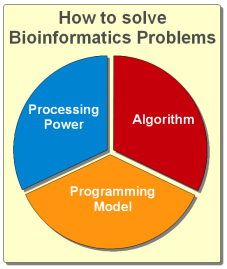
Parallel Processing Solutions in Bioinformatics and in Human Genome Research

Prof. Christian Bach, Prof. Hassan Bajwa, Tamas Ragoncsa,
Department of Technology Management
University of Bridgeport, Bridgeport, CT



1 The Problem

Biomedical science usually deals with huge amount of data. There are complex and costly experiments, which are beneficial to simulate on computers and there are huge data result sets which must be analysed by machines.

One of the most popular topics in biomedical science is human genome research. In many cases to answer a biomedical questions we have to know how certain proteins bind to the human genome. The majority of these problems can be modeled with various sub-sequence search algorithms on the chromosomes of the human genome.

To deal with these problems we need three things: 1. Processing power to run our computations fast enough 2.A programming model, which can give us a framework and various services so we can focus on the original problem instead of getting lost in details. 3. An algorithm, which can give us a well defined sequence of methods and procedures to describe how to solve the problem.

2 Hadoop – HDFS

How can we create a cheap and failure tolerant network from unreliable nodes? The Hadoop framework helps us to do that. The Hadoop project implements HDFS (Hadoop Distributed File System) HDFS is constructed from multiple physical nodes but it looks as one transparent logical file-system for the outside world. It has many beneficial characteristics like: recovery from failure, high throughput, portability across heterogeneous hardware and software.

The system contains a name node and an arbitrary number of data nodes. User data never flows through the NameNode. The NameNode is dedicated to handle meta data of the filesystem and serves as an index. HDFS supports a traditional hierarchical file organization and operations (like create, delete, copy or move a file, create a directory, delete directory etc.) It stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance. When a client reads data HDFS tries to select the closest replica (rack-aware).

3 MapReduce

"MapReduce is a programming model and an associated implementation for processing and generating large data sets." (Dean & Ghemawat, 2004) Its purpose is to provide a framework for software developers to build parallel processing applications without worrying about fault tolerance, distribution, load balancing and parallelization since all of these services are provided by the framework. Hadoop implements the MapReduce framework so these services are also available for Hadoop users.
When working with the MapReduce framework the developer has to build only a "map" and a "reduce" function.

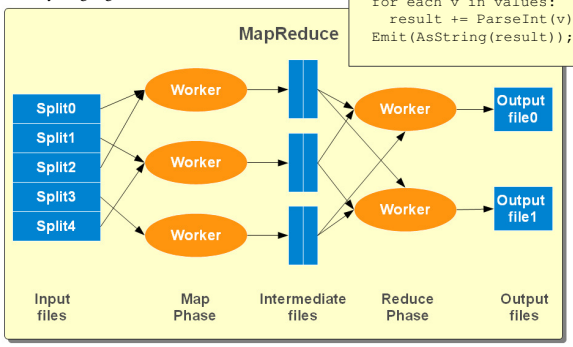
```
map (k1,v1)          → list(k2,v2)
reduce (k2,list(v2))→ list(v2)
```

For example word counting

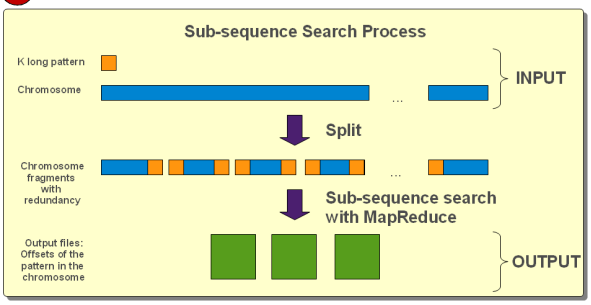
The application will perform the initialization with a dedicated master node. It sets up its map and reduce functions, the location of the input data, the number of splits (how many inputs should be created) so how many map tasks should be performed by the workers and how many output files are required so how many reduce tasks should be performed. The master node will schedule the tasks by assigning them to idle workers.

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```



4 Finding sub-sequences in the human genome



How to find all the occurrences of an arbitrary sequence in the human genome? This problem can be transformed into a mathematical problem: How to find an arbitrary subsequence (substring) in a very long sequence (string) in a timely manner?

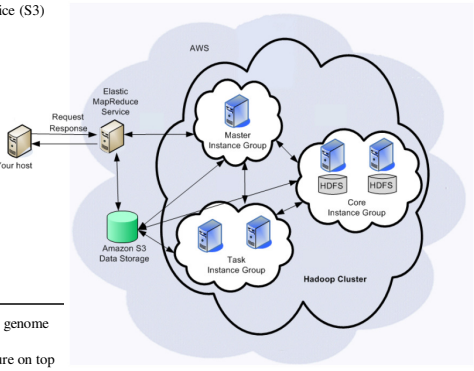
Subsequence
A subsequence of {a} is a sequence {b} defined by $b_k = a_{n_k}$, where $n_1 < n_2 < \cdots$ is an increasing sequence of indices. (Weisstein, 2000)
From a biomedical point of view it is also relevant to find not only exact matches, but some partial matches as well. We apply a simplification to look only for full matches in this research.
There are many existing algorithms available (LCS, KMP, RMAP), so it's easy to build a custom algorithm based on existing ones which is designed to deal with a specific problem.
Using some ideas from RMAP and the technologies described so far, we can build a sub-sequence search process.

```
G:
        ABCDEFGHIJABCDEFGHIJ
P:
        EFG
fragments with positions:
        ABCDEFG 0
        HIJABCD 7
        EFGHIJ 14
fragments with overlaps (this
is the input for Mapreduce):
        ABCDEFG 0
        FGHIJABCD 5
        CDEFGHIJ 12
Expected result (the output
from Mapreduce):
        EFG 4
        EFG 14
```

5 AWS – Elastic Mapreduce

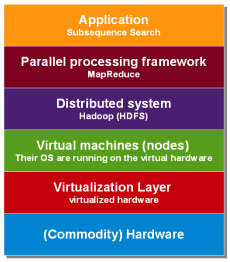
Acquisition of hardware, installation and configuration of the required software and in general the building of a distributed system can be challenging and time consuming. (Not to mention it may cost a lot of money.) Genome search applications, by their nature require huge processing power and can utilize distributed environments that can provide highly parallel computation. Besides that this high computational power is necessary only for a relatively short time (until we execute a particular calculation). These requirements suggest that utilizing a cloud based solution can be a viable option.

Amazon Elastic Compute Cloud (EC2) provides access to arbitrary number of processing units on a very flexible way. Amazon Simple Storage Service (S3) provides storage service for EC2 computing units. Amazon Elastic MapReduce, built on EC2 and S3 provides a complete Hadoop (HDFS and MapReduce) environment pre-installed. This gives us the opportunity to focus only on the problem we are seeking a solution for (searching for subsequences in the human genome).



	Test 1	Test 2
Sequence	Chromosome 1	The whole human genome
Subsequence (pattern)	one 9 bases long	one 9 bases long
Number of EC2 nodes	11 (1 master instance and 10 core instance)	11 (1 master instance and 10 core instance)
Type of EC2 nodes	Small	Small
Execution time	2 min 13.807 sec	8 min 42.100 sec

6 Conclusion



As a conclusion we can state that using currently available technologies it's possible to build human genome research applications quickly and cost efficiently.
The reason why it is quick and needs only minimal amount of programming is the layered architecture on top of which our application sits.

1. Commodity hardware – provides processing power on the physical devices
2. Virtualization Layer – builds a software layer on top of the hardware, so the structure of the physical architecture won't limit the structure of the virtual architecture.
3. Virtual Machines – a set of virtual resources can be allocated to work as one virtual machine and make a regular operating system run on it.
4. Distributed system – a layer that hides the details of the complexity of a multinode system and provides high level services like read/write operations to a distributed file system (HDFS).
5. Parallel processing framework – takes care of the parallel data processing execution. It deals with load balancing, failure recovery and coordination.
6. The application developer has to build only an algorithm that fulfils the requirements of the framework (can be executed with the MapReduce approach in this case) and has to implement only the necessary task (a simple substring search in our case).

References:

- Amazon Elastic MapReduce Developer Guide (API Version 2009-03-31) Retrieved 1/29, 2012, from http://docs.amazonwebservices.com/ElasticMapReduce/
- Apache Hadoop Project. (2008). from http://hadoop.apache.org/
- Bach, C., Bajwa, H., & Erodula, K. (2011). Use of Multi Threaded Asynchronous DNA Sequence Pattern Searching Tool to Identifying Zinc-Finger-Nuclease Binding Sites on the Human Genome.
- Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. Paper presented at the 6th Symp. on Operating Systems Design & Implementation.
- Eppstein, D. (1996). ICS 161: Design and Analysis of Algorithms Lecture notes for February 27, 1996. Retrieved 11/5, 2011, from http://www.ics.uci.edu/~eppstein/161/960227.html
- Leo, S., Santoni, F., & Zanetti, G. (2009). Biodoop: Bioinformatics on Hadoop. Paper presented at the International Conference on Parallel Processing Workshops.
- Schatz, M. (2009). CloudBurst: Highly Sensitive Short Read Mapping with MapReduce. Retrieved 11/6, 2011, from http://sourceforge.net/apps/mediawiki/cloudburst-bio/index.php?title=CloudBurst
- Smith, A. D. (2009). The RMAP software for short-read mapping. Retrieved 11/6, 2011, from http://rulai.cshl.edu/rmap/
- Weisstein, E. W. (2000). "Subsequence.". MathWorld--A Wolfram Retrieved 11/3, 2011, from http://mathworld.wolfram.com/Subsequence.html
- Ragoncsa, T., Bach C., Bajwa H. (2011). The Use of Knowledge Networks in the Biomedical Sciences and Parallel Processing Solutions in Bioinformatics and in Human Genome Research. Master's Thesis, University Of Bridgeport