Dynamic and Fault-Tolerant Clustering For Scientific Workflows

Monish.M.Alapatt[1], S.Chinnathambi[2], Mohan Prasad[3], P. Madhavan[4]

[1, 2, 3] Student, Department Of CSE, Sri Krishna College Of Technology, Coimbatore.

[4] Assistant Professor, Department Of CSE, Sri Krishna College Of Technology, Coimbatore.

Abstract:

Virtualization is being widely used in large-scale computing environments, such as clouds, data centers, and grids, to provide application portability and facilitate resource multiplexing while retaining application separation. The Scientific workflows can be composed of many fine computational granularity tasks. Task clustering has proven to be an effective method to reduce execution overhead and to improve the computational granularity of scientific workflow tasks executing on distributed resources. The existing clusters can be used to manage the massive number of requests that a job will receive, so as to meet the rapid development of Internet application. However, a job composed of multiple tasks may have a higher risk of suffering from failures than a single task job. The proposed system to overcome the job failures under Fault-Tolerant Clustering for Scientific Workflows. The proposed system defined imbalance metrics to quantitative measure workflow. To improve the performance of task clustering. Propose a general task failure modelling framework that uses a Maximum Likelihood estimation-based parameter estimation process to model workflow performance.

Introduction:

Cloud computing is one of the most promising and valuable research directions following utility computing, grid computing and distributed computing currently. It provides services of infrastructure, platform and software for users, and supplies the on-demand services to users through Internet. Infrastructure as a Service (IaaS) is the basis.

Literature Survey:

[1] Cloud computing is revolutionizing the IT industry by enabling them to offer access to their infrastructure and application services on a subscription basis. As a result, several enterprises including IBM, Microsoft, Google, and Amazon have started to offer different Cloud services to their customers. Due to the vast diversity in the available Cloud services, from the whose services they should use and what the basis for their selection is. Currently, there is no framework that can allow customers to evaluate Cloud offerings and rank them based requirements. In this work, we propose a framework and a mechanism that measure the quality and prioritize Cloud services. Such a framework can make a significant impact and will create healthy competition among Cloud providers to satisfy their Service Level Agreement (SLA) and improve

their QoS. We have shown the applicability of the ranking framework using a case study.

[2] with the rapid penetration of mobile devices, more users prefer to watch multimedia live-streaming via their mobile terminals. Quality of service provision is normally a critical challenge in such multimedia sharing environments. In this article, we propose a new cloud-based WMSN to efficiently deal with multimedia sharing and distribution. We first motivate the use of cloud computing and social contexts in sharing live streaming. Then our WMSN architecture is presented with the description of the different components of the network. After that, we focus on distributed resource management and formulate the bandwidth allocation problem in a game theoretical framework that is further implemented in a distributed manner. In addition, we note the potential selfish behaviour of mobile users for resource competition and propose a cheat-proof mechanism to motivate mobile users to share bandwidth. Illustrative results demonstrate the best responses of different users in the game equilibrium as well as the effectiveness of the proposed cheating avoidance scheme. live virtual machine migration policy using Bayes

[3] Green cloud data center has become a research hotspotof virtualized loud computing architecture. Since live virtual machine (VM) migration technology is widely used and studied in cloud computing, we have focused on the VM placement selection of live migration for power saving. We present a novel heuristic approach which is called PS-ABC. Its algorithm includes two parts. One is that it combines the artificial bee colony (ABC) idea with the uniform random initialization idea, the binary search idea, and Boltzmann selection policy to achieve an improved ABC-based approach with better
The other one is that it uses the Bayes theorem to further optimize the improved ABC-based process to faster get the final optimal solution. As a result, the whole approach achieves a longer-term efficient optimization for power saving. The experimental results demonstrate that PS-ABC evidently reduces the total incremental power consumption and better protects the performance of VM running and migrating compared with the existing research. It makes the result of live VM migration more high-effective and meaningful.virtual machine migration for energy-saving in cloud

[4] the field of live VM (virtual machine) migration has been a hotspot problem in green cloud computing. Live VM migration problem is divided into two research aspects: live VM migration mechanism and live VM migration policy. In the meanwhile, with the development of energy-aware computing, we have focused on the VM placement selection of live migration, namely live VM migration policy for energy saving. In this paper, a novel heuristic approach PS-ES is presented. Its main idea includes two parts. One is that it combines the PSO (particle swarm optimization) idea with the SA (simulated annealing) idea to achieve an improved PSO-based approach with the better global search's ability. The other one is that it uses the Probability Theory and Mathematical Statistics and once again utilizes the SA idea to deal with the data obtained from the improved PSO-based process to get the final solution. And thus the whole approach achieves a long-term optimization for energy saving as it has considered not only the optimization of the current problem scenario but also that of the future problem. The

experimental results demonstrate that PS-ES evidently reduces the total incremental energy consumption and better protects the performance of VM running and migrating compared with randomly migrating and optimally migrating. As a result, the proposed PS-ES approach has capabilities to make the result of live VM migration events more high-effective and valuable load balance

[5] The bandwidth of network has been more developed than the speed of processor and EMS memory accessing in view of the development of network technology. More and more bottlenecks appear on the server. The proliferation of Internet portfolio and specialization of network application make load balance to be instant demand. The best way to solve this problem is the technique of network cluster and load balance based on Linux Virtual Server. This paper discusses the work principle and Server/Client way of Linux Virtual Server, which provides a transparent way to extend network throughput and strengthen data process ability and improve network agility and usability. Linux Virtual Server could realize load balance conveniently and effectively by operation distributing to cluster real server according to the connection of transmission lever. This paper also describes load balance and balance strategy of LVS to make cluster technology according with the application demands in detail.

3.1 Existing Work:

The existing works on workflow scheduling in the context of clouds are either on deadline or cost optimization, ignoring the necessity for robustness. Failures are inevitable in such large complex distributed systems. Task-level monitoring system, which retries a job if a task failure is detected. Task clustering methods to improve the performance and reliability of fine-grained tasks. Clustering algorithm that groups bag of tasks based on the runtime, and later based on task file size, CPU time, and resource constraints. Online scheduling algorithm that merges budget, and application deadline. Adaptive scheduling algorithm to group fine-grained tasks according to the processing capacity and the network bandwidth of the currently available esources.

Disadvantages:

- The scheduling overhead and queuing times at resources are high.
- High network bandwidth consume.
- High resources (CPU power, etc.).
- High failure rates.
- The cost of task retry and of the scheduling overheads.
- Queuing overheads, they do not address the fault tolerance problem.

3.2 Proposed System:

The proposed novel workflow scheduling heuristics that are robust against performance variations and fault-tolerant towards failures. The first method retries failed tasks within a job by extracting them into a new job. The second method dynamically adjusts the granularity or clustering size (number of tasks in a job) according to the estimated inter-arrival time of task failures. The third method partitions the clustered jobs into finer jobs (i.e., reduces the job granularity) and retries them.

Advantages:

- Data transfer times and failures probability reduced.

- Efficient execution and reduce scheduling overheads.
- Minimize the overall runtime of a workflow. Failures can be observed from the outputs or logs of a job.
- This system focus on transient failures and we assume that after a finite number of retries these jobs can be completed successfully.

3.4 Workflow system model:

A workflow is modelled as a Directed Acyclic Graph-DAG, where each node in the Directed Acyclic Graph often represents a workflow task, and the edges represent dependencies between the tasks that constrain the order in which tasks are executed. Dependencies typically represent data-flow dependencies in the application, where the output files produced by one task are used as inputs of another task. Each task is a computational program and a set of parameters that need to be executed. The submit host prepares a workflow for execution (clustering, mapping, etc.), and worker nodes, at an execution site, execute jobs individually.

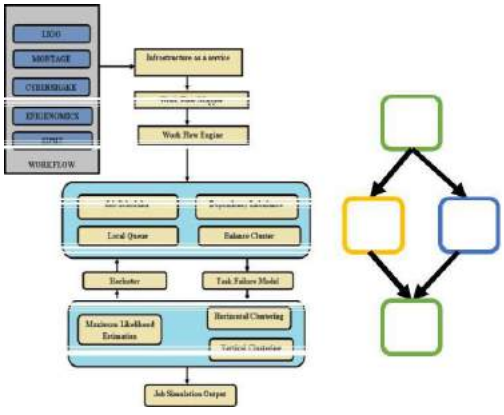

Figure 3.4.1 Structure of DAG and O-DAG

The figure (a) describes for DAG is only perform tasks and there is no additional tasks not added.

Workflow Mapper: generates an executable workflow based on an abstract workflow provided by the user or workflow composition system. It also restructures the workflow to optimize performance and adds tasks for data management and provenance information generation. In this work, workflow mapper is particularly used to merge small tasks together into

A job such that system overheads are reduced, which is called Task Clustering. A job is a single execution unit in the workflow execution systems and it may contain one or more tasks.

Workflow Engine: executes jobs defined by the workflow in order of their dependencies. Only jobs that have all their parent jobs completed are submitted to the Job Scheduler. Workflow Engine relies on the resources (compute, storage, and network) defined in the executable workflow to perform the necessary actions. The time period when a job is free (all of its parents have completed successfully) to when it is submitted to the job scheduler is denoted the workflow engine delay. The workflow engine delay is usually configured by users to assure that the entire workflow scheduling and execution system is not overloaded.

Job Scheduler and Local Queue: manage individual workflow jobs and supervise their execution on local and remote resources. The time period when a task is submitted to the job scheduler to when the task starts its execution in a worker node is denoted the queue delay. It reflects both the efficiency of the job scheduler and the resource availability.

Job Wrapper: extracts tasks from clustered jobs and executes them at the worker nodes. The clustering delay is the elapsed time on the extraction process. Environment that targets a homogeneous computer cluster. The submit host prepares a workflow for execution clustering, mapping, etc.. The jobs are executed remotely on individual worker nodes. a single execution site with multiple compute resources, such as virtual machines on Cloud platforms. Tasks per clustered job (clusters. size), or the number of clustered jobs per horizontal level of the workflow (clusters.num). For simplicity, we set clusters.num to be the same as the amount of available resources. In have evaluated the runtime performance for different clustering granularities. Algorithm 1 shows the pseudo code for Horizontal Clustering. The Clustering and Merge procedures are invoked in the initial task clustering process, while the Re-clustering procedure is invoked when a failed job is detected by the monitoring system.

3.6 Balanced Clustering:

Task clustering has been widely used to address the low performance of very short running tasks on platforms where the system overhead is high, such as distributed computing infrastructures. However, up to now, techniques do not consider the load balance problem. In particular, merging tasks within a workflow level without considering the runtime variance may cause load imbalance (Runtime Imbalance), or merging tasks without considering their data dependencies may lead to data locality problems (Dependency Imbalance). In this section, we introduce metrics that quantitatively capture workflow characteristics to measure runtime and dependence imbalances. We then present methods to handle the load balance problem.

3.5 Fault-Tolerant Clustering

Inappropriate task clustering may negatively impact the workflow make span in faulty distributed environments. The propose three fault-tolerant task clustering methods Selective Re-clustering (SR), Dynamic Re-clustering (DR), and Vertical Re-clustering (VR) that adjust the clustering size (k) of the jobs to reduce the impact of task failures on the workflow execution. These methods are based on the Horizontal Clustering (HC) technique that has been implemented and used in the Pegasus workflow management system (WMS). Horizontal Clustering (HC). Horizontal clustering merges multiple tasks within the same horizontal level of the workflow. The clustering granularity (number of tasks within a cluster) of a clustered job is controlled by the user, who defines either the number of

3.6.1 Balanced clustering methods:

Balanced clustering methods used to improve the runtime and dependency balances in task clustering. The first present the basic runtime-based clustering method, and then two other balancing methods that address the dependency imbalance problem. Horizontal Runtime Balancing (HRB) aims to evenly distribute task runtime among clustered jobs. Tasks with the longest runtime are added to the job with the shortest runtime.

The figure shows an example of HRB where tasks in the first level have different runtimes and should be grouped into two jobs. HRB sorts tasks in decreasing order of runtime, and then adds the task with the highest runtime to the group with the shortest aggregated runtime. Thus, ta and tc, as well

as tb and td are merged together. For simplicity, system overheads are not displayed. the Horizontal Impact Factor Balancing (HIFB) and the Horizontal Distance Balancing (HDB) methods.

The experiments presented hereafter evaluate the performance of our balancing methods in comparison with an existing and effective task clustering strategy named delay aware load balancing using Horizontal and Vertical Clustering (HVC), which is widely used by workflow management systems. We extended the Workflows simulator with the balanced clustering methods and imbalance metrics to simulate a distributed environment where we could evaluate the performance of our methods when varying the average data size and task runtime. The simulated computing platform is composed by 20 single core virtual machines (worker nodes), which is the quota per user of some typical distributed environments.

Two workflows are used in the experiments: LIGO inspiral analysis, and Epigenomics. Both workflows

are generated and varied using the WorkflowGenerator1. LIGO is composed by 400 tasks and its workflow structure is presented; Epigenomics has about 500 tasks and is structured. Runtime (average and task runtime distribution) and overhead (workflow engine delay, queue delay, and network bandwidth) information were collected from real traces production environments then used as input parameters for the simulations.

Montage: Montage is an astronomy application used to construct large image mosaics of the sky. Input images are re-projected onto a sphere and overlap is calculated for each input image. The application re-projects input images to the correct orientation while keeping background emission level constant in all images. Images are added by rectifying them to a common flux scale and background level. Finally the re-projected images are co-added into a final mosaic. The resulting mosaic image can provide a much deeper and detailed understanding of the portion of the sky in question.

LIGO: Laser Interferometer Gravitational Wave Observatory (LIGO) workflows are used to search for gravitational wave signatures in data collected by large-scale measure gravitational waves predicted by general relativity f gravity), in which gravity is described as due to the curvature of the fabric of time and space.

Cybershake: CyberShake is a seismology application that calculates probabilistic seismic hazard curves for geographic sites in the Southern California region. It identifies all ruptures within 200km of the site of interest and converts rupture definition into multiple rupture variations with differing hypocenter locations and slip distributions. It calculates synthetic seismograms for each rupture variance from where peak intensity measures are extracted and combined with the original rupture probabilities to produce probabilistic seismic hazard curves for the site.

Epigenomics: The Epigenomics workflow is a CPUintensive application. Initial data are acquired from the Illumina-Solexa Genetic Analyzer in the form of DNA sequence lanes. Each Solexa machine can generate multiple lanes of DNA sequences.

These data are converted into a format that can be used by sequence mapping software. The workflow maps DNA sequences to the correct locations in a reference Genome. Then it generates a map that displays the sequence density showing how many times a certain sequence expresses itself on a particular location on the reference genome.

SIPHT: The SIPHT-sRNA identification protocol using high-throughput technology workflow conducts a wide search for small untranslated RNAs (sRNAs) that regulates several processes such as secretion or virulence in bacteria. The kingdom-wide prediction and annotation of sRNA encoding genes involves a variety of individual programs that are executed in the proper order using Pegasus. These involve the prediction of r-independent transcriptional terminators, BLAST (Basic Local Alignment Search Tools) comparisons of the inter genetic regions of different replicons and the annotations of any sRNAs that are found.

3.7Algorithm:

The examiner tasks in a level-by-level approach starting from the level with the largest width (number of tasks at the same level, line 2). The intuition behind this breadth favored approach is that we believe it should improve the performance most. Algorithm 1 shows the pseudo code of our balanced clustering algorithm that uses a combination of these balancing methods and metrics. The maximum number of clustered jobs (size of CL) is equal to the number of available resources multiplied by a clustering factor.

Algorithm 1 shows the pseudo code of our balanced clustering algorithm that uses a combination of these balancing methods and metrics. The maximum number of clustered jobs (size of CL) is equal to the number of available resources multiplied by a clustering factor.

3.7.1 Data Aware Balanced Clustering

algorithm

Require: W: workflow; CL: list of clustered jobs; C: the required size of CL;

Ensure: The job runtime of CL is as even as possible

Step 1: procedure CLUSTERING (W; D; C)

Step 2: Sort W in decreasing order of the size of each level

Step 3: for level <the depth of W do

Step 4: TL GETTASKSATLEVEL (w; level). Partition W based on depth

Step 5: CL MERGE (TL; C). Form a list of clustered jobs

Step 6: W W TL + CL . Merge dependencies as well

Step 7: end for

Step 8: end procedure

Step 9: procedure MERGE(TL;C)

Step 10: Sort TL in decreasing order of task runtime

Step 11: for t in TL do

Step 12: J GETCANDIDATEJOB (CL; t). Get a candidate

task

Step 13: J J + t. Merge it with the clustered job

Step 14: end for

Step 15: return CL

Step 16: end procedure

Step 17: procedure GETCANDIDATEJOB(CL; t)

Step 18: Selects a job based on balanced clustering methods

Step 19: end procedure

3.7.2 Selective Reclustering algorithm:

Require: W: workflow; C: max number of tasks per job defined by clusters.size or clusters.num

Step 1: procedure RECLUSTERING(J) . J is a failed job

Step 2: TL GETTASKS (J)

Step 3: Jnew fg

Step 4: for all Task t in TL do

Step 5: if t is failed then

Step 6: Jnew.add (t)

Step 7: end if

Step 8: if Jnew: size () > k_ then

Step 9: W W +Jnew

Step 10: Jnew fg

Step 11: end if

Conclusion:

Transient failures in a distributed environment and assess their influence on task clustering. Proposed delay aware dynamic clustering methods to improve the fault tolerance of task clustering and applied them to five widely used scientific workflows. Experimental results show that makespan when compared to an existing task clustering method used in workflow management systems. In particular, the Dynamic Re-clustering method performed best among all methods since it could adjust the clustering size based on the Maximum Likelihood Estimation of task runtime, system overheads, and the inter-arrival time of failures. The Vertical Re-clustering method significantly improves the performance for workflows that had short task runtimes. The dynamic estimation process, which used data collects during the workflow execution, could further improve the overall runtime in a dynamic environment where the inter-arrival time of failures fluctuated. The load balancing methods to address the load balance problem when clustering workflow tasks. The defined three imbalance metrics to quantitative measure workflow the balanced clustering methods were implemented in the Workflow Sim simulator. The experiments were conducted using two real workflows. The first experiment shows the gain of task clustering by using a naive horizontal clustering technique. Results show that balancing methods can significantly reduce the runtime and data dependency imbalance. For high Horizontal Runtime Variance values, a runtime variance based approach (Horizontal Runtime Balance) performs best over a naive horizontal clustering algorithm. When data dependency is more important Horizontal Impact Factor Balance and particularly Horizontal Distance Balance methods perform better than the naive approach, while Horizontal Runtime Balance performs similarly.

3.8 Future enhancement:

This work focused on the evaluation of fault-tolerant task clustering techniques on homogeneous environments. In the future plan to combine our work with fault tolerant scheduling in heterogeneous environments, i.e., a scheduling algorithm that avoids mapping clustered jobs to failure-prone nodes also intend to combine vertical clustering methods with horizontal clustering methods. For example, vertical clustering can be performed either before or after horizontal clustering, which we believe would bring different performance improvement. The assumed that the inter-arrival time of transient failures is a function of task type, which is one of the major impact factors. In the future plan to consider other factors such as the execution site, which may improve the accuracy of the model. In this paper we assumed that the network bandwidth is the maximum possible data transfer speed between a pair of virtual machines per file. Future work will consider different network models to explore their impact on our fault-tolerant clustering techniques.

References:

[1] G. Singh, M. Su, K. Vahi, E. Deelman, B. Berriman, J. Good, D. S. stering for best effort systems Proc. 15th ACM Mardi Gras Conf., 2008, p. 9

[2] in scientific Proc. IEEE 8th World Congr. Services, 2012, pp. 9 16.

[3] R. Ferreira da Silva, T. Glatard, and F. De - line, nonclairvoyant optimization of workflow activity granularity on Proc. Euro-Par Parallel Process., 2013, vol. 8097, pp. 255 266.

[4] K. Maheshwari, A. Espinosa, M. Wilde, Z. Zhang, I. Foster, S. d data clustering for aggregate use of multiple production Proc. 5thInt. Workshop Data-Intensive Distrib. Comput., 2012, pp. 3 12.

[5] R. Ferreira da Silva, T. Glatard, and F. Desprez, and task granularity in distributed, online, non-clairvoyant Concurrency Comput., Practice Experience,

vol. 26, no. 14, pp. 2347 2366, 2014.

[6] partitioning Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., May 2012, pp. 764 768.

[7] implications of failures in large- in Proc. 10th Workshop JobScheduling Strategies Parallel Process., Jun. 2004, pp. 233 252.

[8] analysis and Proc. Int. Symp. Fault-TolerantComput., 1990, pp. 244 251.

[9] Large-scale study of failures in high- Proc. Int. Conf. Dependable Syst. Netw., 2006, pp. 249 258.

[10] R. K. Sahoo, A. Sivasubramaniam M. S. Squillante, and Y. Zhang, -scalehet in Proc. Int. Conf. Dependable Syst. Netw., 2004, pp. 772 781.

[11] J. Bresnahan, T. Freeman, D. LaBissoniere, and K. Keahey, Proc. Teragrid Conf., 2011, p. 12.

[12] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. s: Mapping scientific workflows onto th Proc. 2nd Eur. AcrossGrid Conf., 2004,pp. 11 20.

[13] R. Duan, R. Prodan, and T. Fahri -time optimisation of Proc. 7th IEEE/ACM Int. Conf. GridComput., 2006, pp. 33 40.

[14] R. Ferreira da Silva and T. - gateway workloadarchive to study pilot jobs, user activity, bag of tasks, task substeps, Proc. 18th Int. Conf. Parallel Process. Workshops, 2013, pp. 79 88.

[15] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. cost of doing science on the cloud: The Proc. ACM/IEEE Conf. Super computer., 2008, p. 50.

[16] G. B. Berriman, G. Juve, E. Deelman, M. Rynge, and J.-S.V ockler, y of Proc. Workshop e-Science ChallengesAstron. Astrophys., 2010, pp. 1 7.