SCHEDULING IN INSTANCE-INTENSIVE COST-CONSTRAINED WORKFLOWS IN A CLOUD

S. Mukute, G. Hapanyengwi, B. Mapako, B. M. Nyambo, A. Mudzagada

Abstract—Cloud computing has been growing tremendously, as it has been accepted recently. Cloud computing has many unique strengths which can be utilized to facilitate workflow execution. This paper investigates the issue of dynamic scheduling in cloud computing with a special attention to the case of instance-intensive cost-constrained workflows. This paper seeks to address the following issues: A study of this 'new' technology taking its gradual migration from Grid computing, their similarities, differences, weaknesses and strengths. Also there will be a study on the metrics of comparing existing scheduling algorithms with a comparison on how they are fairing. Lastly this paper seeks to propose a dynamic scheduling algorithm that will be based on a simulation model. In this research the problem of workflow scheduling has been discussed. The efficiency of the proposed algorithm was tested in comparison with the fixed algorithm that come along with the simulator used (CloudSim 2.1).

Index Terms— Scheduling , Cost, Cloud, Instance- Intensive Workflows, Cost Constrained Workflows

1 INTRODUCTION

Cloud computing can be viewed as an extension of parallel computing and distributed computing. It provides secure, quick, convenient data storage and computing power with the help of the internet. Figure 1, shows an overview of cloud computing. Virtualization, distribution and dynamic extendibility are the basic characteristics of cloud computing. These days most software and hardware have provided support to virtualization. Factors that can be virtualized and managed on a cloud computing platform include software, hardware, operating system and net storage. Efficient scheduling algorithms are required for us to make effective use of the tremendous capabilities of the cloud. There is a need to optimally dispatch workflows to the cloud resources. Scheduling algorithms try to minimize the total completion time of the workflows in the cloud by finding the most suitable resources to be allocated to the workflows [1].

A workflow enables the structuring of applications in a directed acyclic graph form, where each node represents the constituent task and edges represent inter-task dependencies of the applications. A single workflow generally consists of a set of tasks each of which communicates with each other in a sequentially dependent order. Workflow scheduling is one of the key issues in the management of workflow execution [2]. An instance is a single execution occurrence of a workflow at a particular time. Instance-intensive cloud workflows are workflows with a huge number of instances concurrently running on a distributed environment. Examples of instance-intensive workflows include processes like bank cheque processing, insurance claim processing and many other e-business and e-government scenarios. In a bank cheque processing scenario, where millions of cheque-processing transactions need to be processed concurrently each day, while each of them is a rather simple workflow with only a few steps. Considering instance-intensive workflows, the mean execution time, becomes a more important criterion of scheduling instance-intensive workflows than execution time of individual instances [3].

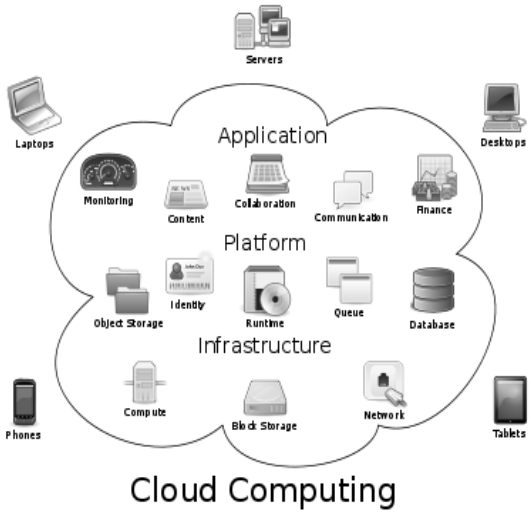

Fig. 1 Overview of cloud computing [1]

Moving workflows to a cloud computing environment enables the utilization of various cloud services to facilitate workflow execution. In contrast to dedicated resources, the resources in clouds are shared and provided to users ''on-demand'', meaning the expenditure on hardware for workflow execution are eliminated. The ''user-centric'' model in a cloud computing environment makes workflow execution more user-friendly thus increasing user satisfaction. The ''pay as you go'' business model in a cloud can reduce the execution cost of workflows

[3]. There is, therefore, a need to migrate workflow executions to cloud computing platforms. However, on a commercial cloud platform, users need to pay for what they use. Assuming these instance-intensive workflows have cost-constraints and are executed on cloud computing platforms, we call them instance-intensive cost-constrained cloud workflows.
In this paper we seek to propose a dynamic scheduling algorithm that will be based on a simulation model.

2 RELATED WORK

Many researchers have come up with different proposals of implementing scheduling algorithms in both distributed systems and cloud computing environments. Listed below are some of the examples:

a) A Compromised-Time-Cost Scheduling Algorithm: Ke Liu, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan, Yun Yang [3]. This algorithm encompassed the issue of instance-intensive cost c0nstrained workflows by reducing the execution cost at the expense of execution time within user designated deadlines. The environment used was SwinDeW-C (**Swin**burne **De**centralised **W**orkflow for **C**loud) as the tool for simulation.
b) A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications: Suraj Pandey, LinlinWu, Siddeswara Mayura Guru, Rajkumar Buyya [4] heuristic based algorithm to schedule applications to cloud resources taking into consideration the computation and communication costs.
c) Improved Cost-Based Algorithm for Task Scheduling: Mrs.S.Selvarani, Dr.G.Sudha Sadhasivam [4]. The idea behind this algorithm was to make an efficient improvement in the mapping of accrued costs. The algorithm also considered computational/communication cost ratio.
d) Resource-Aware-Scheduling algorithm (RASA): Saeed Parsa and Reza Entezari-Maleki [**Error! Reference source not found.**] proposed a new task scheduling algorithm RASA. This algorithm is a fusion of the traditional Max-min and Min-min algorithms. This algorithm was mostly duelling on the execution cost, arrival time and deadline for the tasks without considering the communication costs.
e) Innovative transaction intensive cost-constraint scheduling algorithm: Yun Yang, Ke Liu, Jinjun Chen [**Error! Reference source not found.**]. The thrust on this algorithm was on execution cost and time within specified user designated times.
f) Scalable Heterogeneous Earliest-Finish-Time Algorithm (SHEFT): Cui Lin, Shiyong Lu [8] proposed a workflow scheduling algorithm to schedule a workflow elastically on a Cloud computing environment which they named SHEFT. This algorithm achieved optimization of execution time while achieving elastic scalability at runtime.
g) Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW): Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi [9] worked on multiple workflows and multiple QoS. They have a strategy implemented for multiple workflow management system with multiple QoS. The scheduling access rate is increased by using this strategy. This strategy minimizes the make span and cost of workflows for cloud computing platform.

Mathematical Formulation of Scheduling Problems

Scheduling problems can generally be associated with the popularly and vastly researched Job Shop Problem. The general idea of this problem is summarized by Brucker P [10]:
The job-shop problem can be formulated as follows. Given are m machines $M_1,M_2,...,M_m$ and n jobs $J_1,J_2,...,J_n$. Job Jj consists of n_j operations $O_{ij}(i = 1, ... n_j)$ which have to be processed in the order $O_{1j}, O_{2j}, ... O_{nj}$. It is convenient to enumerate all operations of all jobs by k = 1, ...,N where $N = P_{nj=1\ nj}$. For each operation k = 1, ...,N we have a processing time pk > 0 and a dedicated machine M(k) and k must be processed for pk time units without preemptions on M (k). Additionally a dummy starting operation 0 and a dummy finishing operation N + 1, each with zero processing time, are introduced. We assume that for two succeeding operations $k = O_{ij}$ and $s (k) = Oi+1,j$ of the same job M (k) 6= M (s (k)) holds. Let S_k be the starting time of operation k. Then $C_k = S_k + pk$ is the finishing time of k and (S_k) defines a schedule. A schedule (S_k) is feasible if for any succeeding operations k and s (k) of the same job $S_k + pk \leq Ss(k)$ holds and for two operations k and h with M (k) = M (h)either $S_k + pk \leq Sh$ or $Sh + ph \leq Sk$. One has to find a feasible schedule (S_k) which minimizes the make span $maxN\ k=1\ C_k$.

Scheduling Problems

The author considered k machines M_j (j=1,…,k) that are required to process n jobs. A schedule can be defined as an allocation of ≥1 time intervals to ≥1 machines [10].
Now we consider each job J_i to comprise of predefined number n_i of operations $O_{i1},…,O_{i,ni}$. We also have to note that for each operation O_{ij} , $j\varepsilon n_i$ there is an associated processing requirement Pij. In addition to this, each operation O_{ij} is associated with $\mu ij\varepsilon(M_1,…,M_k)$recalling that $j\varepsilon(1,…,k)$. In simplicity we are saying: if O_{12} is associated μ_{15} it is interpreted as saying operation number 2 of job task 1 has been scheduled to machine number 5. There is also need to define two scenarios

here:
Dedicated machines: all μ_{ij} can be one element set.
Parallel machines [Multipurpose Machines]: All $\mu_{ij} = M_j$ (j=1,…,k)
Finally we have the cost function $f_i(t)$. This parameter may be defined by two parameters due date d_i and weight w_i.

Definition of parameters:

Generally all scheduling algorithms can be depicted as a three-field variable: $\alpha | \beta | \gamma$ [10] where:
The variable α denotes the machine environment (usually the number of machines)
The variable β denotes the job (workflow) characteristics.
The variable γ denotes the optimality criterion in the scheduling algorithm under consideration.
Parameter α is defined further as a string with two parameters $\alpha_1\alpha_2$. If $\alpha_1 = o \rightarrow \alpha = \alpha_2$. The substring α_1 is further decomposed to:

$$\alpha_1 \in (o, P, Q, R. PMPM, QMPM, G, X, O, J, F)$$

In most research works of this level the elements G, X, O, J and F are ignored.
This instance is interpreted as saying each job J_i has a single operation and should be processed on a single specified dedicated machine; hence no scheduling will be employed in this case.
α_1=P: this implies that we have identical parallel machines. Thus we have P_{ij} for job J_i on M_i, $P_{ij}=P_i$ for all $j\varepsilon(1,\ldots,k)$
α_1=Q: we have uniform parallel machines thus we have

$$P_{ij} = \frac{P_i}{s_j}$$

where s_j in this case denotes the processing speed of machine M_j
α_1=R: this is the case where we have unrelated parallel machines that are P_{ij} for job dependant speeds s_{ij} of M_j
α_1=PMPM and α_1=QMPM then we have multi-purpose machines with identical and uniform speeds respectively.
In scheduling algorithms the β parameter is usually represented as a six-fold string thus:

$$\beta = (\beta_1,\ldots,\beta_6).$$

- β_1: determines whether there is preemption or not
- β_2: denotes precedence relations between jobs. This string is usually represented an acyclic directed graph G= (V,A) where $V = (1, \ldots n)$ implying the jobs present and also $(i,k)\epsilon A \leftrightarrow J_i$ must be accomplished before J_k thus β_2 will be set:

$$\beta_2 = prec$$

- $\beta_3 = r_i$, where i = (1,…, n) this notation is interpreted as representing the release dates for each respective job operation J_i.
- β_4: specifies the number of operations for each job J_i
- $\beta_5=d_i$, denotes the deadline for each job J_i
- β_6: denotes the batch processing problems
- γ: is the job optimality criterion.

The finishing time for job J_i is denoted by C_i and the associated cost by $f_i(C_i)$. Basically there are two functions that depict total cost.

$$f_{max}(C) = \max(f_i\ (c_i)|i = 1..k)$$

which is the bottleneck problem and
$\sum f_i(C) = \sum(f_i(C_i)|i = 1,..k)$ the sum objectives.

3 METHODOLOGY

As demonstrated in this document, the numbering for sections upper case Arabic numerals, then upper case Arabic numerals, separated by periods. Initial paragraphs after the section title are not indented. Only the initial, introductory paragraph has a drop cap.

Designing of the proposed algorithm

The following is a listing of the assumptions that will be used in this paper:

- Cost depends on the time slot chewed by each operation O_{ij} on a particular machine
- The set $\mu_{ij}=(M_1,\ldots M_3)$
- Assume the workflow is a predefined query of fixed size in bits
- All workflows as the name implies are similar in size
- Associate with each M_j a predefined bandwidth (which will be represented by speed s_j denoted in the simulation environment by MIPS)
- To M_j associate a corresponding cost c_j which is interpreted as cost per unit time

The parameters used in this simulation will be:
α_1=Q: we have uniform parallel machines thus we have $P_{ij}=P_i/s_j$ where s_j in this case denotes the processing speed of machine M_j for $j\varepsilon(1,\ldots,3)$
β_4: which specifies the number of operations for each workflow J_i for $j\varepsilon(1,\ldots,3)$

$$\gamma = f_{max}(C) = max(f_{ij}(c_{ij})|i,j = 1,2,.)$$

Clear clarification need to be made on the optimal criterion. In this paper we will minimize as much as possible the ultimate cost since attention is from the client side.
Basing on the definitions made above:
We can define

$$P_{ij} = \frac{O_{ij}}{s_j}$$

and this in units is:

$\frac{Mb}{Mb} * s = s$, recalling *s* denotes time

Thus we get a result which reflects time usage on a particular machine on a given operation for a specific workflow.

We define another variable

$$U = \frac{O_{ij}}{s_j} c_j$$

Whose units symbolically becomes:

$$\left(\frac{Mb}{Mb}\right) * s * \frac{£}{s} = £$$

Now we have defined our γ which is cost and much attention will be from the client side. Thus our

$$\gamma = f_{max}(C) = max\left(f_{ij}\left(c_{ij}\right)|i,j = 1,2,.\right)$$

Since the idea behind the proposed algorithm is to *maximize* costs from the client perspective,

$$\gamma = f_{min}(C) = min\left(f_{ij}\left(c_{ij}\right)|i,j = 1,2,.\right)$$

Symbolically:

$$\gamma = min \sum \left(f_{ij}\left(c_{ij}\right)|i,j = 1,2..\right)$$

with $f_{ij}\left(c_{ij}\right) = U = \left(\frac{O_{ij}}{s_j}\right) c_j$

Resource allocation to the various operations examples

Table 1: Matrix assuming sequential allocation

	Operations		
Machine number	**O_1**	**O_2**	**O_3**
M_1	O_{11}	O_{12}	O_{13}
M_2	O_{21}	O_{22}	O_{23}
M_3	O_{31}	O_{32}	O_{33}

A closer look shows that this degenerates to a sequential scheduling algorithm. Also interchanging the workflow allocation will not produce any differences.

Table 2: Matrix assuming sequential allocation

	Operations		
Machine number	**O_1**	**O_2**	**O_3**
M_1	O_{21}	O_{22}	O_{33}
M_2	O_{11}	O_{32}	O_{13}
M_3	O_{31}	O_{12}	O_{23}

The above again is similar on terms of total time regardless of the order used.

*Critical observation: $T(O_{ki}) = T(O_{gi})$ for all k!=g and k ,g ε(1,2,3) and i ε(1,2,3) provided they have been allocated to the same machine M_j.

The matrices can go on and on and the above were done just as an illustration. Also note that it is not assumed that the different operations have similar duration but it is meant to demonstrate the scenario only.

As a matter of simplifying the situation the author suggests:

Assume we let $O_{11}=x$ $O_{12}=y$ $O_{13}=z$

Table 3: Simplification of resource allocation

	Operations		
Machine number	**X**	**Y**	**Z**
M_1	***x***	***y***	***z***
M_2	***x***	***y***	***z***
M_3	***x***	***y***	***z***

Note the order in which the operations are arranged.

From the above matrices we can come up with the following permutations:

$$x_1+y_1+z_1$$
$$x_1+y_1+z_2$$
$$x_1+y_1+z_3$$

Here, x and ***y*** operations are confined to **M_1**

$$x_1+y_2+z_1$$
$$x_1+y_2+z_2$$
$$x_1+y_2+z_3$$

In this case, ***x*** and ***y*** operations are confined to **M_1** to **M_2** respectively

$$x_1+y_3+z_1$$
$$x_1+y_3+z_2$$
$$x_1+y_3+z_3$$

Here, ***x*** and ***y*** operations are confined to **M_1** to **M_3** respectively

Where:

x_i means that operation ***x*** has been allocated to machine ***i***.

y_i means that operation ***y*** has been allocated to machine ***i.***

z_i means that operation ***z*** has been allocated to machine ***i.***

The three set of equations have been done to demonstrate the situation when ***x*** has been allocated fixed to machine M_1 with ***y***

varying but constant at each occurrence.
Since we have defined our criterion function we can proceed to model proposed algorithm.
N.B: The subscripts on s and c denote the particular machine to which a particular instance has been allocated. There are 3 machines and 3 operations thus there are a possible 81 permutations. Thus to simplify the situation 3 parameters i, j and k are introduced to loop through all the possible iterations.

Table 4: A summary of the calculations at each loop

i=0 $U = \frac{x}{s_1} c_1$	j=0 $U = \frac{y}{s_1} c_1$	k=0 $U = \frac{z}{s_1} c_1$
i=1 $U = \frac{x}{s_2} c_2$	j=1 $U = \frac{y}{s_2} c_2$	k=1 $U = \frac{z}{s_2} c_2$
i=2 $U = \frac{x}{s_3} c_3$	j=2 $U = \frac{y}{s_3} c_3$	k=2 $U = \frac{z}{s_3} c_3$

The proposed algorithm in pseudo code

(Procedure 1
While i=0
(for (j=0; j<3;j++)
(for (k=0; k<3;k++)
Do
Calculate U;
Return min (U)
(Increment i+=1;
Repeat procedure 1
)
(Increment 1+=1;
Repeat procedure 1
)
Return min Procedure 1, 2, 3
Schedule as from result above

4. RESULTS AND ANALYSIS

The environment used to simulate the algorithm is an inbuilt component of the CloudSim version 2.1. Results were input through the command prompt line. CloudSim 2.1 is a JAVA coded environment that allows the user to code, compile and run the user's own input programs. The proposed algorithm was fed with following fixed RAM size of 2048MB and fixed WORKFLOW SIZE:10 000MB. The different processing capabilities of the three simulated machines were modelled by the varying MIPS on each instance. We compared our algorithm with the CloudSim 2.1 algorithm. Table 6 shows the results after running 8 simulations for each of the algorithms.
DS (£): This value denotes the accumulated debt after implementing the proposed (**Dynamic Scheduling**) algorithm. **CS (£):** This value denotes the accumulated debt after implementing the inbuilt (**CloudSim**) algorithm. Figure 5 below is a graphical representation of the results obtained.

Table 5: Comparison of results from the proposed algorithm to the CloudSim 2.1 algorithm

Run #	MIPS (M1, M2, M3)	DS(£)	CS(£)
Run1	75, 105, 90	98.134	102.32
Run2	100, 120, 105	104.55	104.99
Run3	150, 135, 115	109.987	112.48
Run4	200, 160, 120	113.26	121.241
Run5	250, 175, 125	116.26	141.624
Run6	350, 250, 130	125.3809	159.3808
Run7	400, 275, 180	184.8837	209.13
Run8	500, 350, 250	212.51	224.83

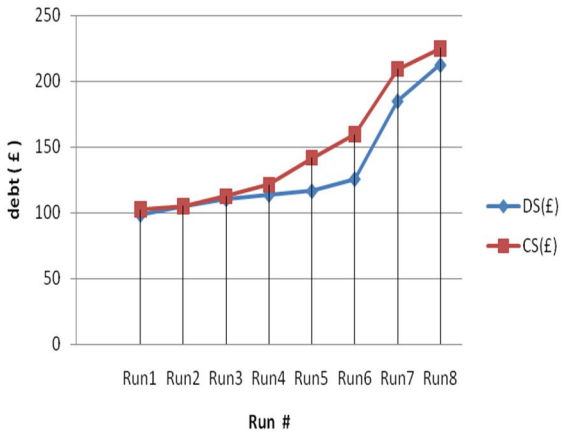

Figure 1: Graphical presentation of the results obtained.

The aim of this research was to make an improvement (reduction) on the cost from the client side of view. The above table and graphical presentation are meant to show the difference in performance from the two algorithms.
Initially, the total cost incurred for processing all the three workflows was almost similar for both DS and CS, as evidence

with the progress from Run 1 to Run 3. As the processing power overally increases (Run3 to Run 4), there is a notable dominance of DS over CS. A sharp increase is noted from Run 4 to Run 6 which however steadies till the end of the simulation (Run 8). This can be attributed to the ability of DS to optimize scheduling of available resources to requesting job (workflows) problems at a particular instance in time unlike CS algorithm which allocates resources regardless of optimum ability but according to previously configured scenarios. The scope of this research is to optimize resources utilization and allocation whilst minimizing the total cost incurred by the client.

5. CONLUSION

In this research the problem of workflow scheduling has been discussed. In day to day practice, industry and academia interact with workflow aware and unaware. The efficiency of the proposed algorithm was tested in comparison with the fixed algorithm that come along with the simulator used (CloudSim 2.1). The results show that the proposed algorithm produces a better efficiency in terms of cost with particular attention to the client side. From the simulated results, it is clear that the proposed dynamic algorithm is superior to the static algorithm to which it was compared. This observation is made in line with cost reduction which obviously is beneficial to the client side. The work presented in this paper can be expanded and improved in many directions. Some of the improvements that can be noted are to incorporate the issue of maximizing the use of bandwidth amongst different clients, making a sensible reduction in the communication time amongst the different virtual machines in consideration and also inclusion of different size tasks that would need to share the available resources.

6. REFERENCES

1. Sujit Tilak , Dipti Patil, A Survey of Various Scheduling Algorithms in Cloud Environment, International Journal of Engineering Inventions, Volume 1, Issue 2 (September)

2. Yogita Chawla, Mansi Bhonsle ,International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 1, Issue 3, September – October 2012

3. K Liu, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan, Yun Yang, A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on a Cloud Computing Platform. The International Journal of High Performance Computing Applications, Sage, Volume 24 Issue 4, November 2010 Pages 445-456

4. Suraj Pandey; LinlinWu; Siddeswara Mayura Guru; Rajkumar Buyya, A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments.24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 20-23 April 2010

5. S.Selvarani, G.Sudha Sadhasivam, Improved Cost-Based Algorithm for Task Scheduling. Computational Intelligence and Computing Research ICCIC 2010 IEEE International Conference on (2010) IEEE, Pages: 1-5

6. Saeed Parsa and Reza Entezari-Maleki, RASA: A New Task Scheduling Algorithm in Grid Environment, World Applied Sciences, pp. 152-160, 2009.

7. K Liu, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan, Yun Yang, An Algorithm in SwinDeW-C for Transaction-Intensive Cost-Constrained Cloud Workflows .Proc of 4th IEEE International Conference on e-Science,Indianapolis, USA, December 2008, (pp :374-375).

8. Cui Lin, Shiyong Lu, Scheduling Scientific Workflows Elastically for Cloud Computing. IEEE, 4th International Conference on Cloud Computing, 2011 (pp 746-747).

9. Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi, Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW). Parallel and Distributed Processing with Applications, 2009 IEEE, (pp: 629 – 634).

10. Peter Brucker. The Job-Shop Problem: Old and New Challenges. The 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications. (MISTA 2007) Paris, France (28 - 31 Aug 2007).

11. Brucker P, Scheduling Algorithms, 4th ed.: Springer, 2004. (pp: 5-6).