International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Survey on Various Load Balancing Approaches in Distributed and Parallel Computing Environment

Mrs. Minal Shahakar [1], Dr. S. A. Mahajan[2], Dr. Lalit Patil [3]

Submitted: 07/05/2023 **Revised:** 15/07/2023 **Accepted:** 06/08/2023

Abstract*:* One of the important issues in distributed and parallel computing is to allocate computational load across different processors or nodes by making better utilization of all the available resources and increasing system performance or throughput. Since this survey mainly reviews and categorize the study on load balancing and task allocation as per the characteristics of distributed systems to make comprehensive taxonomy and to correlate studies on various types of load balancing techniques. Various load balancing techniques helps to meet the QoS requirements and simultaneously maximize throughput by optimum use of the resources. Firstly, this survey briefs the general characteristics of load balancing and distributed systems. On the basis of these characteristics, the survey reviews the study on load balancing and task allocation with respect to resource optimization, various techniques to achieve reliability, load balancing in heterogeneous environment. The resources and workloads must be scheduled in an efficient manner to balance the load. Through the survey, different load balancing techniques are summarized that resolve the issue of task scheduling and load balancing, also in this area the related studies can be understood well on how the general characteristics of distributed systems are satisfied.

Keywords: *Distributed Computing, Load Balancing, Heterogeneous System, Task allocation, Heuristics*

1. Introduction

A software system known as distributed computing makes computing resources accessible to several nodes or computers. The growing computing demands of today's software programmes, made possible by recent technical breakthroughs, are readily able to outpace the hardware that is now available. Applications requiring techniques with high level computing complexity or where a large quantity of information is implicated such as those pertaining to Deep machine Learning, Modeling & Simulation, and other application areas, fall under this category. In these situations, cloud computing solutions provide an excellent means of expanding the computational capacity and achieving gains in both practicality and performance. [2]. Specialized hardware for parallel code acceleration, including Field-Programmable Gate Arrays and General-Purpose Graphics Processing Units, is incorporated into new HPC architectures. In many scientific applications, such techniques are energy efficient as well as efficient in time required for execution, but because the hardware is specialised, they also introduce heterogeneity. [3]. Cluster caches enable load balancing without encoding/decoding overhead or memory redundancy.

The selective partition splits files into a number of partitions based on how well-liked they are; the more partitions a file has, the more popular it is. Servers in the cluster randomly cache file partitions. This strategy offers three advantages. It first improves load balancing by evenly distributing the workload of read requests across cache servers. Second, it boosts the hot files' read parallelism, which enhances I/O performance. Third, simply dividing files into segments does not add storage redundancy or the computational costs associated with encoding and decoding [4]. Clearly, even while doing load balancing at each stage ensures an optimal workload distribution, the overhead caused due to this techniques of balancing load itself may outweigh this advantage. [2].

EDGE computing has the potential to address a number of issues, including computing limitations [27], high energy consumption from end devices, and intolerable propagation delay to the cloud. To provide services with high bandwidth and low latency, edge computing places resource-rich servers near the data source and the end devices. Using established norms, dispatchers collect work units (also known as jobs) and distribute them to servers for processing. This frequently occurs in data centres, server farms, and supercomputers due to the high density of computing resources in these places. The fundamental challenge in achieving high performance is distributing workloads across different servers when size of work units or processing speed vary greatly. Particularly intriguing the fact that when jobs are scheduled statically by using FCFS i.e. First Come First Serve servers, only one lengthy task can significantly increase the average latency if

1 Research Schola, Smt. Kashibai Navale College of Engineering, Savitribai Phule Pune University, India.
ORCID ID : 0000-0003-3449-5489
mhjn.minal@gmail.com
2 Department of Information Technology ,PVG' College of Engg & Tech & GK Pate, (Wani) IOM, Savitribai Phule Pune University, India
sa_mahajan@yahoo.com
3 Department of Information Technology, Smt. Kashibai Navale College of Engineering, Savitribai Phule Pune University, India
lalitvpatil@gmail.com

several shorter tasks must wait in queue behind it. Easy and inexpensive load monitoring [7] Long-term computing load balance is achieved through aware and long-view load balancing by migrating out the data blocks from an overburdened server at various epochs over the course of a period, data blocks contribute more to the server's computing workloads when the server is overloaded and less when it is underloaded [6]. However, a task reassignment algorithm [6] redistributes the backed-up server's workload among other task data servers before CALV is executed.

Balancing workloads is a significant obstacle for distributed and parallel computing. A cache network is a graph consisting of n nodes (cache-enabled servers) and a set of edges (links between them). Files are divided into uniformly sized chunks as part of the cache content insertion process [5]. Coded chunks corresponding to each file are created by linearly merging these chunks. Thereafter, each server will store in its cache the particular encoding sections that are a match for those several files with popular profile. As part of the delivery phase, however on the basis of popular profile the request is placed for the file, n random requests are sent to various servers. After that, a graph is used to determine the shortest path for sending the coded pieces of the requested file to the nodes closest to the request's origin. If the various encoded parts use independent linear combinations, it is simple to demonstrate that at the request origin server each file can be decoded [5]. Use of available resources can be optimised to reduce response times, increase throughput, improve fault tolerance and scalability, raise user satisfaction, reduce operational costs, and reduce wasteful heat and power consumption.

There are two subproblems that make up the load balancing problem:

1. Allowing new requests for provisioning nodes or machines and allocating host space for nodes.
2. System reallocation or migration.

Different load balancing approaches that are mentioned in the literature review and that we will explore in a moment can be used to address the first sub-problem. The second subproblem presents a number of difficulties, including where and when to reallocate resources, which node should be chosen for migration, etc.

The remainder of the paper is organised as follows: The sorts of various load balancing techniques are briefly reviewed in Section II. In Section III, load balancing methods and their advantages and disadvantages are covered. A summary of a heterogeneous system is provided in Section IV. Final thoughts and the potential for further investigation on this subject are presented in Section V, which also finishes the essay.

2. Types of Load Balancing Techniques

Various Fixed and dynamic techniques used for balancing load are the two most common varieties. Static load balancing pre-assigns tasks to processing elements, in contrast to dynamic load balancing, which rebalances tasks as an algorithm executes [2]. In cases where the computations required to complete a task are well-known in advance and remain stable throughout the processing, a static mapping may be the most efficient option. However, in case, computing requirements are not specified in advance before the time of execution or change occurs at runtime [2], than a static or fixed mapping may result into a acute load imbalance. In this situation, balancing load using dynamic techniques is the best option. Avalanches, in their simplest form, spread sand across a lattice, and when applied to queuing tasks in a distributed system, they can result in load-balancing. [8]. When using a static scheduling method, tasks are not prioritised.

The foundation of dynamic load balancing is reallocating jobs among processors as they are being executed. Processors with high loads have their tasks redistributed to those with lower loads, and vice versa. The system's primary objective is to achieve the highest possible efficiency in terms of both resource utilisation and throughput. One of the major drawbacks of dynamic load balancing is the runtime overhead associated with transferring load information among processors, choosing tasks and processors for job transfers, and dealing with communication delays related to the task relocation itself.

Load balancing is a key component in maximising the network's lifespan. Overburdening individual nodes is detrimental to networks, so it's important to allocate tasks effectively. Without an efficient job allocation mechanism, the sensor nodes in a wireless sensor network will only be able to operate independently, wasting resources. Because of their high demand characteristics and constraints, like environmental restrictions, the unstable wireless links and the changing topology, WSNs introduce additional uncertainty and weaknesses for real time applications. [10]. The processing components can be toggled on and off to optimise resource usage based on the current workload. [8]. However, centralised approaches have many problems, such as a bottleneck and a weak spot. Distributed load balancing systems, however, do not face these challenges.

By shifting work from busy to idle processors and underutilised resources, the adaptive solution improves network performance, shortens the time it takes to complete tasks, lessens the amount of energy used by individual nodes, and lengthens the lifespan of the network as a whole. [8] [10].

The following is a concise explanation of why dynamic work allocation is implemented: In order to decrease task execution time, conserve node energy, balance network load, increase network lifetime, guarantee that the task would not fail due to abrupt node failure, and improve the reliability of task management, it is recommended that m tasks be distributed fairly among n sensors [10].

Dynamic load balancing is achieved through the utilisation of a binary matrix encoding form, the reduction of task execution time, the conservation of node energy, the balancing of the network load, and the development of a fitness function for improving scheduling efficiency and system dependability. [10].

3. Literature Survey of Load Balancing Techniques

In the network of scattered datacenters that makes up the cloud computing environment, there are hundreds of servers. As a result, the datacenter controller uses a load balancer to manage tasks that users submit. The load balancer also chooses which computer should be assigned to handle the forthcoming request. The load balancer can

utilise the following algorithms to balance and distribute load equitably.

Cache aware, the cache hierarchy in multicore systems is used by Reorder also known as corder, a straightforward job reordering approach. It primarily supports equitable distribution of compute loads among multicores at the shared memory level and increases cache efficiency by implementing additional vertex order refinements at the private cache level. [1] Several graph applications and datasets are used to adequately evaluate performance of this method. This method works with vertexes identified by outgoing edges, while achieving outstanding benefits in cross-platform scalability and reordering overhead.

The Cellular Automata-CA Model is a dynamic load balancing approach for running spatially-aware applications in parallel. From a qualitative standpoint, it demonstrates the LB by employing a Cellular Automata model adapts to diagrammatically depict the system's enlargement and how strain on the nodes is maintained throughout execution [2]. In this study, we experimented with several different configurations of parallel execution by initiating technique such as message passing interface on an identical node as well as on other nodes in order to achieve load balancing.

UIIMF: This multiobjective load balancing method allows applications to be dynamically customized in accordance with a number of objectives, including time, energy, communications, or combinations of these. It uses various methods, which can diverse the extra time, to achieve the goal of greater resource use at any given time [3]. Energy measurements have been shown to be particularly helpful in situations when workload is regular across iterations, and multi-objective approaches help in irregular workload situations.

Two common methods for dealing with load imbalance are making multiple copies of "hot" files, also known as "hot spots," and breaking large files into smaller ones using storage codes. Due to the memory overhead caused by unnecessary data or complex encoding/decoding, this approach is not ideal. For data parallel clusters, Selective Partitioning Cache is an effective cluster-caching approach. that distributes caching loads evenly and eliminates redundant caching nodes. It selectively divides hot files into multiple divisions based on their sizes and levels of popularity [4] so that read requests can be distributed evenly across different servers. Without frequent load balancing, SP-Cache is unable to handle sudden changes in popularity, such as spikes in access to specific files, even though it eliminates hot spots efficiently while minimising the impact of stragglers.

Comparison of two different strategies used to allocate task to the server with lesser load [5], result into better performance of content placement and delivery using a coded cache scheme. Although computationally this may not be viable in some real world settings, For a mesh network, expressions in closed form are created that almost perfectly balance the load without raising communication costs. Scheduling tasks in distributed, parallel context, towards a heuristics based design and development explores how tasks should be allocated across a network's many nodes.

Computing with a load-aware and long-term perspective (CALV) load balancing technique. Overloaded [6] servers cause CALV to prioritize blocks that contribute the most workload, while underloaded servers cause CALV to prioritize blocks that contribute the least. Load balancing efficiency is improved by CALV thanks to the incorporation of the slow data block transmission mechanism. In order to avoid overloading the destination servers and free up the peak network overhead for data reallocation, it selects a time for each data migration. By enhancing data locality while reducing tasks time, network traffic load, and cost of reallocation, CALV beats competing techniques. .

SITA is a size-based routing technique [7] used in highly dynamic, multi-server distributed queuing systems at scale. It is shown that as the size of the system grows to infinity, the mean waiting time achieved by a size-interval task assignment policy that equalises server loads converges to the minimum mean waiting time achieved by that policy. In spite of this, due to the heterogeneity of servers, SITA performance may decrease for no obvious cause.

Sandpile: This criticality approach for self-organized self-organization is used to dynamically load-balance computing workloads. The issue of scheduling independent jobs is addressed by extending the sandpile model [8]. The sandpile's decentralised execution may automatically adjust the amount of active resources to the specifics of incoming workloads. In order to reduce energy usage and increase service quality, the emerged load balance response of the system is investigated.

TATIM- It demonstrates that the NP-complete Knapsack problem is a variation on the scheduling of tasks with tasks significance for multi tasks transfer learning, requiring that a challenging computation be carried out repeatedly in various scenarios to solve [9]. Using a strategy for allocating cooperative tasks that is data driven, utilises 3.24 times less processing time due to which it is possible to reduce the amount of resources needed for multi tasks transfer learning on the edge.

FTAOA – In this method, each task is given a priority level based on the principle of earliest deadline first, and the task with the highest priority is given priority consideration. A novel fault resistant soft real time job allocation technique is mentioned in relation to wireless sensor networks [10]. The job allocation problem is solved using the DPSO approach in this research effort, and a utilisation function is developed to evaluate the nodes' overall performance.

TCHAP: A performance/energy variance technology-aware partitioning method that is mindful of the underlying cluster heterogeneity. Energy-efficient partitioning [11] is discussed as a software-level technique where task allocation to heterogeneous clusters directly impacts the energy consumption of the entire system. In addition to impacting the power consumption of single-ISA heterogeneous platforms, this problem combines the difficulties of task-aware scheduling with those of energy-efficient partitioning.

PathGraph - For quick iterative graph calculations on very large graphs, task distribution is achieved using a path centric approach. This approach demonstrates path centric abstraction at the computation and storage tiers [12]. The concept of a tasks queue with several stealing points is used in this study to allow work theft from various places

in the task queue. It demonstrates that for a variety of iterative graph algorithms that achieve better balance and speedup on real graphs of different sizes, the path graph defeats x-stream and graph chi.

EXA2PRO - The EXA2PRO approach optimises the distribution of tasks across available resources in heterogeneous computing environments. The runtime supports that oversees the applications-deployment on various nodes in heterogeneous environment and is constructed on top of complex processing models and abstractions that contains lower level platform specific improvements. More HPC applications can now scale to exascale [13] because it aids programmers in taking advantage of heterogeneous computer systems. It's compatible with CPUs, GPUs, and FPGAs, among other processors and accelerators. To test the efficacy of the EXA2PRO framework, four HPC applications from various disciplines were used. Developers can get performance results on accelerators, test scalability on MPI clusters, and efficiently investigate the extent to which each application can effectively use various types of hardware resources thanks to the EXA2PRO framework's ability to automatically deploy and evaluate applications on a wide range of computing architectures.

TaPRA: Algorithms for scheduling tasks to be completed as quickly as possible are used in task allocation. This is essential because it will increase efficiency and effectiveness when working in the cloud. In this research, we focus on the challenge of scheduling parallel jobs that consist of multiple discrete processes. The programmes TaPRA and TaPRA-fast aim to solve this problem by dividing the work between them. To manage a large number of tasks, online schedulers like OnTaPRA are employed [14]. The JCT can be lowered by as much as 430 percent with TaPRA and TaPRA-fast, and by as much as 60 percent on average with the OnTaPRA scheduler. In addition, TaPRA-fast is up to ten times faster than TaPRA, with only a five percent performance loss, making its implementation in practise very attractive.

Ant - Having jobs set up in the same way across different nodes is a major cause of workload inconsistency and subpar performance. Tasks should be set up in a different way to account for the abilities of heterogeneous nodes. In order to find the optimal configurations for each task running on each node, Ant uses a self-adaptive task tuning technique. Through the use of hardware features, Ant partitions a heterogeneous cluster into several smaller, more manageable clusters. The self-tuning procedure is then individually applied to each sub cluster while treating them as homogeneous clusters [15]. Randomly selecting configurations, Ant then configures tasks, gradually improving configurations by emulating those with the highest performance and discarding those with the lowest. During the adaptive task setup phase, Ant uses a genetic algorithm to speed up task tuning and avoid getting stuck in a local optimum.

Resource Allocator: This strategy makes use of fully connected system and graph convolutional system to determine the nearest optimum number of processor cores to speed up the execution of jobs [16]. The fully connected network uses the contigency in flow of work as well as excess factors like size of input, count of processor cores, the memory capacity, and the execution jobs count are serves as input for predicting execution time of job after this the contigency between operations from task scheduling is discovered by graph convolutional networks . The user can choose a close to ideal count of processor cores using the result of predicted job execution time.

Timed Loop Storage – To decentralise the data and utilise the idle capacity of local network lines, one IoT node must queue computational data packets that will later be processed in chunks by other nodes. Extra packets circulate through the network lines until they are needed for processing [17], while the sequenced packets are processed sequentially on the intended IoT device. Each node has a mechanism to handle data transport and prevent packet looping by diverting data at the right time. With this approach, the temporary storage capacity of the connected Internet of Things devices is scalable while still being practical for large datasets thanks to its low 45 Kb primary storage system requirement.

With ROSE, you can oversubscribe without sacrificing performance on a brand new platform designed from the ground up to manage your resources. ROSE enables the simultaneous execution of both latency-sensitive long-running applications and batch jobs with intensive computational requirements (LRAs). Job managers in ROSE can request the launch of speculative tasks in specific machines based on their suitability for oversubscription [18], rather than waiting for resource allocation to be validated by the centralised scheduler. However, by enforcing an admission control mechanism that makes use of performance-aware resource throttle and multi-resource threshold control, the node agents in these computers can prevent resource oversubscription.

Schedulers use adaptive scheduling to plan multiple, concurrent Spark Streaming micro-batch jobs and make proactive adjustments to scheduling parameters in order to optimise resource utilisation and performance in real time. To be more specific, A-scheduler intelligently modifies the degree of job parallelism and resource sharing among jobs based on workload parameters, and schedules many jobs concurrently using different policies based on their data dependencies [19]. The efficiency of the custom Spark Streaming system can be further enhanced with the help of dynamic batching with A-Scheduler. The duration of each batch interval is dynamically adjusted with the aid of an expert fuzzy control approach to account for variations in the streaming workload and the processing speed of the system over time.

In order to better allocate resources, Hypergraph Partition-based Scheduling has been upgraded with HPS+. HPS+ uses an enhanced hypergraph partitioning technique to model the intertwined dependencies between tasks and data as well as between datacenters, all with the goal of decreasing WAN traffic. In addition, it uses a coordination mechanism to allocate network resources in tight accordance with task requirements [20], which helps to reduce the makespan. When compared to conventional methods, HPS+ evaluation shows a 39% reduction in the makespan and a reduction of up to 53% in the number of data transfers.

Two distinct batch allocation strategies are layered and core-based. While the former can yield better performance using the hierarchy pattern as the main technique to create every conceivable batch, the latter may incur higher

computational cost due to the necessity of forming and observing each and every one of them. Using the latter method, which selects core tasks to form batches, suboptimal performance can be achieved with significantly reduced computational cost and complexity [21]. These strategies outperform alternatives in a number of respects, including the total amount paid by requesters, the average salary of workers, the likelihood that the job will be completed successfully, the amount of time spent on task allocation, and so on.

Resource-Constrained Replication Strategies - These strategies are used in a variety of situations, such as when the task's subtasks' completion times are determined by empirical or heterogeneous distributions, or when a job consists of hierarchical tasks that must be completed in a particular order as indicated by a task precedence graph. To examine how to distribute replication resources across the subtasks so that the overall task completion time is minimised, a unique framework is employed [22]. For integer programmes, this research develops a water-filling-like technique and a Lagrange multiplier-based method. It explores the tradeoff between cost and latency from introducing replications in a task graph and demonstrates the efficiency and optimality of algorithms through analysis and simulations.

Field Programmable Gate Array (FPGA): To share resources, the FPGA space is often divided into slots of a fixed size. Particularly when the number of activities rises, this leads to suboptimal resource usage and relatively poor performance. The exploratory capabilities of OpenCL are used to construct a revolutionary, area-sharing methodology that more successfully manages task resource requirements through intelligent clustering, bespoke, task-specific partitioning, and mapping [23]. Workload needs to increase temporal compute density, thus use models with various resource/throughput profiles and choose the most suitable distribution based on the runtime. A comparable task-based virtualization model supports the technique in the system stack.

Resource allocation policies - Three important considerations must be made when allocating a wide variety of system resources to a large number of users: user satisfaction for a lower average user response time; efficiency to maximise system throughput; and fairness among users. Using resource allocation policies for multiple users and multiple application workloads in a hierarchical computing system [24], the task is completed in the allotted time. Three competing policies fairness, greedy efficiency, and fair efficiency are being debated. The simulation results show that the fair efficiency policy may offer competitive efficiency with a level of fairness and user satisfaction that is balanced when compared to the other two resource allocation policies.

Energy Optimized Scheduling: In a non-preemptive heterogeneous environment, a series of periodic real-time tasks are scheduled using an active replication-based framework to ensure that the required reliability and timeliness constraints are met while the energy consumption is kept to a minimum. The issue is first presented as a constraint optimization problem, which offers an ideal resolution but has poor scalability [25]. Therefore, the suggested heuristics in this study work use processor reservations and job reallocation to compute suboptimal solutions effectively in terms of energy usage and scheduleability. Heuristics rely on the interaction between the task-level reliability target, replica reliability, replica number, task reliability, and energy consumption.

4. Heterogeneous System

Load balancing strategies are critical to improving application performance in diverse environments. Application performance can be improved with the help of load balancing heuristics, but this can be a difficult problem to solve if the necessary data is not available until after the application has already begun running. A load-balancing method called "Multi-Objective" because it allows applications to be dynamically tuned for a wide variety of objectives, including but not limited to any factors or their combinations such as communications, energy or time. The dynamic objective method is accommodated, which may change over time, in order to make the best use of the available resources at any given time. This is especially useful when using asynchronous measurements with varying degrees of accuracy on any objective [3].

When jobs are set up consistently across nodes with varying capabilities, performance suffers. Tasks should be set up in a variety of ways to account for the various capabilities of the nodes. In order to find the optimal configurations for individual jobs distributed across multiple nodes, Ant uses a self-adaptive task tuning method [15]. At first, Ant divides a cluster of mixed nodes into smaller, more manageable groups based on shared hardware features. It then applies the self-tuning process separately to each subcluster as if they were their own distinct cluster.

Table 1. Comparison of Various Load Balancing Techniques

Sr. No.	Algorithm	Description	Pros	Cons
1	Corder [1]	The technique works on the basis of pecking order of cache of multitasking computers by rearranging the task.	Encourages the equitable allocation of computing loads among multicores and at the level of private cache.	It only functions on vertices categorized by outgoing edges in terms of cross-platform scalability and reordering overhead.
2	Cellular Automata [2]	Applications linked to space are run concurrently using the CA Model's dynamic load balancing method.	Launching both MPI processes on the same node allowed for parallel execution to take place.	Depending on how well the particular formalization is done.
3	UIIMF [3]	For balancing load, the multiobjective technique is used in environments where the applications may undergo dynamic tuning.	It reveals the areas where the workload varies across iterations.	Effects of various technologies on techniques for dynamic load balancing in parallel applications.
4	Selective Partition [4]	Selective partitioning Cache, a load-balanced, redundant free cluster caching solution, is used by data-parallel clusters.	It separates hot files into various divisions based on their sizes and levels of popularity in order to evenly share the read request burden among different servers.	SP-Cache is unable to quickly respond to fluctuations in short-term popularity.
5	Coded Cache [5]	The employment of a coded content delivery and placement method outperforms the nearest replica strategy in terms of load balancing effectiveness.	It derives closed-form formulations for a grid network that successfully performs load balancing.	Requires a sophisticated coding and decoding method that, in some real-world situations, is not computationally possible.
6	CALV [6]	It uses the Long-View and Load Aware Computing load balancing approach.	To avoid overloading the destination servers and to free up the peak network overhead for data reallocation, it selects a time for each data movement.	Very high overhead is required because a very short time interval must be set for the periodic execution of load balancing.
7	SITA [7]	A subset of methods for size-based routing is used by large-scale multi-server distributed queuing systems with highly variable workloads.	It demonstrates how a Size-Interval Task Assignment strategy can achieve a minimum mean waiting time when all servers have the same processing speeds.	When servers are diverse, they perform arbitrarily poorly.
8	Sandpile [8]	In order to solve the problem of coordinating multiple activities, the sandpile model is applied.	It self-organizes to suit the specifics of incoming workloads, adjusting the amount of active resources.	use an endless supply of energy.

9	TATIM [9]	Through the application of multitask transfer learning, different tasks are combined to improve decision performance.	To support target tasks, it essentially reuses source task parameters or training samples.	Due to the many environmental setups and situations, task relevance is difficult to quantify.
10	FTAOA [10]	A new fault-tolerant allocation algorithm (FTAOA) for soft real-time tasks has been developed.	Each task is given a priority level based on the principle of earliest deadline first, and the task with the highest priority will be taken into consideration first.	uses active backup over-lapping technology in addition, which causes unneeded redundancy.
11	TCHAP [11]	The TCHAP partitioning algorithm is employed.	It combines the issue of task-aware scheduling with the challenge of energy-efficient partitioning on single-ISA heterogeneous platforms.	Affects power demand
12	PathGraph [12]	Fast iterative graph computations are performed using the path-centric method called PathGraph.	Work can be stolen from numerous points in the task queue using a task queue that is built on multiple theft points.	Causing access conflicts and imbalance loads.
13	EXA2PRO [13]	The runtime supports that oversees the applications-deployment on various nodes in heterogeneous environment and is constructed on top of complex processing models and abstractions that contains lower level platform specific improvements.	enables developers to test scalability on MPI clusters, acquire performance findings on accelerators, and productively examine how effectively each programme can utilise various hardware resources.	do not always match the application's computational pattern.
14	TaPRA [14]	It is crucial that each task's completion time be as short as possible when it comes to task allocation in IT because this will boost cloud resource utilisation and productivity.	cuts the job's completion time by 40% to 430%.	It is limited to single job scheduling.
15	Ant [15]	Based on the hardware specifications of the nodes, Ant first separates them into a number of homogenous sub clusters. The self-tuning procedure is subsequently individually applied to each sub cluster while treating them as homogenous clusters.	reproduces the settings from the best-performing tasks and discards the configurations from underperforming jobs to improve task configurations.	In multi-tenant clouds, job skew and varied hardware capabilities caused by interferences may make it difficult to create effective task setups.
16	ReLocag [16]	Two distinct kinds of neural network a fully connected one and a graph convolutional network are implemented in ReLocag approach. Once the graph convolution network has learned the relationship between tasks in a job's workflow, it passes that information along to the fully connected network, which then uses it in conjunction with other factors for prediction of job execution time.	The accuracy in predicting job execution time is increased by 4–14% using ReLocag approach.	Requires More power consumption.

21	Layered batch allocation and core-based batch allocation [21]	The earlier approach, which has the potential for higher performance, primarily forms all practicable batches using the hierarchy structure.	Balances load efficiently.	greater computing cost due to the formation and observation of all potential batches;
22	Resource-Constrained Replication Strategies [22]	The investigation of how to distribute replication resources across the subtasks to reduce the total task completion time makes use of a novel framework.	obtaining the best replication allocations among the tasks/subtasks to reduce the time taken to complete the task/job.	Storage capacity and Processing speed are not considered.
23	Field Programmable Gate Array [23]	The FPGA space is often divided into fixed-sized slots for resource sharing.	superior energy efficiency compared to current methods.	results in less efficient use of resources and generally inferior performance, especially as the number of tasks rises.

Using a random selection method, Ant configures tasks and iteratively improves configurations by emulating the settings of well-performing tasks and discarding the settings of poorly performing tasks [15]. Nonetheless, the main purpose of load balancing is to maximize resource utilization, which speeds up the computation process as a whole [2].

Two tried and true methods known as Dynamic Power Management & Dynamic Voltage and Frequency Scaling that can be used to cut energy use on heterogeneous platforms. Methods for reducing energy consumption at both the hardware and software levels should be used in tandem for optimal efficiency. Allocating tasks to heterogeneous clusters is one software-level strategy where the overall system energy is affected directly [11].

The goal is to prevent resource abuse by enforcing scheduling invariants across applications that run on different platforms. It forgoes efficient resource management in favour of guaranteed fair scheduling and efficient task completion [18].

Heterogeneous Island and Task-Aware Largest-Possible-Job Proposal The first approach [11] clusters tasks into heterogeneous groups at the maximum frequency of each group, and then gradually reduces that maximum frequency without compromising on practicality. They employ a method of energy conservation that takes into account the remaining capacity of the central node in the cluster that is under the most pressure. The following are some thoughts on where node coordination research could go from here, given the many challenges that have been encountered in earlier studies.

• The fixed roles of nodes during task allocation and execution in recent related studies do not accurately reflect reality. Nodes can actually alter their responsibilities inside the system dynamically. The coordination of dynamically transformable nodes in job allocation is thus a research challenge.

• According to recent studies, work distribution is accomplished through maximising throughput and improving resource usage. However, some machines or nodes might form a cluster or a community, and these nodes might carry out some associated work allocation algorithms within a cluster or a community.

Conclusion

In this study, we reviewed a variety of load-balancing approaches and tactics used in distributed computing. A load balancing strategy works in two ways: first, it distributes a lot of concurrent requests or data traffic among several nodes in order to speed up response times; and second, it distributes a single high load over several nodes in order to maximize resource efficiency on each node. As application settings change, task allocation strategies also change. Because cellular automata models are both representative of spatially-related applications and amenable to testing load balancing strategies, they were taken into account when designing the strategy of balancing load dynamic for execution of such approaches simultaneously. An even distribution of work during parallel execution [2] is guaranteed by developing closed form formulas for determining the best segregation of space. We have also spoken about the advantages and disadvantages of various load balancing approaches. So that in the future, more effective load balancing solutions might be created, the difficulties of these techniques are addressed. Different techniques for balancing load discussed in this paper not only serve to balance the load but also to achieve efficient resource use, which increases total throughput and cuts reaction time. Thus, all of these will increase consumer interest in parallel and distributed computing while simultaneously lowering operating expenses.

References

[1] Mrs. Minal Shahakar, Dr. Surenda Mahajan, Dr. Lalit Patil, “Load Balancing in Distributed Cloud Computing: A Reinforcement Learning Algorithms in Heterogeneous Environment”, International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 11, Issue 2, 2023.

[2] YuAng Chen and Yeh-Ching Chung, “Workload Balancing via Graph Reordering on Multicore Systems”, IEEE Transactions on Parallel and Distributed Systems, Vol. 33, No. 5, May 2022.

[3] Andrea Giordano, Alessio De Rango, Rocco Rongo, Donato D’Ambrosio, and William Spataro,

"Dynamic Load Balancing in Parallel Execution of Cellular Automata", IEEE Transactions on Parallel and Distributed Systems, Vol. 32, No. 2, February 2021.

[4] Alberto Cabrera, Alejandro Acosta, Francisco Almeida, and Vicente Blanco, "A Dynamic Multi–Objective Approach for Dynamic Load Balancing in Heterogeneous Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 10, October 2020.

[5] Yinghao Yu , Wei Wang , Renfei Huang , Jun Zhang , and Khaled Ben Letaief, "Achieving Load-Balanced, Redundancy-Free Cluster Caching with Selective Partition", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 2, February 2020.

[6] Mahdi Jafari Siavoshani , Farzad Parvaresh , Ali Pourmiri , and Seyed Pooya Shariatpanahi, "Coded Load Balancing in Cache Networks", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 2, February 2020.

[7] Guoxin Liu, Haiying Shen, and Haoyu Wang, "Towards Long-View Computing Load Balancing in Cluster Storage Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 28, No. 6, June 2017.

[8] Jonatha Anselmi and Josu Doncel, "Asymptotically Optimal Size-Interval Task Assignments", IEEE Transactions On Parallel And Distributed Systems, Vol. 30, No. 11, November 2019.

[9] Juan Luis Jimenez Laredo, Frederic Guinand, Damien Olivier, and Pascal Bouvry, "Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing", IEEE Transactions On Parallel And Distributed Systems, Vol. 28, No. 2, February 2017.

[10] Qiong Chen, Zimu Zheng, Chuang Hu, Dan Wang, and Fangming Liu, "On-Edge Multi-Task Transfer Learning: Model and Practice With Data-Driven Task Allocation", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 6, June 2020.

[11] Wenzhong Guo, Jie Li, Guolong Chen, Yuzhen Niu, and Chengyu Chen, "A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks", IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 12, December 2015.

[12] Ashraf Suyyagh and Zeljko Zilic, "Energy and Task-Aware Partitioning on Single-ISA Clustered Heterogeneous Processors", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 2, February 2020.

[13] Pingpeng Yuan, Changfeng Xie, Ling Liu, and Hai Jin, "PathGraph: A Path Centric Graph Processing System", IEEE Transactions On Parallel And Distributed Systems, Vol. 27, No. 10, October 2016. Design And Development of An Efficient Approach for Task Allocation in Distributed Systems Using Heuristics Environment 26 | SKNCOE, Research Centre - Computer Engineering 2022

[14] Lazaros Papadopoulos, Dimitrios Soudris, Christoph Kessler, August Ernstsson, Johan Ahlqvist, Nikos Vasilas, Athanasios I. Papadopoulos, Panos Seferlis, Charles Prouveur, Matthieu Haefele, Samuel Thibault, Athanasios Salamanis, Theodoros Ioakimidis, and Dionysios Kehagias, "EXA2PRO: A Framework for High Development Productivity on Heterogeneous Computing Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 33, No. 4, April 2022.

[15] Li Shi, Zhemin Zhang, and Thomas Robertazzi, "Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud", IEEE Transactions On Parallel And Distributed Systems, Vol. 28, No. 6, June 2017.

[16] Dazhao Cheng, Jia Rao, Yanfei Guo, Changjun Jiang, and Xiaobo Zhou, "Improving Performance of Heterogeneous MapReduce Clusters with Adaptive Task Tuning", IEEE Transactions On Parallel And Distributed Systems, Vol. 28, No. 3, March 2017.

[17] Zhiyao Hu , Dongsheng Li, Dongxiang Zhang , Yiming Zhang , and Baoyun Peng "Optimizing Resource Allocation for Data-Parallel Jobs Via GCN-Based Prediction", IEEE Transactions On Parallel And Distributed Systems, Vol. 32, No. 9, September 2021.

[18] Anandarup Mukherjee, Pallav Kumar Deb, and Sudip Misra, "Timed Loops for Distributed Storage in Wireless Networks", IEEE Transactions On Parallel And Distributed Systems, Vol. 33, No. 3, March 2022.

[19] Renyu Yang, Chunming Hu, Xiaoyang Sun, Peter Garraghan , Tianyu Wo, Zhenyu Wen, Hao Peng , Jie Xu, "Performance-Aware Speculative Resource Oversubscription for Large-Scale Clusters", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 7, July 2020.

[20] Dazhao Cheng, Xiaobo Zhou, Yu Wang and Changjun Jiang, "Adaptive Scheduling Parallel Jobs with Dynamic Batching in Spark Streaming", IEEE Transactions On Parallel And Distributed Systems, Vol. 29, No. 12, December 2018.

[21] Laiping Zhao, Yanan Yang, Ali Munir, Alex X. Liu, Yue Li, and Wenyu Qu, "Optimizing Geo-Distributed Data Analytics with Coordinated Task Scheduling and Routing", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 2, February 2020.

[22] Jiuchuan Jiang, Bo An, Yichuan Jiang, Senior Member, IEEE, Peng Shi, Zhan Bu, and Jie Cao, "Batch Allocation for Tasks with Overlapping Skill Requirements in Crowdsourcing", IEEE Transactions On Parallel And Distributed Systems, Vol. 30, No. 8, August 2019.

[23] Weng Chon Ao and Konstantinos Psounis, "Resource-Constrained Replication Strategies for Hierarchical and Heterogeneous Tasks", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 4, April 2020.

[24] Umar Ibrahim Minhas, Roger Woods, Dimitrios S. Nikolopoulos, and Georgios Karakonstantis, "Efficient

Dynamic Multi-Task Execution on FPGA-Based Computing Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 33, No. 3, March 2022. Design And Development of An Efficient Approach for Task Allocation in Distributed Systems Using Heuristics Environment 27 | SKNCOE, Research Centre - Computer Engineering 2022

[25] Eunji Hwang, Suntae Kim, Tae-kyungYoo, Jik-Soo Kim, Soonwook Hwang, and Young-ri Choi, "Resource Allocation Policies for Loosely Coupled Applications in Heterogeneous Computing Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 8, August 2016.

[26] Niraj Kumar, Jaishree Mayank, and Arijit Mondal, "Reliability Aware Energy Optimized Scheduling of Non-Preemptive Periodic Real-Time Tasks on Heterogeneous Multiprocessor System", IEEE Transactions on Parallel and Distributed Systems, Vol. 31, No. 4, April 2020.

[27] Myeonggyun Han , Jinsu Park , and Woongki Baek, "Design and Implementation of a Criticality and Heterogeneity-Aware Runtime System for Task-Parallel Applications", IEEE Transactions on Parallel and Distributed Systems, Vol. 32, No. 5, May 2021.

[28] Jiaying Meng, Haisheng Tan, Xiang-Yang Li, Zhenhua Han, and Bojie Li, "Online Deadline-Aware Task Dispatching and Scheduling in Edge Computing", IEEE Transactions on Parallel and Distributed Systems, Vol. 31, No. 6, June 2020.

[29] Hafiz Fahad Sheikh, Ishfaq Ahmad, and Dongrui Fan, "An Evolutionary Technique for Performance-Energy-Temperature Optimized Scheduling of Parallel Tasks on Multi-Core Processors", IEEE Transactions On Parallel And Distributed Systems, Vol. 27, No. 3, March 2016.

[30] Mark A. Oxley, Sudeep Pasricha, Anthony A. Maciejewski, Howard Jay Siegel, Jonathan Apodaca, Dalton Young, Luis Briceno, ~Jay Smith, Shirish Bahirat, Bhavesh Khemka, Adrian Ramirez, and Yong Zou, "Makespan and Energy Robust Stochastic Static Resource Allocation of a Bag-of-Tasks to a Heterogeneous Computing System", IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 10, October 2015.

[31] Peiquan Jin, Xingjun Hao, Xiaoliang Wang, and Lihua Yue, "Energy-Efficient Task Scheduling for CPU-Intensive Streaming Jobs on Hadoop", IEEE Transactions On Parallel And Distributed Systems, Vol. 30, No. 6, June 2019.

[32] Mark A. Oxley, Sudeep Pasricha, Anthony A. Maciejewski, Howard Jay Siegel, Jonathan Apodaca, Dalton Young, Luis Briceno, Jay Smith, Shirish Bahirat, Bhavesh Khemka, Adrian Ramirez, and Yong Zou, "Makespan and Energy Robust Stochastic Static Resource Allocation of a Bag-of-Tasks to a Heterogeneous Computing System", IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 10, October 2015.

[33] Li Chen , Yuan Feng, Baochun Li and Bo Li, "Efficient Performance-Centric Bandwidth Allocation with Fairness Tradeoff", IEEE Transactions On Parallel And Distributed Systems, Vol. 29, No. 8, August 2018.

[34] Haitao Zhang , Xin Geng, and Huadong Ma, "Learning-Driven Interference-Aware Workload Parallelization for Streaming Applications in Heterogeneous Cluster", IEEE Transactions On Parallel And Distributed Systems, Vol. 32, No. 1, January 2021.

[35] Amin Yoosefi and Hamid Reza Naji, "A Clustering Algorithm for Communication-Aware Scheduling of Task Graphs on Multi-Core Reconfigurable Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 28, No. 10, October 2017.

[36] Jalal Khamse-Ashari , Ioannis Lambadaris, George Kesidis, Bhuvan Urgaonkar, "An Efficient and Fair Multi-Resource Allocation Mechanism for Heterogeneous Servers", IEEE Transactions On Parallel And Distributed Systems, Vol. 29, No. 12, December 2018.

[37] Hafiz Fahad Sheikh, Ishfaq Ahmad, and Dongrui Fan, "An Evolutionary Technique for Performance-Energy-Temperature Optimized Scheduling of Parallel Tasks on Multi-Core Processors", IEEE Transactions On Parallel And Distributed Systems, Vol. 27, No. 3, March 2016.

[38] Myeonggyun Han , Jinsu Park , and Woongki Baek, "Design and Implementation of a Criticality-and Heterogeneity-Aware Runtime System for Task-Parallel Applications", IEEE Transactions On Parallel And Distributed Systems, Vol. 32, No. 5, March 2021.

[39] Tamilselvi, T. ., Lakshmi, D. ., Lavanya, R. ., & Revathi, K. . (2023). Digital Companion for Elders in Tracking Health and Intelligent Recommendation Support using Deep Learning. International Journal on Recent and Innovation Trends in Computing and Communication, 11(3), 145–152. https://doi.org/10.17762/ijritcc.v11i3.6331

[40] Dwarkanath Pande, S. ., & Hasane Ahammad, D. S. . (2022). Cognitive Computing-Based Network Access Control System in Secure Physical Layer. Research Journal of Computer Systems and Engineering, 3(1), 14–20. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/36