## Evaluation – Experimental Setup

The main goal of building an optimized BTree is to minimize search time - both for range queries and specific value queries. Additionally we want an optimal Tree for all input types – String, Int, List, Time Series.

Tuning parameters: The optimization parameters for the BTree in this set up are

1. Type of BTree - for a set of inputs -

(i) Text input data :

if inputs data is "String" type - will the search for the data be more efficient

if the tree was represented as Int tree ( build from some hashcode representation of

String input) or should it a String tree ?

(ii) Time series input data

- Search for one col for input dates

- Search multiple columns for input dates

(iii) Input data is a List type i.e comma separated list of values ?

1. Minimizing Random Access (fit tree in L1/2- cache ) - which will be impacted by the fanout

Benchmarking - We will benchmark the performance of Optimal Btree

* Comparing performance of optimized tree vs unoptimized tree for

(a)int input

(b)String input

(c)time series input

* Comparing performance of range queries for varying data size - optimized tree vs unoptimized tree . As data size increase we expect the optimized tree to be a smooth/ slightly curved line, while the performance of an unoptimized tree should increase linearly (?) with data size. The data for this set up will be integer inputs of varying sizes.
* Comparing the time to come up with an optimized option with varying data sizes - This will help to evaluate the performance of GeneticAlgorithm
* Compare the performance of an optimized tree implemented in Scala with the same optimized tree implemented in C - Although the results of this experiment does not directly relate to main goal - i.e the Tree Type ( String, Int ) , that can most efficient search data , but it will help to evaluate the overhead of high level languages , and if that cost can be amortized by the time saved in implemented in a high level language.