International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 4| ISSN: 2456-3307

Cloud Workflow Arrangement with Deadline and Point Period Accessibility

Aravind[1]

[1]MCA, Sri Padmavathi College Of Computer Sciences & Technology, Tiruchanoor, Andhra Pradesh, India

ABSTRACT

Nowadays much attention has been paid on workflow scheduling in service computing environments (cloud computing, grid computing, Web services, etc). Resources are generally provided in the form of services, especially in cloud computing. Allocating service capacities in cloud computing is based on the idea that they're unlimited and may be used atany time. However, available service capacities change with workload and can't satisfy users' requests at any time from the cloudprovider's perspective because cloud services are shared by multiple tasks. Cloud service suppliers provide available time slots for new user's requests based on available capacities. during this paper, we tend to consider workflow scheduling with deadline and time slot availability in cloud computing. An iterated heuristic framework is given for the problem under study that mainly consists ofinitial solution construction, improvement, and perturbation. 3 initial solution construction methods, 2 greedy- and fair-based improvement methods and a perturbation strategy are proposed. Totally different methods within the 3 phases end in many heuristics. Experimental results show that different initial solution and improvement strategies have different effects on solution qualities.

Keywords : Workflow, Scheduling, Time slots, Cloud Computing

I. INTRODUCTION

Schedule Compaction and Deadline Constrained DAG Scheduling 17 In IaaS cloud environment, resources are served in a pay-per-use model and provisioned dynamically. Therefore, resources are provisioned on demand. To satisfy different QoS requirement of users, resource providers offer heterogeneous resources with various processing capabilities and prices. Usually, fast resource costs more money than that of slow resource. Thus, the cost/time trade-off problem becomes a hot topic in the literature. Ideally, users want to run their applications as fast as possible with minimum cost. If applications are not timecritical, a little delay can be tolerated for cost saving. There are already algorithms concerning both time and monetary cost for cloud computing environment. However, most of them thought that leased resources can be completely utilized, and they are charged in an ideal pay-asyou-go model. This is not the case in real production system. As the Amazon EC2 cloud for example, they charge resources by hour. If a resource is terminated before one hour, the cost is still rounded Up to one hour.

There are two common ways for service delivery: (i) an entire application as a service, which can be directly used with no change. (ii) Basic services are combined to build complex applications, e.g., Xignite and

StrikeIron offer Web services hosted on a cloud on a pay-per-use basis . Among a large number of services in cloud computing, there are many services which have same functions and supplied by different cloud service providers (CSPs). However, these services have different non-functional properties. Basic services are rented by users for their complex applications with various resource requirements which are usually modeled as workflows. Better services imply higher costs. Services are consumed based on Service-Level Agreements, which define parameters of Quality of Service in terms of the pay-per-use policy. Though there are many parameters or constraints involved in practical workflow scheduling settings, deadline and time slot are two crucial ones in cloud computing, a new market oriented business model, which offers high quality and low cost information services . However, the two constraints have been considered separately in existing researches. It is necessary to consider both of the constraints jointly because: (i) Deadlines of the workflow applications need to be met. (ii) Unreserved time slots is crucial for resource utilization from the perspective of service providers. (iii) Utilization of time slots in reserved intervals should be improved to avoid renting new resources (saving money).

In this paper, we consider the workflow scheduling problem with deadlines and time slot availability (WSDT for short) in cloud computing. To the best of our knowledge, the considered problem has not been studied yet. Service capacities are usually regarded to be unlimited in cloud computing, which can be used at any time. However, from the CSP's perspective, service capacities are not unlimited. Available service capacities change with workloads, i.e, they cannot satisfy user's requests at any time when a cloud service is shared by multiple tasks. Only some available time slots are provided for new coming users by CSPs in terms of their remaining capacities. For example, each activity in Figure 1 has different candidate services with various execution times, costs and available time slots. For activity 4, there are two candidate services with different workloads. If service 0 is selected for activity 4, the execution time is 4 with the price 6 and available time slots [0, 4) S [9, 14). Time slot [4, 9) is unavailable because there is no remaining capacity.

II. ALGORITHM

The service assignment for each activity in the WSDT depends on both finish times of all predecessors and available time slots of the service. In this paper, an Iterated Local Adjusting Heuristic framework (ILAH) is proposed for the problem under study. ILAH consists of four components: Time Slot Filtering, Initial Solution Construction, Solution Improvement and Perturbation. ILAH starts from an initial solution π. Improving and perturbing operations are performed on π iteratively until the termination criterion is satisfied. The termination criterion is set as α, the number of consecutive iterations without improvement. Let C(π) be the total cost of π.

Algorithm 1: ILAH

```
begin
    Time Slot Filtering;
    Generate the initial solution π by an initial solution
    construction strategy;
    π_best ← π, C(π_best) ← C(π);
    while (termination criterion not met) do
        π ← Improve(π);
        if (C(π_best) > C(π)) then
            π_best ← π, C(π_best) ← C(π);
        Perturbation(π);
    return π_best.
```

Algorithm 2: Time Slot Filtering

Time Slot Filtering Though there are many available time slots, not all of them meet requirements of activities of workflow instances. Some available time slots might not be available for an activity vi even before the service assignment. For example, Ef t(n)>D in the fastest schedule, or the duration of a time slot is less than the execution time of the activity, or the start or finish time is beyond the earliest start or the latest finish time of the activity. By filtering out all impossible time slots, remaining time slots are eligible for activities of the instance, which make workflow scheduling much more efficient.

begin
  for *(each* $v_i \in V$*)* **do**
    Calculate $E_{st}(i)$, $E_{ft}(i)$, $L_{ft}(i)$, $L_{st}(i)$ using equations (8), (9), (10), (11);
    if $(E_{ft}(n) > D)$ **then**
      return *NULL*;
      /* infeasible problem */
  for *(each* $v_i \in V$*)* **do**
    for *(each service* $M_i^j \in \mathbb{M}_i$*)* **do**
      for $k = 0$ **to** $\mathcal{N}_{ij}^s - 1$ **do**
        if $F_{ijk} - B_{i,j,k} < e_{ik}$ *or* $B_{i,j,k} > D$ *or* $B_{ijk} > L_{ft}(i)$ *or* $F_{ijk} < E_{st}(i)$ **then**
          Remove s_{ijk} from $\mathbb{S}_{ij}$;
      if $(\mathcal{N}_{ij}^s = 0)$ **then**
        Remove M_i^j from $\mathbb{M}_i$;
    for *(each* $v_i \in V$*)* **do**
      Generate the service pool $\mathbb{M}_i$ by sorting all candidate services in non-increasing order of costs;
  return $\{\mathbb{M}_i\}$.

Algorithm 3: Greedy Heuristic Algorithm

A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stagewith the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

Input: Initial solution π, the earliest start time of each activity v_i according to π, $est = \{e_{st}(i) | i = 1,...,n\}$.
begin
  $l_{ft}(n) \leftarrow D$, $l_{st}(n) \leftarrow D$, $cost \leftarrow 0$;
  for $i = n-1$ **to** 1 **do**
    Calculate the latest finish time of v_i by $l_{ft}(i) \leftarrow \min_{q \in \mathcal{Q}_i} \{l_{st}(q)\}$;
    Calculate the earliest start time of v_i by $e_{st}(i) \leftarrow \max_{p_i \in \mathcal{P}_i} \{e_{ft}(p_i)\}$;
    Compute $R_{ijj'}^D$ for all candidate available services in $[e_{st}(i), l_{ft}(i)]$ using Equation (15);
    Select the available service $M_i^{j^*}$ with $R_{ijj^*}^D = \max\{R_{ijj'}^D\}$ to replace the current service M_i^j;
    Update the element (i, M_i^j) of π with $(i, M_i^{j^*})$;
    Update $l_{st}(i)$ according to $M_i^{j^*}$;
  return π.

Fair Improvement Heuristic (FIH)

The GIH tries to decrease the cost by assigning a cheaper service that has the maximum Cost Decrease Ratio for each vi . Though the GIH can reduce the total cost, it might reduce the number of cheaper available services of its predecessors and resulted in inferior solutions. For this reason, we presented the Fair Improvement Heuristic (FIH) rule. Instead of substituting Mji of adjustable activity vi with the cheapest available service with the maximum Cost Decrease Ratio, the FIH simply selects the second cheapest available service in [est.(i),lf t(i)] for the substitution. The process is iterated until there is no substitution in iteration. The FIH is formally described. Similar to the analysis process of the GIH.

Input: Initial solution π, the earliest start time of each activity v_i according to π, $est = \{e_{st}(i) | i = 1,...,n\}$.
begin
  $l_{ft}(n) \leftarrow D$, $l_{st}(n) \leftarrow D$, $cost \leftarrow 0$, $flag \leftarrow TRUE$;
  while $flag$ **do**
    $flag \leftarrow FALSE$;
    for $i = n-1$ **to** 1 **do**
      Calculate the latest finish time of v_i by $l_{ft}(i) \leftarrow \min_{q \in \mathcal{Q}_i} \{l_{st}(q)\}$;
      Calculate the earliest start time of v_i by $e_{st}(i) \leftarrow \max_{p_i \in \mathcal{P}_i} \{e_{ft}(p_i)\}$;
      Substitute M_i^j with the second cheapest available service $M_i^{j'}$ in $[e_{st}(i), l_{ft}(i)]$ for v_i;
      Update the element (i, M_i^j) of π with $(i, M_i^{j'})$;
      Update $l_{st}(i)$ according to $M_i^{j'}$;
      $flag \leftarrow TRUE$;
  return π.

Algorithm 4: Maximum Cost Ascending Ratio First

Input: Temporal parameter sets E_{st}, E_{ft}, L_{st}, and L_{ft} for all activities of the considered workflow application calculated by the Time Slot Filtering.
begin
  $S \leftarrow \{1, n\}$, $U \leftarrow \{2,...,n-1\}$, $M \leftarrow \varnothing$;
  $\pi \leftarrow \{(1, M_1^0), (n, M_n^0)\}$;
  while $(|U| > 0)$ **do**
    for $i = 0$ **to** $|U| - 1$ **do**
      Calculate R_i^I of activity v_i according to Equation (14);
      Record the service M_i^j corresponding to R_i^I;
      $M \leftarrow M \cup \{(i, M_i^j)\}$;
    $k \leftarrow \text{argmax}_{i \in U}\{R_i^I\}$;
    $\pi \leftarrow \pi \cup \{(k, M_k^{j'})\}$;
    /* The element $(k, M_k^{j'}) \in M$ */
    $U \leftarrow U / \{k\}$, $S \leftarrow S \cup \{k\}$;
    for *each* $i \in U$ **do**
      Update $E_{ft}(i)$, $E_{st}(i)$ and $L_{ft}(i)$, $L_{st}(i)$;
  return π.

III. CONCLUSION

We have considered workflow programming with deadline and time slots constraints in cloud computing to reduce total costs. the problem was modeled because the WSDT that is more sensible than the DTCTP. We tend to tried that the WSDT had completely different properties from the DTCTP. The ILAH (iterated local

adjusting heuristic) framework was planned for the NP-hard WSDT. 3 initial answer construction ways were developed among that the MCARF and therefore the MACF Showed more practical than the EFTF on initial answer construction. 2 improvement ways, the FIH and therefore the GIH, were introduced that had similar influences on the solution improvement. The FIH was very effective for up poor solutions. By integrating the worst and best initial solution construction ways (EFTF and MCARF) with the two improvement ways, four ILAH-based algorithms were developed. Though the EFTF was the worst initial solution construction strategy, it had been strange that the EFIG showed the best performance. However, the EGIH obtained the worst performance. Additionally, the EFTF wasn't sensitive to instance parameters whereas the EGIH was affected by most of the parameters

IV. REFERENCES

[1]. R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on. IEEE, 2008, pp. 5–13.

[2]. E. L. Demeulemeester, W. S. Herroelen, and S. E. Elmaghraby, "Optimal procedures for the discrete time/cost trade-off problem in project networks," European Journal of Operational Research, vol. 88, no. 1, pp. 50–68, 1996.

[3]. X. Zhang, L. Yang, C. Liu, and J. Chen, "A scalable two-phase top-down specialization approach for data anonymization using map reduce on cloud," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, pp. 363–373, 2014.

[4]. M. Menzel, R. Ranjan, L. Wang, S. Khan, and J. Chen, "Cloud genius: A hybrid decision support method for automating the migration of web application clusters to public clouds," IEEE Transactions on Computers, vol. 64, no. 5, pp. 1336–1348, 2015.

[5]. W. Dou, X. Zhang, J. Liu, and J. Chen, "Hire some-ii: Towards privacy aware cross-cloud service composition for big data applications," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 2, pp. 455–466, 2015.

[6]. C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "Mur-dpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," IEEE Transactions on Computers, vol. 64, no. 9, pp. 2609–2622, 2015.

[7]. A. Verma and S. Kaushal, "Deadline constraint heuristic–based genetic algorithm for workflow scheduling in cloud," International Journal of Grid and Utility Computing, vol. 5, no. 2, pp. 96–106, 2014.

[8]. S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," Future Generation Computer Systems, vol. 29, no. 1, pp. 158–169, 2013.

[9]. S. Abrishami and M. Naghibzadeh, "Deadline-constrained workflow scheduling in software as a service cloud," Scientia Iranica, vol. 19, no. 3, pp. 680–689, 2012.

[10]. A. G. Delavar and Y. Aryan, "Hsga: a hybrid heuristic algorithm for workflow scheduling in cloud systems," Cluster computing, vol. 17, no. 1, pp. 129–137, 2014.

[11]. C. Akkan, A. Drexl, and A. Kimms, "Network decomposition-based benchmark results for the discrete time–cost tradeoff problem," European Journal of Operational Research, vol. 165, no. 2, pp. 339–358, 2005.

[12]. P. De, E. J. Dunne, J. B. Ghosh, and C. E. Wells, "Complexity of the discrete time-cost tradeoff problem for project networks," Operations Research, vol. 45, no. 2, pp. 302–306, 1997.

[13]. T. J. Hindelang and J. F. Muth, "A dynamic programming algorithm for decision CPM networks," Operations Research, vol. 27, no. 2, pp. 225–241, 1979.

[14]. J. Yu, R. Buyya, and C. K. Tham, "Cost-based scheduling of scientific workflow applications on utility grids," in e-Science and Grid Computing, 2005. First International Conference on. IEEE, 2005, pp. 8–pp.

[15]. Y. Yuan, X. Li, Q. Wang, and X. Zhu, "Deadline division-based heuristic for cost optimization in workflow scheduling," Information Sciences, vol. 179, no. 15, pp. 2562–2575, 2009.