



SPE 107561

Real-Time Simulation in Grid Environments: Communicating Data From Sensors to Scientific Simulations

Richard G. Duff, SPE, and Yaakoub Y. El-Khamra, SPE, Louisiana State U.

Copyright 2007, Society of Petroleum Engineers

This paper was prepared for presentation at the 2007 SPE Digital Energy Conference and Exhibition held in Houston, Texas, U.S.A., 11–12 April 2007.

This paper was selected for presentation by an SPE Program Committee following review of information contained in an abstract submitted by the author(s). Contents of the paper, as presented, have not been reviewed by the Society of Petroleum Engineers and are subject to correction by the author(s). The material, as presented, does not necessarily reflect any position of the Society of Petroleum Engineers, its officers, or members. Papers presented at SPE meetings are subject to publication review by Editorial Committees of the Society of Petroleum Engineers. Electronic reproduction, distribution, or storage of any part of this paper for commercial purposes without the written consent of the Society of Petroleum Engineers is prohibited. Permission to reproduce in print is restricted to an abstract of not more than 300 words; illustrations may not be copied. The abstract must contain conspicuous acknowledgment of where and by whom the paper was presented. Write Librarian, SPE, P.O. Box 833836, Richardson, Texas 75083-3836 U.S.A., fax 01-972-952-9435.

Abstract

Across many fields of science and engineering computers now play a significant role in scientific discovery through both large scale simulation and real time data acquisition. As these scientific simulations increasingly require new levels of complexity and fidelity, and leveraging increasing computational capabilities, scientists are migrating their applications back and forth from workstations to supercomputers, high performance clusters, and new distributed grids of computers. For applications which depend on live real world data, the migration to varied and distributed resources provides additional challenges. This migration involves reassigning and testing individual sensor communications and data path integrity for the application before the application is ready for a real time operating environment.

This paper presents one general approach for communicating live real world data to simulations deployed on high-end computational resources. The architecture and design are presented for generic applications, before focusing on a particular scenario for drilling dynamics optimizations. In this scenario, simulations of drilling behavior depend on real time data from on-site (remote) sensors. Sensor data is streamed through to a simulation framework. The framework (available on a desktop computer or a supercomputer) displays relevant information, starts simulations with real time inputs then presents results and recommendations. For our application, we will use the Cactus Code (www.cactuscode.org) open-source high performance scientific computing framework as the simulation framework, and LabVIEW (www.ni.com/LabVIEW) as the data acquisition software. The advances in architecture portability allowed by the framework are summarized, and experiences of system uses are presented. Discussion will include

opportunities found and learnings from using this platform in various environments highlighting drilling optimization.

Introduction

Simulation Framework

The Cactus Code¹ is a framework for high performance scientific computing designed for scientists and engineers. In most cases, high performance scientific computing makes use of supercomputers, which consist of a large number of processors working in tandem (parallel processing) to solve a particular problem. In an effort to reduce the development time involved in parallel programming, researchers have created frameworks to develop scientific computing code that is portable, scalable, efficient and most importantly re-usable. Writing code for such machines has its unique challenges: portability, scalability, efficiency (for computation, communication and input/output) and flexibility. The framework allows scientists and engineers to write modules (in different programming languages if desired) and use them in conjunction with other modules written by other researchers to solve computational problems. The framework provides tools ranging from basic computational building blocks to complete toolkits that can be used to solve numerical relativity problems or computational fluid dynamics problems. Tools developed in the Cactus Code framework run on a wide range of hardware ranging from desktop PC's, large supercomputers, to 'grid' computers. In an effort to expand the available toolset available in the framework, work has begun on the following energy industry tools: a seismic inversion tool, a reservoir simulator and drilling models. The framework and its toolkits are free and publicly available for download from the Cactus Code website.

From an architectural standpoint, the Cactus Code framework consists of a central part (the "flesh") and code modules (the "thorns"). The flesh has very little computational functionality as it serves as a module manager and "glues" code from various modules together to perform specific tasks. The code modules or 'thorns' perform tasks including setting up the computational grid, decomposing the computational grid for parallel processing, setting up coordinate systems, boundary and initial conditions, communication of data from one processor to another, solving partial differential equations, input and output and visualization streaming.² There are code modules that provide simulation control tools, such as the

HTTPD thorn that sets up a web server for the simulation and allows researchers to control a simulation or view sample output from a web interface. The framework also has visualization tools such as iso-surface extraction and dynamic streaming (IsoSurfacer and IoStreamedHDF5 thorns), in addition to several integration modules in popular scientific visualization packages. For all applications the common shared framework is used, customization of the code modules activated causes the execution of different tasks (such as running a drilling prediction or reservoir simulation). For deployment sake this simplifies the utilization of homogeneous resources that users expect to run diverse problems.

Beyond the plethora of tools available, the framework offers scientists developing parallel high performance scientific code a modular and collaborative multilingual development environment. This allows experts in different disciplines to develop their individual codes separately then use them together within the Cactus Code framework to solve challenging problems. Effort and care are put into the maintenance of the framework to preserve generality and portability and remove limitations that might appear when solving new types of problems.

Sensor Platform

Sensors are devices most often used to measure physical phenomena and subsequently output electrical or optical signals. Live sensor data is used in simulations of real world problems such as: predicting weather and storm search, early warning quake simulations and control and maintenance of nuclear reactors.³

While the idea of using sensor data in simulations is not new, it is the authors' opinion that there has not been a consistent effort to seamlessly transfer data and information from sensors to simulations. This opinion was influenced by the perception that simulations rarely yield answers to physical problems in real time, making the use of live sensor data less critical as the results obtained will be old in comparison to the physical system and can best be used for physical model verification rather than prediction.⁴ With the advent of supercomputing, the grid, and high throughput custom computing machines for various applications coupled with advances in algorithms, software design and optimization, it is our belief that accurate real time physical simulations will be widely available in the near future and streaming live real world sensor data to simulations is a critical challenge.

A detail worth highlighting is the difference between sensor data and sensor information. While the former is a pure physical quantity, the latter is the actual physical quantity in addition to information about the sensor, its normal operational parameters, its location and any other information that is deemed useful about the sensor or its deployment (such as battery state, or hours in the field). While streaming numbers over networks is a simple task, when complex information is required by the simulation a challenge emerges.

Sensor data can be retrieved in a multitude of ways. We chose to use a commercial package because of its availability and widespread use in data acquisition. The software we are

currently using for our sensor data collection and streaming is LabVIEW from National Instruments. Since the actual information streaming is performed through a function call to a library we developed, the choice of sensor data collection software would not affect any of the design parameters. The general preference is for data collection software that is responsive, reliable, and capable of automatic relay and we found our vendor solution satisfied those requirements.

Design

Figure 1 shows the communication pattern of the system components. The dashed line represents query calls. The solid lines are the real-time data paths. Our design bridges sensors and simulations by facilitating the transfer of information between the two, based on three components: sensor platform, relay and computational framework. A compromise between batch and streaming data communication methods was made to utilize the advantages of both.⁵ Our design allows for batch and streaming communication, we found that streaming data from the sensor platform to the relay and using batch communication between the relay and the computational framework worked best for our application.

The sensor platform component includes the sensors and their data acquisition software that sends information to the relay component (in our case a database). The application inside the computational framework component is then free to query the relay for measurement information. The relay serves a dual role: it is a communication layer between sensor and simulation and it is a record of the sensor data.

One of the design goals was to maximize portability (and consequently facilitate deployment) of all components. For example, if a user wants to move an application that was going to be too memory or computationally intensive from a desktop PC to another machine or a computational grid, the relay and sensor platform would not require any modification. This allows the user flexibility without a 'run time' or 'configuration' cost at the critical relay or data store spool.

Implementation

Sensor Measurement to Database

A shared library was designed to act as middleware between our sensor platform and the MySQL communication libraries. Figure 2 shows our custom communication module's call behavior. We are able to send sensor information from the sensor platform to the relay at various sampling rates, with or without multiplexing. The modular design of the communication module at the sensor platform also allows for rapid deployment on production systems.

Our LabVIEW code is a demonstration used to test our implementation however it can be used as a sub-vi (by passing in the required arguments) and used as a sensor platform interface to the relay for production systems. Figure 3 is an example of the VI. The arguments include the basic relay information: relay address, database name, user name and password and the data. These arguments are then passed to a shared library function that in turn calls static system library functions. The shared library acts as an interface to the common operations required to stream data to the database.

Since our relay is a MySQL database, these functions are: `mysql_real_connect`, `mysql_real_query` and `mysql_close`. We decided to use a shared library for several reasons: the available LabVIEW database interface was not portable, a desire for a database solution independent interface and flexibility for future implementation of the relay. Our final intention is to use an abstract database interface. This would allow us to stream information to relays that use other database vendors (Oracle, MySQL, Firebird, PostgreSQL and SQLite).

Relay to Simulation

The module `CommMySQL` was developed to interface the computational framework to the relay. Figure 4 shows the essential functions of connecting, querying and disconnecting from the relay that have been implemented in the module. The application has control of when to schedule calls to the relay to minimize communication costs. The implementation allows data to be stored at the relay while the application prepares to query for the next information frame. It is also possible to have multiple instances of the computational framework running in parallel all accessing the same relay to retrieve the same data, different data or data with different time stamps.

Operational Scenario

Figure 5 shows a generic operational scenario for communicating data from sensors to simulations on a computational grid:

1. The sensor platform is streaming information to the relay.
2. The simulation starts on a computational grid and the simulation communication module is activated.
3. The simulation communication module connects to the relay, the hostname and port number parameters can be modified at run time to connect to different relays. (an implementation detail we have in our framework involves the simulation passing its credentials to the database to track the simulation)
4. The simulation communication module queries the relay for relevant information.
5. The relay returns the time stamped information to the simulation through the communication module.
6. The simulation performs its computational task. If the simulation requires more information from the relay it can iteratively query the relay for information at any time.
7. Before the simulation terminates, the simulation communication module disconnects from the relay.

A common reason for attaching sensor information to computational frameworks is to facilitate computation and

simulation of the system, to record phenomenon or monitor performance.

The use case that inspired this system is utilization of data generated on a drilling rig being facilitation more accurate simulation and perdition of behavior.⁶

Drillstring Dynamics Application

During a typical drilling operation the drillstring system is in motion; the drillstring rotates and moves down through the hole it is drilling, but it may also move or vibrate periodically inside the hole. A conceptual understanding of the drillstring system is that the sting is flexible and not robustly rigidly supported making motion in the system ever present and vibration inevitable. Under normal conditions the drilling operator seeks to minimize the occurrence of vibration.⁷ Motion in the drillstring is not necessarily undesirable as in some instances it increases the performance of the system.⁸

For our application scenario the goal was to identify vibration and deformation modes that may occur in the drillstring based on drillstring properties, hole geometry and conditions, and operational parameters. This problem is often addressed,⁹ and the concept of using realtime measurement in a solution is not new.¹⁰ Figure 6 shows a conceptual image of the desired work flow.

When originally installed, the sensors on the rig were not configured to generate information that would be used in computational simulations. Typically a vendor offering a new service may chose to install new sensors, at this point a service provider could tailor the sensor configuration to that desired by the simulation. Our group's goal was to develop a way to use existing sensors to generate a stream of information back to a central information collection platform, which would then be suitable for scientific simulation.

The sensor information is then streamed to the relay where it ready to be requested by a simulation. A user in this case, the driller or drilling performance reviewer, after choosing which modules to activate and the values of the simulation parameters starts the simulation. The simulation modules are activated and modules commence their computational tasks. The application reports the potential behavior, spectra and modes of vibration back to the driller.

The use of the portable computational framework allows the simulation to scale and run on any available hardware. In this case the simulation calculates the expected dynamic frequency set of a specific component of the drill. If knowledge about the behavior of other sections in the drill string is desired, the execution of more calculations can be accommodated. More computation cycles can be used for the solving for more component sections, without changing the code. The system taking advantage of this predicts behavior and frequencies for systems previously thought too large to be simulated in real time. This frequency set can be compared to the encountered sets to help identify conditions of motion and potential hazard times and locations in the dynamic drillstring system.¹¹ This brute force frequency perdition method matching is computationally intensive, without an implementation that required little effort on the part of the

operation and made efficient use of computational resources, the method would perhaps be considered uneconomical.

It is important to recognize that the entire drillstring is not instrumented therefore the simulation gives the driller the potential to predict the nature of motion and risk in the unmeasured locations of the system.

By identifying conditions of motion in the drillstring, the drilling operator has the option of changing operational parameters to improve the performance of the drilling system.¹²

Identified Opportunities

Our choice of simulation framework has a wide array of output methods varying from web servers to streaming three dimensional visualization, at the request of potential users work has begun on other means of output. The output methods we are investigating are alert messages through instant messenger services that work with SMS services and online report generation tools.

A specification for active control of sensors and actuators that could be implemented in a well test scenario has been requested and work has begun on such a specification.

Further optimization of the communication layers to accommodate high bandwidth gigabit interconnects has also begun, along with the utilization of computational grid technologies to allow transparent use of data stores as relays. Translation schemes between our communication layers and the Energistics standards have been identified as a desirable feature.

Another part of the greater project this was developed for is the use of remote resource discovery technologies available through computational grid frameworks. This would allow for completely automated, self arranging data paths that are highly portable and require no user intervention. As opposed to hard coding the data path from the sensor platform to a simulation running on a particular machine, we have implemented a design that allows us to run on various machines without hard coding the data path but we still pass the relay address to the simulation. Using the remote resource discovery technologies, the relay would announce itself to the grid environment and simulations with suitable security credentials would communicate with it without the need to know the address of the relay. This would result in improved information integrity through redundancy of relays.

An opportunity identified by colleagues was the use of a different relay infrastructure. This guided us to start development on a communication layer that is based on abstract database interfaces that allow us to change relay infrastructure without any code modifications.

It is expected that drilling operations will increase the use of computation and simulation as computing resources become more available. This may require more robust communication systems between sensors and compute resources than currently available, making our solution attractive.¹³

Learnings and Conclusions

This proof of concept application has been developed, by leveraging sensor platforms, relays, and computational

frameworks to overcome data collection, conversion, and performance bottle necks in the operation of communicating real time measurements to a robust computational environment.

Deployment of the application within the framework proved successful. Our experience also showed that having a portable computational framework, a portable relay infrastructure and a portable sensor platform allowed rapid deployment on various machines without difficulty in setting up data paths.

Acknowledgments

This research is supported in part by the U.S. Department of Energy (DOE) under Award Number DE-FG02-04ER46136 and by the Board of Regents, State of Louisiana, under Contract Number DOE/LEQSF(2004-07)-ULL. The authors would also like to thank the Center for Computation & Technology CCT.

Reference

1. T. Goodale, G. Allen, G. Lanfermann, J. Masso, T. Radke, E. Seidel, and J. Shalf. "The cactus framework and toolkit: Design and applications", presented at Vector and Parallel Processing - VECPAR '2002 5th International Conference, Porto, Portugal, 26-28 June 2002; published by Springer, 2003
2. G. Allen, T. Dramlitsch, I. Foster, N. Karonis, M. Ripeanu, E. Seidel, and B. Toonen. "Supporting efficient execution in heterogeneous distributed computing environments with cactus and globus." In Proceedings of Supercomputing 2001, Denver, Colorado, 2001
3. Dayong Huang, Gabrielle Allen, Chirag Dekate, Hartmut Kaiser, Zhou Lei, and Jon MacLaren, "GetData: A Grid Enabled Data Client for Coastal Modeling," Proceedings of High Performance Computing Symposium (HPC 2006), April 2006.
4. H. Aslaksen, M. Annand, R. Duncan, A. Fjaere, L. Paez, and U. Tran, "Integrated FEA Modeling Offers System Approach to Drillstring Optimization." IADC/SPE Drilling Conference. Miami, Florida, USA, 21-23 February 2006
5. Mark Sims, Jim Kurose, and Victor Lesser "Streaming versus Batch Processing of Sensor Data." IEEE Communications Society SECON. Amherst, Massachusetts, 2005
6. Nian-Feng Tzeng, Magdy Bayoumi, Hongyi Wu, Edward Seidel, Gabrielle Allen, Dmitri Perkins, Zhou Lei, and Douglas Moreman, "Techniques for Meeting the Communication and Computing Needs in Energy Resource Discovery and Management," (internal Report) submitted to IEEE Distributed Systems Online in July 2006 for possible publication; now under revision for resubmission.
7. Wolf, S.F., Zacksenhouse, M., Arian, A., "Field Measurements of Downhole Drillstring Vibrations." SPE Annual Technical Conference and Exhibition. SPE 14330. Las Vegas, Nevada: Society of Petroleum Engineers, 22-26 September 1985.
8. Eric Maidla, Marc HacıScott Jones, Michael Alexander, Michael Cluchey, Tommy Warren. "Field Proof of the New Sliding Technology for Directional Drilling." SPE 16655 SPE/IADC Drilling Conference. Amsterdam, Netherlands, 23-25 February 2005
9. Y. A. Khulief, H. Al-Naser. "Finite Elements in Analysis and Design." 2005: 1270 - 1288
10. David C-K Chen, Mark Smith, Scott LaPierre,. "Advanced Drillstring Dynamics System Integrates Real-Time Modeling and Measurements" SPE 81093. SPE Latin American and

Caribbean Petroleum Engineering Conference. Port-of-Spain, Trinidad and Tobago: Society of Petroleum Engineers, 27-30 April, 2003

11. Duff, R., "Observation and Modeling of Identified Regimens of Torsional Vibration," accepted for student poster presentation at 2007 AADE (American Association of Drilling Engineers) National Technical Conference and Exhibition, April 2007
12. Fred E. Dupriest, William L. Koederitz . "Maximizing Drill Rates with Real-Time Surveillance of Mechanical Specific Energy" SPE 92194. SPE/IADC Drilling Conference. Amsterdam, Netherlands, 23-25 February, 2005
13. Payne, M.L., Church, C.J."Trends and Strategies for Drilling Computing" SPE 16655. SPE Annual Technical Conference and Exhibition. Dallas, Texas, 27-30 September 1987

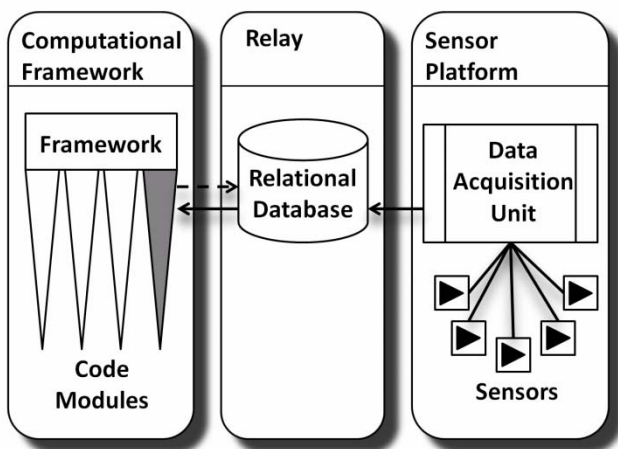


Figure 1: The communication chain of the sensor, platform, relay, grid framework executable. The shaded area in the computational framework is the communications module.

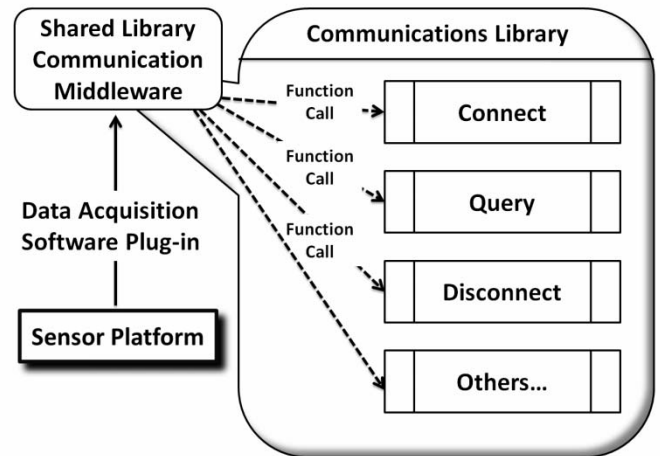


Figure 2: Call architecture in the sensor platform level communications module.

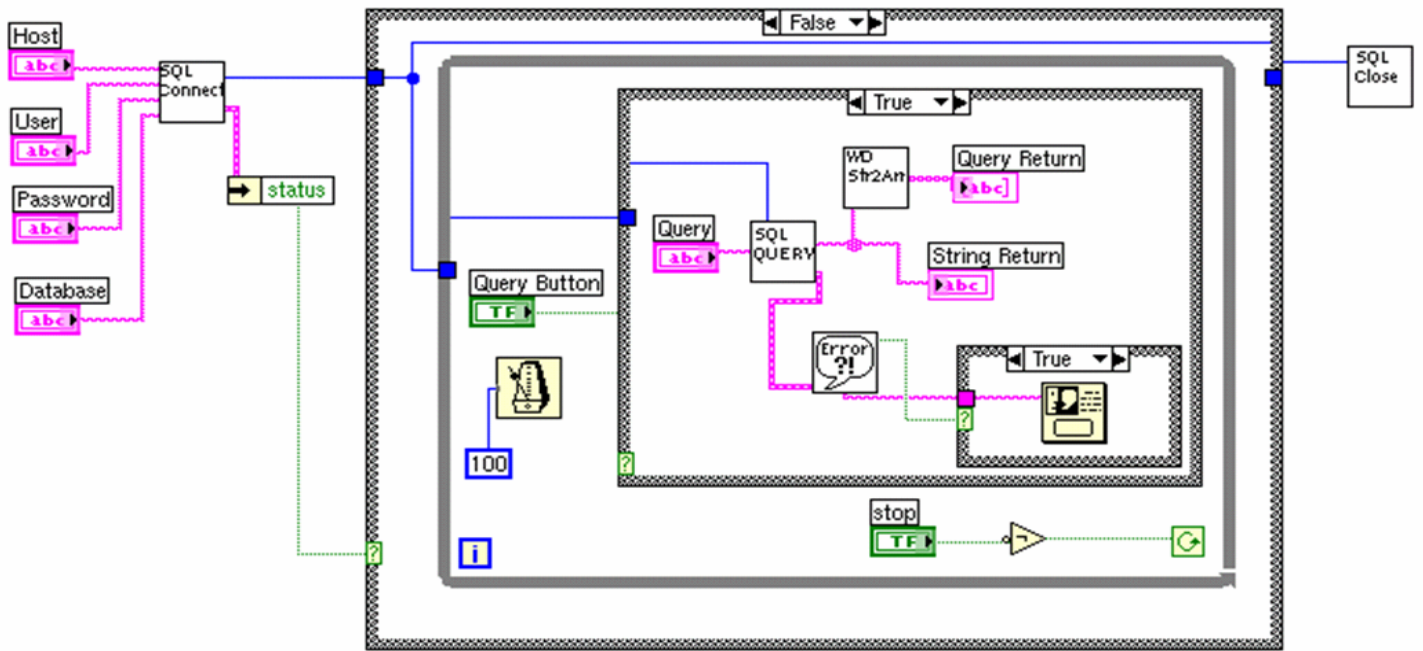


Figure 3: Example VI used for Relay connection.

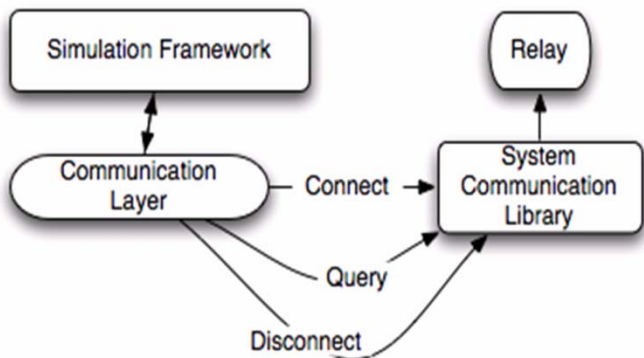


Figure 4: The CommMySQL thorns, illustrated are the framework originating calls to the relay.

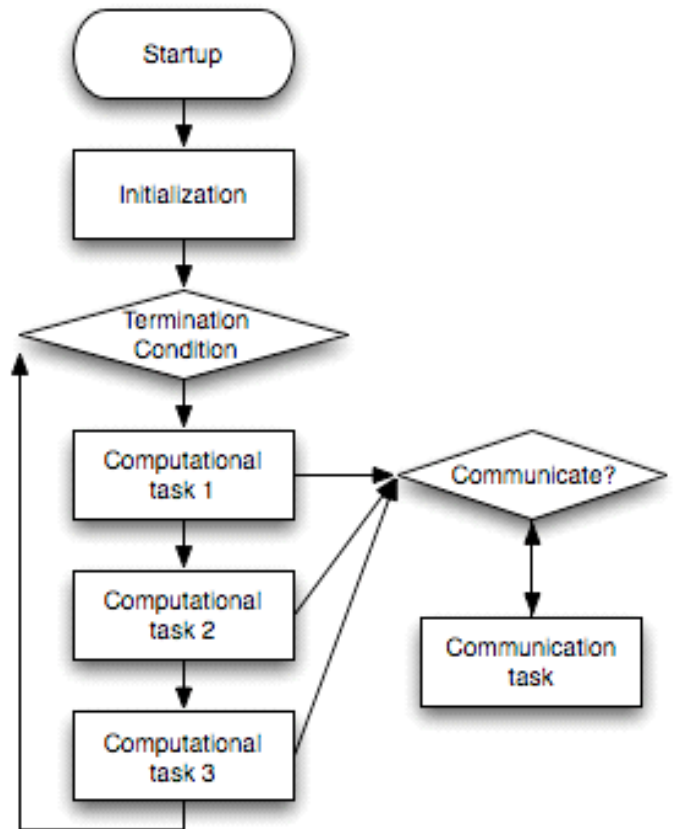


Figure 5: Logic diagram of the order of events from an application perspective.

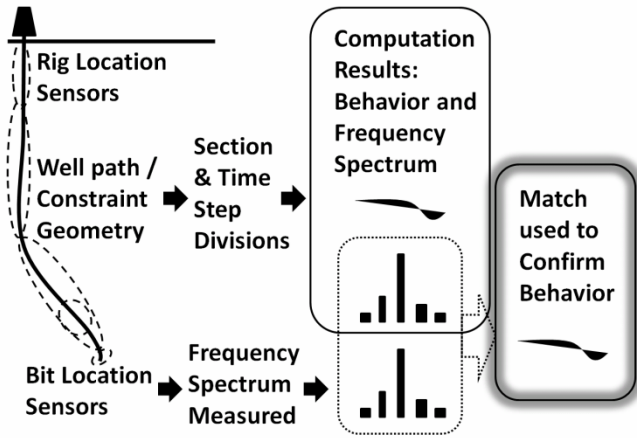


Figure 6: Example application information logic.