1. Introduction

Seismic travel time tomography is a technique for determining the crustal velocity structure of a geographic region. Seismic tomography is a computationally intense procedure based on two major conceptual steps: forward modeling [2][3] and inversion [1]. However, overall computation time is dominated by subtasks associated with inversion. This paper presents a software system, named STg, which has a distributed processing architecture design for an implementation of seismic travel time tomography. STg’s front end, which is partly shown in Figures 1 and 3, enhances usability of the tomography software included as the distributed computational back end. STg front-end graphical interface provides multiple validations of the experimental output data to be used by the tomographic algorithm and decouples processing definition from the associated back-end computations.

STg’s computational back-end, which implements the tomographic algorithm, comprises several executables that implement forward modeling as a single step and the inversion subtasks as a set of independently implemented steps. Through each iteration of the tomographic algorithm, a new set of output models is produced that can be used as the input of the next iteration or as the final models produced by the algorithm after model convergence. STg’s enables users to monitor execution of computational steps and progress of model convergence. Since the inversion procedure is nonlinear, obtaining a converging model can be elusive. This leads to a solution based on trial and error, where the ability to distribute computations to a high performance platform and to monitor processing and convergence progress are key for reducing the time needed to obtain a final tomographic model of the studied region.

2. STg’s Multi-tier Design

2.1 Front End

To provide implementation and execution flexibility to the front and the back ends, both are independently portable to other platforms and invocation of the back-end is controlled by a customizable script. This prevents recompilation of the front end for using completely different computational back ends for comparison or enhancement purposes [5]. However, the inclusion of a back end different from the one discussed in this paper would require the replacement to take as input the XML project definition and to comply with the front-end protocol for monitoring the processing progress.

Figure 1. Front end wizard for project definition

2.2 XML Project Definition

The XML tomographic project definition, as shown in Figure 2, serves as metadata that describes the project inputs, outputs, and computation-control parameters to the back-end. The XML definition does not contain any of the actual data from the experimental measurements, which allows users to define projects reusing experimental data with different modeling parameters [4].
The XML project definition contains the following information in the project setup section:

**Username**: Establishes a one-to-many ownership relationship between users and projects.

**Project Name**: Encapsulates and distinguishes the set of parameters used by different processing runs or projects.

**Project Path**: Provides the location of the experimental measurement data and the project definition. This path also defines the location where the project computational outputs will be stored.

**Iterations**: States the number of iterations that will run. The number of iterations, \( n \), is used for estimating execution workload on the back end and for creating the output hierarchy which is returned in \( n \) folders.

The back end extracts information from the project definition about where to store the results of the computations.

**Server**: The server or hosts, tiers \( t_2 \) through \( t_n \), will act as the computational back end of the project.

**Date**: A timestamp with the creation date of the project definition that is added automatically by the front-end when the user saves a project.

**Active**: This attribute is a Boolean value that indicates whether the project definition is complete, and it contains all the validated parameters required by the back end. If **Active** is false, it implies that the project definition contains partial metadata which is valid, but some of the project information is missing.

**ModelType**: Two different starting configurations can be used to execute the steps of the seismic tomography process. **ModelType** specifies whether the starting configuration is a one-dimensional model or a three-dimensional one.

### 2.3 Computational Back End

The architecture can be based on as few as two tiers. However, the back end can be distributed to a collection of tiers, \( t_2 \) through \( t_n \). The front end, or tier \( t_1 \), is used to collect data about a seismic tomography experiment, referred to as a project, which is used to specify the location of experimental output data and to control execution of the tomographic algorithm, which is an implementation of the Vidale-Hole algorithm for seismic travel time tomography \([1][2][3]\) released by J.A. Hole and documented as a workflow. The back end, distributed to tiers \( t_2 \) through \( t_n \), is a collection of executables that perform the computational steps required by seismic tomography.

The tiers communicate using an XML-based tomographic experiment definition. An inter-tier communication protocol enables the front end to obtain information about processing progress from the back end and retrieve partial results for analysis.

The multi-tier design facilitates the use of parallelization and distributed computing to improve execution time. The back end computations can be made using high-performance or distributed computing platforms such as symmetric multi-processors, beowulf-type clusters, and fully distributed environments.

---

**Figure 2. Project definition**

**Figure 3. Front end process monitoring**

---

### 3. Results

STg’s runs produce crustal velocity models and related outputs which have been validated against results from independent executions of the tomographic software that implements STg’s computational back and a published velocity model \([6]\). The current implementation of STg is being used as the computation platform to process a seismic travel time tomography experiment at the Department of Geological Sciences of the University of Texas at El Paso. This utilization of STg will guide development and enhancements for future releases of this software.

### 4. Conclusions

This paper presents STg – a software system for distributed computation of seismic travel time tomography models. While the design of STg’s front end focuses on usability by simplifying specification of inputs and model processing parameters, the design of ST’s back end is focused on reducing execution time of
the tomographic algorithm by using a high-performance computational platform and on a robust implementation of the tomographic algorithm.

5. Future Work

Future work on STg will be based on user feedback and impact of releasing the software to the seismic tomography community. However, it is envisioned that availability of new computational platforms for the back end and enhancement of the various processing steps of the tomographic algorithm will also justify future software releases.

6. Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Grants CREST HRD–0734825, MRI–0923442, and CNS–0837556.

7. References