RADDADS Grump Jam: Thesis Proposal

Philip Ramsey, Joe Panciera, Rodger Brown

December 17, 2009

1 Problem Definition:

The project attempts to tackle two particular problems - designing and building a persistent high power computing center for evergreen and create a software library to deploy within said center. As the first goal is very open ended, the accomplishments will be measured in two ways: the presence of a functioning cluster and the completion of a manual for system administration and usage of the lab. The particular goal of the library implementation is to create an adaptive and expansive set of functions with major emphasis being placed on ease of understanding and use. The proposed specifications are as follows:

- Produce a distributed data mining library that maintains a 1to1 correspondence between the theoretical relationships and implementation as tasks are mapped across a cluster.
- Maintain an intuitive structure for use of the data mining functions by abstracting away support structure and the hardware implementation, allowing a greater emphasis on scientific computing applications than programmatic usage.
- Produce all the documentation for library usage, lab usage and upkeep, as well as maintain an open door to new input from both students and faculty.
- Create an extensive enough data mining library to be of use to parties in applications in the sciences at evergreen.

2 Data Flow Overview:

As indicated in Figure 1, the User is responsible for directing two types of data. In the first case, it specifies the location of the input data and its format. This data is sent to the Format module which applies the needed parser to that data and sends it along to the DFS. The DFS is responsible for breaking the parsed data into blocks which it then maps out to the particular nodes of the distributed file system. The DFS is also responsible for generating a table

of logical pointers to each of the blocks, specifying the actual addresses in a given node for that block. The second case that the User is responsible for is to send to the Library the sequence of function calls it wishes to have made over the input data. The Library takes this sequence and converts it into a single function definition. This conversion is done by composing a value-based ordering on each function call in the sequence. The Library then melds the data pointers, provided by the DFS, and the function definitions as needed by the System. The System, shown in Figure 2, receives the computation as a whole as a discrete set of functions over specific locations of data in the DFS. The System applies a Parallelization function over this set, as a distribution of the problem over the available nodes. After each node has returned from execution, the System then applies a Sequentialization function to determine the sum of each discrete execution. In the Sequentialization, if it is determined that a given unit of the distribution has not correctly completed execution, it passes the definition of that unit back to the Parallelization function to redistribute. Once the Sequentialization function has received every discrete unit of a given Parallelization function, it sends the returned values to the DFS, as the new data set, and a pointer to the next function in the function definition to the library. This data passing in Figure 2 is represented by Out. Each instance of data passed to the DFS is passed as a pair that identifies 1) the type of data being passed and 2) a pointer to the location of that data (Figure 3). The DFS uses the data type t to determine both where in the distributed file system the data is needed and what specific data pointers the Library will need next.

Data Flow Overview

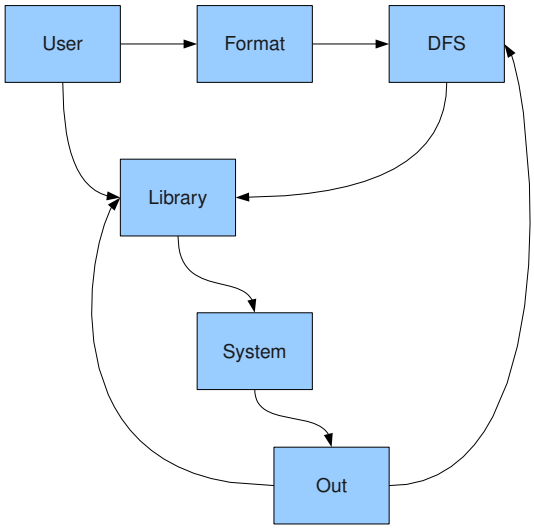

Figure 1

System and Out Data Flow

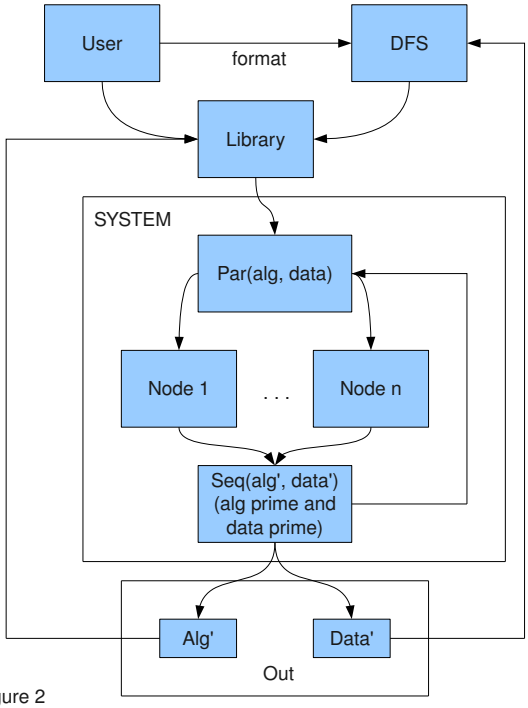

Figure 2

DFS Data Flow

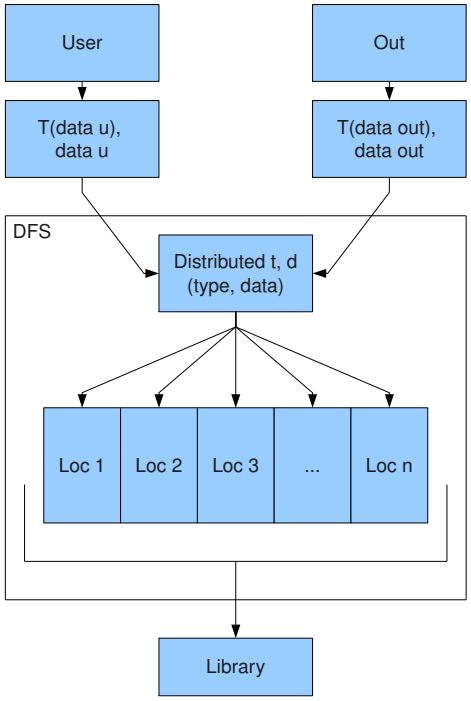

Figure 3

3 Control Flow Overview:

Each instance of implementation commences when a given user program calls any class of functions specified in the library. At this point in time in the user program execution, control is given to the library to define the execution of the called function. Once the Library has defined this execution, control is given to the System to perform the real time execution. During the real time execution, as needed, the System gives control to the DFS so that it can locate and pass back the locations of data pointed to in the function definition. Once the System has completed execution, it gives control back to the library so that the return values may be parsed into the original format, at which point the Library returns control back to the user program. This is illustrated in Figure 4.

Control Flow

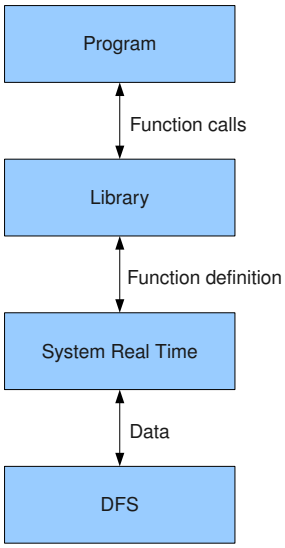

Figure 4

4 Uses Structure Overview:

As the primary point of interaction between the User and the library, the correctness of the Docs is defined as being integral to the utility of the Library. The Docs themselves are only correct in as much as they accurately represent the code as implemented by the Library. For the System to work correctly, it is required that both the functions being loaded into it, and the data pointers provided by the DFS are well formed. The well-formedness of the functions loaded into the System depends on the Classes and Utilities that they are on top of to be well-formed. This is illustrated in Figure 5.

5 Goals:

5.1 Winter

- Have a working cluster and a draft of a system administrator manual for it.

- Write 30 machine learning algorithms (1 per person, per week).
- Generate Java Docs for our current (working) library.
- Draft a class-level description of our library. (continues into spring)
- Research:
  1. Type systems for application level.
  2. New languages to be implemented on the cluster (push, forth, scala, clojure, PH, fortress, chapel, etc.)
  3. Weekly seminar group to discuss our research.
- Apply for grants.

5.2 Spring

- Finish our 30 machine learning algorithms (ideally, most are done at this point).
- Complete Java Docs for our finished library.
- Produce final lab manuals (system administrator and user).
- Open the lab for testing.
- Begin work with other departments at evergreen.
- Write final paper and prepare our presentation.
- Have a giant cluster party (optional).

6 Bibliography

This list will be extended as needed.

1. Chu, Kim, Lin, Yu, Bradski, Ng, and Olukotun. *Map-Reduce for Machine Learning on Multicore.* CS. Department, Stanford University.
2. Fayyad, Piatesky-Shapiro, Smyth. *From Data Mining to Knowledge Discovery in Databases.* AAAI, 1997.
3. Pierce, Turner. *Simple Type-Theoretic Foundations For Object-Oriented Programming.* CS. Department, University of Edinburgh.
4. Kukar and Komonenko. *Machine Learning and Data Mining.* 2007.
5. Kumar, Steinbach, and Tan. *Introduction To Data Mining.* Pearson Education, 2006.

6. Norvig and Russell. *Artificial Intelligence A Modern Approach.* Prentice-Hall 2003.

7. Lauriere, Jean-Louis. *Problem Solving and Artificial Intelligence.* Jean-Louis Lauriere, 1987, 1990.

8. Alpaydin, Ethem. *Introduction To Machine Learning.*