Civ

Mikhail Klimentov, Michael Nguyen, Prateek Sinha,
Yuchen Zeng, Eli Bogom-Shanon

September 25, 2014

1 Language Overview

Civ is a language that is aimed at distributed CSV processing. Provided the IP addresses, ports, and some form of identifier, Civ can send code real time to the slave computers corresponding to the given IPs to compile and run. The intent behind this is to allow rapid data transformations on CSV's resembling matrices, thus fundamental operations such as finding the minimum or maxiumum or averaging data will be provided, as well as mapping arithmetic functions. To conclude in key points:

Distributed Civ works at a local level but comes with built in distributed functionality.

Code Oriented Civ is NOT a system for sending data; it only sends the raw code that gets compiled at the slave site; this means that the master must be able to compile before sending, effectively providing Civ with interpreter-like error-catching.

Matrix Oriented Civ will have a lot of built in matrix operations for data processing at the slave environment. This will be reflected in its data types and syntax.

Extensibility Civ is meant to provide semi-permanent extensibility to its slaves by actively rewriting source code for interpretation/compilation at the remote sites. Different data sets can thus be manipulated in different ways, all from the comfort of your own home.

2 Use Cases

The inspiration behind Civ came from Hadoop and personal experiences with large data sets. One of the common issues that data scientists run into is waiting for processing of large data sets, especially ones that aren't on their local drive. Consequently, Civ is aimed at remedying this by providing built-in distributed support for transforming and analyzing large CSV's from far away. This is the

"Map" portion in "MapReduce". Civ will *not* provide a "Reduce" portion, as the presumption here is that the data sets are independent, and perhaps, the actual code is as well. In the long run, Civ could be used as a central compilation point for code to be issued out to many cloud-based platforms without hogging local resources, thus enabling the user to continue to develop and write for new processes while processes in the cloud continue to run without interruption.

3 Main Features

3.1 Syntax

The style of the language will draw inspiration from C/C++/Java. Curly braces will be plentiful. This means that Civ files can be expressed as one long string, which enables easier sending across to slaves.

3.2 Data Types

Primitives Ints, Floats, Bools, Chars, String, Doubles will all be provided as per most standard languages.

Data Structures Lists will be used as the main implementation of a data structure, and it will be mutable, extensible, and provide constant access time to its elements (thus some sort of indexing will be in place). Lists will also be recursive and nestable, due to matrices being nested lists.

Slave Buffer Because we will be sending raw code in plaintext ASCII-format across a network (often expressed as one line), there is specific syntax for writing code that is meant to be sent, as opposed to compiled and/or run.

4 Source Code Examples

```
Slave slave1 = Slave(127.0.0.1,5000);
Slave slave2 = Slave(127.0.0.1,5001);
print("Hello world!");
List test_data = read("Mydata.txt");

Code x = "int average(List column){
    return sum(column)/column.length;
};"

String y = "int average(List column){return
    sum(column)/column.length;};"

/*Returns the ascii string of how average
is defined that is acceptable by compiler*/
```

```
print(x)

/*Prints out 4*/
print(average([2,4,6])

/*Siimilar to REST POST*/
/*Note that Code objects causes changes to
both the local compiler, AND is available for
sending to a distributed interpreter/compiler.

That being said, the distributed compilers can
also take raw strings to process as code.*/

send(x, slave1);
send(y, slave1);

/*REST POST
and GET return values*/
String r_value = request(slave1, "average data");

/*prints Slave1's dataset average*/
print(r_value);

/*prints this machine's dataset average*/
print(average(test_data));
```

5 Team Roster

Manager Mikhail Kilmentov

Language Gurus Mikhail Kilmentov, Prateek Sinha

Tester Michael Nguyen

Environment Yuchen Zeng, Eli Bogom-Shanon