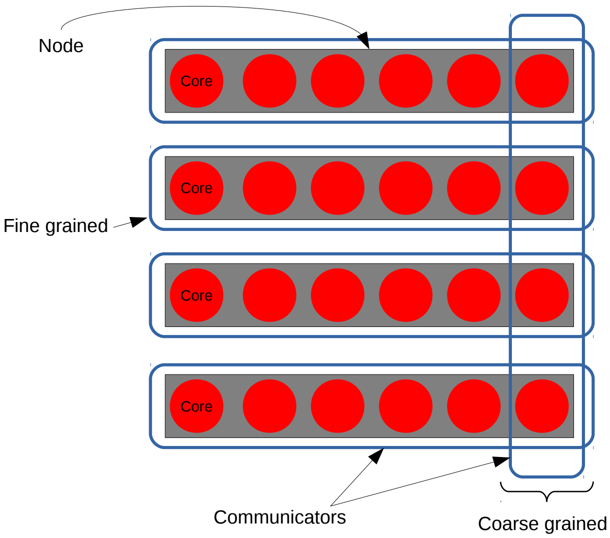


Figure 2: MPI-communicators design. The majority of collective operations are mapped to intra-node communicatorios. Namely, events are distributed using the coarse-grained communicator, while the algorithm from listing 1 is mapped to fine-grained communicators

IV. Program design

The current trend in High Performance Computing (HPC) shows preference for many-core architectures. For example, the fastest computer in the world features a 260 cores processor, while the following nine supercomputers feature processors with 64, 16 or 12 cores [11].

Efficient collective operations are critical for obtaining a good performance of HPC applications. Because intra-node communication is cheaper that multi-node, using collective operations in intra-node communicators instead of multi-node communicators improves performance [12], [13]. Hence, we mapped the majority of required collective operations to intra-node communicators.

From the global communicator, we used the IP address to group ranks in the same node to a new communicator, named *fine-grained*. There are as many fine-grained communicators as nodes, as Figure 2 displays. All the zero ranks from every fine-grained communicators are grouped into a new communicator, named *coarse-grained*.

Events are distributed using the coarse-grained communicator. Also, results are collected by rank zero at that communicator and printed. The search space (i.e. x, y, z and A_0) is divided between all the ranks belonging to the same fine-grained communicator. So, the final `min()` reduce operation displayed in Listing 1 (which is a collective operations) is executed intra-node only.

Table I shows a description of the parameters for seismic-volcanic event location. The last three parameters (stations list, events list and Digital Elevation Model) are easy to generate. They can be retrieved from sources like IRIS[1], but in general, seismic observatories will host repositories of relevant local data.

Table I: Parameters list for the seismic-volcanic event location

Parameter	Description
A_i	Initial amplitude value
A_f	Final amplitude value
dA	Amplitude step
z_{range}	Maximum depth
f	Signal fundamental frequency
β	Seismic wave speed at frequency f
Q	Attenuation factor
Stations list	Specifies station name a location
Events list	Contains signal amplitude for every station at each event
DEM	A digital elevation model in `asc` format

The f, β and Q parameters must be adjusted by the researcher, using her knowledge of the volcano singularities and characteristics. z_{range}, A_i, A_f and dA are actually part of the search space and must be adjusted iteratively as part of the methodology refinement. That means, the search range must be reduced to contain the solution and avoid examining empty space.

V. Experimental evaluation

A. Validation

Turrialba Volcano is a stratovolcano located in the central region of Costa Rica. Its edifice heights 1900 meters, peaking 3340 meters above mean sea level. It belongs to the *Coordillera Central* and shares its basement with the Irazú Volcano [14].

Since 1996, national seismic observatories registered a change in seismicity and emanated gas composition at Turrialba Volcano. Seismic and degasification activity intensified since 2003 and finally the volcano entered into erupting activity in 2007, with peaks of activity in the following years. The *Red Sismolgica Nacional* estimates that around two million people could be affected by the activity of Turrialba Volcano [14].

To test out if our implementation output has physical sense, we processed 430 LP-tremor events that occurred from May to June 2016. Table II specifies the parameter values used in the location process. Figure 3 displays the number of events located at each grid point.

Because there aren't any previous experiments on the field, we relay on the expert opinion to evaluate software correctness. According to seismologists from *Red Sismológica Nacional* and *Observatorio Vulcanológico y Sismológico de Costa Rica*, the number and distribution of events, as displayed in figure 3 matches the expected behavior. The events are aligned from south-west to north-east, and the biggest number of events appears under the summit.

B. Experimental setup

The software was tested at the Kabré supercomputer, hosted by *Centro Nacional de Alta Tecnología*. It consists of 20 nodes interconnected with 1Gb ethernet. Each node has a Intel Xeon Phi 720 with 64 cores at 1.6 GHz and 96 GB of memory.

[1] https://www.iris.edu/hq/

[2] ASTER GDEM is a product of NASA and METI

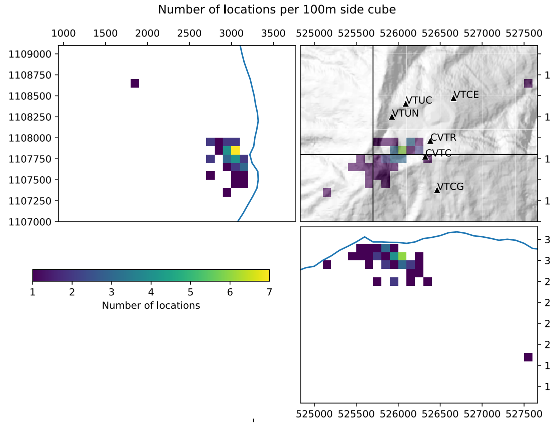


Figure 3: Location of 430 LP-tremor events occurred from May to June 2016. The heatmap shows the number of events located at every 100 m cubic cell.

Table II: Parameters used in the software validation

Parameter	Value
A_i	0.0
A_f	0.007
dA	0.0001
z_{range}	2000
f	2 Hz
β	2300 m/s
Q	50
DEM	Aster GDEM 2005^2, 100 m resolution

C. Scalability

From section III we concluded that the location algorithm is embarrassingly parallel, therefore it should exhibit nearly linear speedup with respect to the number of processing units. Also, for the same reason, the compute time must exhibit linear growth with respect to the number of events it receives as input. Because we use all cores in a single node to explore the search space, in practice, our processing unit isn't the number of cores but the number of nodes.

Table III features the result of measuring compute time with different values of the indipendent variables (number of nodes and number of events). The number of nodes goes from one to six, which is the maximum number of nodes allowed by the Kabré's batch system. We fixed the number of events to 430.

Table III: Elapsed time for different number of nodes and number of events

Nodes	Events	Median Time (s)	Repetitions
6	430	1080	5
5	430	1267	5
4	430	1569	5
3	430	2034	5
2	430	3002	5
1	430	5864	5
6	6	72.01	5
6	12	89.87	5
6	24	114.2	5
6	48	173.3	5
6	96	285.2	5
6	192	504.5	5
6	384	968.2	5

Table IV: Elapsed time predicted by the model in equation (5) and the experimental evaluation

Nodes	Events	Predicted Time (s)	Elapsed Time (s)	Error (%)
3	92	465.8	462.3	0.1
1	272	3286	4002	17.9
5	31	142.7	112.6	26.7
4	348	1231	1035	19.0
6	12	85.70	52.15	64.3
5	157	485.8	450.1	7.9
4	226	820.5	846.4	3.1
5	93	311.5	282.1	10.4
2	315	2090	2186	4.4
6	121	334.5	312.7	7.3

In the other test set, we fixed the number of nodes to 6 and increased the number of events by multiplying the previous by two, starting from 6 events until 384 events. For every test, we made five repetitions. Table III shows the median time of those repetitions. We chose the median time over the mean time to reduce the possible bias introduced by outliers.

To make evident the linear behavior of the implementation, we ploted speedup and elapsed time in Figure 4. For both, we calculated the trend line. This gives us an handy linear model to estimated the elapsed time at any configuration of number of nodes and events:

$$\begin{aligned} epn &= e/(0.89n + 0.17) \qquad (5) \\ t &= 2.36(0.89 \cdot 6 + 0.17)epn + 58.31 \end{aligned}$$

Where:

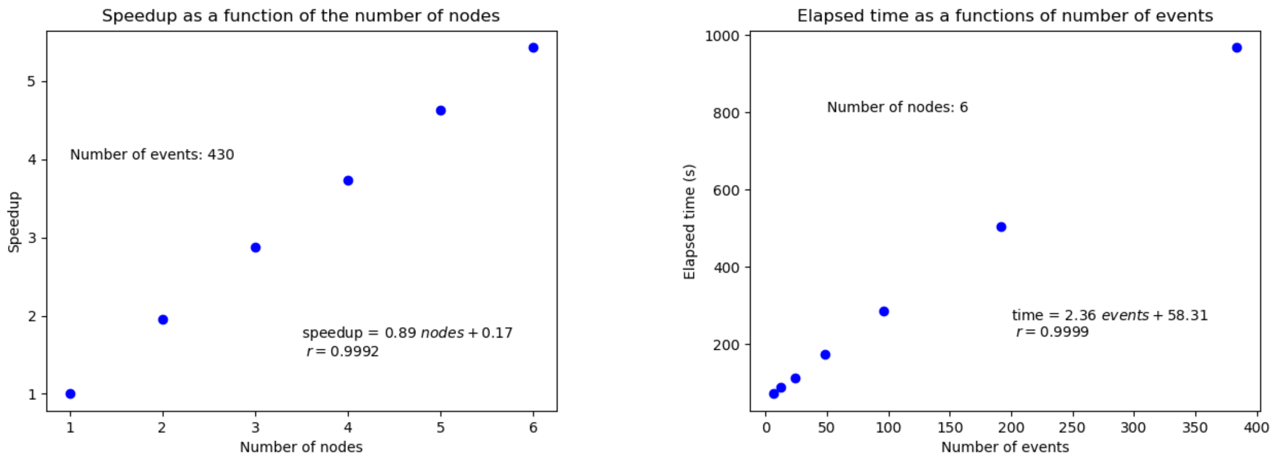
- e is the number of events.
- n is the number of nodes.

In Section VI we will further discuss this model. By now, our interest is to evaluate it's accuracy. In order to do so, we randomly sample ten ordered pairs of number of nodes and number of events. We used a uniform probability distribution and sampling with replacement. Table IV features the result of evaluating the model by the sampled ordered pairs.

We don't perform the evaluation tests in Table IV multiple times. Instead, we argue that by increasing the number of independent test, the percent error will converge to the mean value. This reasoning is justified by the Central Limit Theorem, if we assume that the model error is equal for every point in the experimental domain (from 1 to 6 nodes and from 6 to 430 events). The median error value is 10.4%. That value is accurate enough to estimated the computing time.

VI. Discussion

From the analysis in Section III we discovered that every evaluation is completely independent from the others and the algorithm requires only a `min()` reduce operation at the end. Also, every event location is independent from the others, providing another source of parallelism. If the required time



(a) Speedup as a function of the number of nodes

(b) Elapsed time as a function fo number of events

Figure 4: Speedup and elapsed time of the implementation. The implementation displays linear behavior.

for every evaluation of the Err function (t_{err}) and the `min()` reduction (t_r) is known, we could calculate a lower bound for the required time to solve the location problem for a data set.

Hence, with e the total number of events in the data set and g the number of grid points, we have:

$$T = e\ t_r + g\ t_{err} \tag{6}$$

Conceding that an implementation exhibits perfect linear scaling, the total time would be $T_p = T/w$ where w is the number of *workers*. That means, the work is perfectly distributed among participating workers. Nevertheless, in practice that's very difficult to achieve with an static partition (i.e. the work is distributed before the computation begins). That happens because of two reasons:

- The number of operations is not divisible by the number of workers. This leads to a small load imbalance.
- The operations are not truly independent. Here, the `min()` operations needs the $Err()$ value for every grid point to evaluate. This imposes an order of evaluation that also could lead to load imbalance.

In principle, it is possible to predict the effect of load imbalance with a static distribution of the work. Moreover, because it depends on the residue of dividing the work load among the workers, in many cases it shows a periodic behavior as a function of work load, as figure 5 shows. Nevertheless, restrictions like the order of evaluation make hard to predict that behavior.

The alternative is using a Run Time System (RTS) that distributes the work at run time. A RTS offers flexibility and adaptability, but is more sensible to communication cost between workers physically separated at different machines. For our case, two considerations are relevant:

- With a static workload distribution, time invested in management reduces to zero. While with a RTS, time invested in management, in general, isn't negligible.
- Collective operations, like `min()`, executed through the network can be costly.

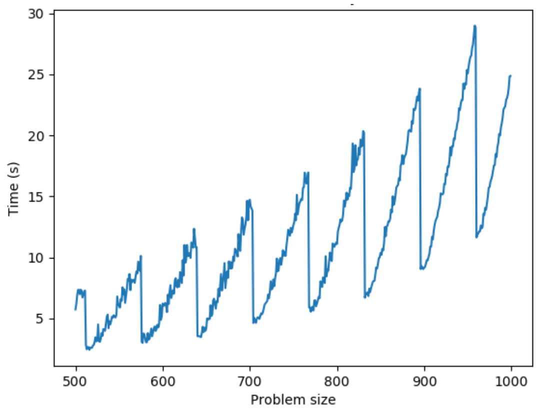


Figure 5: Elapsed time as a function of problem size (load). This is an example of the periodic behavior of load imbalance. The residual work that cannot be equally distributed are assigned to a single worker, thus, increasing total execution time.

This application is simple enough to reach an optimum static workload distribution. If programmed with flexibility in mind, the first stage could consist of workers configuration analysis. Different workload distribution schemes could be evaluated using a more complex model than the derived in this work, in order to find a distribution that minimizes execution time.

A. About the experimental model

The model presented in equations (5) was deduced experimentally. First, assume that exist a function F that takes the number nodes and the number of events and yields the execution time, $F(n, e) \to IR$. Even when F isn't linear with respect to n nor e, if the problem is truly embarrassing parallel,

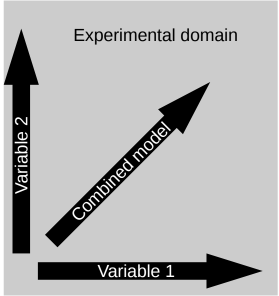


Figure 6: A combined (not necessarily linear) model can be derived from simpler ones that explains how the quantity of interest follows every orthogonal variable.

as we conclude in Section III, Then n and e must be orthogonal variables.

Orthogonality means that the variables could be manipulated individually. In this model, we can fix one and modify the other to find a linear model that explains how F changes as the modified variable changes. Combining those linear models generates a model for F That correctly extrapolates to any point in the experimental domain in which we sampled both variables. Figure 6 visually represents that idea.

Figures 4a and 4b show those simple lineal models that explain how F changes as a function of n with e fixed and the other way around. There are two ways to represent mathematically the meaning of speedup. The first one, and most common, is to interpret speedup as a factor that divides total computing time. The other way is to interpret it as a factor that divides total work. That's the meaning of the first equation in (5). The second equation is a linear fit of how much time it takes to compute *epn* events. The trend was derived using six nodes, so it includes the *speedup* provided by those extra nodes, thus the epn value is increased by the constant factor $0.89 \cdot 6 + 0.17$ to compensate that extra speedup.

Making a case for this methodology, it is very simple to derive and only requires an appropriate sampling. It models, in a best effort fashion, ignored considerations, like network delay, start-up time, load imbalance and similar. Nevertheless, it's valid only for the experimental domain. Theory predicts non-linear behavior of any application for big enough parameters, that is, millions of cores or billions of grid points. This model doesn't captures that behavior.

A more complex model could capture that behavior by taking into account many of the considerations we ignored. Although, it implies measuring individual elements, an extra effort that doesn't pay off if the simpler model accuracy is acceptable.

VII. Related work

The method presented in this paper was proposed and evaluated in [1], [6]. It suits the equipment installed on Turrialba Volcano: an un-arranged seismic network in which every station position was choose based on geophysics, land ownership and practical criteria. [5], [15] present a method to locate seismic-volcanic events using triangular seismic antennas. In that configuration it's possible to use seismic wave *slowness* to estimate its orientation and triangulate its epicenter using a small network of seismic antennas.

[16] present a novel method to locate seismic-volcanic events using the cross-correlation between every pair of stations. The idea is the following, it assumes that $g(\mathbf{x}, t; \mathbf{x_s})$, the linear system from Section II that models ground response, is similar for all stations and the signal is the same, therefore, the signal in every station must be very similar. Hence, the cross-correlation peaks a global maximum when the signal from two stations is in phase. Using the phase difference, i.e. the delay between the peaking in both signals, it's possible to estimate the source location. This method is sensible to anomalies in the velocity model used, as much as the amplitude method presented in our work.

Notwithstanding, mentioned publications don't report about code implementation or its availability. Similar software for the area, for example [17], [18], focus on automatic signal classification. Thence, our work is a contribution to the area.

VIII. Conclusions

In this work we presented the design and implementation of a parallel python application for seismic-volcanic event location. These signals are harder to locate that seismic-tectonic ones because of its source mechanics and heterogeneous propagation medium. The location method used in the implementation extracts the signal amplitude and models its decay as a function of distance. A brute force approach evaluates all possible solutions to find the one that produces the minimum error.

From the method analysis, we concluded it's embarrassingly parallel, requiring only a `min()` reduce operation to find the optimum source location. The design aim to exploit the many-core architecture commonly available in HPC platforms by mapping collective operations to intra-node communicators. We tested the implementation scalability with respect to the number of nodes and the number of events. For both variables it shows linear behavior.

From experimental data we developed a model to predict execution time given the number of events and number of nodes. The model has an accuracy of 90%. We argue that simple experimental models are preferable over complex ones if accuracy is high enough to estimate computing time. In other words, the effect of ignored factors can be represented as the model error.

Acknowledgment

The authors would like to thank Javier Pacheco, from the Observatorio Vulcanolgico y Sismolgico de Costa Rica and Mauricio Mora, from the Red Sismolgica Nacional, for their significant guidance.

References

[1] J. Battaglia and K. Aki, "Location of seismic events and eruptive fissures on the piton de la fournaise volcano using seismic amplitudes," *Journal of Geophysical Research: Solid Earth*, vol. 108, no. B8, 2003. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2002JB002193

[2] T. Diehl and E. Kissling, “Users Guide for Consistent Phase Picking at Local to Regional Scales,” Swiss Federal Institude of Technology Zurich, Tech. Rep., 2008.

[3] H. Kumagai, R. Lacson, Y. Maeda, M. S. Figueroa, T. Yamashina, M. Ruiz, P. Palacios, H. Ortiz, and H. Yepes, “Source amplitudes of volcano-seismic signals determined by the amplitude source location method as a quantitative measure of event size,” *Journal of Volcanology and Geothermal Research*, vol. 257, pp. 57 – 71, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377027313000693

[4] B. Taisne, F. Brenguier, N. M. Shapiro, and V. Ferrazzini, “Imaging the dynamics of magma propagation using radiated seismic intensity,” *Geophysical Research Letters*, vol. 38, no. 4, 2011. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2010GL046068

[5] J. Mtaxian, P. Lesage, and B. Valette, “Locating sources of volcanic tremor and emergent events by seismic triangulation: Application to arenal volcano, costa rica,” *Journal of Geophysical Research: Solid Earth*, vol. 107, no. B10, pp. ECV 10–1–ECV 10–18, 2002. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2001JB000559

[6] G. D. Grazia, S. Falsaperla, and H. Langer, “Volcanic tremor location during the 2004 mount etna lava effusion,” *Geophysical Research Letters*, vol. 33, no. 4, 2007. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2005GL025177

[7] K. Aki and P. G. Richards, *Quantitative Seismology*, 2nd ed. University Science Books, 2002.

[8] K. Mayeda, S. Koyanagi, and K. Aki, “Site amplification from s-wave coda in the long valley caldera region, california,” *Bulletin of the Seismological Society of America*, vol. 81, no. 6, p. 2194, 1991.

[9] S. Koyanagi, K. Aki, N. Biswas, and K. Mayeda, “Inferred attenuation from site effect-correctedt phases recorded on the island of hawaii,” *pure and applied geophysics*, vol. 144, no. 1, pp. 1–17, Mar 1995. [Online]. Available: https://doi.org/10.1007/BF00876471

[10] K. Kato, K. Aki, and M. Takemura, “Site amplification from coda waves: Validation and application to s-wave site response,” *Bulletin of the Seismological Society of America*, vol. 85, no. 2, p. 467, 1995.

[11] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer. (2017) The top 500 list. [Online]. Available: https://www.top500.org/lists/2017/11/

[12] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra, “Performance analysis of mpi collective operations,” in *19th IEEE International Parallel and Distributed Processing Symposium*, April 2005.

[13] R. Thakur and W. D. Gropp, “Improving the performance of collective operations in mpich,” in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, J. Dongarra, D. Laforenza, and S. Orlando, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 257–267.

[14] Red Sismolgica Nacional. (2018) Volcanes de costa rica ii: Turrialba. [Online]. Available: http://rsn.ucr.ac.cr/component/content/article/109-vulcanologia/volcanes-de-costa-rica-ii/32-turrialba?Itemid=225

[15] B. Di Lieto, G. Saccorotti, L. Zuccarello, M. L. Rocca, and R. Scarpa, “Continuous tracking of volcanic tremor at mount etna, italy,” *Geophysical Journal International*, vol. 169, no. 2, pp. 699–705, 2007. [Online]. Available: +http://dx.doi.org/10.1111/j.1365-246X.2007.03316.x

[16] K. L. Li, G. Sgattoni, H. Sadeghisorkhani, R. Roberts, and O. Gudmundsson, “A double-correlation tremor-location method,” *Geophysical Journal International*, vol. 208, no. 2, pp. 1231–1236, 2017. [Online]. Available: +http://dx.doi.org/10.1093/gji/ggw453

[17] M. Masotti, R. Campanini, L. Mazzacurati, S. Falsaperla, H. Langer, and S. Spampinato, “Tremorec: A software utility for automatic classification of volcanic tremor,” *Geochemistry, Geophysics, Geosystems*, vol. 9, no. 4, 2008. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2007GC001860

[18] A. Messina and H. Langer, “Pattern recognition of volcanic tremor data on mt. etna (italy) with kkanalysisa software program for unsupervised classification,” *Computers and Geosciences*, vol. 37, no. 7, pp. 953 – 961, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098300411001294