|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **TITLE** | **ABSTRACT** | **STRATEGY** | **PARAMETERS** | **PROS/CONS** | **CONCLUSION** |
| 1 | A Multi-Resource Load Balancing Algorithm for Cloud Cache Systems. | With the advent of cloud computing model, distributed caches have become the cornerstone for building scalable applications and to speed up applications by avoiding database accesses. Distributed object caches assign objects to cache instances based on a hashing function, and objects are not moved from a cache instance to another unless more instances are added to the cache and objects are redistributed. This may lead to situations where some cache instances are overloaded when some of the objects they store are frequently accessed, while other cache instances are less frequently used.  A multi-resource load balancing algorithm for distributed cache systems aims at balancing both CPU and Memory resources among cache instances by redistributing stored data. Considering the possible conflict of balancing multiple resources at the same time, we give CPU and Memory resources weighted priorities based on the runtime load distributions. A scarcer resource is given a higher weight than a less scarce resource when load balancing. The system imbalance degree is evaluated based on monitoring information, and the utility load of a node, a unit for resource consumption. Besides, since continuous rebalance of the system may affect the QoS of applications utilizing the cache system, our data selection policy ensures that each data migration minimizes the system imbalance degree and hence, the total reconfiguration cost can be minimized. The policy shows a significant improvement in time efficiency and decrease in reconfiguration cost. | Identifies which are the scarcer resources based on the runtime load distributions, and then assigns resources different priorities based on this diagnosis. Some data partitions must be moved from one cache node to another to rebalance the load. During this data migration, reconfiguration cost is to be considered incurred by the extra consumption of resources such as CPU and network bandwidth. A nearest-neighbor-search based policy is proposed to select the bucket to be migrated.  ->Backfill Lowest (BL) determines which resource is most available in the receiver node, and then migrates the job that consumes most of that resource in sender node.  -->Backfill Balance (BB) migrates the job which can minimize the (maximum load/average load) measure for the receiving server.  ->Market Mechanism (MM) to balance multiple resources among nodes with heterogeneous resource capacities; uses a pricing model (∑1K (Ji ∗ Li)/ ∑1KJi ) to calculate the cost per resource of a job.  Ji - job’s consumption of resource i; Li - load of resource i in receiver node before job migration, K is the number of resource types. | 1. Minimize the load imbalance for **CPU and Memory resources** among cache nodes but with different priorities based on the runtime load diagnose. 2. Minimize the system reconfiguration cost by considering the **number of data buckets to be migrated**.   A measure of node’s overall load (utility load) will be helpful to choose a node pair for data migration.  Standard deviation 𝜎𝑐𝑝𝑢 and 𝜎𝑚𝑒𝑚 are used to evaluate the imbalance degree of CPU and Memory usage among cache nodes, respectively. 𝜎𝑐𝑙𝑢𝑠𝑡𝑒𝑟 captures the cluster’s load imbalance degree.  Finding the nearest neighbor (bucket for migration) in amount of resource dimensions (types) using **nearest-neighbor search**. | PROS:  1) The policy of nearest-neighbor search is slightly better in improving the balance degree of less scarcer resource too.  2) When CPU and memory usages are close to each other, (35, 65), NN shows best results for balancing both CPU and memory.  In case CPU and memory usages are (50, 50), since the two resources have equal weight to be balanced, the balancing result for CPU and memory are roughly the same with approximately 50% improvement using policy NN.  3) NN, is much better than the others with the lowest execution time.  CONS:  1) NN shows a slightly higher reconfiguration cost compared with BL. | The algorithm aims at balancing both CPU and memory usage among cache nodes. This is accomplished by migrating data partitions among cache nodes. Algorithm gives different weights to the resources based on the system load distribution. The scarcer a resource is, the higher its weight is. Simulation shows that compared to previous work, NN algorithm attains a better balance for both CPU and memory, and it also reduces the reconfiguration cost, that is, the amount of data buckets to be migrated, exhibiting the lowest execution time. |
| 2 | Proposal for an Optimal Job Allocation Method for Data-intensive Applications based on Multiple Costs Balancing in a Hybrid Cloud Environment. | Due to the explosive increase in the amount of information in computer systems, we need a system that can process large amounts of data efficiently. Cloud computing system is an effective means to achieve this capacity and has spread throughout the world. The focus is on hybrid cloud environments, and method for efficiently processing large amounts of data while responding flexibly to needs related to performance and costs. This method can act as middleware.  For data-intensive jobs, a benchmark is created that can determine the saturation of the system resources deterministically. Using this benchmark, the parameters in this middleware can be determined. This middleware can provide Pareto optimal cost load balancing based on the needs of the user. | When used for CPU processing and disk processing, the middleware determines when the resource has been saturated. Based on this information, this middleware will determine the resource load.  The middleware consists of a monitor unit (checks the status of the resource and determines if saturated) and dispatch unit (receives and distributes jobs). | Load of CPU processing based on CPU usage; and for Load of Disk processing on length of the queue (indicates disk saturation) for the current disk and the number of jobs that are running in these disks.  A threshold can be set for the length of queue for disk processing.  Optimal job placement in hybrid cloud environments in terms of **monetary costs** and **performance**. |  | A method of determining the load based on the required cost and performance to ensure efficient processing load balancing in a hybrid environment cloud. Implementation will introduce a middleware which uses information about CPU processing (CPU usage) and disk processing (length of queue) to provide efficient load balancing if the resources needed to perform a job are scarce. Using this middleware, one can control the Pareto optimal cost balance load distribution without wasting resources. |
| 3 | A Comparative Survey On Load Balancing Algorithms In Cloud Computing | Clouds are high configured infrastructure delivers platform, software as service, which helps customers to make subscription for their requirements under the pay as you go model. Cloud computing is spreading globally, due to its easy and simple service oriented model. Generally cloud is based on data centers which are powerful to handle large number of users. The reliability of clouds depends on the way it handles the loads, to overcome such problem clouds must be featured with the load balancing mechanism. The goal of balancing the load of virtual machines is to reduce energy consumption and provide maximum resource utilization thereby reducing the number of job rejections. Load balancing is the pre requirements for increasing the cloud performance and utilizing the resources. It is a brief discussion on the existing load balancing techniques in cloud computing and further compares them based on various parameters like response time and Data processing time etc. | Load needs to be distributed over the resources in cloud-based architecture, so that each resource does approximately the equal amount of task at any point of time which is performed by a load balancer. The load balancer determines which web server should serve the request.  Static load balancing algorithms require aforementioned knowledge about the applications and resources of the system. The decision to shift the load does not depend on the current state of the system. It distributes the workload based strictly on a fixed set of rules related to characteristics of the input workload.  A dynamic load balancing algorithm depends on the current state of the system. It allows processes to be moved from an over utilized machine to an underutilized machine dynamically for faster execution. | Round Robin Algorithm: passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced.  Throttled Load Balancing Algorithm: It is completely based on virtual machine. The load balancer maintains an index table of virtual machines as well as their states (Available or Busy), if a request arrive throttled load balancer send the ID of available virtual machine to the data center controller and it allocates that virtual machine. | 1) Substantial improvement in performance.  2) Stability maintenance of the system.  3) Increase flexibility of the system so as to adapt to the modifications.  4) Build a fault tolerant system by creating backups. | Load balancing is one of the main challenges in cloud computing. It is required to distribute the dynamic local workload evenly across all the nodes to achieve a high user satisfaction and resource utilization ratio by making sure that every computing resource is distributed efficiently and fairly. |
| 4 | A Load Balancing Model Based on Cloud Partitioning for the Public Cloud. | Load balancing in the cloud computing environment has an important impact on the performance. Good load balancing makes cloud computing more efficient and improves user satisfaction. A better load balance model for the public cloud is based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. The algorithm applies the game theory to the load balancing strategy to improve the efficiency in the public cloud environment. | The cloud (geographically partitioned) has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy.  When a job arrives at the system, the main controller queries the cloud partition where job is located. If this location’s status is idle or normal, the job is handled locally. If not, another cloud partition is found that is not overloaded.  The Load Status table is used by the balancers to calculate the partition status. | The percentage of idle, normal or overload nodes of a cloud partition will decide its status.  The node load degree involves various static (number of CPU’s, the CPU processing speeds, the memory size) and dynamic (memory utilization ratio, the CPU utilization ratio, the network bandwidth) parameters.  The load degree of each node are input into the Load Status Tables created by the cloud partition balancers which get refreshes after each fixed period T.  When the cloud partition is normal, load balancing strategy is based on game theory.  A greedy algorithm is used to calculate fraction of a job assigned to a node for all nodes; leads to Nash equilibrium. | Can switch between the load balancing methods whenever the status changes from idle to normal.  The Round Robin algorithm (ideal status) does not record the status of each connection so it has no status information. In a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes.  The node order will be changed when the balancer refreshes the Load Status Table. However, there may be read and write inconsistency at the refresh period T. When the balance table is refreshed, at this moment, if a job arrives at the cloud partition, it will bring the inconsistent problem. The system status will have changed but the information will still be old. | The load balancing model is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy.  The main controller and the cloud partition balancers refreshes the information at a fixed period. |
| 5 | Pattern Matching Based Forecast of Non-periodic Repetitive Behavior for Cloud Clients. | The Cloud phenomenon brings along the cost-saving benefit of dynamic scaling. As a result, the question of efficient resource scaling arises. Prediction is necessary as the virtual resources that Cloud computing uses have a setup time that is not negligible. An approach to the problem of workload prediction is based on identifying similar past occurrences of the current short term workload history. The Cloud client resource auto-scaling algorithm uses the above approach to help when scaling decisions are made, as well as experimental results by using real-world Cloud client application traces. An overall evaluation of this approach, its potential and usefulness for enabling efficient auto-scaling of Cloud user resources is presented. | Cloud offers on-demand virtualized resources to its users; the possibility of dynamic scaling arises.  In self-similarity web traffic, by identifying resource usage patterns in the past and having a high similarity to the present resource usage pattern, a decision can be made as to the necessity and/or direction of scaling for the present situation.  Predictive approach in hopes of getting an insight into future platform usage make scaling decisions ahead of time, compensating for the resource start-up time.  -> Mathematical model on past server usage.  -> Prediction model on past resource usage (based on the current server load and auto-scaling rules).  AutoRegression of order 1: using a finite history window and identifying appropriate parameters so that a recurring sequence can be obtained to calculate the next values.  Linear Regression: calculates a polynomial approximation of the history of requests.  Rightscale: threshold-based auto-scaling; using a democratic voting system based on the current server load. Each virtual machine has a vote based on its current load level and two thresholds: low threshold corresponds to “scale down” vote and a high threshold corresponds to “scale up” vote. The votes are collected by a central machine and the majority decides the scaling decision for the whole platform. | Scaling decisions are influenced by virtual resource setup time or the migration of existing processes to free resources, but resource usage has the biggest impact. |  | Cloud can adapt the number of resources used based on their actual use by a client. This implies cost saving as resources are not paid for unused period of time. Virtualization supports dynamic scalability. The downside of virtualization is that it has a non-zero setup time that has to be taken into consideration for an efficient use of the platform. It follows that an accurate prediction method is needed in making auto-scaling decisions. A resource usage prediction algorithm uses a set of historic data to identify similar usage patterns to a current window of records that occurred in the past. The algorithm then predicts the system usage by interpolating what follows after the identified patterns from the historical data. |