# Project #1 Hadoop Blast (Due 10/17/2011)

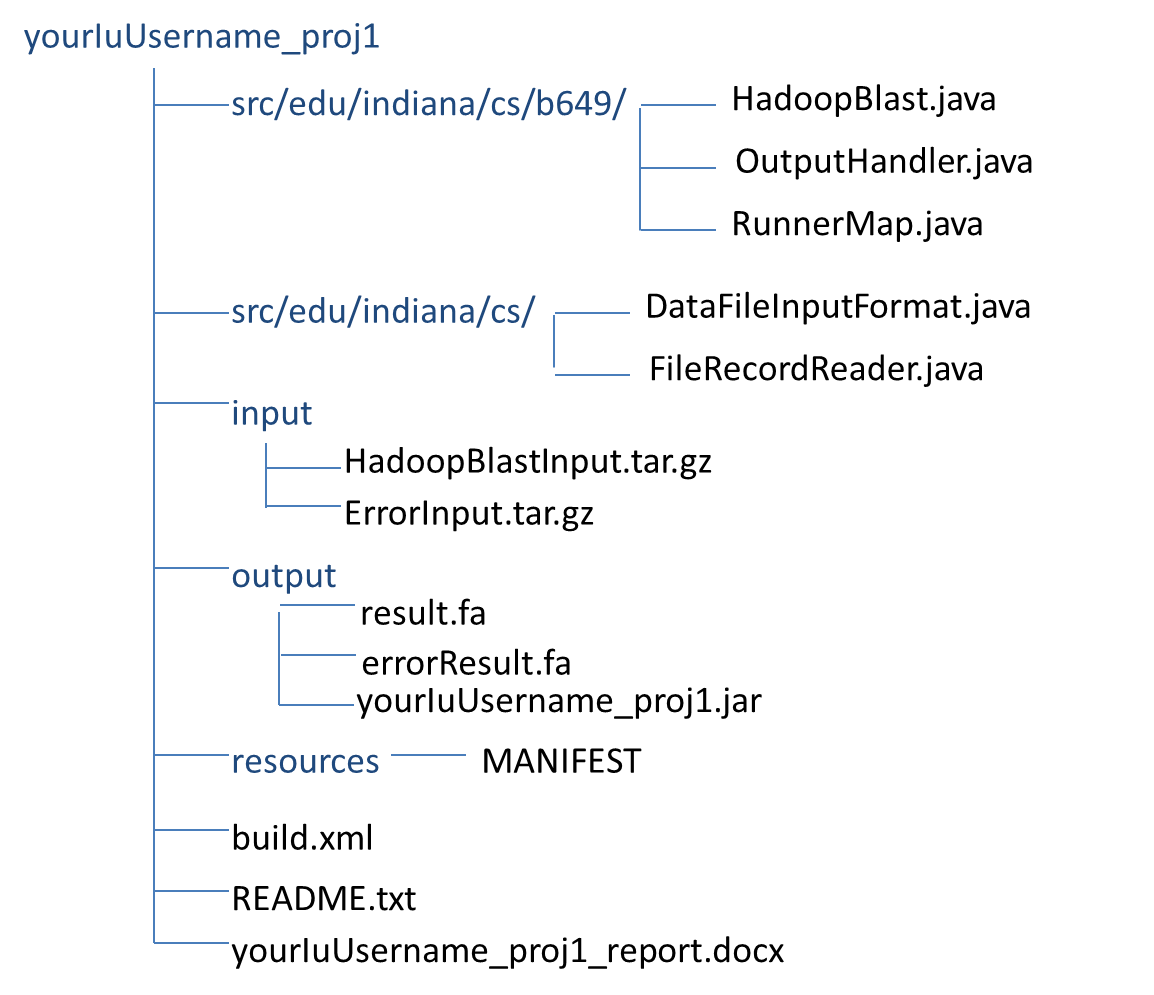
# Introduction

The goal of this project is to write an advance Hadoop program with a bioinformatics application BLAST [1]. Two parts need to be finished: (1) Implement a “map-only” Hadoop BLAST program with the provided sketch code (It is just a simple interface, you can modify it into any custom version.), (2) Written report with speedup performance analysis.

***Hadoop 0.20.203.0***  
We found that the hadoop 0.21.0 has some incompatible issues with overriding the RecordReader class, so, we switch back to hadoop.0.20.203.0 stable version, please download [here](http://salsahpc.indiana.edu/csci-b649-2011/files/hadoop-0.20.203.0-b649-2011-beta.tar.gz) [7].

# Deliverables

You are required to submit a zip package named "yourIuUsername\_proj1.zip", with yourIuUsername replaced by your IU username, e.g. “john\_proj1.zip”. We are attaching a submission template package with this instruction document, which is composed of the following directory structure **(this is same as submission structure)**:



1. Blue strings denote directory names. And please put all your files under a directory "yourIuUsername\_proj1", e.g. "john\_project1".
2. Under directory of "src/edu/Indiana/cs/", "DataFileInputFormat.java" and "FileRecordReader.java" are the customized Hadoop InputFormatClass you will need to implement which generates the key-value pair as <filename, path\_on\_HDFS> with a given HDFS input directory.
3. Under directory of "src/edu/Indiana/cs/b649/", you can create more java file(s) if necessary. Here, "HadoopBlast.java" is the main java class for the entire job; "RunnerMap.java" is the sketch code which you need to implement. Both of them must be included.
4. "input.zip" is your assigned input queries.
5. "result.fa" is a single result collect from different map tasks.
6. Write the execution steps for your program in "README.txt".
7. A report in docx format with speedup performance analysis.

Points will be reduced if the filename and directory structure are different from instructed above.

# Evaluation

The total points of project #1 is 5, where the distribution is as following

* 1. Completeness of your code (2.B. and 2.C.) (2.5 point)
  2. Readability and clarity of README.txt (2.F.) (0.5 point)
  3. Correctness of written report 2.G. (2 point)

# Project Description

In this project, you will firstly understand the usage of NCBI Blast Binary, then write the Hadoop Blast with implementing the FileRecordReader.java and RunnerMap.java, finally test and run your program in two different environments, CS Linux Machines and FutureGrid Eucalyptus. For detail information of using these machines, please refer to the Appendix of Project#0 Part1.

## Blast (NCBI blastx):

NCBI BLAST+ (Basic Local Alignment Search Tool) is one of the most widely used bioinformatics applications written in C++, and the version we are using is v2.2.23. This version is considered as a new version of the software with new features and better performance while BLAST is a legacy software.

There are two versions of BLAST+ program, 32 bit [2] and 64 bit [3], please select the proper version according to the installed OS. Normally, we use Blast+ 64 bit version for our project.

### *How to use it*

We provide a customized Database [4], please download BLAST binary [2-3] and try to execute the standalone BLAST+ with the following command [5]:

bin/blastx -query query\_file -db dataBase\_DIR/dataBase\_NAME -outfmt 6 -seg no -out output\_FileName

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| query\_file | The full path of the query file, e.g. ~/blast/input/celllines\_1.fa |
| dataBase\_DIR | The name of the directory relative to Blast Database, e.g. ~/blast/db/ |
| dataBase\_NAME | The name of formatted Blast Database, e.g. nr |
| output\_FileName | The name of the output file, e.g. out.txt |

cd ~  
mkdir blast  
wget http://salsahpc.indiana.edu/tutorial/apps/BlastProgramAndDB.tar.gz  
tar -zxvf BlastProgramAndDB.tar.gz -C blast  
cd blast  
bin/blastx -query ~/blast/input/input.fa -db ~/blast/db/nr -outfmt 6 -seg no -out out.fa

### *Result*

After running the BLAST binary with a given query, you will get this result with using linux “cat” command

cd ~/blast  
cat out.fa

BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297695302|ref|XP\_002824885.1| 100.00 11 0 0 3 35 12 22 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297677746|ref|XP\_002816750.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297677738|ref|XP\_002816709.1| 100.00 11 0 0 3 35 18 28 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297677736|ref|XP\_002816708.1| 100.00 11 0 0 3 35 13 23 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297662912|ref|XP\_002809930.1| 100.00 11 0 0 3 35 13 23 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297290467|ref|XP\_002803717.1| 100.00 11 0 0 3 35 29 39 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297269450|ref|XP\_002799874.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297269448|ref|XP\_002799873.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297269446|ref|XP\_002799872.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297269444|ref|XP\_002799871.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297269442|ref|XP\_002799870.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|297269440|ref|XP\_002799869.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|296409582|gb|ADH15624.1| 100.00 11 0 0 3 35 11 21 7.0 27.7  
BG3:2\_30MNAAAXX:7:1:981:1318/1 gi|296482166|gb|DAA24281.1| 100.00 11 0 0 3 35 36 46 7.0 27.7

**FileRecordReader – Customized Key-Value Pair**   
In order to make our customized Map input key-value pair as <filename, path\_to\_HDFS> with type as <String, String>, we need to override the hadoop abstract class “org.apache.hadoop.mapreduce.RecordReader”. Mainly, you will need to implement the following function:

* public void initialize
* public float getProgress
* public String getCurrentKey
* public String getCurrentValue
* public boolean nextKeyValue
* public void close

***Hints:*** to achieve this part, understand the function “public void initialize” is important, where “split” contain the current file information (after changing its type from InputSplit to FileSplit). After that, you can simply write a wordcount-like program and check the correction of new key-value pairs.

private Path path;

@Override  
public void initialize(InputSplit split, TaskAttemptContext context)   
throws IOException, InterruptedException {  
}

**RunnerMap**  
This is the Mapper class of Hadoop Blast program, you will implement this part with providing the following functionalities:

* Get the Blast database from distributed cache to local
* Download the input file (each map tasks) with referring the correct key-value pair
* Invoke an external process to run the Blast Binary with provided input file
* Capture the error message (if there is any) from execution of the Binary binary process
* Upload the local Blast output (each map task) to HDFS

### *Distributed Cache*

Distributed Cache [6] distribute the application specific read only data to each worker node locally once before running the first map task on each worker. We here use this function to handle the zipped Blast binary and database “[BlastProgramAndDB.tar.gz](http://salsahpc.indiana.edu/tutorial/apps/BlastProgramAndDB.tar.gz)” [8] to send across multiple worker nodes with one-time setup.

### *Error Message*

To prove that the Blast binary is correctly running, we provide two different dataset; one with correct result, another with error message during the external Blast execution. You are required capture the error message from the “error data set”, and create an “errorResult.fa” file after all.

**Collect output data as single result.fa and errorResult.fa**   
The Hadoop Blast is a “map-only” hadoop program, so, there is no reduce task to collect the final result into a single file; you are not required to write such reducer to handle this part. Therefore, you can simply use hadoop and linux command to collect those output and put them into a single file.

**Blast Performance test**  
In this part of project, you will need to design a performance test with the provided input data set [5]. Mainly, you are required to get the performance in a single node mode and two nodes mode with each node running at least 1 mappers/reducers (maximum of 8/8) on the two different environments — CS and FutureGrid Eucalyptus (c1.xlarge). The performance test should be observed with at least 10 runs. You can change the total amount of mapper and reducer by editing the configuration files “conf/mapred-site\_template.xml”and “conf/mapred-site.xml”. During the test, please mark down the following information and make a table for each mode:

1. Number of mappers on each node
2. Number of reducers on each node
3. Rotal number of nodes (2)
4. Execution time (an average of 10 runs).
5. Speedup (between single node and two nodes mode)

Then, based on these statistic numbers, draw execution time line chart(s) and speedup line chart(s) with explanation.

**References:**

[1] NCBI Blast <http://blast.ncbi.nlm.nih.gov/Blast.cgi>

[2] NCBI Blast+ 32 bit Binary, ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.23/ncbi-blast-2.2.23+-ia32-linux.tar.gz

[3] NCBI Blast+ 64 bit Binary, ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.23/ncbi-blast-2.2.23+-x64-linux.tar.gz

[4] Blast Customized Database, <http://salsahpc.indiana.edu/csci-b649-2011/files/project1/db.tar.gz>

[5] <http://salsahpc.indiana.edu/tutorial/hadoopblast.html>

[6] Hadoop Official Distributed Cache description, <http://hadoop.apache.org/common/docs/current/api/org/apache/hadoop/filecache/DistributedCache.html>

[7] hadoop.0.20.203.0 package, <http://salsahpc.indiana.edu/csci-b649-2011/files/hadoop-0.20.203.0-b649-2011-beta.tar.gz>

[8] Blast Customized Binary and Database, http://salsahpc.indiana.edu/tutorial/apps/BlastProgramAndDB.tar.gz