Map-Reduce Project Proposal

Ben Withbroe and Charlie Imhoff

Overview

Up to this point in our coursework, we've worked with single-threaded implementations of batched algorithms. While we've only been working with relatively small datasets, in many real world data mining settings, the amount of data is too large to be handled by a single process. In these scenarios, the data and the processing is split between many different machines.

With our final project, we'd like to expand our experiences to a distributed model for accomplishing large data mining tasks. Because we lack a large, distributed computing network, we'd like to simulate this kind of programming model via *multi-threaded programming on a single machine*.

In our final project, we seek to create a flexible framework for parallel processing large datasets on a single machine. Specifically, we will implement the basics of the Google Map-Reduce paradigm, providing multi-threaded mapping and reducing of data. As a proof of validity and to get a better sense of the speed improvements attainable through this approach, we'd then use our Map-Reduce framework to reimplement K-Nearest-Neighbors.

Approach

We will use C++ as our programming language. With C++ 11, closures allow for functional-style application of code into other functions. This allows us to create truly generic map and reduce functions. C++ also offers great libraries and systems for multi-threaded processing. The specificity of the memory model allows us to be very clear about how memory is used and shared, which will help us understand the challenges of splitting and combining processing across multiple workers.

The two major elements of our system is our map and reduce function.

```
 - map(datastream, map_closure) --> mapped_datastream
 - reduce(initial_result, datastream, reduce_closure) --> result
```

The "datastream" in this context, refers to an abstraction over the dataset, as we'd like to allow our Map-Reduce implementation to not require the entire dataset be loaded at once on a single thread. To allow this, the datastream interface may look like this:

```
 - item_at(index) --> datapoint
 - count() --> int
```

An important detail for this API is allowing for generic parameters and result types. Without generic types, using our Map-Reduce framework would be limited to only the scenarios we explicitly allowed for (which substantially decreases flexibility).

Resources

- MapReduce: Simplified Data Processing on Large Clusters
- The k-Nearest Neighbor Algorithm Using MapReduce Paradigm
- Lambda Closures in C++
- Templates and Generic Programming in C++

Presentation Logistics

We might be unable to present on Friday, May 26th, as Ben has a chance of being in Ohio for Track and Field nationals.