INTELLIGENT WATER DROP ALGORITHM FOR WORKFLOW SCHEDULING IN CLOUD COMPUTING

Mr.C.B.Sivaparthipan [#1]
Mr. J.Rathinaraja [#2]
Mr.S.Sathish kumar [#3]
[#1]Assistant professor, Department of CSE, SNS College of Technology, Coimbatore, TamilNadu.
[#2]Lecturer, Department of IT, National Institute of Technology, Karnataka.
[#3]Assistant professor, Department of CSE, SNS College of Technology, Coimbatore, TamilNadu.

Abstract- Cloud Computing offers hardware and software as a resource on internet. An IT infrastructure can be setup without any capital investment and reduces the space for infrastructure at premise. Resource scalability and pay per use basis are the remarkable advantages by using cloud computing. Cloud Computing is promoted by the business rather than academic which determines its focus on user applications. In Cloud environment, the processing of an application is done by dividing into small workflows and scheduling them to get the expected result. These workflows are scheduled based on some Quality of Services (QoS). Different users have different QoS Requirements. Scheduling is a very important part of the cloud computing system. This paper introduces an optimized algorithm for scheduling based on Intelligent Water Drop algorithm in cloud computing and its implementation with an objective of minimizing the cost. The algorithm efficiently completes scheduling in the cloud computing environment computing.

Keywords-- Intelligent Water Drop (IWD), workflow scheduling, cloud computing, Quality of Service (QoS), Heuristics.

I. INTRODUCTION

Cloud computing is Internet-connected mode of cluster computing. It is a new type of shared infrastructure, which puts the huge system pool together by the way of operators and the customer. It provides users with a variety of storage and computing resources in the cloud computing environment via Internet, users put a lot of information and processor resources of terminal equipment such as personal computer, mobile phone together for work. Cloud computing is a large-scale distributed computing model, which depends on the economic size of the operator that is abstract, virtualized and dynamic. The main content of cloud computing is managed computing power, storage, platforms and services which assigned to the external user on demand through the Internet. Cloud computing is an emerging computation paradigm with the goal of freeing up users from the management of hardware, software, and data resources and shifting these burdens to cloud service providers [1]. Clouds provide a large pool of resources, including high power computing platforms, data centres, storages, and software services. It also provides management to these resources such that users can access them ubiquitously and without incurring performance problems. Cloud computing provides a more abstract resources and services of these resources and services can be divided into three levels, namely the software as a service, platform as a service and facilities that service.

These services are made available as subscription-based services in a pay-as-you-go model to consumers [3, 2]. Application data can be hosted on different storage resources at the global cloud infrastructure. When one task needs to process data from different data canters, moving the data becomes a challenge [6]. In order to efficiently and cost effectively schedule the tasks and data of applications

among cloud services, end user QoS-based scheduling strategies are implemented, such as those for minimizing make span, minimizing total execution cost andBalancing the load of resources[7]. In this paper, we focus on minimizing the total execution cost of applications on these resources provided by Cloud service providers, such as Amazon Weachieves this by using a swarm based optimization technique called Intelligent Water Drop (IWD).

Intelligent Water Drops (IWD) algorithm is one of the nature-inspired swarm-based optimization algorithms.IWD algorithms imitate some of the processes that happen in nature between the water drops of a river and the soil of the river bed. The paths that a natural river follows have been created by a swarm of water drops. It is often observed that the constructed path seems to be an optimal one in terms of the distance from the destination and the constraints of the environment .The environment is considered as the problem. Each IWD is assumed to flow in its environment from a source to a desired destination. In an environment, there are numerous paths from a given source to a desired destination. If the location of the desired destination is known, the solution to the problem is obtained by finding the best (often the shortest) path from the source to the destination. The solution is obtained by finding the optimum destination in terms of cost or any other desired measure for the given problem.

IWD is emerging technique and it has used in solving various problems. Some of the problems solved using IWD are: Travelling salesman problem, n-queens puzzle, Multidimensional knapsack problem, Automatic multilevel thresholding. IWD also has wide range of applications.

The rest of the paper is organized as follows: Section 2 presents related work. In Section 3, we describe the workflow scheduling problem. In Section 4, we introduce the IWD algorithm. Section 5 presents simulation result. Section 6 concludes the paper and discusses some future work.

II. RELATED WORK

Hamed shah_ hosseini [1] introduced the new problem solving algorithm Intelligent Water Drops or IWD algorithm. They have applied this methodology to travelling salesman problem. It's observed that a river often chooses the optimum path regarding the conditions of its surroundings to get to its ultimate goal which is often a lake or a sea. The ideas are taken from the natural water drops are used in order to develop artificial water drops. The Artificial water drops are then adapted to solve the Travelling salesman problem (TSP).

S.Pandey, et al [2] used PSO which is a self adaptive global search based optimization Technique. They have formulated a model for task-resource mapping to minimize overall cost of execution. The algorithm is similar to other population-based algorithms like Genetic algorithms but, there is no direct re-combination of individuals of the population. Instead, it relies on the social behaviour of the particles. In every generation, each particle adjusts its trajectory based on its best position (local best) and the position of the best particle (global best) of the entire population. This concept increases the stochastic nature of the particle and converges quickly to global minima with a reasonable good solution. Performs better then BRS because of the way it takes into account of communication cost of all tasks, including dependencies between them.

Zhangjun Wu1, Zhiwei Ni1, Lichuan Gu [3] proposed set based concept < task, service > pairs. Each pair means a mapping that task is mapped onto a service. Set-based concept is introduced into PSO to solve combinatorial optimization problems, such as determining RNA secondary structure, travelling salesman problem (TSP) and multidimensional knapsack problem (MKP). This concept has been proved to be promising. Based on the set-based scheme, RDPSO is used to minimize the total computation cost of cloud workflowthis method gives better performance when compared to PSO.

Wei-Neng Chen et al [4] proposed A Novel Set Based PSO (S-PSO) method for the solution of some combinatorial optimization problems in

discrete space is presented. Based on the concept of sets and the possibility theory, S-PSO has two features. First, a set-based representation scheme is designed to characterize the discrete search space as a universal set of elements. Second, each candidate solution in S-PSO corresponds to a crisp subset out of the universal set. As a result, the search behaviour of S-PSO is very similar to that of the original PSO in continuous space.

S. Rao Rayapudi [5] Economic Load Dispatch (ELD) is a method of determining the most efficient, low-cost and reliable operation of a power system by dispatching available electricity generation resources to supply load on the system. The main objective of ELD problem is to decrease fuel cost of generators, while satisfying equality and inequality constraints. an intelligent water drop (IWD) algorithm has been proposed to solve ELD problem with an objective of minimizing the total cost of generation. Intelligent water drop algorithm is a swarm-based nature inspired optimization algorithm, which has been inspired from natural rivers. A natural river often finds good paths among lots of possible paths in its ways from source to destination and finally find almost optimal path to their destination. These ideas are embedded into the proposed algorithm for solving economic load dispatch problem.

III. WORKFLOW SCHEDULING

A workflow is a collection of tasks connected together in a certain order for their execution. Workflows are usually modelled as Directed Acyclic Graphs (DAGs).Information is carried from one task to another task. Each node in a DAG represents a workflow task and directed links indicate the task dependencies.DAG is denoted as G = (V,A) where V is the set of nodes V = $\{T_1,T_2,....T_n\}$ represents the tasks ; A represents the set of arcs that denotes the precedence constraints and the data dependencies between tasks. An arc is in the form of $d_{i,j} =(T_i,T_j) \epsilon A$, where T_i, is called the parent task of T_j, $T_j$is the child task of T_i $d_{i,j}$, is the data produced by $T_i$and consumed by T_j. We assume that a child task cannot be executed until all of its parent tasks have been completed.

Consider that k tasks are to be scheduled on m service instances, we have set of compute sites PC = {1.....j}, a set of storage sites S = {1....i} where the service instances are provided by the GSP (Global Service Provider) e.g. Amazon. The access cost and the transfer cost is fixed by the service provider.

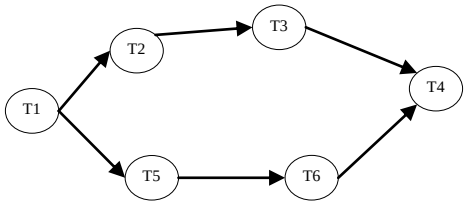

Fig. 1 A workflow with 6 tasks

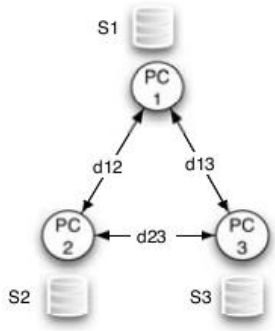

Figure 2. Compute nodes (PC) & storage

Figure 1 depicts the workflow with six tasks which are represented as nodes, in which each tasks can be executed by three service instances.Figure. 2 shows the three compute resources (PC1, PC2, PC3) and the data storage unit (S1, S2,S3), they are fully connected and symmetric.The main goal is to map the workflow tasks to the compute resources efficiently so that the total computation cost is minimized.

Let C_{exe}(M) be the total execution cost of all tasks assigned to a compute resource PC_j (Eq 1).

This value is obtained by adding all the node weights (cost of execution of a task k on a compute resource j) of all tasks assigned to each resource.Let C_{tx}(M) be the total access cost (including the transfer cost) between tasks assigned to a compute resource PC_j and those that are not assigned to that resource in the mapping M (Eq. 2). This value is the product of the output file size (given by the edge weight e_{k1},e_{k2}) from a task k1 ∈ k to task k2 ∈ k and

the cost of communication from the resource where k1 is mapped (M(k1)) to another resource where k2 is mapped (M(k2)). The average cost of communication of unit data between two resources is given by dM(k1),M(k2). The cost of communication is applicable only when two tasks have file dependency between them. For two or more tasks executing on the same resource, the communication cost is zero.

$C_{exe}(M) = \sum_k w_{kj} \forall M(k) = j$ (1)

$C_{tx}(M) = \sum_{k1 \in T} \sum_{k2 \in T} d_{M(k1),M(k2)} e_{k1,k2}$ (2)
$\forall M(k1) = j \; and \; M(k2) \neq j$

$C_{total}(M) = C_{exe}(M) + C_{tx}(M)$ (3)

$Cost(M) = \max(C_{total}(M)) \;\; \forall j \in P$ (4)
$= Minimize(Cost(M))$ (5)

Equation 4 ensures that all tasks are not assigned to a single compute resource. Initially max cost will be required to disribute tasks to all resources. Equation 5 ensures that the total cost is minimal even after the initial distribution through the subsequent minimization of the overall cost . For a given assignment M, the total cost $C_{total}(M)$ for a compute resource PC_j is the sum of execution cost and access cost (Eq. 3). On determining the total cost for all resources, the maximum cost for all the resources is minimized (Eq. 5).

IV. PROPOSED ALGORITHM

A. Natural Water Drops

In nature, flowing water drops are observed mostly in rivers, which form huge moving swarms. The paths that a natural river follows have been created by a swarm of water drops. It is often observed that the constructed path seems to be an optimal one in terms of the distance from the destination and the constraints of the environment.

One feature of a water drop flowing in a river is its velocity. It is assumed that each water drop of a river can also carry an amount of soil. Therefore, the water drop is able to transfer an amount of soil from one place to another place in the front.Three obvious changes happen during the transition:

• Velocity of the water drop is increased.
• Soil of the water drop is increased.
• Between these two points, soil of the river's bed is decreased**.**

Properties of water

- A high speed water drop gathers more soil than a slower water drop.
- The velocity of a water drop increases more on a path with low soil than a path with high soil

A water drop prefers a path with less soil than a path with more soil

B. Intelligent Water Drops (IWDs)

The Intelligent Water Drop, IWD for short, has two important properties

• The soil it carries, denoted by soil (IWD).
• The velocity that it posses, denoted by velocity (IWD).

For each IWD, the values of both properties, soil (IWD) and velocity (IWD) may change as the IWD flows in its environment. From the engineering point of view, an environment represents a problem that is desired to be solved. A river of IWDs seeks an optimal path for the given problem.

C. Intelligent Water Drop Algorithm.

The IWD algorithm employs a number of IWDs to find the optimal solutions to a given problem. The problem is represented by a graph (*N*, *E*) with the node set *N* and edge set *E*. This graph is the environment for the IWDs and the IWDs flow on the edges of the graph. Each IWD begins constructing its solution gradually by travelling between the nodes of the graph along the edges until the IWD finally completes its solution denoted by T^{IWD}. Each solution T^{IWD} is represented by the edges that the IWD has visited. One iteration of the IWD algorithm is finished when all IWDs complete their solutions. After each iteration, the iteration-best solution $T^{IB}$is found. The iteration-based solution $T^{IB}$is the best solution based on a quality function among all solutions obtained by the IWDs in the current iteration. $T^{IB}$is used to update the total-best solution T^{TB}. The total-best

solution $T^{TB}$is the best solution since the beginning of the IWD algorithm, which has been found in all iterations. IWD Algorithm in ten steps:

1. Initialization of static and dynamic parameters

Static parameters are those parameters that remain constant during the lifetime of the IWD algorithm.

N_c- Number of nodes

N_{IWD}- Number of IWD

InitSoil - Initial soil value on each edge

InitVel - IWD Velocity for each IWD.

ρ_n - Local soil updating parameter

ρ_{IWD} - Global soil updating parameter

Dynamic parameters are those parameters, which are dynamic and they are reinitialized after each iteration of the IWD algorithm.

V_c(IWD) - Visited node(Initially empty)

2.Choosing next node based on probability

For the IWD residing in node *i*, choose the next node *j*, which doesn't violate any constraints of the problem and is not in the visited node list V_c(IWD) of the IWD using the following probability $P^i_{IWD}(j)$

$$P^i_{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \notin V_c(IWD)} f(soil(i,j))}$$

such that

$$f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))}$$

Then, add the newly visited node *j* to the list V_c(IWD).

3.Update velocity of each IWD

For each IWD moving from node *i* to node *j*, update its velocity vc(IWD).

$$vel^{IWD}(t+1) = vel^{IWD}(t) + \frac{a_v}{b_v + c_v.soil^2(i,j)} \quad (6)$$

where $vel^{IWD}(t+1)$ is the updated velocity of the IWD.

4.Soil loaded from the path

For the IWD moving on the path from node *i* to *j*, compute the soil $\Delta soil(i,j)$ that the IWD loads from the path by

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s.time^2(i,j:vel^{IWD}(t+1))} \quad \textbf{(7)}$$

suchthat

$$time(i,j:vel^{IWD}(t+1)) = \frac{HUD(j)}{vel^{IWD}(t+1)}$$

where the heuristic undesirability *HUD*(*j*) is defined appropriately for the given problem.

5. Local soil updating

Update the soil *soil*(*i*, *j*) of the path from node *i* to *j* traversed by that IWD, and also update the soil that the IWD carries $soil^{IWD}$ by

$$soil(i,j) = (1-\rho_n).soil(i,j) - \rho_n.\Delta soil(i,j)$$

$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j)$$

6. Finding the Iteration best solution

Find the iteration-best solution $T^{IB}$from all the solutions T^{IWD}found by the IWDs using $T^{IB} = \arg\min_{\forall T^{IWD}} q(T^{IWD})$

7. Global soil updating

Update the soils on the paths that form the current iteration-best solution $T^{IB}$by

$$soil(i,j) = (1+\rho_{IWD}).soil(i,j) - \rho_{IWD} \cdot \frac{1}{(N_{IB}-1)} \cdot soil^{IWD}_{IB}$$

$$\forall (i,j) \in T^{IB}$$

8. Update the total best solution

Update the total best solution $T^{IB}$by the current iteration-best solution T^{IB}using

$$T^{IB} = \begin{cases} T^{TB} & \text{if } q(T^{IB}) \geq q(T^{IB}) \\ T^{IB} & \text{otherwise} \end{cases}$$

9. Increment the iteration

Increment the iteration number by $Iter_{count} = Iter_{count} + 1$. Then, go to step 2 if $Iter_{count} < Iter_{max}$

10. Terminate

The algorithm stops with the total-best solution T^{TB} . Thus the algorithm stops with the total-best solution or when the max iteration specified is reached.

V. SIMULATION RESULTS

The value assigned to a each dimension of a particles are the computing resources indices. Thustheparticlerepresentsa mapping of resource to a task. In our workflow (depicted in Figure 1) each particle is 6-D because of 6 tasks and thecontentofeachdimensionoftheparticlesisthe computer-source assigned to that task. For example a sample particlecould be represented as depicted in Figure 2.The evaluation of each particle is performed by the fitness function given in Eq. 5.

Table 1. Task Assigned to Resource

Task1	Task2	Task3	Task4	Task5	Task6
PC1	PC3	PC2	PC3	PC2	PC1

The TP-matrix, PP-matrix .The valuesshown are an example of 1 instance of the experiment run. We have used three matrices that store the values for:a) average computation cost of each task on each resource(TP-matrix)as depicted in Table 2,b)averagecommunicationcostperunitdata between compute resources (PP-matrix), as depicted in Table 2.The values for PP-matrix resemble the cost of unit datatransfer between resources given by Amazon CloudFront4. We assume PC1 to be in US, PC2 in Hong Kong (HK) andPC3 in Japan (JP), respectively. We randomly choose thevaluesinthematrixforeveryrepeatedexperiment,but keepthese values constant during the IWD iterations.

Table 2. TP Matrix

TP=

	PC1	PC2	PC3
T1	1.80	1.12	1.15
T2	1.60	1.23	1.28
T3	1.50	1.11	1.72
T4	1.80	1.50	1.14
T5	1.19	1.15	1.22
T6	1.10	1.14	1.13

Table 3. PP Matrix

PP=

	PC1	PC2	PC3
PC1	0	0.17	0.21
PC2	0.17	0	0.22
PC3	0.21	0.22	0

Figure 3 plots the convergence of total cost computed by IWD over the number of iterations for different sizes of total data processed by the work flow in Figure1. Initially,the particles are randomly initialized. Therefore,the initial totalcost is always high. This initial cost corresponds to the 0^{th}iteration. As the algorithm progresses, the convergence isdrastic and it finds a global minima veryquickly. The number of iterations needed for the convergence is seen to be 100,for our application environment.

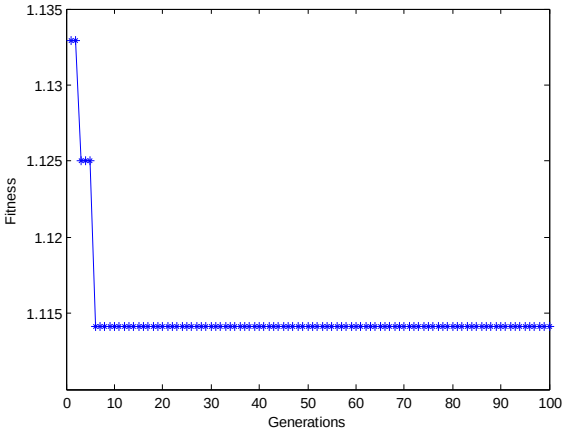

Figure 3.Convergence Graph

VI. Conclusions and Future Work

In this work, we presented a scheduling heuristic based on IWD to minimize the total cost of execution of application workflows on Cloud computing environments. We obtained total cost of execution by varying the communication cost between resources and the execution cost of compute resources. We compared the results obtained by ourheuristic against "Best Resource Selection" (BRS) heuristic. We found that IWD based task-resource mapping can achieve at least three times cost savings as compared to BRS based mapping for our application workflow. In addition, IWD balances the load on compute resources by distributing tasks to available resources.

The heuristic we proposedis generic as it can be used for any number of tasks and re-sources by simply increasing the dimension of the particlesand the numberof resources, respectively. As part of our future work, we

would like to integrate IWD based heuristic into our work flow management system to schedule work flows of real application such as brain imaging.

REFRENCES

[1] Brian Hayes, Cloud computing, Communications of the ACM,Volume 51, Issue 7, July 2008.

[2] Hamscher, Y., et a!., Evaluation of Job-Scheduling Strategies for Grid Computing. LECTURE NOTES IN COMPUTER SCIENCE, 2000: p.191-202.

[3] Buyya R. Economic2based distributed resource management and scheduling for grid computing [0]. Melbourne: School of Computer Science and Software Engineering, Monash University, 2002.

[4] Buyya R, Abramson 0, Giddy J. Nimrod IG: An architecture for a resource management and scheduling system in a glob2 al computational grid [C] IIProceedings

[5] Weidong, H., Y. Yang, and L. Chuang. Qos Performance Analysis for Grid Services Dynamic Scheduling System. in Wireless Communications, Networking and Mobile Computing, 2007. Wi Com 2007. International Conference on. 2007.

[6] Afzal, A., A.S. McGough, and J. Darlington, Capacity planning and scheduling in Grid computing environments. Future Generation Computer Systems 2008. 2008(24): p. 404-414.

[7] Hamed Shah-Hosseini,Optimization with the Nature-Inspired Intelligent Water Drops Algorithm,Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G.C.,Tehran,Iran.