ISSN 2007-9737

Resource Search in HPC Systems using Lévy Flights

Apolinar Velarde Martínez

Instituto Tecnológico El Llano Aguascalientes,
Departamento de Sistemas y Computación,
Mexico

apolinar.vm@llano.tecnm.mx

Abstract. Parallel applications represented by Directed Acyclic Graphs (DAGs) as Parallel Task Graphs (PTGs) requiring high execution times with large amounts of storage, are executed on High Performance Computing (HPC) Systems such as clusters. For the execution of these applications, a scheduler performs the scheduling and allocation of the resources contained in the HPC System. One of the activities of the scheduler is the search for idle resources that are geographically dispersed in the clusters to schedule and allocate them to the tasks. The search for idle resources in the clusters is a process that consumes time and system resources due to the geographical distances that must be traveled and the repetitive and permanent execution within the system. A sequential search for resources causes the scheduling and allocation of resources in the HPC System to be slowed down and paused, and increases the waiting times of the tasks that remain in the queue. The techniques that shorten the location times of resource and perform more exhaustive searches in dispersed geographic spaces can reduce the times generated by sequential searches. The open access paper: Scheduling in Heterogeneous Distributed Computing Systems Based on Internal Structure of Parallel Tasks Graphs with Meta-Heuristics presents the Array Method, a scheduler for scheduling and allocation resources in an HPC System. Array Method uses a sequential search process for the idle resources that are geographically dispersed in the clusters and save their location and characteristics in an array, which is updated every time idle resources are located in the clusters. Considering the above, this paper presents a search for idle resources using Lévy random walks, a technique used for searching resources in large geographical spaces; this technique avoids sequential node-by-node searches of each cluster, and promote short and long range searches over the entire geographical extent of the HPC Systems. To obtain experimental results, sequential resource search algorithm versus Lévy random walks with the synthetic loads, different clusters and different numbers of resources per cluster as proposed in the open access paper aforementioned, are used. Obtained results show Lévy random walks locates more idle resources in less time, optimizes the times of the resource searches in the clusters and update the array of available resources more frequently. With more idle resources found, the total execution times of the tasks are reduced.

Keywords. High performance computing systems, clusters, Lévy flights, scheduling resources, allocation resources.

1 Introduction

Parallel tasks represented by Directed Acyclic Graphs (DAGs) referenced as Parallel Task Graphs (PTGs) and constituted by a set of subtasks of scientific and enterprise applications, require large amounts of processor and storage space during their execution in High Performance Computing Systems (HPC Systems) [1], such as clusters [10], in order to achieve desired level of service [16] to users. Clusters or cluster computing refers to the use of interconnected computers geographically distributed [19, 8] as a high-performance computing platform and has become wide-spread [10, 3].

Often the computers used in a cluster are “commodity” computers, that is, low-cost personal computers as used in the home and office [3]; they consist of PCs heterogeneous, running Linux, and connected with Ethernet, and referred to as Beowulf clusters; since the processors are

independent they are examples of the MIMD (Multiple Instruction, Multiple Data) or SPMD (Single Program, Multiple Data) model [7]. In this work, the processor refers to an individually programmable computer resource [5] that can be assigned to a PTG and in the rest of the paper are referred to as nodes, processing elements (PE) and resources. Resource is a collection of components that can be scheduled to perform an operation by an application [15]; some traditional examples of resources are CPU cores for computing, memory spaces for storage, network links for transferring, electrical power, and so on [18].

When Parallel Tasks Graphs in a cluster are executed, a scheduler performs the scheduling and the allocation resources [10]. The scheduling is performed in two phases: first, a search for idle resources in all clusters is performed to determine the positions of the PEs remain idle; the search for idle resources can be performed by a resource allocation control which is a mechanism that manages and control resources in a cluster system [10]. Second a search for the best allocations to optimize the execution times of the tasks is performed.

In [21], the open access article, an array-based scheduler (called the Array Method) for scheduling and resource allocation in clusters is explained. For scheduling, the Array Method uses an array of idle resources or processors (resource array) that is updated as follows, each time a cluster is added to the target system, a task finishes executing or a PE of a cluster is added as a system resource; for locating idle resources, a sequential search is performed on each of the clusters of the HPC System.

The search is a scan that is performed for each of the processors belonging to the clusters, generally initiated on a central node and executed by a Software Agent (SA); the SA initiates the scan on all the processors belonging to the cluster; upon finishing with a cluster, the SA advances to the neighboring clusters and starts again a similar scanning process, processor by processor; every so often SA informs the central node the number of visited processors and activates a process to update the idle resource matrix (resource matrix). The SA resides in the central node of the HPC Systems and because resources can change from a busy state to an idle state at any time, SA should start the sequential search for idle resources in the clusters and update the resource matrix periodically and continuously. Both processes become fundamental and necessary. Finally, for the assignments of idle resources to tasks, a sequential search is executed in the resource matrix and the process of searching is initiated for the best assignments.

Considering the results of the execution times generated by the matrix method and an analysis of each process executed by the scheduler, the most time-consuming process is the sequential scanning of idle resources in the clusters. In order to improve the resource search times in the clusters that constitute the target system, this paper present a technique of idle resources search with Lévy flights, a special class of random walks that have been investigated in different works [19, 15, 22, 4, 13, 23, 11] as a resource search technique in locations geographically distributed; in the following paragraphs the development and implementation of this technique is justified. The term Lévy random walks is used to refer Lévy flights by context in which this technique is used in this research work.

1.1 Justification of the Proposed Method

The use of a technique of resource search in a HPC System, different from sequential search with acceptable service levels, is justified by the next conditions:

— Search for idle resources, is a process that is executed periodically, constantly and permanently that results in a higher consumption of time and resources of the target system, therefore, it must optimize the search for free resources (locate the largest amount of resources each time it is executed).

— Resource searches must be executed in different clusters; the clusters are geographically distributed [19, 8], to meet the demands of geographically distributed applications [1].

— When searching for resources on the target system, it should not be limited by a sequential search on adjacent clusters, but should be able to migrate to non-adjacent clusters if neighboring clusters are disabled.

— Because the allocation of new nodes in the infrastructure can occur at any time [8]. Each time a cluster is added to the system, the distances for searching and extracting features from the target system resources gradually increase; with the ever increasing size of systems, the task scheduling problem has become more challenging and complex [15].

— The search for resources is considered a vital process to avoid sending a subtask to an inactive node of a cluster, and to assume that the subtask starts its execution. This can occur because a resource may join or leave the network at any time due to dynamic nature of resources [19].

For the aforementioned, in this research work an idle resource search technique with Lévy flight is proposed and experiments that compares the proposed technique with sequential search algorithm for idle resources are accomplish. The synthetic loads proposed in [21] with different number of clusters and different number of resources per cluster was used during the experiments. The results obtained show that the Lévy random walks allow more idle resources to be taken in shorter periods of time, optimize the search times of the resources in the clusters and allow the array of available resources of the scheduler to have constant updates in shorter periods of time (as explained in the experiments section).

Note that the purpose of this research work is the performance analysis of the sequential search of resources versus Lévy flights search and not scheduling tasks in HPCS.

This paper is organized as follows: in section 2, four additional definitions to the Basic Definitions section of [21] are proposed; in section 3, problem statement is established, section 4 addresses the works related to this research; in section 5, a very brief explanation of Array Method, proposed in [21] is presented; section 6, how Lévy random walks are used in this work and how is implemented the search for idle resources in HPC System using Lévy random walks are described; the experiments performed are presented in section 7; finally the conclusions and future works can be found in sections 8 and 9 respectively.

2 Basic Definitions

In this paper, the section of basic definitions of [21] is used and the following new definitions are proposed.

Definition 1. The target system consists of C_l clusters, $C_1, C_2, ..., C_l$ where l is the number of clusters contained in the HPC System. Each cluster contains m heterogeneous processors with n processing cores, then $C_{l,m,n}$ is cluster k, processor m, processing core n. In this way each $C_{l,m,n}$ represents a resource that is identified as $R_{l,m,n}$ and can be used by the scheduler.

Definition 2. Let to consider definition 2 from [21]; it follows that each PTG will request η_i resources, which must be extracted from the scheduler's resource array, then the next condition is established:

$$\forall\ \eta_i\ \exists\ R_{l,m,n}. \tag{1}$$

For the PTG to complete its execution.

Definition 3. A Lévy flight is a type of random walk in which the increments are distributed according to a "large tail" probability distribution. Specifically, the distribution used can be approximated by a power law of the form [23]:

$$P(l)\ \propto\ l^{-\mu}\ with\ 1 < \mu \leq 3, \tag{2}$$

where μ is a constant parameter of the distribution known as the exponent or scaling parameter.

Definition 4. Let a software agent SA, which locates any idle $R_{l,m,n}$ found in any C_l using equation 2, then registers it in an array of resources and condition 1 is satisfied.

3 Problem Statment

Let a scheduler running on an HPC System consisting of C_l clusters, $C_1, C_2, ..., C_l$ where l is the number of clusters of the HPC System. Each cluster contains m heterogeneous processors with n processing cores; for searching $R_1, R_2, ..., R_n$ idle resources, a search engine S operated by a software agent (SA), is executed using equation 2.

Then: respecting the PTGs execution constraints, β must be executed continuously and periodically at times $t_1, t_2, ..., t_n$, where t_n, is the time where the scheduler is active in the target system. Any $R_1, R_2, ..., R_n$ idle resource located by S must be stored in an array of dimension $N \times M$ and be available for use at any time t_n.

4 Related Works

In a plethora of literature, research works that use idle resources for scheduling and allocation of tasks in an HPCS, assume a set of resources that are available in a resource pool as in [19, 15]; based on required characteristics, the resource discovery mechanism searches and returns the addresses of the resources that match with the provided descriptions [15], in [1] resources are scheduled prior to path setup request, the allocator has a resources' metadata [2]; in [20] assume that each user contributes a certain number of machines (resources) to its common pool of machines (resources) in the cloud, [8] only proposes a resource management system manage a pool of processing elements (computers or processors) which is dynamically configured (i.e., processing elements may join or leave the pool at any time); other research works such as [14] establishes a model where capabilities of the machines are known: the number of cores, the amount of memory, the disk space, and the host OS running, and supposes that a large data center and cloud systems already have significant monitoring tools that provide near-real-time updates of various systems to their controllers. In [18] the resources are considered to be available on demand, charged on a pay-as-you-go basis, and in one aspect, cloud providers hold enormous computing resources in their data centers, while in the other aspect, cloud users lease the resources from cloud providers to run their applications; [9] the cloud service end user can use the entire stack of computing services, which ranges from hardware to applications. [17] supposses a cloud of resources where tasks are scheduled, the resources are heterogeneous and characterized by power and cost constraints. With the resources, metrics such as makespan, throughput [15], waiting time [19, 16, 10], overall mean task response time [8], load balancing [12] are sought to improve in HPC Systems.

In contradistinction to the works described above, in this research work any set of resources for the schedule are not assumed, but proposes the development of idle resources search engine operated by a software agent; software agent executes a resource discovery procedure [19, 15], which allows visiting a set of available clusters in the shortest possible time.

Considering Lévy random walks [22, 4, 6] as a prominent area of research in various disciplines from ecology to physics [23], and a special class of random walks whose stride lengths are not constant but are selected from a probability distribution with a power law [22, 4, 13, 23], in addition to the hypothesis that Lévy random walks are optimal when exploring unpredictably distributed resources [13] have been proposed in this work as a strategy of search for idle resources in an HPC System. Similarly, Levy's random walks are used in this work as a search strategy, because due to the divergence of the variance, extremely long jumps can occur and the typical trajectories are self-similar in all scales, showing groups of jumps shorter interspersed [6].

5 Array Method

The array method [21], is a dynamic scheduler for scheduling and allocating tasks in an HPC system. The operation of this scheduler is based on a set of arrays that store the results generated from each function. For each of the arrays, iterative processes or loops are executed. A very brief review of the data structures used by the Array Method is presented below.

— The resource matrix. The values of this matrix are obtained through an iterative process of resources searching in the HPC system.

— The matrix of characteristics of the PTGs. The PTG characteristics array stores the characteristics extracted from each PTG using the Depth First Search Algorithm (DFS Algorithm)

— The allocation matrix, is a dynamic array that stores the values produced by the resource allocation algorithm.

— The matrix of task start times. Once the algorithm determines the best assignment for the PTG, start times for each of the PTG subtasks are calculated; these values are stored in the matrix of task start times.

The software agent SA, is executed over all clusters; all characteristics of idle resources found are stored in resource matrix, i.e., when the SA is executed, it only updates the resource matrix so the planning and allocation process can use idle resources.

The Software Agent execution has been defined so far. On the next paragraphs Lévy random walks and its SA interaction with, will be defined; how this interaction operates over the clusters are defined too.

6 Lévy Random Walks

This section defines how Lévy random walks are used in this research work. The exponential increase in step length gives the Lévy flight the property of being scale invariant, and they are used to model data exhibiting clustering. Lévy flights are a special class of random walks whose step lengths are not constant, but are selected from a probability distribution with a power law [22, 4, 13, 23]. The random walk is represented by a succession of random variables X_n with $n \in N$ known as a discrete stochastic process. The sequence $X_n, n \in N$ forms infinite sequences $X_0, X_1, ..., X_i$ with $X_i \in Z$. That is, if a run starts at state 0, at the next time the agent can move to position $+1$, with probability p or to position -1 with probability q, with $p + q = 1$. Position $+1$ is considered the PE to the left of the current PE and position -1 is considered the PE to the right of the current PE.

In this research work we analyze the case of a software agent which starts from a specific cluster and PE, and moves in stages or steps along the target system, at each step it moves a unit distance to the right or to the left, with respectively equal probabilities. For the movement of the agent, let ζ_n as the nth motion of the agent and P as the probability function then,

1. $P(\zeta_n = +1) = p$ the agent moves one PE to the right of the current position.
2. $P(\zeta_n = -1) = q$ the agent moves one PE to the left of the current position.

ζ_n are considered as the independent random variables.

It is further assumed that the agent moves k steps in total, before registering idle resources in the resource matrix (see section 5). For these experiments, Lévy flights are only used as a search procedure for idle resources, not as a prediction algorithm. The steps for the execution of the software agent are explained on next section (a reduced algorithm is provided in Table 1).

6.1 Search for Idle Resources in HPC System using Lévy Random Walks

For the resource allocation in HPC Systems, most of research works (as Related Works section explains) assume a centralized resource manager that has a complete vision of network topology, as well as networking and computing resources status; this assumption is not valid for large-scale worldwide grid networks; practically, grid network comprises geographically distributed heterogeneous resources interconnected by multi-domains networks [1]. In this paper, before performing the scheduling and allocation resources, a resource search engine is proposed in HPC Systems, using Lévy random walks to extract the characteristics of each processing element, assuming that ignoring computational resources capacity and availability may affect the overall performance

significantly specially in computational intensive applications [1]. To generalize the problem, lets consider the figure 1, which has a set of geographically dispersed clusters linked by wireless and wired communication; figure 1 is the most similar to actual architecture used in the experimentations. For reasons of space, details about network bandwidth, multidomain environments, interdomain, and intradomain topology are not specified, in addition to how the different domains interact to provide end-to-end connectivity.

On figure 1 circles represent processing elements; filled circles show occupied PEs, circles filled with lines represent PEs without functionality, while unfilled PEs are idle at time $t + 1$ and can be located by the software agent. Links between processing elements within a cluster and links between clusters are represented by solid lines. The three dots above the solid lines show the possible addition of new clusters to the target system.

The example showed in the Figure 1 up operates like this: the resources search for in the HPC System using Lévy random walks, is shown with a dotted line; the Lévy random walk executed by software agent using algorithm showed in Table 1, starts in cluster 1, executes a set of steps for searching the PEs on this cluster, migrates to cluster 2, 3, 4 and 5; on the cluster 5 a long jumps is executed. The search process is repeated for the time set in the algorithm; at the end of the random walk, twenty idle resources are identified by the SA, resource matrix is updated and resources are available for scheduling and allocation.

Considering the algorithm proposed in [11], Table 1 shows a reduced structure of the resource search algorithm using Lévy's random walks, operated by the software agent. On next sections the results of the experiments performed are described.

7 Experiments

Target System. Experiments are performed on 450 desktop computers and a server farm with 10 servers. Clusters between 5, 10 and 20 computers were built and distributed in different buildings within the campus.

Algorithm 1. Algorithm for resources search

$Input:\ Starting\ position\ for\ search\ in\ HPC$
$Output:\ List\ of\ idle\ resources\ in\ HPC$
$Begin$
  $Assigns\ search\ time\ by\ user;$
  $Identify\ starting\ position\ of\ SA\ in\ HPC;$
    $Location\ of\ last\ search;$
    $Start\ search\ from\ cluster\ identified\ as\ starting\ cluster;$
  $While\ (Search\ time\ exists)$
    $Calculate\ movement\ of\ SA\ using\ equation\ 2;$
    $Identify\ and\ verify\ status\ of\ PE\ where\ SA\ is\ to\ be\ moved;$
    $If\ (PE\ Status\ ==\ Idle)$
      $Save\ position\ and\ characteristics\ of\ PE;$
    End_if
  End_While
  $Update\ resource\ matrix$
end

The server farm is considered the initial cluster and where the task queue remains.

Agent Software. The software agent is programmed with C language, and its movements are allowed through C shell programming within the target system. Once the agent is transported to a PE, the execution of the agent is performed with a Linux Cron utility programmed in each EP to detect the arrival or presence of the agent in the PE.

Considering the purpose of this survey is the performance analysis of the resources search techniques, two metrics for the experiments in this paper are used:

— The percentages of resources located after established number of executions by each algorithm (Lévy random walks and sequential search) in the target system.

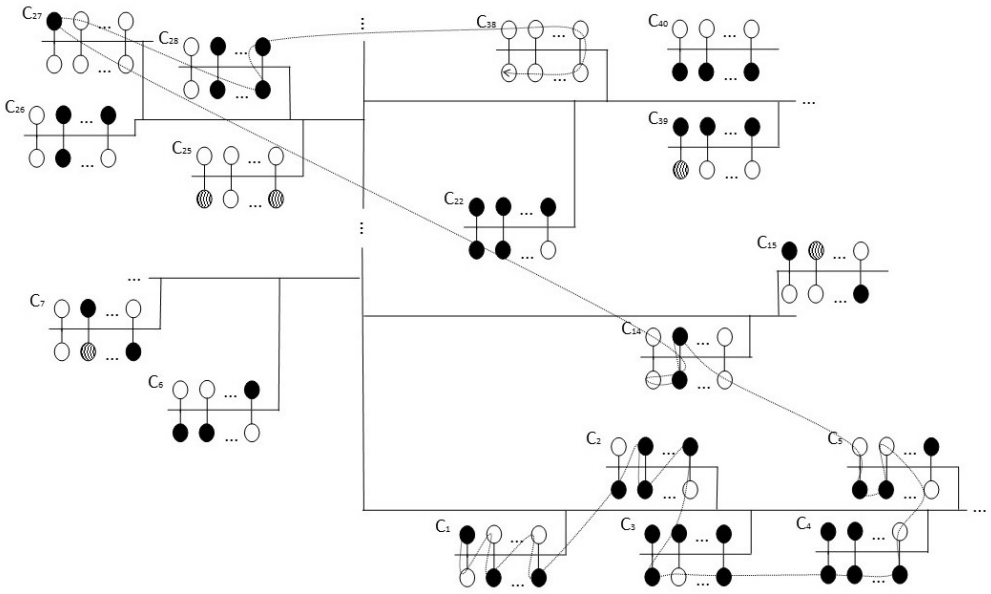

Fig. 1. HPC Systems with clusters showing a search for idle resources using Lévy random walks

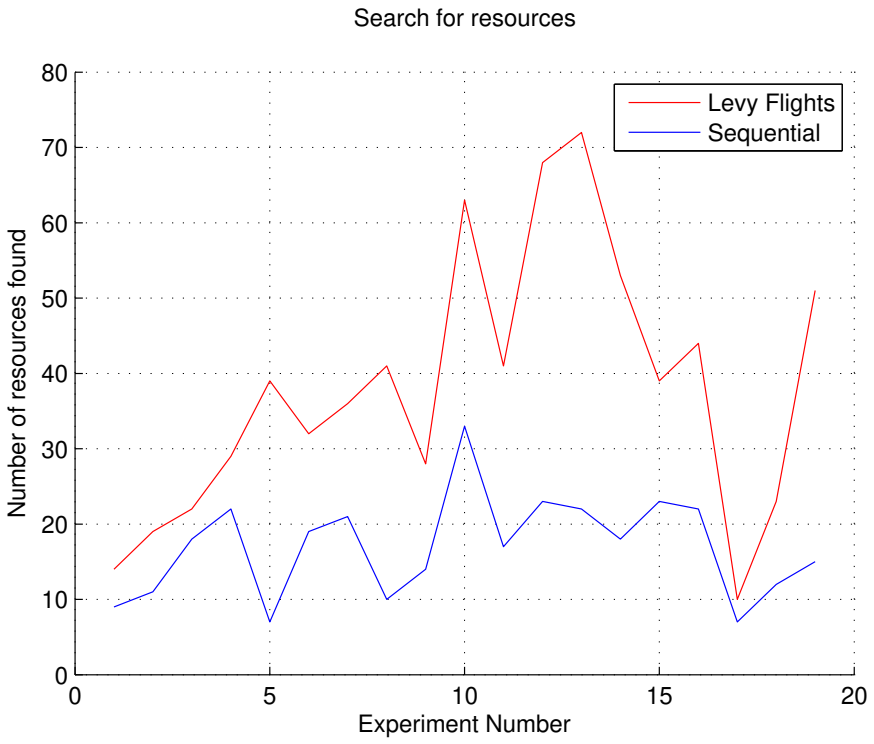

Fig. 2. The percentage of resources located by search engine operated by SA in the target system

— The disaggregation of tasks in the HPC System clusters, during scheduling and resource allocation, i.e., how tasks are allocated in the target system according to the resource search technique used, sequential or Levy flights.

Each experiment and the results obtained are explained in the following paragraphs.

The percentage of resources located by each algorithm in the target system at time t. In these experiments two types of searches are executed as follows: from the server farm and from where the last search ended; with these experiments it is possible to measure the ability of each technique to locate the resources in the target system, in time t.

For each of these experiments, numbered 1 to 20, a set of 100 tests is run; in 50 tests the starting position of the search is in the initial cluster and 50 tests the starting position is the location of the last search; alternating the experiments, a test is run with the first type of search and then with the second type of search.

For experiment 1, the tests are started with a time t of 5 minutes, 100 experiments are run, the result of each of the 100 runs is the number of resources found at time t (5 minutes); the total sum of the resources found in each run is divided by 100, and the result is reported for this experiment. For a second experiment the time t is 10 minutes and the 100 runs, the results are obtained in the same way as experiment 1. For each experiment 5 minutes are incremented. The results of all experiments are shown in graph 2.

Results (Experiment 1). The results obtained in these experiments show that the random walk search is more efficient for locating geographically distributed resources, selected with a probability distribution; this search method allows moving to clusters that may be without assigned tasks (completely idle).

During the execution of this search method, subgroups with different amounts of free resources were found in the clusters, before the migration of the SA from one cluster to another; with these groups, the system can send tasks with a number of subtasks that are equal to the subgroups of resources found, allowing the execution of the tasks with their subtasks in the same cluster.

In contrast to first experiment, we propose an experimentation related to the way in which the tasks are disaggregated into the resources found by each type of search executed: the sequential search and the search using Lévy flights. The following paragraphs explain this experimentation.

Disaggregation of the tasks in the HPC System clusters, during the resource allocation process. Once the resources are located the resource matrix and the matrix of characteristics are updated; the next step of the algorithm is to assign the tasks to the available resources.

For this experimentation, the total execution time of the tasks was measured considering that Lévy random walks searches can locate resources with longer geographic distances. When the subtasks of a task are executed in geographically distant clusters with long distances, the communication times between tasks increase gradually, depending on the speed of the network devices of the target system. The workloads of [21] were chosen for this experiment.

The tasks of each workload were scheduled and allocated to the resources that each algorithm found in the HPC System. The completion times of the tasks of each workload were measured considering the communication time between subtasks of each task. The results of these experiments are shown in graph 3.

Results (Experiment 2). Results obtained in the experiments of scheduling and allocation of localized resources by each method have shown the outperformance of the Lévy random walks, since lower task execution times are obtained.

More resources that this technique can locate in the resource search executions, allow to make more task assignments in geographically distributed clusters; assigning tasks with their subtasks grouped in the same cluster can decrease communication times and the task execution is faster. When scheduling is executed, the scheduler can have more available resources for assignment, and allocation can be done in close clusters of resources.

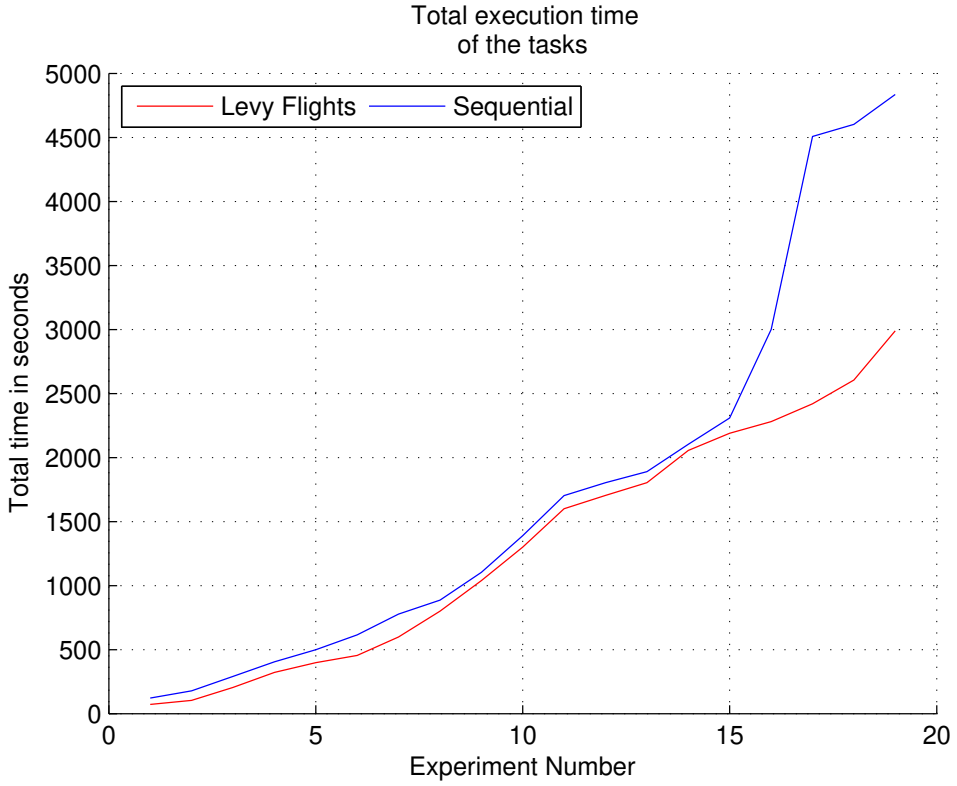

Fig. 3. The percentage of resources located by each search engine in the target system at time t

8 Conclusion and Future Work

The emulation in computational environments of scale-invariant phenomena observed in biological systems has led to propose solutions to problems of the non-deterministic polynomial time type. In this research work we conducted an exploratory study to propose a solution that allows the search for idle resources in an HPC System, specifically in a cluster system using Lévy random walks.

The justification of this work is the development of a software agent which migrates between clusters of an HPC system and simulates Lévy random walks. With this software agent a comparison is made with two performance metrics between a sequential search of resources and Lévy random walks: the number of resources located in a given time, and in contrast the total execution time of the tasks using the resources located by each technique is measured.

In the experiments with the first metric, Lévy random walks allow visiting clusters in geographic distances that a sequential resource-by-resource search would take a longer amount of time. With the second metric, the total execution time of the tasks using the resources obtained with each algorithm of search has been evaluated; this total time is measured from the assignment of the resources to the task and the dispatch of the task, until the task finishes its execution and returns to the central node.

In the experiments, it is verified that, the total times generated with the resources obtained with the Lévy random walks' algorithm, are not so far from the total times obtained with the resources located in the sequential search, when the workloads contain less than 500 PTGs for their processing; shorter total execution times of the tasks are highlighted when the workloads are dense.

Finally, it should be noted that the resources, architecture and configuration of the computer network can affect the performance of the algorithms when the configurations are not adequate.

8.1 Future Works

Research projects under development preceding this research work, include experiments with real application workloads using the same number of clusters in real environment, in order to measure task execution times and performance of the underlying computer network.

Second work, includes design and programming of the Array Method using hybrid programming with OpenMP and MPI (Message Passing Interface), in order to perform experiments with sequential executions and parallel executions; these execution comparisons will allow evaluating the performance of the Array Method.

Acknowledgments

This project is funded by Tecnológico Nacional de México TecNM. I express special thanks for support to Instituto Tecnológico El Llano Aguascalientes.

References

1. **Abouelela, M., El-Darieby, M. (2012).** Multidomain hierarchical resource allocation for grid applications. Journal of Electrical and Computer Engineering, Vol. 2012.
2. **Alkhalaileh, M., Calheiros, R., Nguyen, Q., Javadi, B. (2019).** Dynamic Resource Allocation in Hybrid Mobile Cloud Computing for Data-Intensive Applications. pp. 176–191.
3. **Baronchelli, A., Radicchi, F. (2013).** Lévy flights in human behavior and cognition. Chaos Solitons & Fractals, Vol. 56.
4. **Casanova, H., Desprez, F., Suter, F. (2010).** On cluster resource allocation for multiple parallel task graphs. Journal of Parallel and Distributed Computing, Vol. 70, No. 12, pp. 1193–1203.
5. **Chechkin, A., Metzler, R., Klafter, J., Gonchar, V. (2008).** Introduction to the Theory of Lévy Flights. pp. 129–162.
6. **Eijkhout, V., van de Geijn, R., Chow, E. (2016).** Introduction to High Performance Scientific Computing.
7. **El-Zoghdy, S., Nofal, M., Shohla, M., El-sawy, A. (2013).** An efficient algorithm for resource allocation in parallel and distributed computing systems. International Journal of Advanced Computer Science and Applications, Vol. 4.
8. **Gawali, M., Shinde, S. (2018).** Task scheduling and resource allocation in cloud computing using a heuristic approach. Journal of Cloud Computing, Vol. 7.
9. **Hussain, H., Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., Qureshi, M. B., Zhang, L., Wang, Y.-J., Ghani, N., Kolodziej, J., Zomaya, A. Y., Xu, C.-Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J. E., Kliazovich, D., Bouvry, P., Li, H., Wang, L., Chen, D., Rayes, A. (2013).** A survey on resource allocation in high performance distributed computing systems. Parallel Comput., Vol. 39, pp. 709–736.
10. **Kamaruzaman, A., Zain, A., Yusuf, S., Udin, A. (2013).** Levy flight algorithm for optimization problems – a literature review. Applied Mechanics and Materials, Vol. 421.
11. **Kumar, R., Chaturvedi, A. (2021).** Improved Cuckoo Search with Artificial Bee Colony for Efficient Load Balancing in Cloud Computing Environment. pp. 123–131.
12. **Murakami, H., Feliciani, C., Nishinari, K. (2019).** Lévy walk process in self-organization of pedestrian crowds. Journal of The Royal Society Interface, Vol. 16, pp. 20180939.
13. **Pillai, P., Rao, S. (2016).** Resource allocation in cloud computing using the uncertainty principle of game theory. IEEE Systems Journal, Vol. 10, pp. 637–.
14. **Qureshi, M., Dehnavi, M., Min Allah, N., Qureshi, M., Hussain, H., Rentifis, I., Tziritas, N., Loukopoulos, T., Khan, S., Xu, C.-Z., Zomaya, A. (2014).** Survey on grid resource allocation mechanisms. Journal of Grid Computing, Vol. 12, pp. 399–441.
15. **Qureshi, M., Qureshi, M., Fayaz, M., Mashwani, W., Brahim Belhaouari, S., Hassan, S., Shah, A. (2020).** A comparative analysis of resource allocation schemes for real-time services in high performance computing systems. International Journal of Distributed Sensor Networks, Vol. 16, pp. 2020.

16. **Qureshi, M., Qureshi, M., Fayaz, M., Zakarya, M., Aslam, S., Shah, A. (2020).** Time and cost efficient cloud resource allocation for real-time data-intensive smart systems. Energies, Vol. 13.
17. **Ren, Z., Zhang, X., Shi, W. (2015).** Resource Scheduling in Data-Centric Systems. Springer New York, New York, NY, pp. 1307–1330.
18. **Shukla, A., Kumar, S., Singh, H. (2019).** An improved resource allocation model for grid computing environment. International Journal of Intelligent Engineering and Systems, Vol. 12, pp. 104–113.
19. **Tang, S., Lee, B., He, B. (2016).** Fair resource allocation for data-intensive computing in the cloud. IEEE Transactions on Services Computing, Vol. PP, pp. 1–1.
20. **Velarde, A. (2020).** Scheduling in heterogeneous distributed computing systems based on internal structure of parallel tasks graphs with meta-heuristics, pp. 1–22.
21. **Viswanathan, G., Afanasyev, V., Buldyrev, S., Murphy, E., Prince, P., Stanley, H. (1996).** Lévy flight search patterns of wandering albatrosses. Nature, Vol. 381.
22. **Wilkinson, B., Allen, M. (2005).** chapter Parallel programming techniques and Applications Using Networked Workstations and Parallel Computers. Pearson Education, pp. .
23. **Zhao, K., Jurdak, R., Liu, J., Westcott, D., Kusy, B., Parry, H., Sommer, P., McKeown, A. (2015).** Optimal Levy-flight foraging in a finite landscape. Journal of the Royal Society, Interface / the Royal Society, Vol. 12.

Article received on 25/06/2021; accepted on 07/10/2022.
Corresponding author is Apolinar Velarde Martinez.