SOFTWARE REQUIREMENTS SPECIFICATION

for

Ensemble Toolkit for Earth Sciences: Seismic Inversion

Version 0.1.0 approved

Prepared by <Authors>

Princeton University and Rutgers University

April 17, 2017

Contents

Revision History

Name	Date	Reason For Changes	Version
1	16-04-2017	Initial version	0.1.0

1 Introduction

1.1 Purpose

The purpose of the document is to capture, in detail, the requirements of the "Seismic Inversion" track of the "Ensemble Toolkit for Earth Sciences" project. This document will include all the system requirements, interface, interactions and limitations of this track. . This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system.

1.2 Document Conventions

The requested features are listed in section 4 and the non-functional requirements are listed in section 5. Each of these requirements have a priority from the set {HIGH, MEDIUM, LOW}. Based on the number of requirements and their priority, a timeline will be created with each requirement and its expected time-to-completion.

1.3 Intended Audience and Reading Suggestions

The document is created and iterated by both the users and developers together. This document is intended for the developers and project manager to get a complete understanding of the requirements of the system as expected by the user.

An early usecase document is available at [1]. [2] presents the current status and overall goal of the seismic inversion application. [3] provides a quick introduction to the current capabilities of the Ensemble Toolkit.

1.4 Project Scope

<This might be better suited for Princeton >.

The project has the following goals:

- increase the ultimate resolving power of global inversions through more efficient use of resources
- open new avenues of scientific research,in particular, anisotropic inversions
- a package is to be developed and shared with the wider seismology community

1.5 References

[1] https://docs.google.com/document/d/1EjUKJxsNNISBwZ9IXERyiCAISKb25mEwjhyi1gePCDk/edit?usp=sharing

[2] https://github.com/radical-collaboration/hpc-workflows/blob/master/docs/presentations/20161026_ET_Seismo.pdf

[3] https://github.com/radical-collaboration/hpc-workflows/blob/master/docs/presentations/Ensemble%20Toolkit%20-%20Quick%20Overview.pdf

[4] http://seisflows.readthedocs.io/en/latest/

[5] https://bitbucket.org/mpbl/simpy

2 Overall Description

This chapter provides an overview of the system. It will describe the context of the system and why its development is deemed necessary. It will broadly list the functions the system is expected to perform along with its interaction with other systems as well as the user. This chapter will also describe the intended users and the system environment for the current version of the system. It will conclude with a list of assumptions and foreseen constraints.

2.1 Product Perspective

The system will be designed to execute the seismic inversion workflow, as represented in Figure 2.1, at scale on HPCs, more specifically on DoE Titan machine. The various sequential and concurrent stages of the workflow and the associated data movement will be captured using the Ensemble ToolkitAPI.

Existing solution executes the various stages of the workflow in parts: Seisflow[4] is the tool used to perform the simulations on an HPC and Simpy[5] is used to preprocess and postprocess the simulation data before the next set of simulations. This document captures the requirement of a product that will be an end-to-end solution addressing many of the missing features in the existing solution.

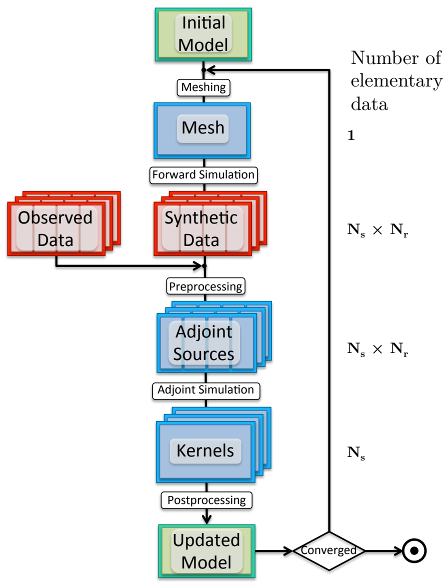

Figure 2.1: Seismic inversion workflow

This system will be designed in order to expose, to the user, two components: an API and a database. The API will enable the user to describe the application workflow completely. The database will provide access to the state of the various tasks in the workflow. This will enable the user to observe performance metrics or perform various analysis in real time. Once the application is described via the API, the internal components will translate the application into executable units and manage their execution across multiple resources. Figure 2.2 provides a graphical representation of the system.

2.2 Product Functions

The major functions the product is expected to perform is the following:

- an API sufficient to capture the description of the complete workflow
- capacity to execute O(1000) tasks concurrently
- ability to manage task failures due to several reasons
- ability to provide real time execution statistics and user access to intermediate data

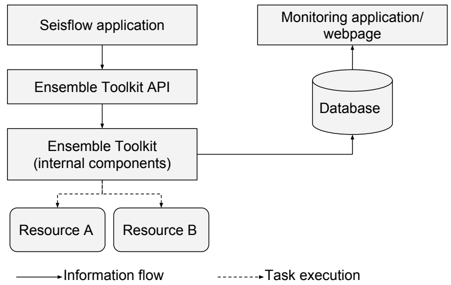

Figure 2.2: High-level design of the Ensemble Toolkit for Earth Sciences

2.3 User Classes and Characteristics

The 0.1.0 version of the product is intended to be used only by the Princeton and Rutgers groups. Later versions will be developed and packaged to be distributed to a wider community.

2.4 Operating Environment

The primary OS the product will be tested on is Linux with possible support of MacOS. The primary HPCs the product will be tested on is the ORNL Titan, ORNL Rhea and TACC Stampede.

2.5 Design and Implementation Constraints

The entire framework, including the API exposed to the user, will be python based but will support any task kernel intended for a Linux OS.

The current version will be using RADICAL Pilot (RP) as a runtime system and will be constrained to the performance limitations of the same. ¡TBA: Relevant RP performance numbers/plots¿.

The forward and adjoint simulations will use the SPECFEM tool, available at `http://specfem3d-globe.readthedocs.io/en/latest/`. The preprocessing and postprocessing will use the pypaw tool, available at `https://github.com/wjlei1990/pypaw`.

2.6 User Documentation

All the relevant code will be available on Github. Ensemble Toolkit code will be available at `https://github.com/radical-cybertools/radical.ensemblemd`. User manual for

Ensemble Toolkit will be made available in `http://radicalensemblemd.readthedocs.org/en/latest/`. The code for the end-to-end workflow will be available on Github and user manual on `www.readthedocs.org`. The exact URLs will be updated as the project matures.

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

The following assumptions have been made:

- Currently, the framework requires no interactivity. If a manual halt is necessary, this requires starting (as opposed to resuming) from the last checkpoint or backup.
- GPU-only simulation is sufficient.

The framework will have a dependency on the following tools:

- RADICAL Pilot
- Specfem3D
- pypaw (and its dependencies)

3 External Interface Requirements

3.1 User Interfaces

TBD

3.2 Hardware Interfaces

The interaction between the product and the hardware, both on the local and remote ends, is abstracted from the user. The hardware interaction is complex and can be deemed out of scope for the current document.

3.3 Software Interfaces

TBD

3.4 Communications Interfaces

There exists communication amongst the various internal components of the system as well as communication between the system and external entities such as the database or the HPC. Although these are crucial, these are hidden from the user, within the system.

4 System Features

This chapter enumerates the features the system is expected to support. Each of these features have requirements that will be given a priority and will be implemented in that order.

4.1 Execution of O(1000) concurrent tasks on Titan

4.1.1 Description and Priority

Ensemble Toolkitshould be able to support O(1000) concurrent tasks on Titan. It is important to note that depending on the stage of the workflow, all resources might not be in use at the same time. This is due to the tasks requiring various core counts.
Priority: High

4.1.2 Stimulus/Response Sequences

No specific stimulus required.

4.1.3 Functional Requirements

This feature requires the underlying runtime system, RADICAL Pilot, to be functional at O(1000) tasks where each task may have different core counts.

4.2 GPU-only simulations

4.2.1 Description and Priority

The forward and adjoint simulations need to be able to use GPUs. Currently, it suffices to execute in a GPU-only mode (as opposed to GPU+CPU).
Priority: High

4.2.2 Stimulus/Response Sequences

This will be the default scenario. No specific stimulus required.

4.2.3 Functional Requirements

This feature has multiple dependencies, listed below:

- Req 1: RADICAL Pilot capability to utilize GPUs on Titan and Stampede at the required scale.
- Req 2: Compilation of the specfem tool with GPUs on Titan and Stampede. Once compiled, the GPU specific executable is to be tested for correct execution via PBS/SLURM script.

4.3 Real-time monitoring: Database

4.3.1 Description and Priority

The user is required to have access to the following in real time:

- state of all tasks
- the input data used by the tasks
- the location of the intermediate and output data of these tasks

Using a database, hosted either locally or remotely, will be required to keep the information listed above. A database can serve the following purposes:

- The user can connect to this database and analyze the information as preferred.
- The state of the system can be recovered using the database even if the master process is crashed.

Priority: High

4.3.2 Stimulus/Response Sequences

None.

4.3.3 Functional Requirements

Depending on the database that is chosen (redis,mongodb,?), the framework will require a component exclusively to keep the state of the entities up to date in the database.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

TBD

5.2 Safety Requirements

None

5.3 Security Requirements

TBD

5.4 Software Quality Attributes

5.4.1 Robustness

This system is expected to be robust and capable of handling task failure. The failed tasks may either be automatically resubmitted for execution or handed over to the user (say via callbacks).

- Expected error rate: ?
- Tolerable error rate: ?

5.4.2 Reliability

The system is perform reliably over long durations of time at scale on the intended HPCs. Given no variations in the execution environment, the system should not be faulty state at any point.

6 Other Requirements

6.1 Compiling binaries with specific OpenMPI

The mpi-based executable kernels SPECFEM and pypaw are required to be compiled against the EnTK/RP specific OpenMPI on Titan.

6.2 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>
TBD

6.3 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>
TBD

6.4 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>