Assignment

Subject: P&DC

Instructor: Sir Asif Laghari

Student: **Tanveer Hassan**

Id: **CSC-22F-142**

Topic: Load Balancing and Fault Tolerance in Parallel and Distributed Computing

Introduction

Parallel and Distributed Computing (P&DC) is an important part of the contemporary system of computing when big problems are addressed with the division of the tasks between several processing units or nodes. Such nodes can be either in one system (parallel computing) or in multiple machines linked together by a network (distributed computing). Such systems have high performance, scalability and efficiency but they also come with critical challenges. Load balancing and fault tolerance are two of the most problematic issues.

Load balancing is used to make the workload spread even across all the computing nodes such that the workload does not overload any single node and leave the rest idly waiting. Fault tolerance is however used to make sure that the system still operates properly even when it is coupled with the failure of some of the parts. This task is devoted to three key areas: the dynamic load balancing of the nodes, checkpointing and recovery systems, and fault-tolerant parallel systems. Combined, the concepts enhance the reliability, performance and availability of the system.

1. Distributed Systems Load Balancing

1.1 Concept of Load Balancing

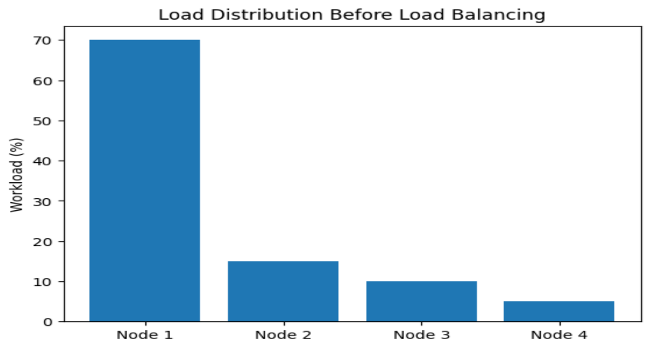
Load balancing is defined as the process of allocating the computational loads to a set of nodes to maximize resource usage, reduce the execution time and prevent the performance bottlenecks. Workloads in P&DC systems are not predictable and are also time varying. When the allocation of tasks is not done, then some nodes will go unused and others will be overloaded hence inefficiency.

There are two major types of load balancing:

- Static Load Balancing
- Dynamic Load Balancing

In Static load balancing, workload and system behaviour are predictable and therefore the tasks are assigned in advance. Nonetheless, in actual systems, dynamic changes frequently occur, and hence, the static approaches are not productive.

Load Distribution Before Load Balancing



Load Distribution After Load Balancing

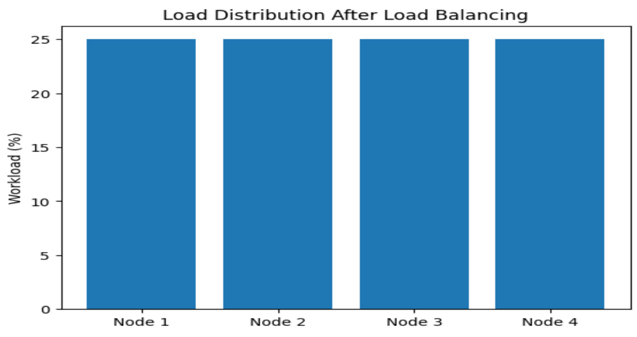


Figure: Comparison of system workload before and after applying load balancing

2. Elastic Load Balancing between Nodes

2.1 Definition and Importance

Dynamic Load Balancing (DLB) is a process that is run at runtime and it involves the redistribution of tasks among the nodes depending on their workload and performance at a given time. As opposed to the unchanging methods, the dynamic load balancing constantly updates the conditions of the system and changes as needed.

There is a dynamic load balancing requirement in:

- Cloud based computing ecosystems.
- Distributed databases
- Scientific simulations
- Big data processing systems

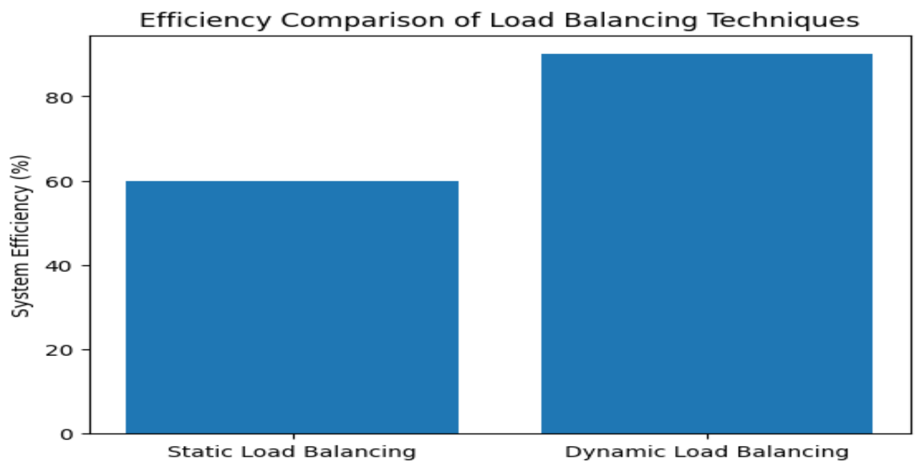


Figure: Efficiency comparison between static and dynamic load balancing techniques.

2.2 Working Mechanism

The following are the steps that are generally undertaken in dynamic load balancing:

1. **Monitoring -** Every node gives a report of its load condition (CPU usage, memory usage, network latency).
2. **Decision Making -** A focal or decentralized controller makes the decision on whether load redistribution is necessary.
3. **Migration of tasks -** Migrating tasks to underloaded nodes are done.
4. **Continuation of execution -** Resumes are not considered as tasks that modify the correctness of the systems.

2.3 Strategies of Dynamic Load Balancing

To carry out dynamic load balancing it is possible to use:

Centralized Approach - load allocation is done by a central node.

Distributed Approach - Every node is involved in the decisions of the load balancing.

Hierarchical Approach - Nodes are arranged into levels which are more scalable.

Both methods have some benefits and disadvantages. Centralized systems are simpler to handle but have the advantage of being a bottleneck whereas distributed systems have a greater fault tolerance but higher communication overhead.

2.4 Benefits of Dynamic Load Balancing

Improved system throughput

- Improved use of resources.
- Reduced response time
- Ability to change work load.
- Increased scalability

3. Fault Tolerance of Parallel and Distributed Systems

3.1 Understanding Fault Tolerance

Fault tolerance refers to the capability of a system to fully remain functional despite failures. In distributed environments, failures are widespread because of hardware problems, software bugs, power cuts or network failures.

The fault tolerance is necessary because:

- Distributed systems are composed of a large number of autonomous components.
- Large-scale systems are prone to failures.
- Down-time of system may lead to loss of financial and data.

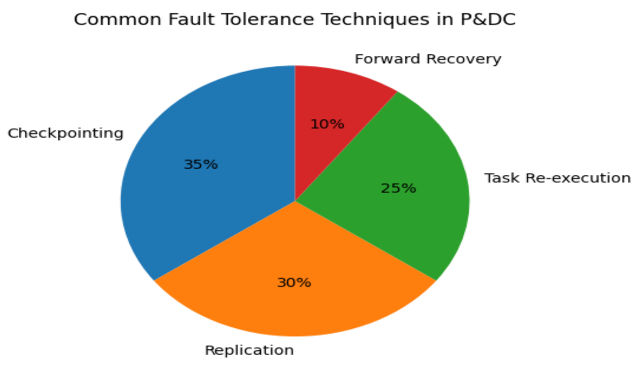


Figure: The typical fault tolerance methods applied in parallel and distributed systems.

4. Checkpointing and Recovery Strategies

4.1 Concept of Checkpointing

Checkpointing Checkpointing is a form of fault tolerance where the system saves the state of an application on a regular basis as it executes. In case of a failure, the system is able to resume at the last checkpoint saved rather than resuming the system at the start.

A check point usually consists of:

- Process state
- Memory contents
- Program counter
- Communication state

4.2 Types of Checkpointing

a) Coordinated Checkpointing.

The process of coordination is to occur at checkpoints across all the processes. This approach does not create inconsistencies but can stop the whole system.

b) Uncoordinated Checkpointing.

All process checks on an individual basis. Although it minimizes the overhead of coordination, it can have the domino effect, where several checkpoints will become ineffective.

c) Checkpointing brought about by Communication.

A hybrid solution that would impose checkpoints depending on communication patterns to provide consistency.

4.3 Recovery Mechanisms

- Such recovery measures are:
- **Rollback Recovery** - System is rolled back to a former checkpoint.
- **Forward Recovery** - System recovers the errors and goes ahead with the execution.
- **Replication-Based Recovery** - Multiple copies are used to perform the same task ensuring that failure by one does not impact on execution.

4.4 Benefits of Checkpointing and Recovery

- Minimizes computation loss
- Enhances reliability of the systems.

- Reduces downtime
- Ensures data consistency

5. Fault Tolerant Parallel Frameworks

5.1 Fault-Tolerant Frameworks Requirements

The recent applications of modern applications are likely to be based on frameworks that automatically balance the loads as well as the fault tolerance. These frameworks make development easier as they conceal complexity in the system to the developers.

5.2 Overview to the Sustainably Popular Fault-Tolerant Frameworks

a) Apache Hadoop

Hadoop is fault tolerant in terms of replicating data and re-execution of tasks. In case a node goes offline, the tasks are reallocated to different nodes.

b) Apache Spark

Spark enhances resilience to fault through Resilient Distributed Dataset (RDDs). Lineage information may be applied to re-calculate the RDDs in cases where failures occur.

c) MPI (Message Passing Interface)

Checkpointing libraries including BLCR are used in MPI-based systems to provide fault tolerance in high-performance computing systems.

d) Kubernetes

At the container level, Kubernetes offers fault tolerance, which consists of re-spun failed containers and redistribution of workloads among the nodes.

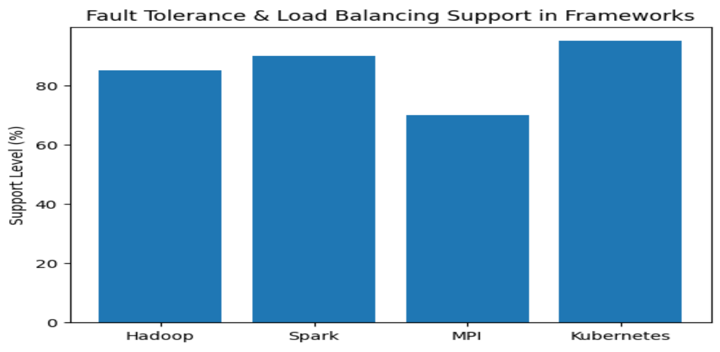


Figure: Distributed frameworks that have fault tolerance and load balancing.

5.3 Benefits of Fault-Tolerant Frameworks.

- The automatic failure detection.
- Task re-execution
- Improved scalability
- Less complicated development.
- High availability

6. Relationship between Load Balancing and Fault Tolerance

Fault tolerance and load balancing are very similar. Proper load balancing decreases the stress levels on the system and this decreases the chances of failure. Equally, fault tolerance controls redistribute tasks of the failed nodes which indirectly balances load. As a combination, they guarantee high performance, reliability, and efficiency of distributed systems.

Conclusion

Parallel and Distributed Computing systems consist of the basic elements of load balancing and fault tolerance. Dynamic load balancing makes sure the resource is efficiently used by responsive to varying workload and checkpointing and recovery mechanisms guard against systems failure by reducing the loss of computation. Resilient parallel systems like Hadoop, Spark and Kubernetes offer default-based systems to resolve failure and load balancing.

These techniques will only become even more important as distributed systems continue to grow larger and more complicated. The concept of load balancing and fault tolerance is critical in the process of constructing a reliable, scalable and high performance computing system in the current application.

Sites to Read and Rewrite

The following original sources will be read, understood and paraphrased to your own word:

1. **GeeksforGeeks – Load Balancing**
   - https://www.geeksforgeeks.org/load-balancing-in-distributed-system/
2. **IBM Documentation – Fault Tolerance**
   - https://www.ibm.com/docs/en/cloud-paks/cp-integration/2023.2?topic=concepts-fault-tolerance

3. **Apache Hadoop Documentation**
    - https://hadoop.apache.org/docs/
4. **Apache Spark Fault Tolerance**
    - https://spark.apache.org/docs/latest/rdd-programming-guide.html
5. **Distributed Systems – Tanenbaum**
    - https://www.distributed-systems.net/
6. **MIT OpenCourseWare – Distributed Systems**
    - https://ocw.mit.edu/courses/6-824-distributed-systems/

References

1. Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed Systems: Principles and Paradigms*. Pearson.
2. Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). *Distributed Systems: Concepts and Design*. Addison-Wesley.
3. Apache Software Foundation. *Hadoop Documentation*.
4. Apache Software Foundation. *Spark Programming Guide*.
5. IBM Cloud Documentation – Fault Tolerance Concepts.
6. MIT OpenCourseWare – Distributed Systems Course Materials.