Performance

Parallelization of the retrieval and rendering of the data proved to be the key element in achieving satisfactory speed of the application. Multi-threaded analysis of the queries by the server enables the system to support multiple users simultaneously. The implementation of commonly used libraries enables the use of content distribution networks, i.e., Google Libraries API, for the collection of widgets used by the application. This decreases the total amount of code that needs to be maintained and updated.

Future development

1. Real time live interface to streaming waveforms from a Object Ring Buffer (ORB).
2. Automate the installation of server-side library dependencies.
3. Promote the development of new clients that can use the server as a gateway to the databases.

Download

You can download the code used in this presentation from the online Git repository hosted by Github. http://github.com/antelopeusersgroup/antelope_contrib

References


Acknowledgements

Funding for this project was provided by the Incorporated Research Institutions of Seismology (IRIS) and the Institute for Crustal Studies at the University of California Santa Barbara (UCSB).

Introduction

The Waveform Server is an interactive web-based interface to multi-station and multi-channel high-density seismic waveforms stored in Center for Seismic Studies (CSS) 3.0 schema relational databases (Newman et al., 2009). In the last twelve months, based on expanded specifications and current user feedback, both the server-side infrastructure and client-side interface have been extensively rewritten. In addition to supporting the single database model, the Python server-side code has been fundamentally modified to retrieve data using Python Deferred Objects in a multi-threaded architecture and access data stored in cluster-based databases. This allows interactive web-based access to high-density (> 200 Hz) waveform data that can span multiple years, the common lifespan of broadband seismic networks.

The client-side interface expands on its use of simple JSON-based AJAX queries to now incorporate a variety of User Interface (UI) improvements including standardized calendars for defining time ranges, applying-off-line data calibration and unit representation, and time-zone correction.

Development

JSON

JavaScript Object Notation (JSON) is a lightweight text-based open standard designed for human-readable data interchange. Originally derived from the JavaScript programming language, now it is language-independent with parsers available for many programming languages. It lacks semantical meaning by definition but JSON has a much smaller grammar and maps more straightforwardly to the data structures used in modern programming languages. This translates to faster parsing and processing of the data structure.

Dbcentral

Many institutions have divided their datasets into multiple independent databases, for example databases segmented by day, week, month or year. The dbcentral schema provides the logic to reference those databases. To properly handle this paradigm the server fend schema provides the logic to reference those databases. To properly handle this paradigm the server model consists of a single database model, the Python server-side code has been fundamentally modified to retrieve data using Python Deferred Objects in a multi-threaded architecture and access data stored in cluster-based databases. This allows interactive web-based access to high-density (> 200 Hz) waveform data that can span multiple years, the common lifespan of broadband seismic networks.

Deferred Objects

A key concept in the Twisted application model is the deferred object. This object encapsulates an instruction that will produce a result in the future and contains a series of functions that will be applied to the returned value. Our new implementation of the server leverages on the use of deferred objects and thread function calls to spawn a child process for every asynchronous request. The Deferred object then hands the results of each callback or error function to the next function in the chain and returns results to the client allowing parallel processing of asynchronous requests. This multiplexed analysis of queries allows a non-blocking processing of client requests.

AJAX

All data and metadata is retrieved using the Asynchronous JavaScript and XML (AJAX) technique from the client to the server. All of this background bidirectional communication does not interfere with the behavior of the loaded page. AJAX leverages the HTML and CSS mark-up and style information to dynamically display, and allow the user to interact with, the information presented. The asynchronous, callback-style retrieval of data allows the client to populate each waveform independently, dramatically decreasing the total loading and rendering time of the application.

Deferred Objects

A key concept in the Twisted application model is the deferred object. This object encapsulates an instruction that will produce a result in the future and contains a series of functions that will be applied to the returned value. Our new implementation of the server leverages on the use of deferred objects and thread function calls to spawn a child process for every asynchronous request. The Deferred object then hands the results of each callback or error function to the next function in the chain and returns results to the client allowing parallel processing of asynchronous requests. This multiplexed analysis of queries allows a non-blocking processing of client requests.

AJAX

All data and metadata is retrieved using the Asynchronous JavaScript and XML (AJAX) technique from the client to the server. All of this background bidirectional communication does not interfere with the behavior of the loaded page. AJAX leverages the HTML and CSS mark-up and style information to dynamically display, and allow the user to interact with, the information presented. The asynchronous, callback-style retrieval of data allows the client to populate each waveform independently, dramatically decreasing the total loading and rendering time of the application.

Deferred Objects

A key concept in the Twisted application model is the deferred object. This object encapsulates an instruction that will produce a result in the future and contains a series of functions that will be applied to the returned value. Our new implementation of the server leverages on the use of deferred objects and thread function calls to spawn a child process for every asynchronous request. The Deferred object then hands the results of each callback or error function to the next function in the chain and returns results to the client allowing parallel processing of asynchronous requests. This multiplexed analysis of queries allows a non-blocking processing of client requests.

AJAX

All data and metadata is retrieved using the Asynchronous JavaScript and XML (AJAX) technique from the client to the server. All of this background bidirectional communication does not interfere with the behavior of the loaded page. AJAX leverages the HTML and CSS mark-up and style information to dynamically display, and allow the user to interact with, the information presented. The asynchronous, callback-style retrieval of data allows the client to populate each waveform independently, dramatically decreasing the total loading and rendering time of the application.

Deferred Objects

A key concept in the Twisted application model is the deferred object. This object encapsulates an instruction that will produce a result in the future and contains a series of functions that will be applied to the returned value. Our new implementation of the server leverages on the use of deferred objects and thread function calls to spawn a child process for every asynchronous request. The Deferred object then hands the results of each callback or error function to the next function in the chain and returns results to the client allowing parallel processing of asynchronous requests. This multiplexed analysis of queries allows a non-blocking processing of client requests.