International Journal of Computer Applications (0975 – 8887)
Volume 86 – No 10, January 2014

Asymmetric Key-Value Split Pattern Assumption over MapReduce Behavioral Model

Ravi Prakash G.
Department of CS/IT
Alliance University
Bangalore, India

Kiran M.
Member, Technical Staff
Data Analytics Research Lab
Bangalore, India

Saikat Mukherjee
Senior Software Engineer
HP India
Bangalore, India

ABSTRACT

Actual Quantifiability is a concept in MapReduce that is based on two assumptions: (1) every mapper is cautious, i.e., does not exclude any reducer's key-value split pattern choice from consideration, and (2) every mapper respects the reducer's key-value split pattern preferences, i.e., deems one reducer's key-value split pattern choice to be infinitely more likely than another whenever it premises the reducer to prefer the one to the other. In this paper we provide a new approach for actual quantifiability, by assuming that mappers have asymmetric key-value split pattern about the reducer's key-value utilities. We show that, if the uncertainty of each mapper about the reducer's key-value utilities vanishes gradually in some regular manner, then the key-value split pattern choices it can quantifiably make under common conjecture in quantifiability are all actually quantifiable in the original MapReduce with no uncertainty about the reducer's utilities.

Keywords

Mapper, reducer, key-value, asymmetric, split pattern, utilities, MapReduce, behavioral, actual quantifiability

1. INTRODUCTION

MapReduce deals with the ways the mappers may reason about its reducers before making a decision. More precisely, in MapReduce mappers base its key-value split pattern choices on the conjectures about the reducers' behavior, which in turn depend on its conjectures about the reducers' conjectures about other reducers' behavior, and so on [5]. A major goal of MapReduce in this work is to study such conjecture hierarchies, to impose reasonable conditions on these, and to investigate its split pattern behavioral implications.

A central idea in MapReduce is common conjecture in quantifiability, stating that a mapper premises that its reducers choose quantifiably, and so on. In our view, one of its most natural refinements is the concept of actual quantifiability. Actual Quantifiability is based on the following two conditions: The first states that mappers are cautious [1] [8], meaning that they do not exclude any reducers' key-value split pattern choice from consideration. The second condition states that whenever premise that a key-value split pattern choice a is better than another key-value split pattern choice b for a reducer, then the probability assign to b must be at most α times the probability assign to a. Under α-actual quantifiability there is common conjecture in the event that every mapper is cautious and satisfies theα-actual trembling condition. A key-value split pattern choice is called actually quantifiable if it can be chosen under α-actual quantifiability for every $\alpha > 0$ [4] [7].

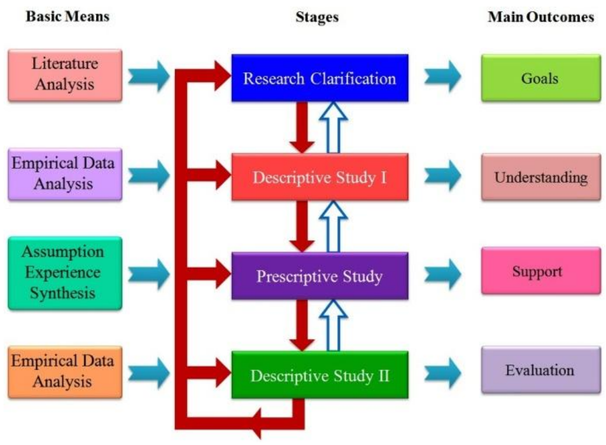

Fig 1: Research Plan: Basic Means, Stages, Main Outcomes

2. RESEARCH CLARIFICATION

The usual interpretation of actual quantifiability assumes that reducer makes mistakes, but that deem more costly mistakes much less likely than less costly mistakes. In this paper we offer a rather different approach for actual quantifiability. Instead of assuming premise reducer to make mistakes, we rather suppose that have uncertainty about its utility function, while believing that it chooses quantifiably. We thus consider a MapReduce with asymmetric key-value split pattern. Our main result states that, if we let uncertainty about the reducer's utility go to zero in some regular manner, then every key-value split pattern choice that can quantifiably be made under common conjecture in quantifiability in the MapReduce with asymmetric key-value split pattern, will be actually quantifiable in the original MapReduce, in which there is no uncertainty about the reducer's utilities.

In the MapReduce with asymmetric key-value split pattern, we impose some regularity conditions on the mappers' conjectures about the reducer's utility functions which can be summarized as follows: First, for every outcome in the MapReduce, the conjecture that mapper i has about mapper j's utility from this outcome, is always normally distributed with its mean at the "original" utility in the original MapReduce. As a consequence, mapper i deems any utility function possible for mapper j, and hence every split pattern choice for mapper j can be optimal for some utility function deemed possible by i. Together with the condition that i premises in j's quantifiability, this actually makes sure that mapper i deems every key-value split pattern choice possible for mapper j, thus mimicking the cautiousness condition described above. Secondly, i's conjecture about j's utility function should be independent from its conjecture about j's conjecture hierarchy. This makes intuitive sense since j's conjecture hierarchy is analytic property of this mapper, whereas its utility function is not analytic property [6]. Therefore there is no obvious reason to expect any correlation between these two characteristics. Thirdly, i's conjecture about j's utilities from different outcomes in the MapReduce should be independent from each other.

The paper is organized as follows: In Section 3 we introduce our MapReduce programming model [3] [2], for MapReduce with asymmetric key-value split pattern, we formalize the idea of common conjecture in quantifiability for these MapReduce, and show that common conjecture in quantifiability is always possible (Descriptive Study I). In Section 4 we introduce our MapReduce programming model for MapReduce with symmetric key-value split pattern, and present the concept of actual quantifiability for these MapReduce (Prescriptive Study). In Section 5 we state our main result, establishing the connection between common conjecture in quantifiability in the MapReduce with asymmetric key-value split pattern in the presence of small uncertainty about the reducer's utility function, and actual quantifiability in the original MapReduce (Descriptive Study II). In Section 6 we provide some concluding remarks.

3. DESCRIPTIVE STUDY I

3.1 MapReduce Programming Model

Throughout this paper we restrict attention to MapReduce operations with two sets of mapper. Let $\delta = (C_i, w_i)_{i\epsilon I}$ be a finite, static MapReduce where $I = \{1, 2\}$is the set of mappers, C_i is the finite set of key-value split pattern choices of mapper i, w_i is mapper i's utility function. The function w_i assigns to every pair of key-value split pattern choice $(c_1, c_2) \epsilon C_1\times C_2$ a utility $w_i(c_1, c_2) \epsilon F$.

In a MapReduce with asymmetric key-value split pattern, mappers do not only uncertainty about the reducer's key-value split pattern choices; they also have uncertainty about the reducer's utility function. Hence a conjecture hierarchy should not only specify what the mapper premises about the reducer's key-value split pattern choice but also what it premises about the reducer's utility function. Not only this, it should also specify what the mapper premises about the reducer's conjecture about its own key-value split pattern choice and utility function, and so on. A possible way of modeling such conjecture hierarchies is by means of the following necessary and sufficient condition.

Necessary and sufficient condition 3.1 (*MapReduce programming model*). A finite MapReduce programming model for δ with asymmetric key-value split pattern is a tuple $M = (S_i, v_i, K_i)_{i\epsilon I}$ where (1) S_i is the set of key-value types for mapper i. (2) $v_i : S_i \rightarrow \theta(C_j\times S_j)$ is the conjecture assignment taking only finitely many different probability distributions on $\theta(C_j\times S_j)$ and (3) k_i is the utility assignment that assigns to every $s_i \epsilon S_i$ a utility function $k_i(s_i) : C_1\times C_2 \rightarrow F$.

By $\theta(P)$ we denote the set of probability distributions on P. Therefore, in a MapReduce programming model, each key-value type s_i has a conjecture about mapper j's split pattern choice-key-value type combinations. And hence, in particular, it has a conjecture about j's split pattern choice. But, as mapper j's key-value type also specifies its utility function and its conjecture about i's split pattern choice, mapper i also has some conjecture about mapper j's utility function, and about mapper j's conjecture about its own split pattern choice, and so on. In this way one can derive a complete conjecture hierarchy for every given key-value type.

Note that each key-value type s_i can be identified with a pair $(k_i(s_i), v_i(s_i))$, where $k_i(s_i)$ is its utility function and $v_i(s_i)$ is its conjecture hierarchy. Since we required the conjecture assignment to take only finitely many different probability distributions, the MapReduce programming model contains only finitely many different conjecture hierarchies.

3.2 Limitations on the MapReduce Programming Model

Our goal will be to model the situation where the mappers have uncertainty about the reducer's utility function, but where this uncertainty "vanishes in the limit". In order to formalize this we need to impose additional limitations on the MapReduce programming model.

Recall that every key-value type s_i can be identified with a pair $(k_i(s_i), v_i(s_i))$,where $k_i(s_i)$ is s_i's utility function and $v_i(s_i)$ is its conjecture hierarchy. Denote by K_i the set of all possible utility functions, and by V_i the set of all conjecture hierarchies in the MapReduce programming model $M = (S_i, k_i, v_i)_{i\epsilon I}$. The first condition we impose is that $S_i = K_i\times V_i$, that is, for every possible utility function we can think of, and every conjecture hierarchy in the model, there exists a key-value type in the model with exactly this combination of utility function and conjecture hierarchy. Therefore in a sense we assume that the key-value type is rich enough.

Secondly, we assume that s_i's conjecture about j's utility from (c_1, c_2) is statistically independent from its conjecture j's utility from $(ć_1,ć_2)$ whenever $(c_1, c_2) \neq (ć_1,ć_2)$ and that this conjecture is also statistically independent from its conjecture about j's conjecture hierarchy.

Finally we assume that s_i's conjectures about j's utilities from the various outcomes in the MapReduce are all induced by a unique normal distribution. More formally, s_i's conjecture about j's utility from (c_1, c_2) is given by a normal distribution with its mean at $w_j(c_1, c_2)$ – the "true" utility of mapper j in the original MapReduce. Therefore, all these conjectures are distributed identically around the mean. By collecting all these conditions we arrive at the following necessary and sufficient condition.

Necessary and sufficient condition 3.2 (*σ–regular MapReduce programming model*). Let D be the normal distribution on F with mean 0 and variance $\sigma^2 > 0$. Then a MapReduce programming model $M = (S_i, v_i, k_i)_{i\epsilon I}$ is σ–regular if for both mappers i, (1) $S_i = K_i\times V_i$, (2) for every key-value type $s_i \epsilon S_i$, its conjecture about j's utility from (c_1, c_2) is statistically independent from its conjecture about j's utility from $(ć_1,ć_2)$ whenever $(c_1, c_2) \neq (ć_1,ć_2)$ and its conjecture about j's utilities is statistically independent from its conjecture about j's conjecture hierarchy, and (3) for every key-value type $s_i \epsilon S_i$, and every split pattern choice-pair (c_1, c_2), the conjecture of s_i about j's utility from (c_1, c_2) is given by D, upto a shift of the mean to $w_j(c_1, c_2)$.

3.3 σ-Quantifiability

In this subsection we will define common conjecture in quantifiability inside a MapReduce programming model with asymmetric key-value split pattern. In addition, if we require the MapReduce-programming model to be σ-regular for a given normal distribution with mean 0 and variance σ^2, then we obtain the concept of σ-quantifiability. We first need some more notations. For given key-value type s_i and key-value split pattern choice c_i, let $k_i(s_i)(c_i)$ be the expected utility for key-value type s_i from choosing c_i, given its conjecture $v_i(s_i)$

about the reducer's key-value split pattern choice, and given its utility function $k_i(s_i)$.

Necessary and sufficient condition 3.3 (*Quantifiable key-value split choice*). A key-value split pattern choice c_i is quantifiable for s_i if $k_i(s_i)(c_i) \geq k_i(s_i)(ć_i)$ for all $ć_i \in C_i$.

We will now define common conjecture in quantifiability. In words it says that a mapper premises that its reducer makes quantifiable key-value split pattern choices, and premises that its reducer premises that it makes quantifiable key-value split pattern choices, and so on.

Formally, for every $\hat{S}_i \subseteq S_i$, let

$$(C_i \times \hat{S}_i)^{quant} = \{ (c_i, s_i) \in C_i \times \hat{S}_i : c_i \text{ is quantifiable for } s_i \}.$$

Necessary and sufficient condition 3.4 (*Common conjecture in quantifiability*). For mappers i we define subsets of key-value types $S_i^1, S_i^2, \ldots$ in a recursive way as follows:

$$S_i^1 := \{ s_i \in S_i : v_i(s_i)\,[(C_j \times S_j)^{quant}] = 1 \},$$

$$S_i^2 := \{ s_i \in S_i : v_i(s_i)\,[(C_j \times S_j^1)^{quant}] = 1 \},$$

$$\vdots$$

$$S_i^l := \{ s_i \in S_i : v_i(s_i)\,[(C_j \times S_j^{l-1})^{quant}] = 1 \},$$

$$\vdots$$

Key-value type s_i expresses common conjecture in quantifiability if $s_i \in \cap_{l \in \mathbf{N}} S_i^l$. A key-value type σ–quantifiable if it expresses common conjecture in quantifiability with a σ–regular MapReduce programming model.

Necessary and sufficient condition 3.5 (*σ–quantifiable key-value type*). Let $M = (S_i, v_i, k_i)_{i \in I}$ be a σ–regular MapReduce programming model. Every key-value type $s_i \in S_i$ that expresses common conjecture in quantifiability is called σ–quantifiable.

Now we show that σ–quantifiable key-value types always exist.

Proposition 3.1 (σ–quantifiable key-value types always exist). Consider a finite static MapReduce $\delta = (C_i, w_i)_{i \in I}$, and some $\sigma > 0$. Then there is a σ–regular MapReduce programming model $M = (S_i, v_i, k_i)_{i \in I}$ for δ where all key-value types are σ–quantifiable.

3.4 Limit Quantifiability

In this subsection we focus on those key-value split pattern choices, which can quantifiably be made under common conjecture in quantifiability when the uncertainty about the reducer's utility vanishes. This will lead to the concept of limit quantifiability. We first need an additional necessary and sufficient condition.

Necessary and sufficient condition 3.6 (*Constant key-value type and utility assignments*). A key-value sequence of MapReduce programming models $((S_i^n, v_i^n, k_i^n)_{i \in I})_{n \in \mathbf{N}}$ has constant key-value type and utility assignments if $S_i^n = S_i^m$ and $k_i^n = k_i^m$ for all n and m, and for mappers i.

We are now ready to say the concept of limit quantifiable key-value split pattern choice.

Necessary and sufficient condition 3.7 (*Limit quantifiable split pattern choice*). Consider a finite static MapReduce $\delta = (C_i, w_i)_{i \in I}$ with mappers. A key-value split pattern choice c_i is limit quantifiable if there is a key-value sequence $(\sigma_n)_{n \in \mathbf{N}} \to 0$, and a key-value sequence $(M^n)_{n \in \mathbf{N}}$ of σ_n–regular MapReduce programming models with constant key-value type and utility assignments, such that in every M^n there is a σ_n-quantifiable key-value type s_i^n with utility function w_i, for which key-value split pattern choice c_i is optimal.

4. PRESCRIPTIVE STUDY

4.1 MapReduce Programming Model

Let $\delta = (C_i, w_i)_{i \in I}$ be a finite, static MapReduce with mappers. In a MapReduce with symmetric key-value split pattern mappers do not have uncertainty about the reducer's utility function. Therefore a conjecture hierarchy only needs to specify what a mapper premises about the reducer's key-value split pattern choice, what it premises about the reducer's conjecture about its own key-value split pattern choice, and so on. Therefore the MapReduce programming model will be simpler compared to the case of asymmetric key-value split pattern.

Necessary and sufficient condition 4.1 (*MapReduce programming model*). A MapReduce programming model forδwithsymmetric key-value split pattern is a tuple $M = (\Omega_i, \rho_i)_{i \in I}$ where (1) Ω_i is the finite set of key-value types for mapper i, and (2) ρ_i: $\Omega_i \to \theta(C_j \times \Omega_j)$ is the conjecture assignment.

Therefore, in a MapReduce programming model, each key-value type τ_i has a conjecture about mapper j's key-value split pattern choice-key-value type combinations. And hence, in particular, it has a conjecture about j's key-value split pattern choice. But, as mapper j's key-value type also specifies its conjecture about mapper i's key-value split pattern choice, mapper i also has some conjecture about mapper j's conjecture about its own key-value split pattern choice, and so on. In this way one can derive a complete conjecture hierarchy for every given key-value type.

For given key-value type τ_i and key-value split pattern choice c_i we define $w_i(c_i, \tau_i)$ as the expected utility for key-value type τ_i from choosing c_i given its conjecture $\rho_i(\tau_i)$ about its reducer's key-value split pattern choice (and *given* its "fixed" utility function w_i). Key-value type τ_i is said to *prefer* key-value split pattern choice c_i to key-value split pattern choice $ć_i$ when $w_i(c_i, \tau_i) > w_i(ć_i \tau_i)$. We say that a key-value type τ_i *considers possible* some reducer's key-value type τ_i*if* $\rho_i(\tau_i)(c_j, \tau_j) > 0$ for some $c_j \in C_j$. Now we introduce the key condition in actual quantifiability, which is the α–actual trembling condition. Intuitively it says that (1) a mapper should deem possible all reducer's key-value split pattern choices, and (2) if a mapper premises key-value split pattern choice a is better than key-value split pattern choice b for the other mapper, then it should deem key-value split pattern choice a much more likely than key-value split pattern choice b.

Necessary and sufficient condition 4.2 (*α-actual trembling condition*). Let $\alpha > 0$. A key-value type τ_i satisfies the α-actual trembling condition if (1) for each τ_j that τ_i deems possible, $\rho_i(\tau_i)(c_j, \tau_j) > 0$ for all $c_j \in C_j$, and (2) for every τ_j that τ_i deems possible, whenever τ_j *prefers* c_j to $ć_j$, then $\rho_i(\tau_i)(ć_j, \tau_j) \leq \alpha \cdot \rho_i(\tau_i)(c_j, \tau_j)$.

Therefore, the first condition says that whenever τ_i deems some key-value type τ_j possible, τ_i also assumes every key-value split pattern choice is possible for τ_j. Actual Quantifiability is based on the event that the key-value types should not only satisfy the α-actual trembling condition themselves, but also express common conjecture in the event that key-value types satisfy the α-actual trembling condition.

Necessary and sufficient condition 4.3 (*α-actually quantifiable key-value type*). A key-value type τ_i is α-actually quantifiable if: τ_i satisfies the α-actual trembling condition, τ_i only deems possible reducer's key-value types τ_j which satisfy the α-actual trembling condition, τ_i only deems possible reducer's key-value types τ_j which only deem possible mapper i's key-value types τ'_i which satisfy the α-actual trembling condition, and so on.

Actually quantifiable key-value split pattern choices are those key-value split pattern choices, which can quantifiably be made by α-actually quantifiable key-value types for all α.

Necessary and sufficient condition 4.4 (*Actually quantifiable split pattern choice*). A key-value split pattern choice c_i is α-actually quantifiable if there is a MapReduce programming model and a α-actually quantifiable key-value type τ_i within it for which c_i is optimal. A key-value split pattern choice c_i is actually quantifiable if it is α-actually quantifiable for all $\alpha > 0$.

5. DESCRIPTIVE STUDY II

5.1 Statement of the Main Result

For a static MapReduce we analyzed two contexts, one with asymmetric key-value split pattern and another with symmetric key-value split pattern. In the context with asymmetric key-value split pattern, where mappers have uncertainty about the reducer's utility, we introduced the concept of a limit quantifiable key-value split pattern choice. In the context with symmetric key-value split pattern, where mappers have no uncertainty about the reducer's utility, we discussed the concept of an actually quantifiable key-value split pattern choice. In our main result we connect these two concepts.

Proposition 5.1 (*Limit Quantifiability implies actual quantifiability*). Consider a finite static MapReduce with mappers. Every limit quantifiable key-value split pattern choice for the context with asymmetric key-value split pattern is a actually quantifiable key-value split pattern choice for the context with symmetric key-value split pattern.

5.2 Illustration of the Main Result

By means of an example we provide some intuition for our main result. More precisely we show how a quantifiable key-value type in the context of asymmetric key-value split pattern can be transformed into a actually quantifiable key-value type in the context of symmetric key-value split pattern. Also we show that when σ goes to zero then α goes to zero as well.Let us start with the context of asymmetric key-value split pattern. Let D be the normal distribution with mean 0 and variance σ2. From the Proposition 3.1 we know that there exists a regular MapReduce programming model M = (Si, vi, ki)i∊I where every key-value type is quantifiable and all the key-value types have the same conjecture hierarchy. Therefore, key-value types only differ by their utility function. For each of the key-value types s1 of mapper 1 we denote by ρ1 the conjecture about mapper 2's key-value split pattern choice, and for each key-value type s2 let ρ2 be the conjecture about mapper 1's key-value split pattern choice. As we assume that all the key-value types have the same conjecture hierarchy, ρ1 and ρ2 are unique.

For both mappers i let Oi be the probability distribution on mapper i's utility functions generated by D. Since the MapReduce-programming model is σ-regular every key-value type sj has the conjecture Oi about i's utility function. Let Ki(ci, ρi) be the set of utility functions for mapper i such that the key-value split pattern choice ci is optimal under the conjecture ρi about the reducer's key-value split pattern choice. Since every key-value type si expresses common conjecture in quantifiability, the probability it assigns to a reducer's key-value split pattern choice cj is exactly the probability it assigns to the event that j's utility function is in Kj(cj, ρj) which is Oj(Kj(cj, ρj).

Since D has full support, it follows that all these probabilities are positive. Now we turn to the context of symmetric key-value split pattern. We construct a MapReduce programming model with a single key-value type τ1 for mapper 1 and a single key-value type τ2 for mapper 2. Let the conjecture of τ1 about the mapper 2's key-value split pattern choice be given by the ρ1 constructed above, and similarly for the conjecture of τ2. Therefore, the conjecture about the reducer's key-value split pattern choice has not changed by moving from the context with asymmetric key-value split pattern to the context with symmetric key-value split pattern.

6. CONCLUDING REMARKS

We premise that actual quantifiability is a very natural concept in MapReduce, but it has not yet received the attention it deserves. In this paper we have established a new approach for actual quantifiability from the viewpoint of MapReduce with asymmetric key-value split pattern. In MapReduce with asymmetric key-value split pattern we define a key-value split pattern choice as limit quantifiable if it can quantifiably be made under common conjecture of quantifiability when the uncertainty vanishes gradually in some regular way. We show the existence of such key-value split pattern choices. We then establish that each limit quantifiable key-value split pattern choice in the MapReduce with asymmetric key-value split pattern is actually quantifiable for the context with symmetric key-value split pattern.

7. ACKNOWLEDGMENTS

This research has benefited from the financial support of Vision Group on Science and Technology, Department of Information Technology, Biotechnology and Science & Technology, Bangalore-560 001. This research has also been enriched by many intangible contributions from my fellow students and friends. Finally, thanks to the referees for their careful reading and constructive criticism.

8. REFERENCES

[1] Kiran M., Saikat Mukherjee and Ravi Prakash G., Characterization of Randomized Shuffle and Sort Quantifiability in MapReduce Model, International Journal of Computer Applications, 51-58, Volume 79, No. 5, October 2013.

[2] Amresh Kumar, Kiran M., Saikat Mukherjee and Ravi Prakash G., Verification and Validation of MapReduce Program model for Parallel K-Means algorithm on Hadoop Cluster, International Journal of Computer Applications, 48-55, Volume 72, No. 8, June 2013.

[3] Kiran M., Amresh Kumar, Saikat Mukherjee and Ravi Prakash G., Verification and Validation of MapReduce Program Model for Parallel Support Vector Machine Algorithm on Hadoop Cluster, International Journal of

Computer Science Issues, 317-325, Vol. 10, Issue 3, No. 1, May 2013.

[4] Aniruddha Basak, Irina Brinster and Ole J. Mengshoel. MapReduce for Bayesian Network Parameter Learning using the EM Algorithm, Proc. of Big Learning: Algorithms, Systems and Tools, 1-6, December 2012.

[5] Berli'nska, J., Drozdowski, M.: Scheduling divisible MapReduce computations. J. Parallel Distrib. Comput 71(3), 450-459 (2011).

[6] Emanuel Vianna, Giovanni Comarela, Tatiana Pontes, Jussara Almeida, Virgilio Almeida, Kevin Wilkinson, Harumi Kuno, Umeshwar Dayal. Analytical Performance Models for MapReduce Workloads, Int J Parallel Prog 41:495-525 (2013).

[7] Erik B. Reed and Ole J. Mengshoel. Scaling Bayesian Network Parameter Learning with Expectation Maximization using MapReduce, Proc. of Big Learning: Algorithms, Systems and Tools, 1-5, December 2012.

[8] Ravi Prakash G, Kiran M and Saikat Mukherjee, On Randomized Preference Limitation Protocol for Quantifiable Shuffle and Sort Behavioral Implications in MapReduce Programming Model, Parallel & Cloud Computing, Vol. 3, Issue 1, Pages 1-14, January 2014.