Using parallel computing for seismo-volcanic event location based on seismic amplitudes

Guillermo Cornejo-Suárez†, Leonardo Van der Laat‡§, Esteban Meneses∗†, Javier Pacheco§, Mauricio M. Mora‡§

†School of Computing, Costa Rica Institute of Technology
‡Advanced Computing Laboratory, Costa Rica National High Technology Center
§Escuela Centroamericana de Geología, Universidad de Costa Rica

Email: gcornejo@cenat.ac.cr, leonardo.vanderlaat@ucr.ac.cr, emeneses@cenat.ac.cr
javier.pacheco.alvarado@una.cr, mauricio.mora@ucr.ac.cr

Abstract—Volcanoes are very complex geophysical systems where fluids of different nature interact with porous rock at different physical conditions and within a complex matrix of conduits. Two types of seismicity are generated by this complex interactions. The first type is characterized by fracture of the elastic media, in which we have the volcano-tectonic events (VT) that produce two distinctive phases: a compressional phase (P wave) and a shear wave (S wave) that travels with different velocities within solid media. The second type is characterized by low frequencies, in which we have a wide variety of long period events (LP) and volcanic tremors. These signals are produced by fluid motion within restricted paths and have normally emergent onsets and no distinctive P or S wave phases. Classical earthquake source location procedures take advantage of the distinctive phases and their different propagation velocity. However, for LP events and tremors, those procedures can not be used. Therefore, complex algorithms have to be applied, demanding much more computer resources and time than the classical location methods.

In this work, we present the analysis and design of a parallel approach for locating the source of LP and volcanic tremor signals. Besides tectonic or rock fracture seismic signals, volcanoes produce signals radiated from the motion of internal fluids within constraint conduits. These signals are very different from tectonic ones, which make them impossible to study with standard methods [1].

In classical seismology, in order to locate a seismic event source, the 3D waveform recorded at every seismic station is reduced to a single value. This procedure could be understood as a feature extraction. For example, tectonic signals are located using the arrival times of P and wave to every station. Then, using a geophysical model of seismic wave velocity, the hypocenter is located as the most likely point to produce that difference in the time of arrival between the P and S components.

Tectonic signals are easy to pick because its waveform has an abrupt beginning and the event last a few seconds (bottom part of Figure 1). On the other hand, other volcanic signals are emergent, they ramp up slowly from background noise and can last days (top part of Figure 1). The standard method for seismic event hypocenter location depends on clearly identifying the beginning of the P and S components, therefore, it cannot be used on volcanic signals [1], [2].

The workaround method involves extracting another feature from the signal at seismic records and then simulating how seismic propagation affects that feature. 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 hypocenter [1], [3]–[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 approach uses an optimization technique like hill climbing or random walk. Nevertheless, it’s still possible to increase grid resolution until computing time turns unacceptable. The second approach, in its simplest form, uses brute force to evaluate every grid point, but divides the work within many processors, effectively dividing computing time by the number of processors. We chose the second alternative because it’s simpler to implement and ensures finding the point of minimum error. Later implementations could explore mixed approaches and use the one presented here as a reference.

In this work, we build a parallel implementation of the method proposed in [1] (explained in Section II). 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. Second, we present a parallel design.
The seismo-tectonic signal would be the thunder while the seismo-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 wave that expands in a dissipative medium, loses amplitude as a function of distance. Authors in [1] suggest to use (3) to model amplitude decay as function of distance:

$$A(r) = A_0 e^{-Br/r}$$

where:
- $A(r)$ is the wave amplitude at a distance $r$ from the source.
- $A_0$ is the source amplitude.
- $B$ is a constant defined by $(\pi f)/(Q\beta)$
- $f$ is the source signal fundamental frequency.
- $Q$ is the quality factor, it models how much energy is transformed into heat every cycle.
- $\beta$ 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 $e^{-Br/r}$ is the anelastic attenuation: how fast energy is transformed into heat. Notice that time doesn’t appear explicitly in (3), as wanted. Also, the full waveform at every seismometer is reduced to a single value $A(r)$, obtained by the maximum peak or the root mean square value.

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

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

Sometimes, the shallow layers below the seismometer amplify the seismic signal. The site effect factor is a transfer function that models that amplification [9]–[11].

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

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

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 physically meaning amplitude values. Finally, the pair of source position and amplitude value that produces the minimum error is chosen as the most probable hypocenter for the event.

### III. Algorithm Description

Succinctly, the method consists on finding the tuple $(x, y, z, A_0)$ that minimizes the function $Err(x, y, z, A_0, A_{obs})$, as defined in (3) and (4). The brute
force approach evaluates \( Err \) for every possible combination of parameters within the physical and geographic limits. Then, at the end, a \( \min() \) reduce operation is applied. The pseudo-code in Algorithm 1 displays a plain implementation of the algorithm.

**Algorithm 1** Source location algorithm

Require: \( err: \) map of tuple to float
Require: \( Xrange: \) list of \( x \) values
Require: \( Yrange: \) list of \( y \) values
Require: \( Zrange: \) list of \( z \) values
Require: \( Arange: \) list of amplitude values
Require: \( Stations: \) a list of stations

for \( x \) in \( Xrange \) do
  for \( y \) in \( Yrange \) do
    for \( z \) in \( Zrange \) do
      for \( A0 \) in \( Arange \) do
        error := 0
        obs := 0
        for \( s \) in \( Stations \) do
          \( r := s.distance(x, y, z) \)
          \( A := A0 \cdot e^{(-B \cdot r)}/r \)
          error += (\( A - s.amplitude \))^2
          obs += (\( s.amplitude \))^2
        end for
        err[(x, y, z, A0)] = sqrt(error/obs)
      end for
    end for
  end for
end for

\( loc := \min(err) \)

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.

IV. PROGRAM DESIGN

This section briefly introduces MPI, which is used to build the software, and then explains the design.

A. The Message Passing Interface

The Message Passing Interface standard defines a small group of functions to implement parallel programs using the message passing computational model. In this model, every process has the same program text, but the runtime system that initiates the system assigns an unique ID to every process, known as *rank*. Because they're separated processes, they have independent address spaces [12], [13]. Figure 2 shows a graphic representation.

With the rank number, it's easy to implement point-to-point communication, using a minimal interface:

send(\( message, destination, tag \))
receive(\( message, source, tag \))

Destination and source are rank numbers. The \( tag \) value is used to match every send operation with its corresponding receive operation. The rank ID is given to every process that belongs to the same *communicator*. Communicators are an MPI abstraction, it groups together process and assigns an unique ID to everyone. They are useful to implement *collective operations*, which are operations involving multiple ranks.

For example, calculating the arithmetic average of a big array using point-to-point communications would require for loops to distribute the data and gather the partial results. Instead, using collective operations over a communicator, the same calculation could be implemented with a scatter operation to distribute the array and a sum reduce operation:

\[
\text{local array} := \text{comm.scatter(array)}
\]
\[
\text{partial sum} := \text{local array.sum()}
\]
\[
\text{total sum} := \text{comm.reduce(local_array, SUM)}
\]

Hence, collective operations allow us to write concise code without using verbose point-to-point communications.

B. Proposed design

As explained in Section III, every evaluation of \( Err(x, y, z, A_0, A_{obs}) \) is independent, and could be independently distributed among the ranks. We don't use scatter operations to distribute the requirements \( (x, y, z \) and \( A \) ranges), instead, we use the rank number to deduce those values for every rank. Thence, reducing data transmission.
### Table I: Parameters list for the seismo-volcanic event location

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_i$</td>
<td>Initial amplitude value</td>
</tr>
<tr>
<td>$A_f$</td>
<td>Final amplitude value</td>
</tr>
<tr>
<td>$\Delta A$</td>
<td>Amplitude step</td>
</tr>
<tr>
<td>$\tau_{\text{range}}$</td>
<td>Maximum depth</td>
</tr>
<tr>
<td>$f$</td>
<td>Signal fundamental frequency</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Seismic wave speed at frequency $f$</td>
</tr>
<tr>
<td>$Q$</td>
<td>Attenuation factor</td>
</tr>
<tr>
<td>Stations list</td>
<td>Specifies station name a location</td>
</tr>
<tr>
<td>Events list</td>
<td>Contains signal amplitude for every station at each event</td>
</tr>
<tr>
<td>DEM</td>
<td>A digital elevation model in.asc format</td>
</tr>
</tbody>
</table>

At the end, the $\min()$ operation in line 18 of Algorithm 1 is implemented as a collective operation. Rank zero reads events from a catalog and distributes the event data (amplitude values at every station) using a broadcast collective. This is illustrated in Algorithm 2.

#### Algorithm 2 Program design

**Require:** events: catalog of events

1. {Deduce x, y, z and A ranges from rank number}
2. for event in events do
3.    event = communicator.broadcast(event)
4.    local_loc := {Call Algorithm 1}
5.    location := communicator.reduce(local_loc, MIN)
6. end for

Table I shows a description of the parameters for seismo-volcanic event location. The last three parameters (stations list, event 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.

The $f$, $\beta$ and $Q$ parameters must be adjusted by the researcher, using her knowledge of the volcano singularities and characteristics. $\tau_{\text{range}}$, $A_i$, $A_f$ and $\Delta A$ 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 places without physical meaning.

### 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 Coordindilla 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

\(1\)https://www.iris.edu/hq/ a public service to share seismology-related data

Fig. 3: Location of 430 LP-tremor events recorded from May to June 2016 on Turrialba Volcano. Small triangles represent seismic stations, red dots represent hypocenters. We used a 10 m resolution DEM\(^2\).

Red Sismológica Nacional estimates that around two million people could be affected by the activity of Turrialba Volcano [14].

To test out if our implementation’s output has physical meaning, we processed 430 LP-tremor events that occurred from May to June 2016. Figure 3 displays the location of every event and the parameters used in the location procedure.

Because there aren’t any previous experiments on the field, we rely on the expert opinion to evaluate software correctness. According to seismologists from Red Sismológica Nacional and Observatorio Vulcanológico y Sismológico de Costa Rica, event’s distribution 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 32 nodes interconnected with 1Gb ethernet. Each node has an Intel Xeon Phi 720 with 64 cores at 1.3 GHz and 96 GB of memory.

We used the Intel Python distribution version 3.5.2, which links to the Intel MPI implementation from Parallel Studio 2017 update 4. The cluster runs CentOS 7 with Linux Kernel version 4.8.5-16.

#### 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.

\(2\)ASTER GDEM 2005 is a product of NASA and METI
To test out these theoretic predictions, we fixed the number of events to the maximum (430 events) and set the number of nodes from one to six nodes. That means using from 64 to 384 cores. We carried out the experiment twenty times for each configuration. Figure 4a displays the resulting curve.

Similarly, to test if the implementation behaves linearly with respect to the workload, we fixed the number of nodes to six and set the number of events (the load) from 6 to 430 events. Again, we carried out the experiment twenty times for each configuration. Figure 4b displays the resulting curve.

VI. DISCUSSION

Finding the most probable source location of an event is a fundamental step in understanding the process dynamics that generates the phenomenon. For example, in classic seismology, a hypocentral projection is a 3D plot of hypocenters from different events. This figure reveals the geometry of relevant anomalies, like faults and subduction planes. An example can be found at [15].

Seismo-volcanic sources are different in nature from seismo-tectonic ones, nevertheless, the same logic applies when locating its hypocenters. Projecting them could reveal regions of intense seismic activity, which could be correlated with other observations like deformation, water jets and eruptions [6].

Because the complexity of the seismo-volcanic source and the heterogeneous medium, we are forced to use different location methods from the ones used in classical seismology. Even when the method analyzed in this work isn’t mathematically complex, it requires huge amounts of computational power. This justifies a parallel programming approach to solve the location problem.

To harvest the most from a parallel machine, the implementation should exhibit linear scalability. Not all algorithms could be implemented that way. Fortunately, it’s the case for the method presented in [1], as we showed in Section III. From that analysis, we implemented the design proposed in Section IV-B. To test for linearity, we ran that implementation with different configurations of number of cores and number of events, the results are detailed in Section V.

First, we fixed the number of events and manipulated the number of cores. By doing so, we demonstrated that the scaled speedup grows linearly with respect to the number of cores. Besides showing almost linear growth, a perfectly scalable system would have an efficiency of one. It means that if the system receives \( N \) workers it would exhibit an speedup of \( N \) times.

In order to discover our implementation’s efficiency, we calculate a linear regression. The slope of that curve could be interpreted as the average efficiency. From Figure 4a we can conclude that our implementation show linear scalability with an average efficiency of 91%.

In a similar way, from the analysis in Section III we know that the execution time must grow linearly with respect to the number of events. Therefore, to test it out experimentally, we fixed the number of cores and manipulated the number of events. As Figure 4b displays, the elapsed time grows linearly with the number of events. A linear regression over the test has an r-square value of almost one.

It is important to notice that the lines in Figures 4a and 4b aren’t the linear regression, but the median value. Our goal was to give a reference of data dispersion while helping to visualize the linear trend.

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 chose based on geophysics, land ownership and practical criteria. The authors in [5], [16]
present a method to locate seismo-volcanic events using triangular seismic antennas. In that configuration it’s possible to use seismic wave slowness to estimate its orientation and triangulate its hypocenter using a small network of seismic antennas.

In [17] they present a novel method to locate seismo-volcanic events using the cross-correlation between every pair of stations. The idea is the following, it assumes that \( g(x, t; x_0) \), the linear system from (1) that models ground response, is similar for all stations and the source 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 are 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 [18], [19], focus on automatic signal classification. Thence, our work is a contribution to the area.

VIII. CONCLUSIONS

In this paper we presented the analysis of an algorithm for seismo-volcanic source location using the amplitude decay. We demonstrated that the algorithm could be implemented in parallel and achieve almost linear growth. Also, we proposed a design using the Message Passing Interface.

We evaluated an implementation developed in the Python Programming Language. As predicted by the theory, it scales almost linearly with respect to the number of cores and the number of events. We used that program to locate events in the Turrialba Volcano, Costa Rica.

The algorithm analysis as well as the design and the implementation are contributions to the field of volcanic seismology.

ACKNOWLEDGEMENTS

This research was partially supported by a machine allocation on Kabrè supercomputer at the Costa Rica National High Technology Center and by the following research projects: Geofísica y Geodinámica Interna del Arco Volcánico de Costa Rica (113-B5-A00), Apoyo de asistentes a la Sección de Sismología, Vulcanología y Exploración Geofísica (113-A1-716), Vigilancia Sísmica de Costa Rica (113-B5-704) and the Red en sismología computacional para el estudio de los volcanes activos en Costa Rica, all financed by the Vicerrectoría de Investigación of the Universidad de Costa Rica. This research was also possible thanks to the resources coming from the Ley Nacional de Emergencias y Prevención del Riesgo N° 8488. We thank the technical staff of OVSICORI-UNA and Red Sismológica Nacional (RSN: UCR-ICE), particularly Luis Fernando Brenes and Jean Paul Calvo, for seismic stations maintenance.

REFERENCES


Supporting Information

The code described in this work and other useful software for computational seismology is freely available at this github organization: https://github.com/orgs/CNCA-CeNAT.