Vol.2015-MPS-105 No.16
2015/9/30

# Parameter Optimization Framework on Apache Hadoop using Steady-State Genetic Algorithm

PHYO THANDAR THANT[†1] AKIYOSHI SUGIKI[†2]
MASAHARU MUNETOMO[†2]

**Abstract:** This paper proposes a Hadoop parameter optimization framework based on a Genetic Algorithm (GA) search strategy. In the proposed framework, steady-state GA is applied to obtain optimal configuration of Hadoop parameters on various workloads. We can also identify the parameters that are necessary to tune for a particular type of application based on the optimized parameter list. The proposed framework has been implemented and tested on an updated Hadoop 2.7.1 based 24 node cluster.
**Keywords**: Hadoop, Configuration Optimization, Genetic Algorithm, Search-based Strategy

## 1. Introduction

The digital universe is expanding with data at 40% per year and is projected to continue this trend into the next decade, as a result of widespread adoption of computing devices in our everyday lives. The major sources of these big data include public web, social media, business applications, mobile and sensor applications, machine log data and scientific and simulation data. The use of big data is also becoming crucial for leading companies to outperform their peers [1].

The most popular technique for big data processing is using the MapReduce parallel data processing framework across thousands of machines. Apache Hadoop is an open-source MapReduce implementation and it exposes hundreds of configuration parameters [2]. Parameter Tuning is a daunting and time consuming task because it is very difficult to identify the parameters that can impact the performance of a particular application [3]. Although several approaches are available, there are some limitations and further research is still necessary because of the complex nature of Hadoop configurations. Moreover, the Hadoop framework is always being updated, posting another challenge for ideal parameter optimization.

This paper proposes a parameter optimization framework in which steady-state Genetic Algorithm (GA) iteratively searches for the fittest parameter combination and outputs an optimized configuration parameter list for a particular type of application. We chose a GA approach because it conducts an aggressive search that results in near-optimal configuration. Further, this search based optimization approach is also suitable for the dynamic nature of the Hadoop framework. In this system, processing execution time is considered as the fitness evaluation values and the fittest one is the parameter configuration values that can give the shortest execution time. Our experimental results show that, we can also identify the effective parameters for a particular type of application which can improve later configuration decisions for big data applications.

## 2. Proposed Framework

This section presents the proposed framework in detail. The objective function of the proposed framework is first explained, followed by the flow of the optimization system, system encoding scheme and the associated optimization algorithm.

### 2.1 Objective Function

The target optimization problem is minimization of the total execution time to finish the MapReduce tasks stated below:

$$\min E_T(p_1,p_2,p_3,\dots,p_m)$$

where,

$E_T$ = execution time of each MapReduce task
$p_j$ = configuration parameter list to control each MapReduce task ($1 \le j \le m$)

### 2.2 Flow of the Optimization System

This section explains the steps in the steady-state GA optimization system. First, an initial population step generates random initial chromosome parameter lists to initiate the process. Next, the fitness values of the initial chromosomes are evaluated. Then, parents are selected from the initial search pool and, based on the parents, successive evolution continues until the predefined number of generations is reached. Finally, the system gives the fittest chromosome within the evolution process. The optimal chromosome parameter values will be recommended to use for future processing in the system. The algorithm for this process is presented in section 2.4.

### 2.3 System Encoding Scheme

The system uses a binary encoding scheme, the most common representation of chromosomes in GA. In this system, the genes in the chromosomes are in the form {0,1} with each gene,- a single bit or two consecutive bits, representings its respective Hadoop configuration parameter values. Further, strings of 24 parameter values (genes) are coded as 34-bit binary chromosome strings that are then used in the parameter optimization process of the proposed system.

### 2.4 Parameter Optimization Algorithm

**Algorithm: Steady-State GA based parameter optimization**

```
  Input   : population size N, number of generations G
  Output  : optimized chromosome solution $C_{opt}$
Step 1: Generate random initial population, IP
    1.1 Generate binary chromosome randomly $C_i$
        (I=1,2,…N)
```

†1 Graduate School of Information Science and Technology,Hokkaido University
†2 Information Initiative Center, Hokkaido University

```
        1.2 Calculate the fitness of each random chromosome,
            $f_i = E_T(C_i)$
Step 2: Evolve the Population (P, G)
        2.1 Select 2 parents at Random ($P_1$, $P_2$)
        2.2 Perform two-point dynamic crossover operation on
            parents ($P_1$, $P_2$) with probability, $P_c = 1$ to produce
            ($C_1$, $C_2$)
        2.3 Perform mutation operation on random points
            applied to ($C_1$, $C_2$) (probability, $P_m=0.01$)
        2.4 Calculate fitness of new offspring ($C_1$, $C_2$) and
            replace the best two of ($P_1$,$P_2$,$C_1$,$C_2$) with the worst
            two chromosomes in the population, P
Step 3: Repeat step 2 until the predefined number of generations
        is reached
Step 4: Output the optimal chromosome solution, $C_{opt}$
```

## 3. Performance Evaluation

### 3.1 Experimental Setup

This section presents the test environment and test workloads used to evaluate the optimization framework.

**Test Environment**

Figure 2 shows the Hadoop cluster testing environment, implemented using 24 virtual machines on our Academic cloud system. It comprises one NameNode that serves as the front end for the cluster system and 23 Datanodes. Each node comprises 4GB RAM, an Intel Xeon E7 CPU, and 100GB of storage. Hadoop 2.7.1 and jdk 1.8.0 are used in the cluster development.

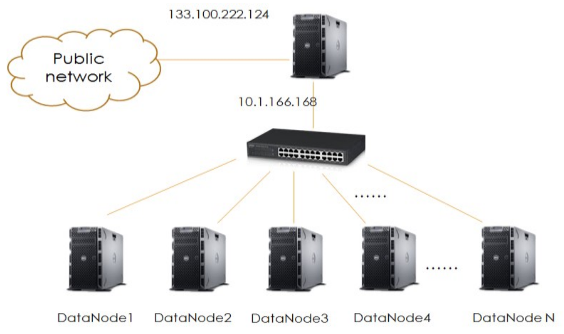

Figure .2. Hadoop Cluster Test Environment

**Test Workloads**

Several benchmarks are available to evaluate the processing performance of a Hadoop cluster environment. Among them, we used workloads from HiBench 4.0 for this Hadoop parameter optimization framework. Table I shows the selected workloads used to evaluate the effectiveness of the proposed framework.

Table I. Workload List

| Type | Workload |
|---|---|
| MicroBenchmark | - Sort<br>- TeraSort |
| Machine Learning | - PageRank |

**3.2 Results**

Figures 3-5 show the gradual optimization process for terasort, sort and pagerank workloads in different generations using the proposed system. The system eventually outputs the optimized solution. On the basis of that solution, effective parameters that can pave the way to optimized configuration of future big data processing applications are identified.

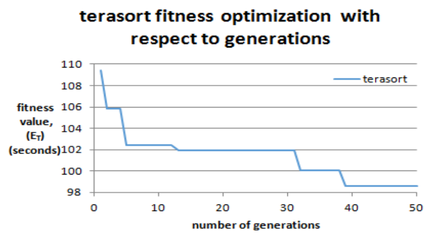

Figure .3. Fitness optimization on terasort

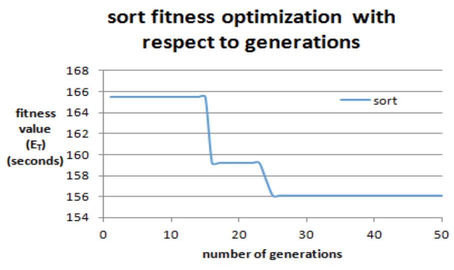

Figure .4. Fitness optimization on sort

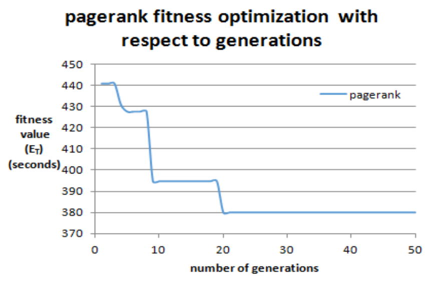

Figure .5. Fitness optimization on pagerank

## 4. Conclusion and Future Work

Big data processing places is one of the most challenging tasks in the current IT arena. Efficient big data processing is possible with the use of Hadoop's open-source MapReduce implementation. However, Hadoop exposes hundreds of configuration parameters and so parameter tuning is a daunting and time consuming task because it is very difficult to identify the parameters that can impact the performance of a particular application. This framework performs Hadoop parameter tuning using a steady-state GA based search strategy. Currently, the system utilizes parameters from HDFS and MAPRED configuration files. Our experimental results show that the system is effective. In future work, performance tuning of YARN parameters and comparison of the system's performance to that of similar research works will be conducted.

## References

[1] H. Herodotou, H.Lim, G. Luo, N. Borisov, L. Dog, "Starfish: A Self tuning system for Big Data Analytics," 5th Biennial Conference on Innovative Data Systems Research (CIDR'11), Asilomar, California, USA, 2011.

[2] G. Liao, K. Datta, T. L. Wilke, "Gunther: Search Based Auto Tuning of MapReduce," Intel Labs, Hillsboro, Oregon, USA, LNCS8097, pp 406-419, Springer-Verlag Berlin Heidelberg, 2013.

[3] M. Li, L. Zeng, S. Meng, "MRONLINE: MapReduce Online Performance Tuning" HPDC 14, June 23-27 Vancouver, Canada, 2014.