# This report is concerned with a literature survey on paper (1) [1], paper (2) [2] and paper (3) [3]. All of the papers focus on management of resources on cloud environment where multi-stage jobs requiring service from multiple resources are considered as workloads. The job requests are characterized by end-to-end SLA [10] with an earliest start time, execution time and soft deadline specified by the user. *MapReduce* [4] is a programming model proposed by Google which requires multiple stages of execution. Devising effective matchmaking and scheduling approaches in a cloud environment considering *MapReduce* jobs with *SLA* as workloads is the common focus of attention in the all three papers. In the following paragraphs, common background information applicable to paper (1), paper (2) and paper (3) is discussed first, the individual contribution of the papers are identified next and the report concludes with outlining the observations and insight gained from the research papers.

A cloud environment typically deals with two kind of requests: *OD* (On Demand) requests which are served on best effort basis and *AR*(Advance Reservation) requests which are associated with *SLA* comprising earliest start time, execution time and deadline specified by user. Because of the constraints applied by the *SLA* on the *AR* requests, research typically focuses on devising effective resource management techniques considering *AR* requests as workloads .To serve the requests, a cloud resource manager needs to perform two important operations: *matchmaking* and *scheduling*. *Matchmaking* is the process of assigning resource or resources from the resource pool to an incoming request. Once a number of requests are assigned to a specific resource, *scheduling* algorithm determines the order in which the requests will be executed. Both *matchmaking* and *scheduling* are computationally hard problems because they not only need to comply with the users requirement captured in *SLA* but also need to achieve desired system objectives. Although significant work has been done for conventional jobs on the cloud which requires single resource(single-stage workflow) ,comparatively less research exists considering requests which requires processing from multiple resources (multi-stage workflow) with SLAs. The research presented in paper (1), paper (2) and paper (3) approaches the problem of devising effecting matchmaking and scheduling techniques for multi-stage workflow requests by considering *MapReduce* requests with SLAs. *MapReduce* programming model is proposed by **Google** and has two key phases : a *map* phase and a *reduce* phase. The map phase generates intermediate key/value pairs from set of input key/value pairs and pass it to the *reduce* phase. The *reduce* phase processes these intermediate key/value pairs and produces meaningful output. Typically, a MapReduce task contains a set of *map* tasks and a set of *reduce* tasks. A similar implantation like *Hadoop* (Open source *MapReduce* implementation by apache) has been used to model the systems in the research presented in paper (1), paper (2) and paper (3) where the number of *map* and *reduce* task capacity which can run in parallel, is fixed by a number.  Lastly the *MapReduce* jobs concerned with all of three researches have *soft deadline*s which implies the requests are allowed to miss their deadline though the prime objective is to minimize the number of such late jobs.

First we discuss paper(1) [1] titled "***Engineering Resource Management Middleware for Optimizing the Performance of Clouds Processing MapReduce Jobs with Deadlines***". In this paper a resource management middleware is engineered that can effectively perform matchmaking and scheduling on a closed system running a fixed number of *MapReduce* jobs. With an objective of reducing resource management overhead and achieving high system performance , the problem of *matchmaking* and *scheduling* are formulated using *MILP* (Mixed Integer Linear Programming) and *CP* (Constraint Programming). Three approaches are suggested: 1. *MILP* model implemented and solved using LINGO [5] 2. *CP* model implemented using MiniZinc/FlatZinc [6]and solved using Gecode [7] 3. *CP* model implemented and solved using IBM ILOG CPLEX Optimization Studio (CPLEX) [8]. The *MILP* model uses a time indexed formulation [11]. The input to the *MILP* and *CP* model is comprised of a set of resources *R* on which to execute a set of jobs *J*, and that a set *T* contains all tasks of all jobs in *J*. Both of the model has some decision variable defined and some constraint applied. The expression of constraints in *CP* model are different from that of *MILP* model but serves the same role. Both of the models have a common objective function which states that the number of jobs which miss their deadlines should be minimized. It has been identified in the paper that formulation of constraints in the *MILP* model is less intuitive and requires complex mathematical formulas in comparison to that of the *CP* model.

The performance of three approaches are compared using experiments with different workloads and parameters .The performance metrics used are: completion time, processing time, number of jobs that miss their deadlines, and size of workload (number of tasks) that the approach could successfully handle. The primary objective is to minimize the number of jobs missing their deadline. The secondary objective is to minimize the completion time. Various systems and workload parameters are used such as number of jobs, earliest start time of jobs, deadline of jobs, number of map and reduce tasks, task execution times, number of resources, and capacity of reduce and map slots. Both *MILP* or *CP* model generates optimal solutions, and therefore all three approaches are able to produce optimal output schedules with regards to minimizing the number of jobs missing their deadlines. The experiment results show that *approach 3* achieves the lowest processing time compared to *approach 1* and *approach 2*, however, *approach 3* produces the highest completion times. The reason being the solver does not prioritize on the minimization of completion time*.* Additionally, *approach 3* was the only approach that was able to handle larger workloads. The research concluded that *approach 1* and *approach 2* are most useful to systems with small workloads and are offline (Processing time is of less concern) and *approach 3* would be useful to use within the implementation of a resource manager that performs matchmaking and scheduling of an open stream of *MapReduce* jobs with end-to-end *SLAs*. The inclusion of more advanced features such as data locality and speculative execution and experiment with real workload in a more complex environment are stated as future work direction.

Next we discuss paper (3) [3] titled " ***A Constraint Programming-Based Resource Management Technique for Processing MapReduce Jobs with SLAs on Clouds***". This research is motivated by the superiority of *CP* based approach (in terms of the lower processing overhead and ability to handle larger workload) identified in paper (1). In case of the research presented in paper (1), a closed system with fixed number of *MapReduce* jobs is considered whereas paper(3) considers an open system with a stream of job arrivals. The main contribution of paper(3) includes: **1)** CP formulation and implementation using IBM ILOG CPLEX Optimization Studio [5], 2) CP-based resource management algorithms (MRCP-RM algorithm) and *MapReduce* constraint programming based resource manager (MRCP-RM).

A high level overview of MRCP-RM goes as follows: 1) User submits *MapReduce* jobs to the system. 2) Jobs are placed in a job queue. 3)If MRCP-RM is not busy and there are jobs in the queue then MRCP-RM will map and schedule jobs to the computing environment/resource. Mapping and scheduling is done by invoking MRCP-RM algorithm which incrementally builds on previous solution (if available) by generating an *OPL*[9] model with new constraints added for each of the tasks that have started but not completed executing, 4) New *OPL* model is then solved by CPLEX’s CP Optimizer to create an updated task to resource mapping and scheduling

The research identified that separating matchmaking and scheduling process in the MRCP-RM results in performance improvement by reducing time required by the CP optimizer to generate and solve the *OPL* model. Another performance improvement mechanism was implemented to start matchmaking and scheduling jobs only when their earliest start timehave arrived, or are close to arriving. This reduces the number of tasks that MRCP-RM has to remap and reschedule each time it is run

For the experimentation and performance evaluation, paper (3) compares MRCP-RM to MinEDF-WC [10] using simulation. The focus is on the relative performance of MRCP-RM in comparison to MinEDF-WC. The performance is measured using factor-at-a-time experiments where one parameter is varied and other parameters are kept at their default values. The performance metrics considered are processing overhead, average job turnaround time, number of late jobs, percentage of jobs that are late. Different job ordering strategies used such as order by job id, earliest deadline first, and least laxity did not make a significant difference. Various system and workload parameters are used such as number of map and reduce tasks, task execution times, earliest start time of jobs, deadline of jobs, arrival rate of jobs, and number of resources and capacity. Synthetic Facebook workloads are used in the experiments. Overall experiment results show that with comparison to MinEDF-WC, MRCP-RM can efficiently perform matchmaking and scheduling with lower average job turnaround time, and smaller percentage of late jobs. Also, MRCP-RM has a small overhead relative to overhead time. Future research directives include experiment using systems with additional resources, considering monetary costs for resource usage, reduce matchmaking and scheduling times when the arrival rate is high, and handling more complex workflows with user-specified precedence relationships.

Lastly, we discuss paper (3) [3] titled "***Resource Management Techniques for Handling Requests with Service Level Agreements***". A novel budget based resource management technique for handling *MapReduce* jobs with SLAs on a system subject to open stream of request arrivals is introduced in this paper. The proposed *MapReduce* Budget-based Resource Manager (MRBB-RM) uses three algorithms: **1**) Job and task mapping algorithm, **2**) Deadline budgeting algorithm, **3**) Job remapping algorithm. *Algorithm 1* is invoked first. *Algorithm 1*’s first step is to invoke *Algorithm 2*, which decomposes job *j’*s deadline in to components and gives each of *j*’s tasks a sub-deadline. After each task get a sub deadline , *Algorithm 1* continues by mapping the tasks on to the resources. Lastly, if *j* is not able to be mapped before its deadline, *Algorithm 3* is invoked to remap *j* and set of jobs which caused *j* to miss it's deadline. *Algorithm 3*  is invoked twice in the implementation to minimize the number of jobs missing their deadline.

Also, another point to note that recourse management algorithms perform matchmaking and scheduling in a single step (termed as mapping).

MRBB-RM is evaluated by comparing it with MinEDF-WC [10]. Following performance metrics are used: average job matchmaking and scheduling time, percentage of late jobs and average job turnaround time. The performance is measured using factor-at-a-time experiments. Various system and workload parameters used are arrival rate, earliest start time, deadline, number of map and reduce tasks, execution time of map and reduce tasks, number of resources and capacity. Synthetic Facebook workloads are used in the experiments. The evaluations show that MRBB-RM outperformed MinEDF-WC in terms of turnaround time in all cases, and had a lower or comparable percentage of late job for lower arrival rate. MRBB-RM was able to generate high quality mapping in terms of turnaround time and percentage of late jobs while incurring very low overhead. MRBB-RM proved to have a small overhead relative to job turnaround time which shows the effectiveness of MRBB-RM. Extension of the experiments with real systems, and extension of the algorithms to work with jobs that have more than two stages and tasks that are characterized by precedence relationships are stated as future directive.

Devising effective resource management technique in a cloud environment considering multi stage jobs (*MapReduce* jobs) with SLAs as workload is the common concern for all the papers. Paper (1) and paper (3) approaches the problem as an optimization problem by devising *LP* and *CP* models producing optimal output. The insight gained from paper (1) about the superiority of CP based approach motivated paper (3) to devise a CP based model for an open system. Paper (2) introduces a novel deadline budgeting algorithm which produces high quality mapping with low overhead. Paper (2) approaches the *matchmaking* and *scheduling* in single step whereas paper(3) separates the process. It is to be noted that, all of the researches use synthetic workload and simulated experiments. More insights could be gained if real workloads are used. Also it would be interesting to see the Resource Allocation vs. Resource Consumption trend for such open systems using the proposed strategies with heavy workloads.

**References**

1. Lim, N, Majumdar, S., Ashwood-Smith, P., “Engineering Resource Management

Middleware for Optimizing the Performance of Clouds Processing MapReduce Jobs with

Deadlines”, Proc. 5th ACM/SPEC International Conference on Performance Engineering

(ICPE), Dublin, March 2014.

1. Lim, N., Majumdar, S., Ashwood-Smith, P., “Resource Management Techniques for

Handling Requests with Service Level Agreements, Proc. 2014 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'14),

Monterey, July 2014.

1. Lim, N, Majumdar, S., Ashwood-Smith, P., “A Constraint Programming-Based Resource

Management Technique for Processing MapReduce Jobs with SLAs on Clouds”, Proc. 2014.International Conference on Parallel Processing (ICPP-2014).

1. Dean, J. and Ghemawat, S. 2004. MapReduce: Simplified data processing on large clusters. *International Symposium* *on Operating System Design and Implementation* (December2004). 137–150.
2. Lindo Systems Inc. Lindo Systems – Optimization Software. Available: http://www.lindo.com/.
3. [6] NICTA. MiniZinc and FlatZinc. Available: http://www.MiniZinc.org/.
4. Gecode. Generic Constraint Development Environment. Available: http://www.gecode.org/.
5. IBM. IBM ILOG CPLEX Optimization Studio. Available: http://www-03.ibm.com/software/products/us/en/ibmilogcpleoptistud
6. IBM. 2009. IBM ILOG OPL Language Reference Manual. White Paper. IBM Corporation (2009).
7. A. Verma, L. Cherkasova, V.S. Kumar, and R.H. Campbell, “Deadline based workload management for MapReduce environments: Pieces of the performance puzzle,” Network Operations and Management Symposium, pp.900-905, 16-20 April 2012
8. Van den Akker, J. M., Hurkens, C., and Savelsbergh, M. 2000. Time-indexed formulations for machine scheduling problems: Column generation. INFORMS Journal on Computing. 12.2 (2000). 111-124