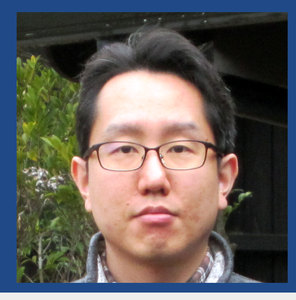
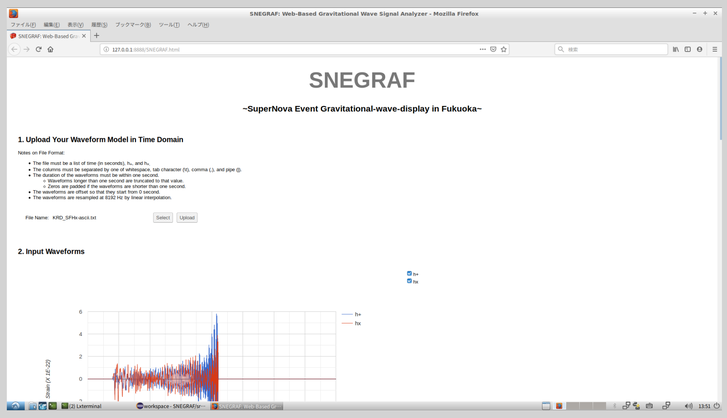
The Recent Developmental Status of SNEGRAF: a Web-Based Gravitational Wave Signal Analyzer

Satoshi Eguchi, Shota Shibagaki, Kazuhiro Hayama, and Kei Kotake

Department of Applied Physics, Faculty of Science, Fukuoka University



Abstract



Unveiling physical processes in a supernova is one of challenging topics of modern physics and astrophysics since that event is due to particle physics on a stellar scale and tightly related to nucleosynthesis in Universe. Multi-messenger astronomy, a combination, such as of electromagnetic-wave, gravitational-wave, and neutrino observations, will be a breakthrough to the puzzle. To boost the research, we released a web-based gravitational wave signal analyzer "SuperNova Event Gravitational-wave-display in Fukuoka (SNEGRAF)" last year (Eguchi et al. 2018). We are now working on an integration of the application with the RIDGE pipeline, which is for a coherent network analysis between the LIGO, VIRGO, and KAGRA observations (Hayama et al. 2007), and implemented in MATLAB. In the basic design phase, we decided to wrap RIDGE with a simple Python script and make it listen for connections from SNEGRAF. This design enables these two programs to be hosted on different servers independently, and minimizes the cyber risks of RIDGE. In this poster, we report the current developmental status of our system.

1. Multi-Messenger Astronomy and Our Scientific Goals

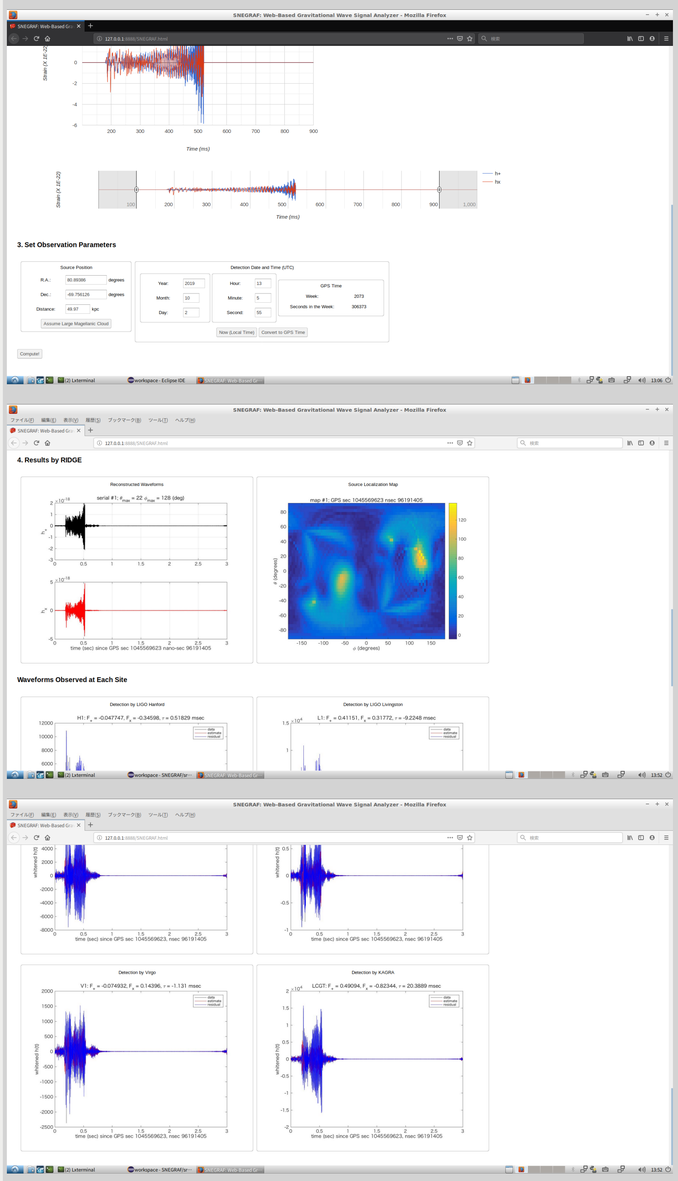
A supernova explosion is one of the most energetic phenomena in Universe, and it takes place in the last stage of stellar evolution. **In Fukuoka University, we have a special team aimed to reveal the physical mechanisms of core-collapse supernovae (CCSNe) from both sides of theories (numerical simulations) and observations**. The central engine of a CCSN is in an extremely condensed state and photons can hardly escape, hence new observational methods utilizing neutrinos and gravitational waves, to which such dense materials are even transparent, are crucial; these observations combined with classical electromagnetic ones are referred to as "**multi-messenger astronomy**".

2. Coherent Network Analysis and the RIDGE Pipeline

When compared to events of neutron star mergers and black hole ones, our understanding of the gravitational waves radiated during a CCSN is rather limited. **Coherent network analysis (CNA) technique**, which coordinates individual observations by different worldwide gravitational-wave detectors, provides us adequate information even about a transient source including a CCSN since the method can reduce the impact of temporal and unmodelled detector noises on the data (Chatterji et al. 2006). **RIDGE** is an implementation of CNA algorithms, and its robustness has been confirmed by intensive reviews **from the aspect of "physics"** (Hayama et al. 2007; Hayama et al. 2015).

3. SNEGRAF

"**SuperNova Event Gravitational-wave-display in Fukuoka (SNEGRAF)**" is a web application to analyze gravitational-wave signals provided by our team in order to boost the astrophysics of supernovae, and was presented in ADASS2018 (Eguchi et al. 2018). Although SNEGRAF at that time could just display FFT results and the signal-to-noise ratio together with the analytic sensitivity curve of KAGRA from the users' inputs, our intensive improvements of the software made over the past year, for **an integration of SNEGRAF with the RIDGE pipeline**, allows the users to perform much more realistic simulations of gravitational-wave detections with LIGO, Virgo, and KAGRA based on their model waveforms.

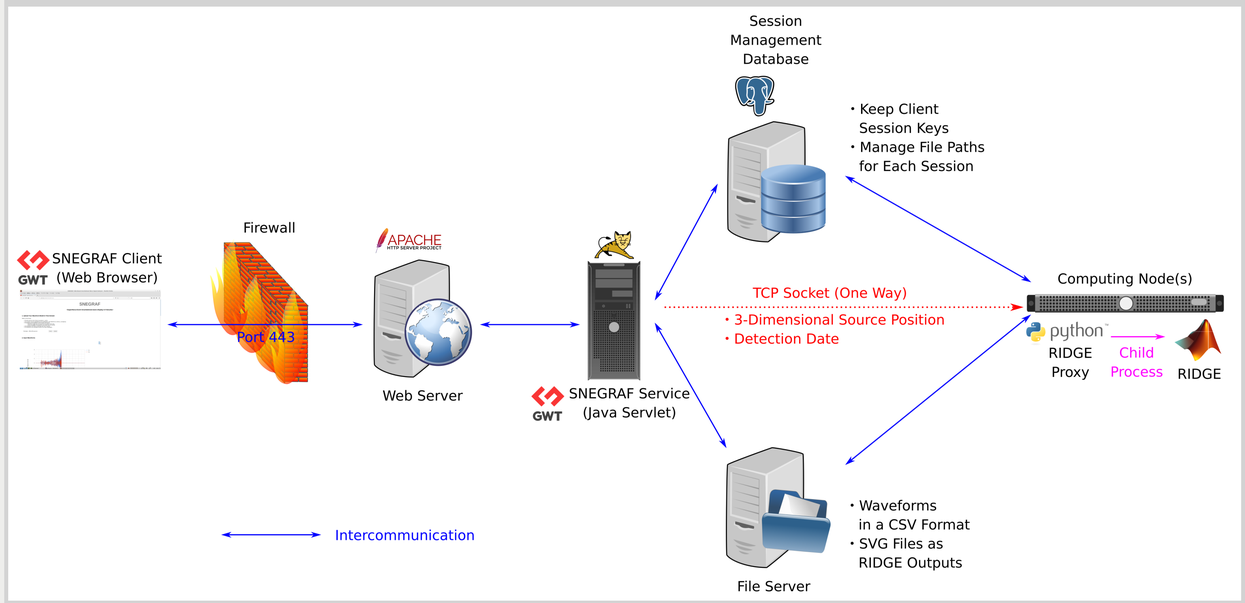


The left panels show screenshots of the latest development version of SNEGRAF. Once a user uploads his/her theoretical waveforms, a panel asking the "source" information (right ascension, declination, and the distance to the source) and "detection" date and time appears (top panel). While RIDGE internally uses a GPS time representation, we made SNEGRAF accept Coordinated Universal Time (UTC) with consideration of leap seconds, since UTC is the standard in "traditional" (or electromagnetic) astronomy; the user can easily check the feasibility of follow-up observations especially with X-ray and gamma-ray satellites, which have different observation windows depending on a source direction, in case that such an event should really occur.

A click on "Compute!" button invokes RIDGE with the above parameters together with the uploaded waveforms. In the process, RIDGE convolutes the waveforms with the response and idealized Gaussian noises of each detector, performs a waveform reconstruction, and makes a source localization sky map. Then these results including "virtually observed" signals by respective detectors are sent back to SNEGRAF as scalable-vector-graphics (SVG) files; SNEGRAF presents them as static images on the screen (middle and bottom pannels).

We are now in the final phase of unit testing for both software; we will be able to make the brand-new SNEGRAF public in this winter (by the end of this year).

4. System Design



The above figure shows a diagram of our system. For rapid development and maximum utilization of existing software resources, we adopted GWT (previously known as Google Web Toolkit) for a framework of SNEGRAF (Eguchi et al. 2018); GWT generates both sever-side (Java servlet) and client-side (JavaScript) codes from one Java source file.

The RIDGE pipeline is written in MATLAB. From the viewpoint of software engineering, RIDGE has not been in long-term continuous operation as a software service. In addition, a MATLAB application always runs on MATLAB kernel even when it is packaged by MATLAB Compiler; we have to be aware of risks similar to SQL injection attacks against the kernel if we make RIDGE public to the Internet directly. To these concerns, we took a very simple approach: **an isolation of RIDGE from both the Internet and a web server at hardware level**.

An application server running the SNEGRAF servlet and **a computing node running RIDGE can communicate with each other** just through a file server, which stores the users' inputs as character-separated-values (CSV) files and SVG files produced by RIDGE, and **through a one-way** transmission-control-protocol **(TCP) socket to send source parameters as CSV strings, from the application server to the computing node**. Elsewhere than the file server, any files exist just as Base64 encoded strings. Since we would not like to make any changes to RIDGE for awaiting a connection from the servlet, we implemented a simple proxy server which invokes RIDGE as a child process in a worker thread in Python, and run it on the computing node. This design brings us a favorable by-product; with another few tens of lines for a simple round-robin scheduler in SNEGRAF, we can easily scale the system out just by adding computing nodes.

5. Miscellanies

- The interactive chart panel to display uploaded waveforms was implemented with GWT Charts, a class library of Google Charts for GWT environments. GWT Charts seems not to be maintained for a while; the API loader interface for Google Charts changed long, long time ago, but it was not reflected in GWT Charts. On January 1 of this year, GWT Charts temporally stopped working by a maintenance by Google, possibly due to disablement of old loader codes. We completely rewrote the core classes of GWT Charts to load the Google Charts modules, and made a pull request to upstream. Please access my GitHub repository if you are interested in:
  `https://github.com/satoshieguchi/gwt-charts/tree/new_api_loader_for_pullreq`.
- We searched for a library to covert time between UTC, International Atomic Time (TAI), and GPS time; we found some implementations (one was by essential Global Navigation Satellite Systems projects: `http://gnsstk.sourceforge.net/index.html`), but all of them used Julian Days (JDs) with double precision for the internal leap-second tables, and also treated a second as a double-precision number. Our work to port the algorithm to Java (or GWT) led to errors due to floating point (rounding errors). We implemented time conversion classes from scratch, where the table uses Modified Julian Days (MJDs) in units of seconds as integers. We should remember that FPU registers in an IA-32 CPU have an 80-bit width but 64-bit precision in AMD64 architecture, when we work with old software.

References

- Chatterji S., Lazzarini A., Stein L., Sutton P. J., Searle A., Tinto M., 2006, PhRvD, 74, 082005
- Eguchi, S., Shibagaki, S., Hayama, K., & Kotake, K. 2018, in ADASS XXVIII
- Hayama K., Mohanty S. D., Rakhmanov M., Desai S., 2007, CQGra, 24, S681
- Hayama K., Kuroda T., Kotake K., Takiwaki T., 2015, PhRvD, 92, 122001