Scientific Workflows Using Dynamic and Fault-Tolerant Clustering

1R.Ranitha, 2A.Monika, 3L.Marutharasi
1Assistant Professor/CSE
Kings College of Engineering, Thanjavur, Tamil Nadu

Abstract: **In this project, we conduct a theoretical analysis of the impact of transient failures on the runtime performance of scientific workflow executions. We propose a general task failure modelling framework that uses a maximum likelihood estimation-based parameter estimation process to model work flow performance. 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. However, a job composed of multiple tasks may have a higher risk of suffering from failures than a single task job. We further propose three fault-tolerant clustering strategies to improve the runtime performance of workflow executions in faulty execution environments. Experimental results show that failures can have significant impact on executions where task clustering policies are not fault-tolerant, and that our solutions yield makespan improvements in such scenarios. In addition, we propose a dynamic task clustering strategy to optimize the workflow's make span by dynamically adjusting the clustering granularity when failures arise.**

INTRODUCTION

The task runtime may be shorter than the system overhead, the period of time during which miscellaneous work other than the users computation is performed. Task clustering methods, merge several short tasks into a single job such that the job runtime is increased and the overall system overhead Clustering strategies ignore or underestimate the impact of failures on the system, despite their significant effect on large-scale distributed systems such as grids and Clouds. We focus particularly on transient failures since they are expected to be more prevalent than permanent.

The most common technique is to retry the failed job. Retrying a clustered job can be expensive since completed tasks within the job usually need to be recomputed; there by resource cycles are wasted. Three fault-tolerant task clustering methods to improve existing task clustering techniques in a faulty distributed execution environment. 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 according to the estimated inter-arrival time of task failures. The third method partitions the clustered jobs into finer jobs and retries them.

OBJECTIVES:

- In this project, we develop the Job Scheduling Process using on Adaptive Clustering Algorithm.
- There has been a lot of work done on clustering task graphs on Multi-processing system in order to minimize compilation time.
- A tool for launching, configuring, monitoring a set of interdependent VMs are introduced.
- This algorithm used for re-clustering method.

DOMAIN EXPLANATION

Cloud Computing:

Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.

A cloud service has three distinct characteristics that differentiate it from traditional hosting. It is sold on demand, typically by the minute or the hour; it is elastic -- a user can have as much or as little of a service as they want at any given time; and the service is fully managed by the provider (the consumer needs nothing but a personal computer and Internet access). Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet and a weak economy, have accelerated interest in cloud computing.

A cloud can be private or public. A public cloud sells services to anyone on the Internet. (Currently, Amazon Web Services is the largest public cloud provider.) A private cloud is a proprietary network or a data center that supplies hosted services to a limited number of people. When a service provider uses public cloud resources to create their private cloud, the result is called a virtual private cloud. Private or public, the goal of cloud computing is to provide

easy, scalable access to computing resources and IT services.

Platform-as-a-service in the cloud is defined as a set of software and product development tools hosted on the provider's infrastructure. Developers create applications on the provider's platform over the Internet. PaaS providers may use APIs, website portals or gateway software installed on the customer's computer. Force.com, (an outgrowth of Salesforce.com) and GoogleApps are examples of PaaS. Developers need to know that currently, there are not standards for interoperability or data portability in the cloud. Some providers will not allow software created by their customers to be moved off the provider's platform.

In the software-as-a-service cloud model, the vendor supplies the hardware infrastructure, the software product and interacts with the user through a front-end portal. SaaS is a very broad market. Services can be anything from Web-based email to inventory control and database processing. Because the service provider hosts both the application and the data, the end user is free to use the service from anywhere. Cloud Computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. This technology allows for much more efficient computing by centralizing data storage.

A foundation: Nearly every application uses some platform software on the machine it runs on. This typically includes various support functions, such as standard libraries and storage, and a base operating system. A group of infrastructure services: In a modern distributed environment, applications frequently use basic services provided on other computers. It's common to provide remote storage, for example, integration services, an identity service, and more.

A set of application services: As more and more applications become service-oriented, the functions they offer become accessible to new applications. Even though these applications exist primarily to provide services to end users, this also makes them part of the application platform. (It might seem odd to think of other applications as part of the platform
Due to the complex resource to performance relationship in a cloud environment, it is hard to build a mathematical system model that can be used in classical feedback control. Our First attempt to address the issue of the lack of accurate server model is to use a model-free control approach. We extend out previous work on self-tuning fuzzy controller (STFC) for the QoS assurance in physical web servers to the problem of resource allocation in a dynamic cloud environment. By introducing extra layers of output amplication and exible fuzzy rule selection, the proposed STFC outperform other popular controllers in CPU resource allocation. We also present the design of DynaQoS, an adaptive and multi-purpose QoS provisioning framework based on STFC. we further improve the automatic resource management in three aspects. First, we extend the automated resource allocation to manage multiple resources. Second, in addition to guaranteeing cloud users' SLA, we also optimize the system- wide performance of the hosting server. Finally, to avoid the performance degradation and oscillation caused by the resource reconfigurations, we optimize the process of reconfiguration operations. We present VCONF, a reinforcement learning-based auto-configuration agent for physical nodes. Reinforcement learning is in nature a model- free and adaptive method and it's well in the system-wide resource allocation. To deal with multiple resources and improve the scalability and adaptability of the learning approach, we propose a model-based reinforcement learning algorithm that makes efficient use of collected system information.

PROBLEM DEFINITION

EXISTING SYSTEM:

Existing work Most of existing works are Task Clustering technique. Task clustering technique on a larger set of workflows widely used scientific applications and evaluating the performance impact of the variance of the distribution of the task runtimes. This system overheads, and the inter-arrival time of failures on the work-flow's Makespan turnaround time to execute all workflow tasks.
Disadvantages:

- Turnaround time to execute all workflow tasks failure estimations with different inter-arrival times of failures.
- Clustering size to balance the cost of task retry and of the scheduling overheads.

PROPOSED SYSTEM:

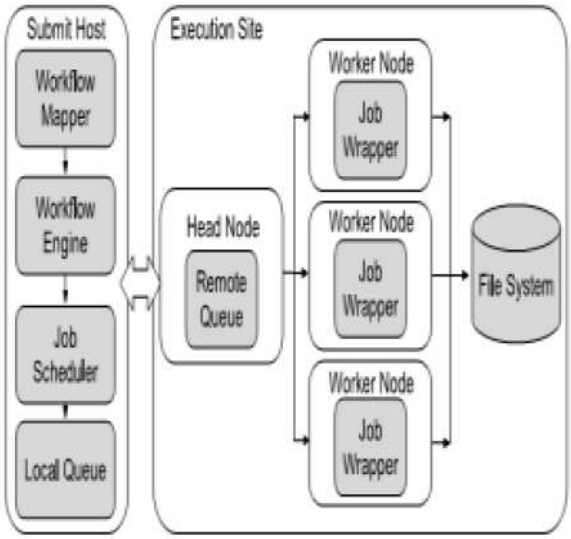
In the proposed system, we propose a general task failure modeling. We propose three fault-tolerant task clustering methods to improve existing task clustering techniques in a faulty distributed execution environment. This method retries failed tasks within a job by extracting them into a new job. This method dynamically adjusts the granularity or clustering size

according to the estimated inter-arrival time of task failures. This method partitions the clustered jobs into finer jobs.

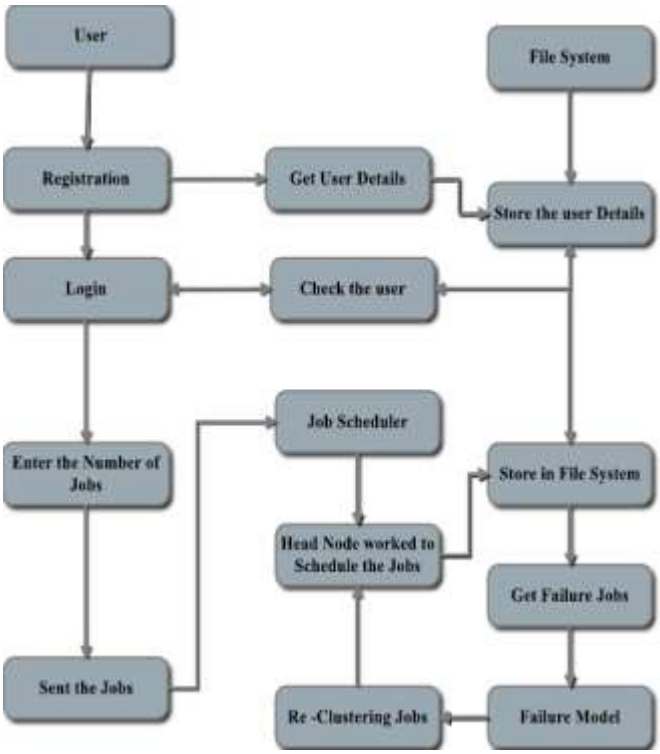
Advantages:

- Three dynamic clustering methods to improve the fault tolerance of task clustering
- 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 MODELING AND DESIGN
SYSTEM ARCHITECTURE:



Flow Diagram:



MODULE DESCRIPTION

USER PROFILING:

• In this module user registration and login into the cloud.
• After Registrations all the details store in the file system.
• User login to the cloud and sent a job .

JOB SCHEDULER:

• In this module, job Scheduler Manage individual workflow jobs and supervise their execution on local and remote resources.
• The time span between the job submission to the scheduler and the beginning of its execution on a worker node is denoted as the queue delay.
• This delay reflects both the efficiency of the scheduler and the resource availability.

Failure Model:

In this Failure model, reclustering method that merges failed tasks into new clustered jobs in which the clustering size.
We use three clustering methods are Horizontal, Vertical, Dynamic clustering techniques.

FILE SYSTEM:

• In this section, file system is storage of the cloud.
• File system worked into the store the user registration details and the user files details.
• User want to view the uploaded file into the cloud, file system find the user selected file and sent to user.

ALGORITHM HORIZONTAL CLUSTERING ALGORITHM:

- It 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, the number of tasks per clustered job (clusters, Size), or the number of clustered jobs per horizontal level of the workflow (clusters.num).
- We set clusters number to be the same as the amount of available resources. We have evaluated the runtime performance for different clustering granularities.

SELECTIVE RECLUSTERING ALGORITHM:

- Clustering technique, on the other hand, merges only failed tasks within a clustered job into a new clustered job.
- The Clustering and Merge procedures are the same. In the first attempt, the clustered job, composed of four tasks, has three failed tasks (t1, t2, t3).
- Three failed tasks are merged into a new clustered job j 2 and retried.
- This approach does not intend to adjust the clustering size k, although the clustering size may become smaller after each attempt since subsequent clustered jobs may have fewer tasks.

FEASIBILITY STUDY

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements. The feasibility carried out mainly in three sections namely.

• **Economic Feasibility**
• Technical Feasibility
• Behavioral Feasibility

Economic Feasibility

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department if sufficient for system development.

Technical Feasibility

This study center around the system's department hardware, software and to what extend it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

Behavioral Feasibility

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user's request, instead of getting a report, which doesn't contain much detail. In which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user's request, instead of getting a report, which doesn't contain much detail.

CONCLUSION

In this work, we modeled transient failures in a distributed environment and assess their influence on task clustering. We proposed three dynamic clustering methods to improve the fault tolerance of task clustering and applied them to five widely used scientific workflows. Experimental results showed that the proposed methods significantly improve the workflow's Makespan when compared to an existing task clustering method used in workflow management systems. In particular, the dynamic reclustering 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. This work focused on the evaluation of fault-tolerant task clustering techniques on homogeneous environments.

FUTURE ENHANCEMENT

In the future, we 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.
We 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.

We 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, we 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

- W. Chen and E. Deelman, "Fault tolerant clustering in scientific workflows,", 2012.
- R. Ferreira da Silva, T. Glatard, and F. Desprez, "Online, non-clairvoyant optimization of workflow activity granularity on grids ",2013.
- K. Maheshwari, A. Espinosa, M. Wilde, Z. Zhang, I. Foster, S.Callaghan, and P. Maechling, "Job and data clustering for aggregate use of multiple production cyberinfrastructures," 2012.
- R. Ferreira da Silva, T. Glatard, and F. Desprez, "Controlling fair-ness and task granularity in distributed, online, non-clairvoyant workflow executions," Concurrency Comput., 2014.
- W. Chen and E. Deelman, "Integration of workflow partitioning and resource provisioning," 2012.
- J. Bresnahan, T. Freeman, D. LaBissoniere, and K. Keahey, "Managing appliance launches in infrastructure clouds,", 2011.

