|  |
| --- |
| **SYSC 5207 – Distributed Systems Engineering** |
| Fall 2014      Project Report(Part 1) on    ***Resource Management Techniques for Handling Errors in User***  ***Estimated Job Execution Times***  Phuong Hoang; Majumdar, S.; Zaman, M.; Srivastava, P.; Gael, N., "Resource management techniques for handling uncertainties in user estimated job execution times," Performance Evaluation of Computer and Telecommunication Systems (*SPECTS 2014*), International Symposium on , vol., no., pp.626,633, 6-10 July 2014  Submitted To  Dr. Shikharesh Majumdar  By  **Rudraneel chakraborty (7563772, 100992549)**  **Ferhan Jamal (100953487)** |

The main research focus of this paper is to improve system performance of a cloud or grid environment by effectively handling the problem of erroneous user estimated run times of requests. A new concept *SAR* (Soft Advance Reservation Request) has been introduced and two *SAR* based algorithms are proposed to handle the errors associated with user estimated job execution times.

The requests submitted to the cloud environment are categorized into two types: *OD* (On Demand) requests with no strict deadlines and *AR* (Advance Reservation) requests which are associated with an earliest start time, user estimation of job execution time and a deadline for the job completion. This research considers requests to be *AR* requests. A number of studies [1, 2] identified that user estimation of job execution times are highly inaccurate. Overestimation and underestimation of job execution time effects the matchmaker performance [2, 3] greatly. The concept of *SAR* introduced in this paper handles the problem of erroneous user estimation where the strict requirement for all the *AR* requests to meet their deadlines is relaxed and a predetermined proportion of *AR* requests are allowed to miss their deadlines. Two proposed *SAR* based algorithms aims to artificially reduce the user estimated job execution times by considering previous history of user requests submitted to the system, and take matchmaking/scheduling decisions based on these artificially reduced estimates of job execution times. The following matchmaking process comprised of six steps is applied upon request arrival

* **Step 1**: The matchmaker decides whether the user estimate of request runtime or a system generated estimate of runtime for the request should be used for matchmaking.
* **Step 2**: Go to step 3 if the system generated runtime is used. Go to step 4If the user estimated runtime is used.
* **Step 3**: The system generated estimate of runtime of the request is calculated using the information from previous requests.
* **Step 4**: *AS* criterion [4] is used to determine the resources that can serve the request. If the AS criterion is satisfied on a resource it implies that the request can be accepted and allocated to the resource.
* **Step 5**: An allocation algorithm is used to select one resource If multiple candidate resources satisfy *AS* criterion
* **Step 6**: If none of the resources can satisfy the *AS* criterion, the request is rejected.

Among the two algorithms, *Algorithm 1* aborts an executing request if its execution time t elapsed on the system has exceeded the user provided estimate. With *Algorithm 2*, a request with an underestimated runtime (irrespective of user estimated or system generated) is not aborted. Instead, an unfinished request is preempted and allowed to run later when there is a free time slot in the resource schedule until the request is completed.

The simulation results produced by the research shows significant performance improvement using the proposed strategy. Used metrics are : arrival rate, laxity(*L*), request-runtime(*S*), proportion of allowable missed deadline (*P­miss*), window size(*W*),margin and number of resources(*R*). The two proposed algorithms are compared with a system with no error(*NE*) which is the ideal case and a system with no error handling technique , thus no modification of user estimated run time(*NM*) . In all of the cases a significant performance improvement is observed using the proposed *algorithm 1* and *algorithm 2* over *NM*. *Algorithm1* and *algorithm 2* performs similarly for all cases except for the high values of *ʎ*, *S* and *L* where *algorithm 2* performs better which signifies benefit of avoiding job abortions at higher values of these parameters. Useful utilization *UU* is seen to increase with increasing values of *ʎ* and *L*. On the other hand , probability of blocking *Pb* increases with higher *ʎ* values and decreases with higher values of *L*. Increasing values of margin results in deteriorating performance at higher value for both of the proposed algorithms but remains unchanged initially, which suggests to use lower values of margin for performance improvement. In case of increasing the number of resource *R*, the performance seems to improve because of reduced contention , but a knee point is observed. Increasing the value beyond the knee point does not produce performance improvement. Additionally, changing local scheduling policy or allocation algorithm doesn't have significant affect on performance improvement for the proposed algorithms. The scope of using other matchmaking algorithms and real workloads are addressed as future research direction.

To conclude , real systems are subject to both *OD* and *AR* requests. Also the use of *AS* *criterion* in this research gives the freedom to use both transparent and opaque resources which results in improvement in revenue rate and rate of accepted jobs[4] .A more realistic and broader view can be investigated if both *AR* and *OD* requests are considered and both transparent and opaque resources are included in the resource pool.