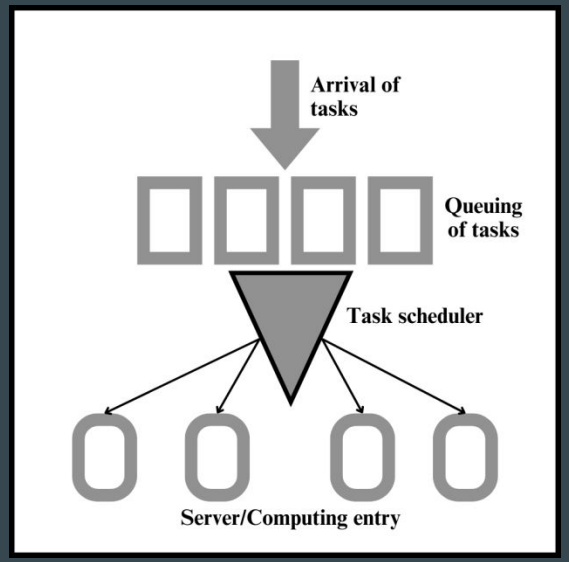
A Novel Task Scheduling Algorithm in Heterogeneous Multi-Cloud Environment

•••

What is **cloud task scheduling?**

- **Task scheduling** is the process of allocating tasks or jobs to virtual machines (VMs) that already exist using the resources available in the cloud environment.
- Scheduling is necessary to bring **efficiency in the processing of user requests**, **reduce task completion time** that is makespan, and **manage cloud resources** in a better way.



Cloud Task Scheduling – The Difficulties

- Cloud task scheduling is an NP-hard/NP-complete problem depending on the use case.
- Existing task scheduling algorithms offer **varied performance results** depending on application **cases and conditions**.
- The good scheduling algorithms help **maximize resource utilization** and **minimize the total execution time.**
- This research aims to improve upon the currently employed techniques by reducing the **makespan** and **cloud utilization** of some leading algorithms.

Some definitions

- **Makespan**: Makespan is the total time taken by the resources to complete the. executing of all tasks. It is a measure of the total throughput of the heterogeneous computing systems.
- **Cloud utilization:** The time for which each available cloud computing resource remains busy with a task is called cloud utilization. In this paper, we calculate the average cloud utilization of the proposed algorithm. It is the average time for which resources remain busy and is given as

$$U = \frac{\sum_{i=1}^{m} M(C_i)}{m}$$

Existing algorithms - SBTS

- **Smoothing Based Task Scheduling (SBTS)**: Has two phases: smoothing and scheduling.
  Smoothing divides tasks into various batches depending on their execution time.
  Then, they are allocated to the available clouds with the aim of minimizing makespan (Panda et al. 2014).

Existing algorithms - CMAXMS

- **Cloud Max-Min Scheduling (CMAXMS)**: The algorithm begins with a bunch of unscheduled tasks and then generates an **expected completion time (ECT) matrix** based on the available resources.
  It then assigns the **task with maximum completion time** to the cloud machine with the minimum completion time (MCT).

Existing algorithms - CMMS

- **Cloud Min-Min Scheduling (CMMS)**: Min-min task scheduling is based on Minimum Completion Time (MCT).
  It operates in two phases: first, the algorithm **calculates the ECT matrix** depending on the available resources.
  Then, the task with the **overall minimum expected completion time** is assigned to the corresponding cloud.

Suggested Algorithm - RGTS

- Random Grouped Task Scheduling (RGTS): The algorithm has two phases, namely Random grouping, and scheduling.
  In the first phase, all tasks are allocated to clouds based on their minimum execution time.
  In the second phase i.e., the grouping phase, all the tasks are divided into various batches or groups randomly and then the groups form a schedule among themselves randomly intending to minimize makespan.
- The algorithm is intended for use in IaaS since it involves calculations performed in static context.

Suggested Algorithm - RGTS

- Consider a set of m clouds C = {C_1, C_2, C_3,…, C_m}
- Consider a set of n task T = {T1, T2, T3,..., Tn}
- An ETC matrix represents the execution time of a task on a cloud which can be represented as:

$$ETC = \begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_l \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_m \end{array} \\ \begin{Bmatrix} ETC_{11} & ETC_{12} & \cdots & ETC_{1m} \\ ETC_{21} & ETC_{22} & \cdots & ETC_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ ETC_{l1} & ETC_{l2} & \cdots & ETC_{lm} \end{Bmatrix} \end{array}$$

ETC_{ij}, $1 \leq i \leq l$, $1 \leq j \leq m$ denotes the execution time needed to execute task Ti on cloud C_j

- A random task group formation follows the calculation of ETC matrix.
- The model is designed in such a way that every task Ti is assigned to Cj and every task must belong to a task group that was formed randomly.

Pseudocode of RGTS algorithm

1. The task (T_n) and cloud (C_m) machine list is taken as user input.
2. An ETC matrix is $ETC_{i,j}$ is generated.
3. Independent tasks T_i are assigned to clouds C_j according to their minimum E_{ij}
4. Random grouping is done as per $G=\{G_1, G_2,..., G_k\}$ with k as user input.

Pseudocode

Input:

1. A set of *n* independent tasks
2. A set of *m* cloud
3. An *ETC* matrix

Output:

1. Makespan
2. Cloud utilization

1. A set of n tasks $T = \{T_1, T_2, T_3,..., Tn\}$ and clouds $C = \{C_1, C_2, C_3,..., C_m\}$ are taken and $ETC_{i,j}$ matrix is generated taking the total execution time $E_{i,j}$
2. Assign $T_i \rightarrow C_j$, where $E_{i,j}$ is minimum. $(1 \le i \le n, 1 \le j \le m)$
3. Groups are formed, $G=\{G_1, G_2....., G_K\}$ where K is user input
4. *for* i=0 to n, assign rand $(T_i \rightarrow G_k)$
   $G_1, G_2, ..., G_K$ are executed randomly
5. Makespan and cloud utilization $M = max(\Sigma\ ETC\ (i,1) \times F\ (i,1), \Sigma\ ETC\ (i,2) \times F\ (i,2),..., \Sigma\ ETC\ (i,m) \times F\ (i,m))$ *and*

$$U = \frac{\sum_{i=1}^{m} M(C_i)}{m}$$ is generated.

Pseudocode of RGTS algorithm

6. Independent tasks T_i are assigned to groups G_k

7. $G_1, G_2, \ldots, G_k$ are then executed randomly.

8. Makespan is calculated using

$$M = max(\Sigma\, ETC(i,1) \times F(i,1), \Sigma\, ETC(i,2) \times F(i,2), \ldots, \Sigma\, ETC(i,m) \times F(i,m))$$

9. Average cloud utilization is calculated using

$$U = \frac{\sum_{i=1}^{m} M(C_i)}{m}$$

Pseudocode

Input:

1. A set of *n* independent tasks
2. A set of *m* cloud
3. An *ETC* matrix

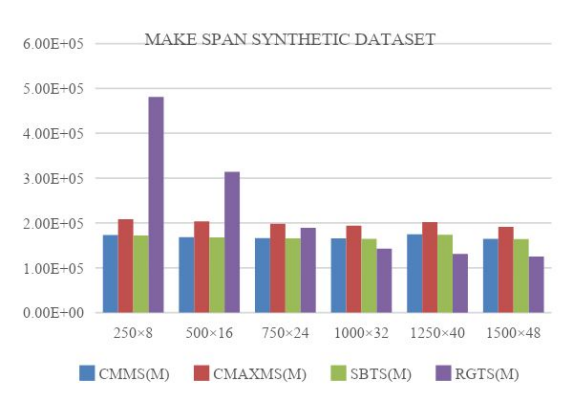
Output:

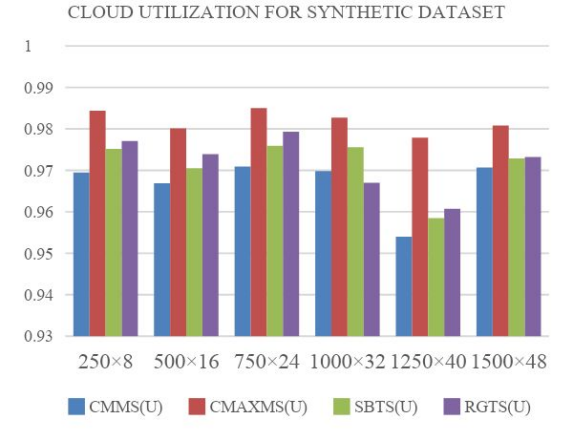
1. Makespan
2. Cloud utilization

1. A set of n tasks $T = \{T_1, T_2, T_3, \ldots, Tn\}$ and clouds $C = \{C_1, C_2, C_3, \ldots, C_m\}$ are taken and $ETC_{i,j}$ matrix is generated taking the total execution time $E_{i,j}$
2. Assign $T_i \rightarrow C_j$, where $E_{i,j}$ is minimum. $(1 \le i \le n, 1 \le j \le m)$
3. Groups are formed, $G = \{G_1, G_2, \ldots, G_K\}$ where K is user input
4. *for* i=0 to n, assign rand $(T_i \rightarrow G_k)$
   $G_1, G_2, \ldots, G_K$ are executed randomly
5. Makespan and cloud utilization $M = max(\Sigma\, ETC(i,1) \times F(i,1), \Sigma\, ETC(i,2) \times F(i,2), \ldots, \Sigma\, ETC(i,m) \times F(i,m))$ *and*

$U = \frac{\sum_{i=1}^{m} M(C_i)}{m}$ is generated.

Benchmark Results in Synthetic Dataset





Instance	CMMS(*M*)	CMAXMS(*M*)	SBTS(*M*)	RGTS(*M*)
250×8	1.73E+05	2.08E+05	1.72E+05	4.81E+05
500×16	1.68E+05	2.04E+05	1.68E+05	3.14E+05
750×24	1.66E+05	1.98E+05	1.65E+05	1.89E+05
1000×32	1.66E+05	1.94E+05	1.65E+05	1.43E+05
1250×40	1.75E+05	2.02E+05	1.74E+05	1.31E+05
1500×48	1.64E+05	1.92E+05	1.64E+05	1.25E+05
Instance	**CMMS(*U*)**	**CMAXMS(*U*)**	**SBTS(*U*)**	**RGTS(*U*)**
250×8	0.9695	0.9844	0.9752	0.9271
500×16	0.9669	0.9801	0.9705	0.9390
750×24	0.9709	0.9850	0.9759	0.9693
1000×32	0.9698	0.9827	0.9756	0.9670
1250×40	0.9540	0.9779	0.9585	0.9607
1500×48	0.9707	0.9808	0.9729	0.9732

Benchmark Results for Braun et al's dataset

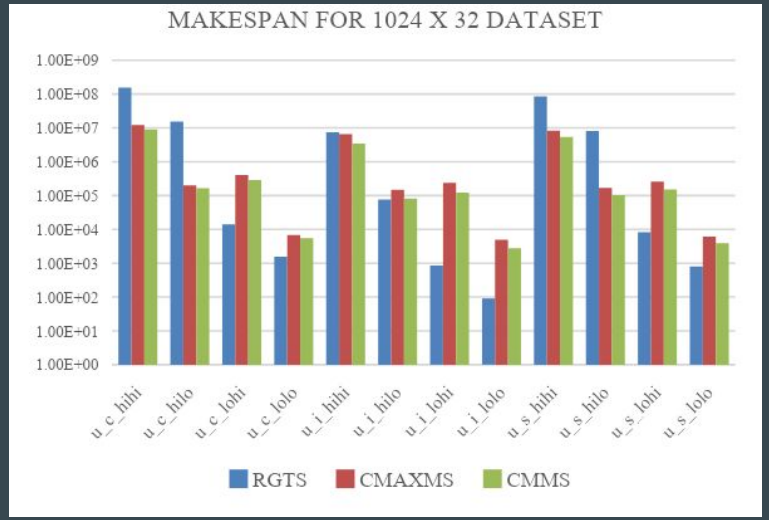
Instance	RGTS	CMAXMS	CMMS
u_c_hihi	1.54E+08	1.22E+07	9.00E+06
u_c_hilo	1.55E+07	2.01E+05	1.64E+05
u_c_lohi	1.42E+04	4.08E+05	2.88E+05
u_c_lolo	1.57E+03	6.87E+03	5.51E+03
u_i_hihi	7.46E+06	6.61E+06	3.48E+06
u_i_hilo	7.66E+04	1.47E+05	8.23E+04
u_i_lohi	8.54E+02	2.40E+05	1.23E+05
u_i_lolo	9.11E+01	4.92E+03	2.75E+03
u_s_hihi	8.48E+07	8.37E+06	5.40E+06
u_s_hilo	8.10E+06	1.67E+05	1.04E+05
u_s_lohi	8.34E+03	2.58E+05	1.52E+05
u_s_lolo	8.02E+02	6.07E+03	3.92E+03

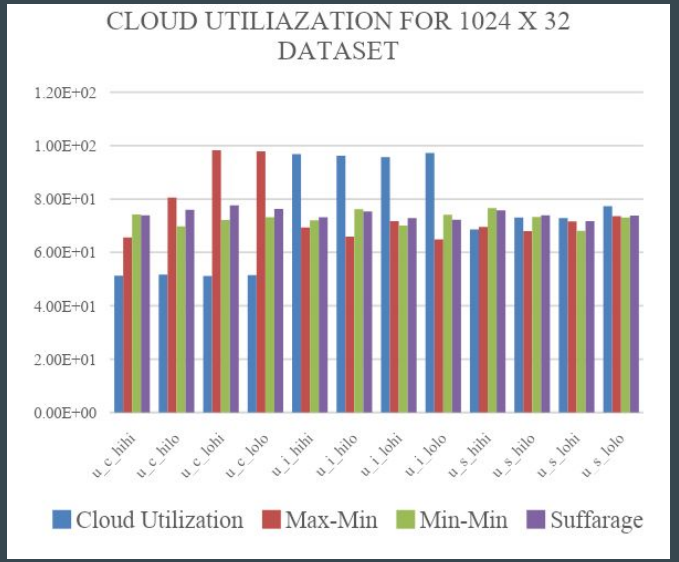
Makespan for 1024 x 32 dataset

Instance	RGTS	Max-Min	Min-Min	Suffrage
u_c_hihi	5.13E+01	6.56E+01	7.42E+01	7.38E+01
u_c_hilo	5.17E+01	8.05E+01	6.98E+01	7.60E+01
u_c_lohi	5.12E+01	9.83E+01	7.21E+01	7.77E+01
u_c_lolo	5.15E+01	9.79E+01	7.32E+01	7.63E+01
u_i_hihi	9.68E+01	6.93E+01	7.20E+01	7.32E+01
u_i_hilo	9.62E+01	6.59E+01	7.61E+01	7.54E+01
u_i_lohi	9.57E+01	7.17E+01	7.01E+01	7.28E+01
u_i_lolo	9.72E+01	6.48E+01	7.41E+01	7.22E+01
u_s_hihi	6.86E+01	6.95E+01	7.65E+01	7.58E+01
u_s_hilo	7.31E+01	6.80E+01	7.33E+01	7.39E+01
u_s_lohi	7.29E+01	7.16E+01	6.81E+01	7.18E+01
u_s_lolo	7.74E+01	7.36E+01	7.30E+01	7.38E+01

Cloud utilization for 1024 x 32 dataset

Benchmark Results for Braun et al's dataset





Benchmark Results for Braun et al's dataset

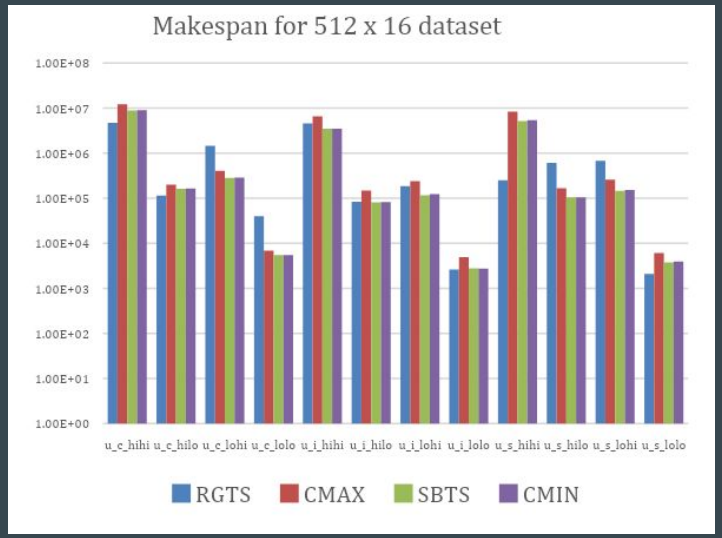
Instance	RGTS	CMAX	SBTS	CMIN
u_c_hihi	4.74E+06	1.22E+07	8.82E+06	9.00E+06
u_c_hilo	1.15E+05	2.01E+05	1.63E+05	1.64E+05
u_c_lohi	1.45E+06	4.08E+05	2.83E+05	2.88E+05
u_c_lolo	3.98E+04	6.87E+03	5.51E+03	5.51E+03
u_i_hihi	4.55E+06	6.61E+06	3.49E+06	3.48E+06
u_i_hilo	8.34E+04	1.47E+05	8.05E+04	8.23E+04
u_i_lohi	1.85E+05	2.40E+05	1.16E+05	1.23E+05
u_i_lolo	2.62E+03	4.92E+03	2.78E+03	2.75E+03
u_s_hihi	2.51E+05	8.37E+06	5.17E+06	5.40E+06
u_s_hilo	6.05E+05	1.67E+05	1.05E+05	1.04E+05
u_s_lohi	6.74E+05	2.58E+05	1.46E+05	1.52E+05
u_s_lolo	2.10E+03	6.07E+03	3.77E+03	3.92E+03

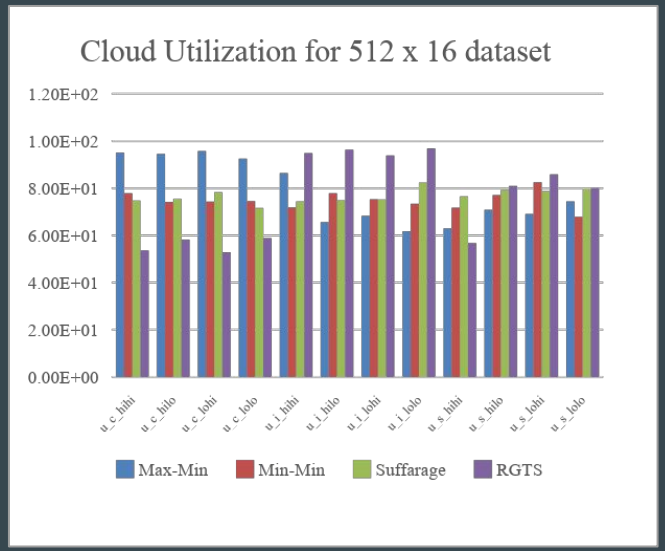
Makespan for 512 x 16 dataset

Instance	Max-Min	Min-Min	Suffrage	RGTS
u_c_hihi	9.51E+01	7.78E+01	7.47E+01	5.36E+01
u_c_hilo	9.45E+01	7.42E+01	7.55E+01	5.82E+01
u_c_lohi	9.57E+01	7.43E+01	7.83E+01	5.28E+01
u_c_lolo	9.25E+01	7.45E+01	7.16E+01	5.88E+01
u_i_hihi	8.64E+01	7.19E+01	7.44E+01	9.48E+01
u_i_hilo	6.56E+01	7.79E+01	7.48E+01	9.63E+01
u_i_lohi	6.83E+01	7.53E+01	7.52E+01	9.38E+01
u_i_lolo	6.17E+01	7.34E+01	8.24E+01	9.69E+01
u_s_hihi	6.30E+01	7.17E+01	7.65E+01	5.67E+01
u_s_hilo	7.08E+01	7.71E+01	7.93E+01	8.10E+01
u_s_lohi	6.91E+01	8.25E+01	7.87E+01	8.59E+01
u_s_lolo	7.44E+01	6.78E+01	7.96E+01	8.01E+01

Cloud utilization for 512 x 16 dataset

Benchmark Results for Braun et al's dataset





Conclusion

- In synthetic datasets, RTGS delivers 50% better results for makespan and cloud utilization in select scenarios.
- In Braun et al's standardized 1024 x 32 benchmark dataset, it shows a 50% better makespan result comparison and a 65% better result in the case of cloud utilization in select scenarios.
- In Braun et al's standardized 512 x 16 benchmark dataset, it shows a 65% better makespan result comparison and a 70% better result in the case of cloud utilization in select scenarios.