Optimising Sequence Alignment in Cloud using Hadoop and MPP Database

Senthilkumar Vijayakumar, Anjani Bhargavi, Syed Azar Ahamed and Uma Praseeda

TATA Consultancy Services LTD
Bangalore
India
Problem Statement

- Huge datasets are pouring out of high-throughput genome centres, but with the advent of ultra high-throughput sequencing, genotyping and other functional genomics in different laboratories we are facing a highly challenging new era in petabyte scale data handling.

- GenBank®, the NIH genetic sequence database containing a collection of all publicly available DNA sequences has approximately 126,551,501,141 bases in 135,440,924 sequence records in the traditional GenBank divisions and 191,401,393,188 bases in 62,715,288 sequence records in the WGS division as of April 2011. The storage and retrieval of this massive data poses a growing problem.

- A highly available, protected, secure, parallel processing database is required to address this issue. One has to consider storage, in addition to having the required capacity, it has to be made sure that its available to the compute(network), and that sufficient I/O to do anything in real-time is present. Software language and implementation become critical factors when dealing with terabytes of data. With such high-intensity computing, power (getting enough), cooling, etc. become real issues.

- One more motivating factor for developing new solutions in bioinformatics is the fact that the existing local and global alignment algorithms slow down when processing huge data and matching a query sequence with large database of sequences.

- This pattern matching is of critical importance in the study of gene mutations and analysis of rare diseases. This paper brings to light all these areas of advancements. An attractive Cloud Computing solution may very well provide massively scalable computational power and green credentials too, so long as the off-site compute is located where renewable sources of energy are used preferentially.

- A massively parallel processing database in the form of Greenplum, coupled with the computational brilliance of Hadoop, built on the foundation of Cloud and virtualization with an optimized FASTA algorithm is “the next generation solution”.
The Next Generation Solution

- Cloud Computing solution may very well provide massively scalable computational power
- Green credentials
- Pay for what you use

Openstack Cloud

Greenplum MPP Database

Hadoop

- Greenplum MPP database infrastructure coupled with computational capabilities to provide faster querying, analysis and management of the large unstructured datasets will be definitely the future for new generation sequence analysis.

- The complex nature of the FASTA algorithm, coupled with data and computational parallelism of Hadoop grid and MPP database for querying from big datasets containing large sequences, improves performance and optimizes querying from big datasets.
Openstack Cloud Infrastructure

Open source software for building private and public clouds.
OpenStack: The Open Source Cloud Operating System

- OpenStack Software delivers a massively scalable Cloud Operating System.
- OpenStack comprises of three core projects like Compute, Object Storage and Image Service.
  - **OpenStack Compute**: Open Source software and standards for large-scale deployments of automatically provisioned virtual compute instances.
  - **OpenStack Object Storage**: Open Source software and standards for large-scale, redundant storage of static objects.
  - **OpenStack Image Service**: Provides discovery, registration, and delivery services for virtual disk images.
Server1 runs all the components of Nova, Glance, Swift, Keystone and Horizon (OpenStack Dashboard).

Server2 runs nova-compute, Hadoop and Greenplum instances as well.
Openstack Dashboard – Instances & Volumes

Instances

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>IP Address</th>
<th>Size</th>
<th>Status</th>
<th>Task</th>
<th>Power State</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-ww.demo.com</td>
<td>10.4.128.20</td>
<td>4GB RAM</td>
<td>2 VCPU</td>
<td>10.0GB</td>
<td>Active</td>
<td>None</td>
</tr>
<tr>
<td>test-ww.demo.com</td>
<td>10.4.128.19</td>
<td>4GB RAM</td>
<td>2 VCPU</td>
<td>10.0GB</td>
<td>Build</td>
<td>Spawning</td>
</tr>
<tr>
<td>myserve</td>
<td>10.4.128.18</td>
<td>2GB RAM</td>
<td>1 VCPU</td>
<td>10.0GB</td>
<td>Active</td>
<td>None</td>
</tr>
<tr>
<td>myserver</td>
<td>10.4.128.16</td>
<td>2GB RAM</td>
<td>1 VCPU</td>
<td>10.0GB</td>
<td>Active</td>
<td>None</td>
</tr>
</tbody>
</table>

Volumes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Size</th>
<th>Status</th>
<th>Attachments</th>
<th>Actions</th>
</tr>
</thead>
</table>

2012 Storage Developer Conference. © TATA Consultancy Services LTD. All Rights Reserved.
What is FASTA Algorithm

- FASTA takes a given nucleotide or amino-acid sequence and searches a corresponding sequence database by using local sequence alignment to find matches of similar database sequences.

- The FASTA program follows a largely heuristic method which contributes to the high speed of its execution. It initially observes the pattern of word hits, word-to-word matches of a given length, and marks potential matches before performing a more time-consuming optimized search using a Smith-Waterman type of algorithm.
Why do we need Local Alignments

- To compare a short sequence to a large one.
- To compare a single sequence to an entire database.
- To compare a partial sequence to the whole.
- Identify newly determined sequences.
- Compare new genes to known ones.
- Guess functions for entire genomes full of ORFs of unknown function.
Basic Idea

- A good alignment contains subsequences of absolute identity (short lengths of exact matches):
  - First, identify very short exact matches.
  - Next, the best short hits from the first step are extended to longer regions of similarity.
  - Finally, the best hits are optimized.
FASTA

- Derived from logic of the dot plot
  - compute best diagonals from all frames of alignment

- The method looks for exact matches between words in query and test sequence
  - DNA words are usually 6 nucleotides long
  - protein words are 2 amino acids long
First, look for all identities between small “word” = ktup and every sequence in database. Ktup size determines how many letters must be identical (e.g., 3)
FASTA Algorithm...

- Ktup matches can be depicted in a matrix; diagonals indicate matches. For every library sequence, the 10 best diagonals are constructed from the ktup matches using a distance formula.

```
+---+---+---+---+---+---+---+
|   | a | g | c | t | g | a | c | g | c | a |
+---+---+---+---+---+---+---+---+---+---+---+
| C |   | - |   | - | - |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
| T |   | - |   |   | - |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
| G | - |   | - | - |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
| C | - |   | - | - | - |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
| A | - |   | - | - | - |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
| C | - |   | - | - | - |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
| A | - |   | - | - | - |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+
```
FASTA Algorithm...

- The top 10 diagonals are rescored using substitution matrices and identities smaller than ktup; each of these rescored diagonals is called an initial region.
Initial regions are then joined using a joining penalty (like a gap penalty). The highest score of the joined initial regions is then the score for that library sequence. The library sequences are ranked by this score.
FASTA Algorithm…

(a) Find runs of identical words

(b) Re-score using PAM matrix
Keep top scoring segments
FASTA Format

- simple format used by almost all programs
- [>] header line with a [hard return] at end
- Sequence (no specific requirements for line length, characters, etc)

>URO1 uro1.seq  Length: 2018  November 9, 2000 11:50  Type: N  Check: 3854 ..
CGCAGAAAGAGGAGGCGCTTGCCCTTCAGCTTGTTGGGAATCCCGAAGATGGCCAAAGAC
AACTCAAATCTTTTTGCTTGGCTTCCAGGGCCTGCTGATTTTTGGAAATGTGATTATTGGTTT
GCGGCATTGCCCCTGACTGCGGAGTGCACTCTTCTTCTGTAACAAACACAGCGCTCTACC
CACTGCTTGAAGCCACCCGACAACGATGACATCTATGCGGCTGGATCGGATCGGATATTTTG
TGGGCCATCTCGCCTCTTCTGCTGCTGTTCTAGGCATTGTAGGCATCATGAATGGGCTT
Assessing Alignment Significance

- Generate random alignments and calculate their scores
- Compute the mean and the standard deviation (SD) for random scores
- Compute the deviation of the actual score from the mean of random scores

\[ Z = \frac{\text{meanX}}{\text{SD}} \]

- Evaluate the significance of the alignment
- The probability of a Z value is called the E score
Local Vs. Global Alignment

- The Global Alignment Problem tries to find the longest path between vertices (0,0) and (n,m) in the edit graph.

- Local Alignment—better alignment to find conserved segment
## FASTA Diagonal Computation

$$L(T, W)(J) = \{5, 8\} \quad L = i-j = 4-5 = -1; \quad S(-1) = 0+1 = 1; \quad L = 4-8 = -4; \quad S(-4) = 0+1 = 1;$$
**FASTA Diagonal Computation**

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
<th>W</th>
<th>R</th>
<th>T</th>
<th>W</th>
<th>S</th>
<th>T</th>
<th>W</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>F</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ L(W, T) \{J\} = \{9\} \]

\[ L = j - j = 5 - 9 = -4; \quad S(-4) = 1 + 1 = 2; \]
## FASTA Diagonal Computation

<table>
<thead>
<tr>
<th>J</th>
<th>F</th>
<th>L</th>
<th>W</th>
<th>R</th>
<th>T</th>
<th>W</th>
<th>S</th>
<th>T</th>
<th>W</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$L(F, L) (J) = \{1\}$

$L = j - j = 7 - 1 = 6$; $S(6) = 0 + 1 = 1$;

2012 Storage Developer Conference. © TATA Consultancy Services LTD. All Rights Reserved.
### FASTA Diagonal Computation

<table>
<thead>
<tr>
<th>$S_l$</th>
<th>-10</th>
<th>-9</th>
<th>-8</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S_l$</th>
<th>-10</th>
<th>-9</th>
<th>-8</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Other Software using Hadoop in Bioinformatics

- Cloudburst
- Crossbow
- Contrail
- Myrna
- GSEA (Gene Set Enrichment Analysis)
- CloudBLAST
Hadoop – Map Reduce
Map-Reduce Operation – Genome Sequencing

Sequences from various nodes are fed to Map Phase for Data Pre-processing in Parallel.

Exchange of intermediate results between nodes.

Hadoop guarantees: Keys ordered. Values are not ordered can use secondary keys if desired parallel processing as well.

<table>
<thead>
<tr>
<th>tcsnode-0-1-2</th>
<th>tcsnode-0-1-3</th>
<th>tcsnode-0-1-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAAGGACGTGACAA</td>
<td>TATTAATGCAATGAG</td>
<td>ACGTCAACGTGATCG</td>
</tr>
<tr>
<td>TTATCATCGGAAGCT</td>
<td>TGAGTTGCACTTAAG</td>
<td>TTTATCTCTCGAAAACGT</td>
</tr>
<tr>
<td>TAGATCACGTTTTTA</td>
<td>ATAGGGACCATGTAG</td>
<td>ATTCCATAGTGAGTG</td>
</tr>
<tr>
<td>CCGGACGAAACCA</td>
<td>AGTGGTGCACTGAT</td>
<td>TTATCGTATGCTAG</td>
</tr>
</tbody>
</table>

Map Phase

Shuffle Phase

(Key, Value)
- (ACGT, CAAGGACGTGACAA)
- (TGCA, TATTAATGCAATGAG)
- (ACGT, TAGATCACGTTTTTA)

(Key, Value)
- (ACGT, CAAGGACGTGACAA)
- (ACGT, TAGATCACGTTTTTA)
- (ACGT, CCATAGCGTACGTC)

Reduce Phase

Reduce Output

<table>
<thead>
<tr>
<th>(Key, Value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TGCA, TATTAATGCAATGAG)</td>
</tr>
<tr>
<td>(TGCA, AGTGGTGCACTGAT)</td>
</tr>
<tr>
<td>(TGCA, TATTAATGCAATGAG)</td>
</tr>
<tr>
<td>(TGCA, TGAGTTGCACTTAAG)</td>
</tr>
<tr>
<td>(TGCA, AGTGGTGCACTGAT)</td>
</tr>
</tbody>
</table>
Map Operations

- **Map Processing**: HDFS splits the large input data set into smaller data blocks (64 MB by default) controlled by the property `dfs.block.size`. Data blocks are provided as an input to map tasks. The number of blocks to each map depends on `mapred.min.split.size` and `mapred.max.split.size`. The block data is split into key value pairs based on the Input Format. The map function is invoked for every key value pair in the input. Output generated by map function is written in a circular memory buffer, associated with each map. The buffer is 100 MB by default and can be controlled by the property `io.sort.mb`.

- **Spill**: When the buffer size reaches a threshold size controlled by `io.sort.spill.percent` (default 0.80 or 80%), a background thread starts to spill the contents to disk. While the spill takes place map continues to write data to the buffer unless it is full. Spills are written in round-robin fashion to the directories specified by the `mapred.local.dir` property, in a job-specific subdirectory. A new spill file is created each time the memory buffer reaches to spill threshold.

- **Partitioning**: Before writing to the disk the background thread divides the data into partitions (based on the partitioner used) corresponding to the Reducer where they will be sent.

- **Sorting**: In-memory sort is performed on key (based on `compareTo` method of key class). The sorted output is provided to the combiner function if any.

- **Merging**: Before the map task is finished, the spill files are merged into a single partitioned and sorted output file. The configuration property `io.sort.factor` controls the maximum number of streams to merge at once; the default is 10.

- **Compression**: The map output can be compressed before writing to the disk for faster disk writing, lesser disk space, and to reduce the amount of data to transfer to the Reducer. By default the output is not compressed, but it is easy to enable by setting `mapred.compress.map.output` to true. The compression library to use is specified by `mapred.map.output.compression.codec`. Output file partitions are made available to the Reducers over HTTP. The number of worker threads used to serve the file partitions is controlled by the task `tracker.http.threads` property—this setting is per tasktracker, not per map task slot. The default of 40 may need increasing for large clusters running large jobs.
Reduce Operations

- **Copy**: Each map task's output for the corresponding Reducer is copied as soon as map task completes. The reduce task has a small number of copier threads so that it can fetch map outputs in parallel. The default is 5 threads, but can be changed by setting the mapred.reduce.parallel.copies property. The map output is copied to the reduce tasktracker's memory buffer which is controlled by mapred.job.shuffle.input.buffer.percent (specifies the proportion of the heap to use for this purpose). When the in memory buffer reaches a threshold size (controlled by mapred.job.shuffle.merge.percent), or reaches a threshold number of map outputs (mapred.inmem.merge.threshold), it is merged and spilled to disk. As the copies accumulate on disk, a background thread merges them into larger, sorted files. This saves some time in subsequent merging.

- **Sort**: This phase should actually be called the *Merge phase* as the sorting is done at the map side. This phase starts when all the maps have been completed and their output has been copied. Map outputs are merged maintaining their sorting order. This is done in rounds. For example if there were 40 map outputs and the merge factor was 10 (the default, controlled by the io.sort.factor property, just like in the map’s merge) then there would be 4 rounds. In first round 4 files will be merged and in remaining 3 rounds 10 files are merged. The last batch of files is not merged and directly given to the reduce phase.

- **Reduce**: During reduce phase the reduce function is invoked for each key in the sorted output. The output of this phase is written directly to the output filesystem, typically HDFS.
Parameters affecting Performance in Hadoop

**dfs.block.size: File System Block Size**

Default: 67108864 (bytes) = 64 MB

**Suggestions:**
- Small cluster and large data set:
  - Default block size will create a large number of map tasks.
  - e.g. Input data size = 160 GB and dfs.block.size = 64 MB then the minimum no. of maps = (160*1024)/64 = 2560 maps.
- If dfs.block.size = 128 MB minimum no. of maps = (160*1024)/128 = 1280 maps. If dfs.block.size = 256 MB minimum no. of maps = (160*1024)/256 = 640 maps. In a small cluster (6-7 nodes) the map task creation overhead is considerable. So dfs.block.size should be large in this case but small enough to utilize all the cluster resources.

**mapred.compress.map.output:**

**Map Output Compression**

Default: False

**Pros:** Faster disk write, saves disk space, less time in data transfer (from Mappers to Reducers).

**Cons:** Overhead in compression at Mappers and decompression at Reducers.

**Suggestions:** For large cluster and large jobs this property should be set true. The compression codec can also be set through the property mapred.map.output.compression.codec (Default is org.apache.hadoop.io.compress.DefaultCodec).
Parameters affecting Performance in Hadoop (Cont..)

**io.sort.mb: Buffer Size (MBs) for Sorting**

Default: 100

**Suggestions:**
- For large jobs (the jobs in which map output is very large), this value should be increased keeping in mind that it will increase the memory required by each map task. So the increment in this value should be according to the available memory at the node.
- Greater the value of io.sort.mb, lesser will be the spills to the disk, saving write to the disk.

**io.sort.factor: Stream merge factor**

Default: 10

**Suggestions:**
- For large jobs (the jobs in which map output is very large and number of maps are also large) which have large number of spills to disk, value of this property should be increased.
- Increment in io.sort.factor, benefits in merging at Reducers since the last batch of streams (equal to io.sort.factor) are sent to the reduce function without merging, thus saving time in merging.

**mapreduce.job.reuse.jvm.num.tasks:**

**Reuse single JVM**

Default: 1

**Suggestions:** The overhead of JVM creation for each task is around 1 second. So for the tasks which live for seconds or a few minutes and have lengthy initialization, this value can be increased to gain performance.

**mapreduce.reduce.parallel.copies:**

**Threads for parallel copy at Reducer**

Default: 5

- **Description:** The number of threads used to copy map outputs to the Reducer.

**Suggestions:** For large jobs (the jobs in which map output is very large), value of this property can be increased keeping in mind that it will increase the total CPU usage.
Parameters affecting Performance in Hadoop (Cont..)

**Parameters:**

- **mapred.map/reduce.tasks.speculative.execution**
  - **Enable/ Disable task (map/reduce) speculative execution**
  - **Default:** True

  **Pros:** Reduces the job time if the task progress is slow due to memory unavailability, hardware degradation.

  **Cons:** Increases the job time if the task progress is slow due to complex and large calculations. On a busy cluster speculative execution can reduce overall throughput, since redundant tasks are being executed in an attempt to bring down the execution time for a single job.

  **Suggestions:** In large jobs where average task completion time is significant (> 1 hr) due to complex and large calculations and high throughput is required the speculative execution should be set to false.

- **mapred.tasktracker.map/reduce.tasks.maximum:**
  - **Maximum tasks (map/reduce) for a tasktracker**
  - **Default:** 2

  **Suggestions:**
  This value should be set according to the hardware specification of cluster nodes and resource requirements of tasks (map/reduce).

  e.g. a node has 8GB main memory + 8 core CPU + swap space
  Maximum memory required by a task \(\sim 500\text{MB}\)
  Memory required by tasktracker, Datanode and other processes \(\sim (1 + 1 + 1) = 3\text{GB}\)
  Maximum tasks that can be run = \((8-3) \text{GB/500MB} = 10\)
  Number of map or reduce task (out of the maximum tasks) can be decided on the basis of memory usage and computation complexities of the tasks.

  The memory available to each task (JVM) is controlled by **mapred.child.java.opts** property. The default is \(-Xmx200\text{m}\) (200 MB). Other JVM options can also be provided in this property.
EMC Greenplum MPP Database
Query Optimization

- There can be many different ways to get an answer from a given query. The result would be same in all scenarios.

- DBMS strive to process the query in the most efficient way (in terms of ‘Time’) to produce the answer.

  Cost = Time needed to get all answers

- The estimation should be accurate and easy. Another important point is the need for being logically consistent because the least cost plan will always be consistently low.
Cost of physical and other Costs associated

- **Cost of physical plans** includes processor time and communication time. The most important factor to consider is disk I/Os because it is the most time consuming action.

- **Some other costs associated are:**
  - Operations (joins, unions, intersections).
  - The order of operations.

⚠️ **Why?**
  - Joins, unions, and intersections are associative and commutative.
  - Management of storage of arguments and passing of it.

Factors mentioned above should be limited and minimized when creating the best physical plan.
Parallel Dataflow Engine

- At the heart of the Greenplum Database is the Parallel Dataflow Engine.
- The Parallel Dataflow Engine is an optimized parallel processing infrastructure that is designed to process data as it flows from disk, from external files or applications, or from other segments over the gNet interconnect. The engine is inherently parallel – its spans all segments of a Greenplum Database cluster.
- Greenplum’s Parallel Dataflow Engine is highly optimized at executing both SQL and MapReduce, and does so in a massively parallel manner. It has the ability to directly execute all necessary SQL building blocks, including performance-critical operations such as hash-join, multi-stage hash-aggregation and arbitrary MapReduce programs.
Cost-based query Optimization: Transformation & Implementation

**Transformation**

**QUERY:**
SELECT p.pname, d.dname
FROM Patients p, Doctors d
WHERE p.doctor = d.dname
AND d.dgender = 'M'

**Implementation & Plan selection based on costs**
Greenplum Database's parallel query optimizer is responsible for converting SQL or MapReduce into a physical execution plan. It does this using a **cost-based optimization algorithm** in which it evaluates a vast number of potential plans and selects the one that it believes will lead to the most efficient query execution.

Unlike a traditional query optimizer, **Greenplum's optimizer takes a global view of execution across the cluster, and factors in the cost of moving data between nodes in any candidate plan.**

The benefit of this 'global' query planning approach is that it can use global knowledge and statistical estimates to build an optimal plan once and ensure all nodes execute it in a fully coordinated fashion. This leads to far more predictable results than the alternative approach of 'SQL pushing' snippets that must be replanned at each node.
## Cost measure and Parallelism

<table>
<thead>
<tr>
<th>Plan A</th>
<th>Plan B</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 IOs</td>
<td><strong>site 1</strong></td>
</tr>
<tr>
<td></td>
<td><strong>site 2</strong></td>
</tr>
<tr>
<td></td>
<td><strong>site 3</strong></td>
</tr>
<tr>
<td>50 IOs</td>
<td>70 IOs</td>
</tr>
<tr>
<td>50 IOs</td>
<td></td>
</tr>
</tbody>
</table>

Influence on response time: max (I/O), not ∑ I/O

Same holds for CPU time
Query Profiling

- Greenplum Database devises a query plan for each of the given query. Choosing the right query plan to match the query and data structure is absolutely critical for good performance. A query plan defines how the query will be executed in Greenplum Database’s parallel execution environment. By examining the query plans of poorly performing queries, you can identify possible performance tuning opportunities.

- The query planner uses the database statistics it has to choose a query plan with the lowest possible cost. Cost is measured in disk I/O and CPU effort (shown as units of disk page fetches). The goal is to minimize the total execution cost for the plan.
Response Time

- The total elapsed time took to run text query 'ACGT' was **22.548 milliseconds**.
- The sequential scan operation had only one segment (seg0) that returned rows, and it returned just 1 row. It took **0.255 milliseconds** to find the first row and 0.486 to scan all rows. The gather motion operation then received 1 row (segments sending up to the master). The total elapsed time for this operation was **0.537 milliseconds**.
- Through this optimized query plans in Greenplum database sequences can be fetched in much faster and efficient manner than traditional databases.

```
EXPLAIN ANALYZE SELECT * FROM sequence WHERE sequences = 'ACGT';

QUERY PLAN

Gather Motion 2:1 (slice1) (cost=0.00..20.88 rows=1 width=13)  recv: Total 1 rows with 0.305 ms to first row, 0.537 ms to end.  -> Seq Scan on 'sequence' (cost=0.00..20.88 rows=1 width=13) Total 1 rows (seg0) with 0.255 ms to first row, 0.486 ms to end.
  Filter: name::text ~~~'ACGT'::text
22.548 ms elapsed
```
EXPERIMENTAL RESULTS AND ANALYSIS
Greenplum MPP Vs Conventional Database

- In our experiment, to compare the performance of MPP database with the Conventional database (PostgreSQL), a query was executed to read sequences each of 1024KB. The number of sequences to be fetched from database was also changed and performance was noted.

- It was observed that the time in fetching the sequences from the database decreased by 57.22% in case of Greenplum (optimised query plans) with Hadoop when compared to PostgreSQL.
During our experiment we tried aligning sequences each of different sizes for a set of 30 sequences. The analysis shows that time increases as the pairwise alignment depends on the size of the sequences. As per the investigations also it was found that as the number of nodes increases, the alignments that can be executed in parallel also increases. Hence the time for alignment decreases.

Figure shows that a combination of Greenplum MPP Database with Hadoop outperformed Hadoop in alignment of sequences. It was noted that the time taken for alignment of sequences decreased by 30.55% in case of Greenplum with Hadoop as compared to Hadoop with PostgreSQL.
During our experiment we tried aligning sequences each of different sizes for a set of 30 sequences. The analysis shows that time increases as the pairwise alignment depends on the size of the sequences. As per the investigations also it was found that as the number of nodes increases, the alignments that can be executed in parallel also increases. Hence the time for alignment decreases.

Figure shows that a combination of Greenplum MPP Database with Hadoop outperformed Hadoop in alignment of sequences. It was noted that the time taken for alignment of sequences decreased by 30.55% in case of Greenplum with Hadoop as compared to Hadoop with PostgreSQL.
Bioinformatics use cases such as gene sequencing when implemented with conventional hardware and traditional sequencing algorithms prove to be expensive due to the hardware requirements and they take a backseat when it comes to performance. With the above experiments and the results, we are now in a position to realise the value of Cloud computing, Hadoop and Greenplum MPP Database in the area of bioinformatics.

There is an observed average decrease of 48% (approx) in the sequence computations with respect to time while using Greenplum with Hadoop. Our approach achieves near-optimal results and is distinguished by its robustness, responsiveness to highly dynamic query workloads, and negligible overhead. We implemented this mechanism using Greenplum MPP Database, a full-featured parallel and distributed database system. Apart from this we have the advantage of scalability and efficiency brought about by Hadoop. Cloud provides green infrastructure and a source of sequence for scientific analysis from all over the world.

The ideas presented in this paper are targeted to catalyse the research and development studies in the area of bioinformatics. The existing Drugs and treatment methods, aimed at the study of sequence alignment of DNA, RNA, or protein will benefit tremendously by the technologies discussed in this paper.
Senthil Vijayakumar is an IT Analyst with Tata Consultancy Services and has 5+ years of work experience in Information Storage Systems (NAS and SAN Products). Couple of years he has been associated with Cloud Computing and Big Data Analytics Initiative Project and his area of interest includes Green IT, High Performance Computing, and Cloud Computing Technologies. He recently involved in designing and developing solutions in the area of Computational Biology, Neurosciences using Cloud Computing Technologies and Big Data Analytics.

He has authored, published and presented his work at several conferences, including IEEE Computer Society, Storage Developer Conference organized by SNIA and International Society for Research in Science and Technology, USA. Paper titled “Optimizing Sequence Alignment in Cloud using Hadoop and MPP Database” Author’s of the paper - Senthilkumar Vijayakumar, Anjani Bhargavi, Uma Praseeda and Syed Azar Ahamed accepted and presented in the conference IEEE CLOUD 2012, 5th International Conference on Cloud Computing June 24-29 2012, Hyatt Regency Waikiki Resort and Spa, Honolulu, Hawaii, USA. He has filed 2 patents (pending) in the area of Green IT and Cloud Computing Technologies. He holds a Bachelors Degree in Electronics and Communication Engineering and a Masters in Communication Systems from PSG College of Technology, Coimbatore, India.

Anjanibhargavi Ragothaman is holding a designation of Systems Engineer at TATA Consultancy Services, Bangalore. She has been working in the Storage Domain as Storage System Performance Analyst for 3.8 years and has been working with a Leading Storage Vendor. Her interests include Neurosciences, Biomedical Imaging, Computational Biology, Virtualization, Storage System Analysis and Cloud Computing. She has been the co-author for the paper published upon Cloud Computing in Bioinformatics in IEEE Computer Society. She has completed her Bachelor of Technology at Amrita School of Engineering, Coimbatore. Currently she is pursuing her Masters at Rutgers University, New Jersey.
Author’s Profile

**Syed Azar Ahamed** is holding a designation of Technical Analyst at TATA Consultancy Services, Bangalore. He has been working in the Storage Domain as storage system performance analyst for 4 years and has been working with a Leading Storage Vendor. His interests include Virtualization, Big Data Analytics, Storage Systems Analysis and Cloud Computing. He has been the co-author for the paper published upon Cloud Computing in Bioinformatics in IEEE Computer Society. He has completed her Bachelor of Technology at Amrita School of Engineering, Bangalore, India.

**Uma Praseeda** has been with Tata Consultancy Services for 5+ years and in TCS she is working as a IT Analyst with a Leading Storage Vendor on their NAS and SAN storage products. She has been associated with Stress QA Test Execution and NAS Solutions Team of NAS and SAN storage products. Her interest includes Cloud Computing, Storage Systems and Virtualization. She has been the co-author for the paper published upon Cloud Computing in Bioinformatics in IEEE Computer Society. She holds a Bachelors Degree in Engineering in Information Technology from LBS Institute of Technology, Trivandrum Kerala India.
The authors thank Mr Rajinder Kalsi, Mr Naveen Bolalingappa, Mr Nagaraj Ijari, Mr Nirmal Kumar and Mr Anand Behere, TATA Consultancy Services Ltd for their continuous encouragement and support in setting up the Cloud Computing lab at TATA Consultancy Services Ltd. This research is a consequence of TCS-Innovation Labs research collaboration on Cloud Computing. **We would also like to extend our sincere thanks to the reviewers.**

Senthil, Anjani, Azar and Uma, Authors, TCS Bangalore, INDIA.
Thank You

senthilkumar.vijayakumar@tcs.com
anjani2.b@tcs.com
syed.azar@tcs.com
umapraseeda.pk@tcs.com