High Performance Computing in Resource Poor Settings: An Approach based on Volunteer Computing

Adamou Hamza[1], Azanzi Jiomekong[2]
University of Yaounde I, Faculty of Sciences, Yaounde, Cameroon;
IRD, Sorbonne Université, UMMISCO, F-93143, Bondy, France

***Abstract*—High Performance Computing (HPC) systems aim to solve complex computing problems (in a short amount of time) that are either too large for standard computers or would take too long. They are used to solve computational problems in many fields such as medical science (for drug discovery, breast cancer detection in images, etc.), climate science, physics, mathematical science, etc. Existing solutions such as HPC Supercomputer, HPC Cluster, HPC Cloud or HPC Grid are not adapted for resource poor settings (mainly for developing countries) because their fees are generally beyond the funding (particularly for academics) and the administrative complexity to access to HPC Grid creates a higher barrier. This paper presents an approach allowing to build a Volunteer Computing system for HPC in resource poor settings. This solution does not require any additional investment in hardware, but relies instead on voluntary machines already owned by the private users. The experiment has been made on the mathematical problem of solving the matrices multiplication using Volunteer Computing system. Given the success of this experiment, the enrollment of other volunteers has already started. The goal being to create a powerful Volunteer Computing system with the maximum number of computers.**

Keywords*—*Volunteer computing; resource poor settings; high performance computing; matrix multiplication

I. Introduction

High performance computing [1], [2], [3], [4], [5] is really important in most scientific and industrial sectors. It helps scientists gain valuable insights to boost innovation and discovery in almost all areas of science ranging from life sciences [6], [7], [8] to quantum mechanic [9] and large scale data mining [10], [11]. In the industrial sector, high performance computing gives companies a significant competitive advantage in reducing the costs of development cycles and in producing higher quality products and services [12]. Societal challenges, including preventing and managing natural disasters, early detection and treatment of disease, and forecasting climate evolution require very high computing power [13].

In the past, dedicated supercomputers [14], [15] powerful machines made up of a large number of processors, were only used to build a HPC system. Despite the processing capacity and the speed of calculation of these computers, they do not offer the commodity price advantage, especially for small businesses and academics [13], [16]. Then, the following solutions were proposed : HPC Clusters [17], [18], [19], HPC Cloud [20], [21], and HPC Grid [19], [22]. HPC Cluster is a system composed of computers connected to a local area network, while HPC Cloud is a system involving computers connected in a private or public network. Finally, HPC Grid is composed of computers connected in wide-area networks such as Internet.

The proposed solutions cannot be applied in low resource areas because of its costs. The HPC Cluster and the HPC Cloud require financial resources beyond the reach of most institutions in developing countries. For the HPC Grid, administrative complexity in obtaining the necessary permissions to use this system can be a higher barrier [21].

In developed countries, HPC have provided computing resources to projects at a huge scale [23], often resulting in major scientific discoveries and invention whether at the universities or businesses level. However, in resource poor or limited settings, many academics and small business do not always have enough resources to access to the HPC. Resource poor or constrained settings are defined as a local where the capability to provide HPC is limited to basic critical resources, including desktops and laptops. Resource poor settings can be stratified by no resources and limited resources. In resource limited settings, people use their personal computers for computation. This article focuses on the limited resources category. These include scientists in research teams with significant computational needs, individuals in developing countries with no access to other alternatives.

It should be noted that in many developing countries, the wide range of commercial centers of computers has made available many devices. For instance, in the Department of Computer Science at the University of Yaounde I in Cameroon, there are about 1,000 computers owned by lecturers and students; the Internet connection is widespread in Cameroon. These computers are idle most of the time (e.g., when students are in class or sleeping at night, etc.), this therefore constitutes a source of computing power available and largely reusable. If their owners are willing to actively lend CPU time and memory, these devices can be used as distributed computing infrastructure at no cost. This is called a Volunteer Computing (VC) system [24], [25], [26], [27]. A HPC Volunteer Computing system is obtained through community engagement by setting up a system of volunteer machines. Their goal is similar to HPC Grid, which is to gather distributed computing resources and federate them to solve large computational problems. The difference between these two systems is that the resources come from non-dedicated computers, underutilized and controlled by their owners (volunteers). This approach requires no additional hardware investment, but relies on

devices already owned by users and their communities.

This article presents an approach to build a Volunteer Computing system for HPC in resource poor settings. Initially, the Section II describes the main solutions to HPC problems. Then, the Sections III and IV follow with our solution and the experience of this solution respectively. Finally, the Section V concludes and opens future directions.

II. High Performance Computing (HPC)

High Performance Computing uses the principle of parallel computing to address the high computing requirements of applications by dividing them into smaller ones that can be processed simultaneously on different computing units. These computing units can reside on the same computer (Supercomputer) or on multiple computers connected by a network forming HPC Cluster, HPC Cloud, HPC Grid and HPC Volunteer Computing. These solutions are presented in this order in the rest of this section.

A. HPC Supercomputers

One of the best known type of HPC solution is the Supercomputer [14], [15], [28]. A Supercomputer contains hundreds, thousands or even millions of computing units forming a massively paralleled processor organized in a network of processors [29], [30]. For instance, the *Summit - IBM Power System AC922*[1] contains $2,414,592$ cores. These processors work together to solve large computational problems as efficiently and quickly as possible.

Supercomputers allow us to obtain a great computing power. However, with the entry fees [13], [15], acquiring a Supercomputer is almost impossible for scientists and engineers working in resource poor settings. Because of this limitation, other HPC solutions that do not require a fully dedicated computer have been developed. These solutions are HPC Cluster, Cloud, Grid and Volunteer Computing.

B. HPC Cluster

A HPC cluster is a computing system in which independent computers are connected by a high performance local area network in order to solve complex problems [18], [19], [31], [32]. Each machine in the HPC cluster is a complete computer consisting of one or more CPUs or cores, memory, disk drives and network interfaces. HPC cluster is generally owned by a single administrative entity. The software used to manage clusters give users the illusion that they are with a single large computer when in reality the cluster may consists of hundreds or thousands of individual machines [33]. A cluster is much more cost-effective than a single supercomputer of comparable speed [13], [16].

An example of a computer cluster is the dedicated physical cluster at HP Labs Singapore (HPLS)[2]. This cluster is connected to a Gigabit Ethernet network on a single switch. Each server, with 2 CPU sockets (populated with a 6 core CPU), results in twelve physical cores per machine.

[1]https://www.top500.org/system/179397

[2]https://xrds.acm.org/article.cfm?aid=2000789

C. HPC Cloud

A High Performance Computing Cloud [20], [21], [34], [35] is an on-demand availability of computing power, without direct active management by the user. It provides dynamic and scalable computing power through resources organized in data centers distributed around the world. By purchasing on-the-go and not as an asset, HPC Cloud delivers consistent, scalable results, minimizing the initial costs of computing infrastructure. A cloud can be private (operating for a single organization) or public (open for public use). Since 2016, many IT companies such as Amazon Web Services (AWS)[3], Microsoft Azure cloud[4], Google Cloud Platform[5] have offered HPC Cloud.

The advantage of the HPC cloud over other types of HPC is that it allows the user to immediately access computing resources without the approval of an allocation committee and the service can be provided without human interaction with the service provider. Software can be used without the need to purchase a license or install it, and users do not need to have strong software/infrastructure management skills [21], [36]. However, in the context of resource poor settings, the main drawback of HPC Cloud is that it relies on dedicated hardware managed centrally, which implies a minimum cost which can be high for academics and small businesses [21].

D. HPC Grid

A Grid consists of many computing resources (from multiple administrative domains) connected to a network (e.g., The Internet) working together to solve large problems requiring HPC. The whole system is called a HPC Grid [19], [22], [37]. The HPC Grid differs from the HPC Cluster or Cloud in that it uses many computers, but with a much more distributed nature. Some HPC Grids span the world while others are located within a single organization.

The HPC Grid capabilities are generally managed by a precise organization and computational resources are provided by various supporting institutions, such as companies, research groups, laboratories, and universities. Large Grid computer facilities are often used by a large number of users to solve intensive scientific, mathematical, and academic problems. However, the administrative complexity (obtaining necessary authorizations) allowing the public to access a HPC Grid is a real barrier for academics and small businesses. [25].

An example of Grid is the French Grid'5000[6]. This Grid is distributed over 8 sites, 31 clusters and 12,328 cores. Each site hosts a cluster and all sites are connected by high speed network. It aims to provide a highly reconfigurable, controllable and monitorable experimental platform for research in large-scale parallel and distributed systems.

E. HPC Volunteer Computing

In HPC Volunteer Computing (VC) [24], [25], [27], [38], [39], computer users/owners, who are members of the general public contribute their computing resources to solve HPC

[3]https://aws.amazon.com

[4]https://azure.microsoft.com

[5]https://cloud.google.com

[6]https://www.grid5000.fr/w/Grid5000:Home

problems. It is based on two pillars: the first is the allocation and management of large computing tasks; the second is the participation of a large number of individuals volunteer who offer their computing resources to a project. Compared to HPC Cluster, HPC Cloud and HPC Grid, HPC VC removes financial and administrative barriers. The costs are supported by volunteers, which cover the acquisition, operation, and maintenance of the computing devices.

HPC Volunteer Computing has produced many remarkable scientific results over the last decade. The most popular Volunteer Computing systems are: SETI@home[7][8] [40] for searching the extraterrestrial intelligent life and Folding@home[9] [41] for statistical calculations of molecular dynamics trajectories for models of biological systems. The Folding@home project is a good example of how important scientific results can be produced with VC for affordable problems for other HPC schemes. For instance, in 2008, it was used to study mutations of influenza hemagglutinin [23].

The main challenges of the Volunteer Computing approach is the capabilities of personal devices, the need to encourage and maintain volunteer engagement, and the automatic management of volunteer unreliability [42], [43]. Despite the previous challenges, Volunteer Computing is the solution for HPC in resource poor settings. In fact, in many developing countries, the wide range of commercial centers of computers has made available a lot of devices. These devices spend a lot of time without being used. They can then be used free of charge as a distributed computing infrastructure. It requires no investment in additional hardware, rather relies on devices that generally belong to users and their community, and favours simple tools that can be implemented part-time by a single developer. Section III, will be concerned with a methodology for deploying and using HPC to improve computing in resource poor settings while in Section IV, it will be shown in experiments that this solution can solve the problem of HPC in resource poor settings.

III. An approach based on Volunteer Computing for HPC in resource poor settings

The lack of HPC resources is a challenge for scientists, engineers and businesses in resource poor settings. However many countries have an Internet or local network and many computers are owned by individuals. For example, in the computer science department of the University of Yaounde I in Cameroon, there are around 1,000 computers belonging to students and lecturers. Generally, these computers are not used full time. For example, students spend a lot of time a week attending classes, sleeping, or taking breaks. These computers can be used during their idle time to build a platform for Volunteer Computing. Considering the resource poor settings constraints, this section presents an approach (summarized in Fig. 1) for HPC in these environments. This approach is divided into three main activities: management activity (Section III-A), processing activity (Section III-B) and support activity (Section III-C).

[7]https://setiathome.berkeley.edu/
[8]https://www.seti.org/
[9]https://foldingathome.org/

A. The Management Activity

The management activity uses information about the daily work of volunteers (e.g., the performance of the system and of each volunteer) to supervise the system and increase efficiency. This is the essential key to the success of the Volunteer Computing system. For instance, information about the volunteers, the types of tasks performed (e.g. matrix multiplication, polynomial multiplication), the name of each task, the duration of execution of the tasks, the success or failure of the execution of the tasks and the dates of their execution, will allow us to consider the resource poor settings context. The management activity involves the following activities: design of the Volunteer Computing system (Section III-A1), planning (Section III-A2), coordination (Section III-A3), staffing (Section III-A4), motivation (Section III-A5), control (Section III-A6) and quality assurance (Section III-A7).

1) Design of the Volunteer Computing System: The first and main activity of this approach is to set up the Volunteer Computing system. Resource poor settings generally suffer from power outages and malfunctions in Internet connectivity. In our approach, potential volunteers are firstly users who are often connected to the Internet, secondly those who need the system later and finally friends, family members or communities. Each volunteer has information about the others. A server is designed and contains the most up-to-date information about the volunteers, sent by the latter. All other local information for volunteers is updated by the server. If the server crashes (due to a power outage or a failure), the election algorithm [44], [45] is used to choose the volunteer that will replace the server by waiting that the problem is solved. In our case, it will consist of choosing the volunteer who connects the most to the Internet. Overall, the proposed VC is defined by the equation 1, where:

- S is the server. In our approach, the server is used to centralize all the volunteer information in the system in order to restore it whenever a volunteer needs it. Thus, the server will be used to store all the information on the system, the performance of the system, and of each volunteer in the system, the logs on the computation performed, the type of tasks already completed and its performances, tasks in progress or waiting. These information are essential for efficient management of the system. The server will use the feedback to help volunteers estimate the execution time each time a volunteer wants to perform a task by the system.
- V_i a volunteer computer. It receives tasks, performs them and returns the result. It can submit tasks to other volunteers, retrieve the results and merge to obtain the final result. It informs the server of all the computation activities carried out, in progress or pending. It receives updated information from other volunteers from the server.
- P_i the profile of the volunteer V_i. The profile P_i contains information on the elements that can be used to determine its computing capacity. These are: CPU speed (cpu_speed), number of cores (num_core), memory size (mem_size), disk size (disk_size), operating system name (os_name), its location and its

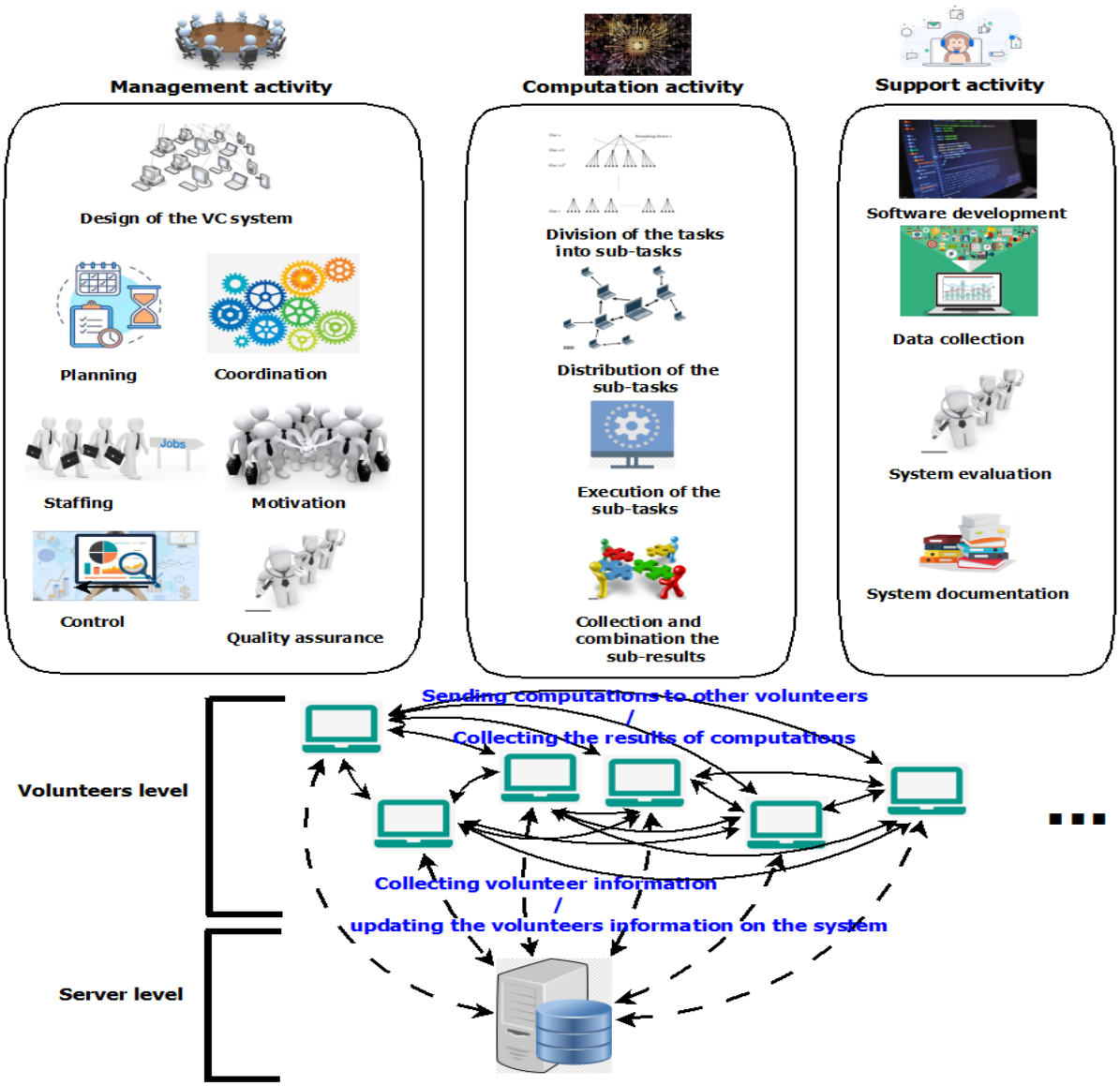

Fig. 1. The Volunteer Computing approach

availability (avail). These information will be used by each volunteer to identify to which volunteers they can send their computations to, what types of computations to send and when.

$$VC = [S, (V_1, P_1), ..., (V_i, P_i)],$$
$$P_i = [cpu\_speed, num\_core, mem\_size, disk\_size, os\_name, location, avail]. \quad (1)$$

A software named VCSoftware is used in the system. It is composed of the volunteersManager module and the serverManager module:

- volunteersManager module: The volunteerManager module is used at the volunteer level. A new volunteer signs up to the system by filing out their profile in a form, installing the VCSoftware and activating the volunteerManager on their machine. The volunteersManager is used to collect information about each volunteer and send it to the server, receives tasks from other volunteers, performs and returns the result, allocate tasks to other volunteers, collect and merge the results and send feedback of computation to the server. The volunteer contribution level is determined by the setting of their profile. The participant can choose to contribute permanently or only when the computer is idle, or can decide whether to contribute or not when the computer is running on battery/inverter. If the server is down, this software informs the computer owner and all other volunteers, the goal being to design the new server.
- serverManager module: The serverManager module is used at the server level. It allows to periodically collect information on the profile of each volunteer and to record a log of all the tasks performed by the system.

This information will allow to recommend a profile to a volunteer (e.g. computation time taking into account the idle period of the computer), to know the performance of each volunteer and the performance of the whole system. When the server is down, the volunteer elected as server while waiting for the problem to be solved activates this module.

2) Planning: Planning activity is the foundation of management. It refers of determining the future course of action towards the desired objective of the system. The planning activity must anticipate and precede all the other functions of the management activity and permit to meet the challenges of environmental changes (e.g., arrival and departure of volunteers, server failure, power/Internet outage etc.). In our VC approach the planning activity is at two levels:

- At the server level, planning involves continuous assessment of the strengths and weaknesses of the VC system using information about volunteers and the results of computation; identification of the actions assigned to a task already performed by the system. It requires all the information collected on the volunteers during the support activity (see Section III-C). This information will allow the server to know the idle time of each volunteer computer, when each one connects to the Internet, to evaluate the performance of each volunteer and the performance of the whole system. For example, information about a task already executed by the system can be used to efficiently execute it the next time the same type of task is submitted.
- At the volunteer level, planning requires the information from other volunteers. Stored on the server, this information is sent to each volunteers when they connect to the system. This information will be used to decide which volunteer to send a task taking into account the computing power offered by each volunteer and the period of connection to the system: which task to send, when to send the task, to whom the task can be sent, the duration of each task sent to a volunteer and the duration of the whole task. At the volunteer level, good planning will help to effectively address the challenges of environmental change (e.g., when a volunteer's computer cannot complete a sub-task).

3) Coordination: Like the planning activity, the coordination activity is done at two levels:

- At the server level, the server identifies the volunteers available for the computation, the computation power they offer and the duration for which they remain connected to the Internet. It also stores information about the different types of tasks already performed. This information will be used to predict the time and resources required to complete a task. Any change in volunteer information (computing power) should result in organizational changes. Information about leaving/joining the system by a volunteer must be considered.
- At the volunteer level, the coordination activity involves organizing the sub-tasks to be performed, as well as the time and resources necessary to carry them out. It will use the information obtained from the other volunteers to identify the resources needed to achieve a given task.

4) Staffing: The staffing activity includes recruiting good volunteers, selecting a group of volunteers to perform a task, and evaluating the volunteers registered in the system. Since volunteers registered in the system are not paid, the manager must be careful during their recruitment. In a resource poor settings, we recommend starting with people who need High Performance Computing and who do not have enough financial resources; and encourage them to invite members of their community to register. Information on volunteer profiles will be used to identify the good volunteers profiles.

5) Motivation: The motivation activity consists of attracting volunteers to contribute to the system and those in the system to increase their contributions. Since volunteers are not paid, a motivational environment must be created. In our case, the possibility of having access to a High Performance Computing is a great motivation for students and researchers. Data collected on the use of the system by other volunteers must be made available to the public, mainly students, engineers and researcher in order to encourage them to join the system. Performance data desired by the system will be made available to volunteers enrolled in the system to encourage them to participate and to encourage members of their communities to participate.

6) Control: The control activity aims to guarantee that scheduled tasks are completed as planned. During the control activity, the performance evaluation of each volunteer and the whole system is made in order to identify weaknesses and strengths. Planning is the basis of control. It focuses on the tasks that are performed and the results of those tasks. It plays an important role in ensuring the efficiency and effectiveness of the VC system. The information collected during the use of the VC system will help to measure the performance of each volunteer but also to identify gaps. At the server level, the comparison between planned performance and actual performance is analyzed to know if there are deviations, and the reasons of the deviations are analyzed. At the volunteer level, each result of a sub-task collected from volunteers is used to evaluate it. Real time information will allow quick control, which will reduce the costs of planning errors.

7) Quality assurance: The quality assurance activity guarantees that the quality of each computation is satisfactory (took the expected time, returned the expected results).

B. Computation Activity

To perform a task with the VC, that task must be divided into sub-tasks and each sub-task sent to volunteers. After the execution by the volunteers, the results and the results logs are collected by the volunteer who initiated the computation. Globally, computation activity involves: dividing the task into sub-tasks, distributing of sub-tasks to volunteers, performing of sub-tasks by volunteers, collecting and combining the results.

1) Division of the task into sub-tasks: This step consists of dividing the task into smaller and independent sub-tasks (preferably atomic sub-tasks). Consider T as a given task to

be performed by a volunteer. Then, $T = t_1, t_2, ..., t_n$ where n is the number of independent tasks that compose the task T. Since t_i are independent, they can be executed in parallel.

2) Distribution of the sub-tasks amongst volunteers: Given the data obtained from the server on the system (computing power offered by each volunteer, time to connect to the Internet of each volunteer, etc.), the volunteersManager will identify the volunteers that can participate in the computation. These are the computers most likely to be available until the end of a given sub-task. Subsequently, the computation will be sent to these volunteers when they connect to the system. Depending on the number of volunteers available and the computation time they provide, many sub-tasks can be sent to a volunteer.

The volunteer computer that sent the tasks to others will send a log file to the server to inform it that a job has started. The server will broadcast this information to all other volunteers. If a machine starts a new task, it must consider the existing tasks running in the system.

3) Execution of the sub-tasks: The volunteersManager on the volunteer machine that receives a task will start within the period specified in the volunteer profile. If many tasks have been assigned to a volunteer, the volunteersManager will perform the oldest first. At the end, a log records information on the execution time and the execution status (finished or failed).

4) Collection of the sub-results: After completion, the volunteersManager collects the results and the result logs. If some computations failed, more efficient volunteers are identified and these computations are sent to them. At the end of the computation, the results logs are sent to the server.

5) Combination of the sub-results: The last activity of the computation activity is the combination of the sub-results obtained from the volunteers.

C. Support Activity

The support activity involves the series of activities performed at the same time as the management and the computation activities. It aims to facilitate management and computation by providing all the needed tools and information. The VCSoftware software is developed during this activity. Overall, this activity includes the software development (section III-C1), data collection on the system and each volunteer computer (section III-C2), system evaluation (section III-C3) and system documentation (section III-C4).

1) Software development: The software development activity involves the development/updating of the VCSoftware that will be used by the server and the volunteers. This tool is composed of two main modules: the serverManager module and the volunteersManager module.

The serverManager will help acquire data sent by volunteers to the server. These data are those provided by the volunteer during registration, but also other information on the idle time of the volunteer computer and the time of connection to the Internet. The Information collected from the volunteers will allow the server to: make suggestions of profiles to the volunteers. The volunteer can use the suggestion or not. For example, during the holidays, the idle period is not necessarily the same as during the working period. Then, during the holidays, the server can suggest to volunteers to update their profile.

The volunteersManager is the module which, on the volunteer side, will: periodically collect information on the volunteer and send it to the server; allow identification of volunteer computers that can perform a given task, send tasks to volunteers and collect the results; reception of tasks and their scheduling according to other tasks already received; and sending logs on the execution of the tasks and the performance of each volunteers.

2) Data collection: The data collection activity is done both at the server and at the volunteer level. At the server level, the data collected is used to evaluate the performance of the system and each volunteer. The performance of the whole system is calculated based on the number of volunteers registered in the system, the tasks performed by the volunteers, and when the volunteers will connect to the Internet.

At the volunteer computer level, data is collected in two cases: firstly, the volunteer fills out a registration form in which information about the device and its availability for computation is provided. Then, an automatic data collection takes place (using volunteersManager module). This can be done periodically (hourly or daily) and will concern the idle time of the computer and the time when the computer is connected to the Internet. This data is used to: suggest profile updates to the volunteer and evaluate each volunteer. Each time the server receives updated information from a volunteer, this information is aggregated with existing information and forwarded to other volunteers.

3) System evaluation: The system evaluation involves the evaluation of the server and each volunteers. Indeed data collected during the computations will allow the server to know if the system is efficient and if it provides relevant computations. For this purpose, during the execution of each task, the name, the type, the execution time of the task, the date of execution and the status (execution failed or not) are recorded in a log file and sent to the server.

4) System documentation: The system documentation activity is the vital and continuous activity of our approach. In fact, during this activity, the documents are produced to help the manager and the volunteers to use the system. It provides all information on the capabilities and characteristics of the system. It helps users understand the system and its essential reference materials. VC documentation is updated every time there are changes in the system. The documentation activity is at two levels: the manager level and the volunteer level.

At the manager level, the documents will allow the manager to install and configure the server. It will also give tips to the manager on how to motivate the volunteers to register, contribute and maximize system performance as well as how to customize the software so that it works best for each volunteers (e.g., profile suggestion given data collected from volunteers). The manager can also write Frequently Asked Questions (FAQ) to help volunteers.

At the volunteer level, documents are used to inform the volunteer about the system, and describe what it is intended to do and how it works. Overall, it explains to volunteers how to

install the VCSoftware in order to contribute/use the system and how to submit computations and collect the results. To facilitate access to non IT volunteers, a short video is a good way to show how to install the VCSoftware and another on how to submit tasks and collect results. The documentation about the contribution of other users is provided to newcomers in order to motivate the latter.

IV. Experimentation

For Volunteer Computing to be adopted, volunteer devices must provide enough computing power to solve associated computing problems. The first step in the development of our VC system consists of a pilot phase, which is the development and experimentation of a VC system. This section shows how this system has been built and used to solve the problem of matrix multiplication. In the following, the problem of matrix multiplication is first described in section IV-A. Section IV-B follows with the VC_UY Volunteer Computing system built at the University of Yaounde I in Cameroon. Finally, Section IV-C presents the experimentation of VC_UY on the matrix-matrix multiplication.

A. Matrix Multiplication

Matrix multiplication [46], [47], [48], [49] is an operation that produces a matrix product from two input matrices. Each matrix product entry is the dot product of a row in the first matrix and a column in the second (see equation 2). Matrix multiplication has many applications. It is the basic computational kernel for many algorithms of machine learning systems and recommendation systems. Matrix-vector multiplication is the core kernel of the PageRank algorithm [21], [48]. Matrix multiplication is often a computational bottleneck because generally, matrices are very large with dimensions which can easily reach hundreds, thousands and even millions [46], [47], [48], [49].

Three popular algorithms for matrix multiplication have been proposed in the literature: the iterative algorithm, the recursive algorithm and the Strassen algorithm. Let us consider two square matrices A and B given below.

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix},$$

$$\mathrm{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{bmatrix}$$

The matrix multiplication of A and B gives the matrix C obtained by using the equation 2.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{2}$$

From the previous equation, a simple iterative algorithm can be constructed using loops on the indices i, j from 1 to n. This algorithm takes time on the order of n^3. Using the Divide and Conquer approach, the matrices A, B are partitioned into blocks. Then, the multiplication gives:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

The Divide and Conquer approach works for all square matrices whose dimensions are powers of two. For matrices which do not respect this condition (e.g., matrix-vector multiplication), fill the missing rows and columns with zeros. The complexity of the Divide and Conquer approach is the same as the iterative algorithm, i.e. n^3. In fact, this approach requires 8 blocks multiplications to calculate the product matrix which still requires n^3 running time [46], [47], [49].

Another matrix multiplication approach is called Strassen algorithm. The Strassen algorithm [46], [50] is a recursive Divide and Conquer approach. For each recursive call, the input matrices are divided into 4 blocks but only 7 blocks multiplications are needed. Compared to the iterative and the Divide and Conquer approach, the Strassen approach needs only 7 block matrix multiplications, which involves a time complexity of $n^{2.807}$ [46], [47], [49]. When the matrices are large, the execution time of the matrix multiplication can be very long. Since matrix multiplication can be divided into sub-matrix multiplication, this task can be parallelized. The next sections will present how our Volunteer Computing System was built and used for matrix multiplication. All experiments will be performed using the Strassen algorithm.

B. VC_UY: The Volunteer Computing System of the University of Yaounde I

In order to overcome the HPC problems faced by researchers and students from the university of Yaounde I in Cameroon, it was decided to build a VC system named VC_UY. This section presents the summary of the pilot phase in two main points: the recruitment of volunteers and the designing of the VC system.

1) Recruitment of Volunteers: During the volunteer recruitment phase, the master students at the Department of Computer Science of the University of Yaounde I were met. Then, the HPC concepts and the problems encountered in the building of HPC platforms in resource poor settings were explained. Our approach based on volunteering was presented and their participation as volunteers was asked. Ten students were selected from those who generally connect to the Internet at least twice a day and whose computers can be available for computations for at least one hour per day. Table I presents the profile of each volunteer.

TABLE I. Characteristics of volunteer computers

RAM	CPU	SWAP	Operating system	Disk space
4GB	Core i3, 2.4GH	1GB	LINUX	50GB
4GB	Core i3, 2.4GH	2GB	WINDOWS	100GB
4GB	Core i4, 2.4GH	2GB	LINUX	100GB
4GB	Core i5, 2.7GH	1.5GB	LINUX	100GB
4GB	Core i5, 2.4GH	1.5GB	LINUX	50GB
4GB	Core i5, 2.4GH	1GB	LINUX	50GB
4GB	Core i3, 2.4GH	2GB	WINDOWS	50GB
4GB	Core i4, 2.4GH	750MB	LINUX	100GB
4GB	Core i4, 2.6GH	1GB	LINUX	100GB
4GB	Core i4, 2.4GH	2GB	LINUX	100GB

2) VC_UY system design: Once the volunteers were recruited, our system was designed as follows: the machine (CPU core i5 and 4 GB of RAM) was used as a server. The VCSoftware[10] was deployed on the server and on each volunteers.

As presented in Section III, the server and volunteers run different modules to exchange information and perform tasks. For the purpose of experimentation, all our source code was written using the Python programming language[11], IPython Application Programming Interface[12] and the Django framework[13]. IPython is an API for parallel and distributed computing. It enables to develop, execute, debug and monitor interactively all types of parallel applications. The architecture of serverManager and volunteersManager modules is completely based on the architecture of IPython API. The Django framework has enabled the implementation of a user-friendly web interface for volunteers (registration, consultation of information about other volunteers, etc.) and the manager (for monitoring using a dashboard).

The serverManager module consists of the front-end developed using the Django framework and the back-end developed using the python programming language and the IPython API. The main feature of the front-end is to support all managements activities (Section III-A) by presenting relevant information on a dashboard. On the back-end side, the IPython API allows the server to listen to the network and collect information on volunteers. For the purpose of experimentation, a script has been written[14] allowing to collect information on volunteers; and a software developed for the visualization of the contribution of each volunteers and the performance of the whole system.

The volunteersManager module consists of the front-end and the back-end. At the front-end, the Django framework presents a dashboard describing all the other volunteers to the volunteer. At the back-end side, the volunteer machine performs the following operations: sends computation to other volunteers; listens to the requests on the network, executes the code, and returns results back to related volunteers; accepts tasks, performs them; collects the results and sends them back to the volunteers; sends profile information to the server. For the purpose of experimentation, scripts have been written (available on github[15]) and permitting to send matrices to volunteers, perform the Strassen matrix multiplication algorithm and collect results.

C. Experimentation of Matrix-Matrix Multiplication on VC_UY

To test the VC_UY Volunteer Computing system, the Strassen matrix-matrix multiplication algorithm was implemented using Python and IPython API. Matrix multiplication was used with different input sizes on one computer (CPU Core i5 and 4GB of RAM) and the whole VC system. If A and B are block-partitioned matrices, the block dimension of the resulting matrix C is determined by considering the number of volunteers and the input block dimensions. A volunteer responsible for each resulting block retrieves all the necessary blocks from A and B to execute a multiplication operation locally. Fig. 2 shows the performance of the execution on one machine. Fig. 3 presents the matrix multiplication on VC_UY system, and Fig. 4 presents the contribution of each volunteers in the computing. Fig. 4 shows that although the volunteers used have close computing power, their contribution to the calculation varies according to their availability.

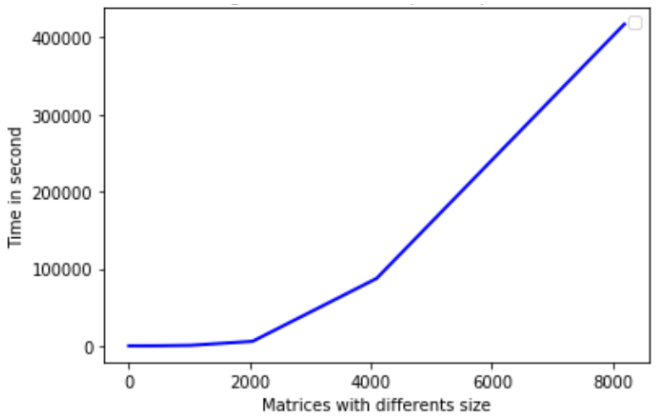

Fig. 2. Experimenting the Strassen algorithm for matrix-matrix multiplication on one machine Core i5 of CPU and 4GB of RAM

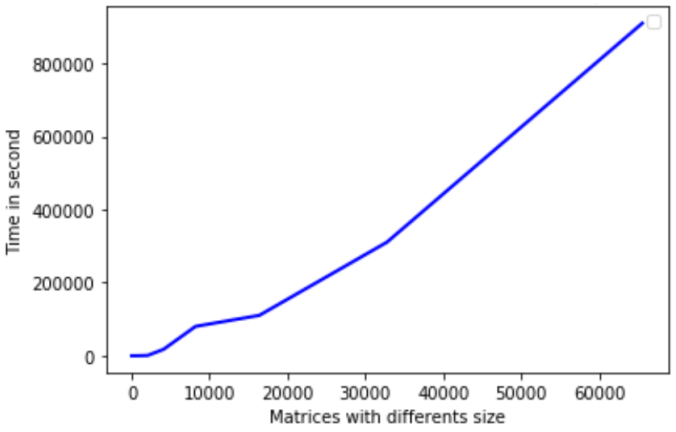

Fig. 3. Experimenting the Strassen algorithm for matrix-matrix multiplication on our Volunteer Computing system

As demonstrated by [25], [26], [51], the experiments conducted in this section shows that Volunteer Computing systems can provide a significant amount of computing power. Then, it is an important option for HPC problems in resource poor settings.

V. Conclusion

This article presented an approach allowing to build a Volunteer Computing system for HPC in resource poor settings. The experiments were made on the mathematical problem of solving matrix multiplication. Volunteers were recruited amongst students at the University of Yaounde I in Cameroon.

[10]https://github.com/admhamza/VC_UY
[11]https://www.python.org/
[12]https://ipyparallel.readthedocs.io/
[13]https://www.djangoproject.com/
[14]https://github.com/admhamza/VC_UY/blob/master/automatisation_collecte_informations.py
[15]https://github.com/admhamza/VC_UY/blob/master/calcul_distribue_dans_un_reseaux.py

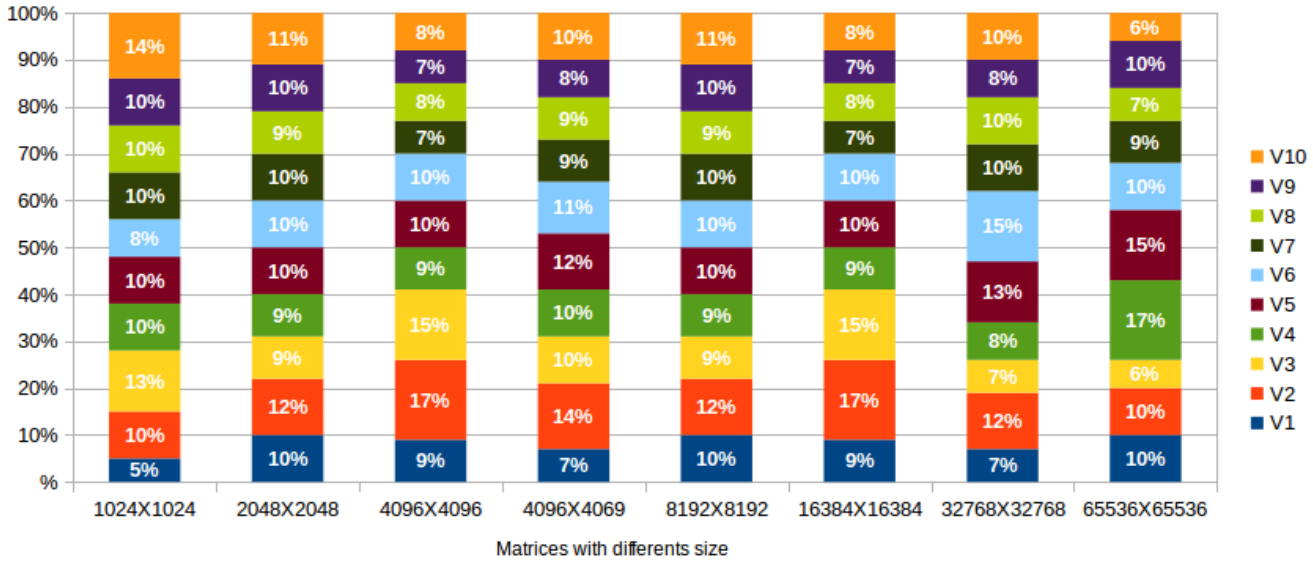

Fig. 4. Contribution of each machine. V_i represents the i^{th} volunteer

Experiments showed that a VC system can be used to enhance HPC in resource poor settings.

This work opens doors for significant possibilities in developing countries where the computing resources are generally limited. Given the success of the experiments, the recruitment of other volunteers from the University of Yaounde I is in progress. The goal being to create a powerful Volunteer Computing system with a maximum number of computers. During the experiment, potential volunteer machines were selected manually according to the data collected. This can be a difficult task if there are hundreds of volunteers registered in the system. Thus, also planned is the exploration and implementation of automatic methods for predicting volunteer machines for a given task.

ACKNOWLEDGMENTS

Our gratitude goes to all students who accepted to participate in this project, in particular Mr. Romeo Koati who recruited volunteers and participated to the development of the software.

REFERENCES

[1] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1989.

[2] Zahid Ansari, Asif Afzal, Moomin Muhiuddeen, and Sudarshan Nayak. Literature survey for the comparative study of various high performance computing techniques. *Int J Comput Trends Technol (IJCTT)*, 27(2):80–86, 2015.

[3] Fatima El Jamiy, Abderrahmane Daif, Mohamed Azouazi, and Abdelaziz Marzak. An effective storage mechanism for high performance computing (hpc). *International Journal of Advanced Computer Science and Applications*, 6(10), 2015.

[4] Zhiwei Xu, Xuebin Chi, and Nong Xiao. High-performance computing environment: a review of twenty years of experiments in China. *National Science Review*, 3(1):36–48, 01 2016.

[5] Ranjit Rajak. A comparative study: Taxonomy of high performance computing (hpc). *International Journal of Electrical & Computer Engineering (2088-8708)*, 8, 2018.

[6] Olaf Schenk, Helmar Burkhart, and Hema Reddy. Towards personalized medicine: High-performance computing in the life sciences. *ERCIM News*, 2008(74), 2008.

[7] Shaoliang Peng. High performance computational biology and drug design on tianhe supercomputers. In *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2016, Shenzhen, China, December 15-18, 2016*, page 7, 2016.

[8] Bertil Schmidt and Andreas Hildebrandt. Next-generation sequencing: big data meets high performance computing. *Drug discovery today*, 22(4):712–717, 2017.

[9] Georg Hager and Gerhard Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. Chapman and Hall / CRC computational science series. CRC Press, 2011.

[10] Daniel A Reed and Jack Dongarra. Exascale computing and big data. *Communications of the ACM*, 58(7):56–68, 2015.

[11] Bo Tang, Zhen Chen, Gerald Hefferman, Shuyi Pei, Tao Wei, Haibo He, and Qing Yang. Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Transactions on Industrial informatics*, 13(5):2140–2150, 2017.

[12] Anwar Osseyran and Merle Giles. *Industrial applications of high-performance computing: best global practices*, volume 25. CRC Press, 2015.

[13] Ezell Stephen and Atkinson Robert. *The Vital Importance of High-Performance Computing to U.S. Competitiveness*. INFORMATION TECHNOLOGY INNOVATION FOUNDATION, 2016.

[14] N. R. Adiga, G. Almasi, G. S. Almasi, Y. Aridor, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, and M. Blumrich et al. An overview of the bluegene/l supercomputer. In *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 60–60, Nov 2002.

[15] Erich Strohmaier, Hans Werner Meuer, Jack J. Dongarra, and Horst D. Simon. The TOP500 list and progress in high-performance computing. *IEEE Computer*, 48(11):42–49, 2015.

[16] Douglas Eadline. *High Performance Computing For Dummies*. Wiley Publishing, Inc., USA, 5th edition, 2009.

[17] Chee Shin Yeo, Rajkumar Buyya, Hossein Pourreza, Rasit Eskicioglu, Peter Graham, and Frank Sommers. *Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers*, pages 521–551. Springer US, Boston, MA, 2006.

[18] Zhe Fan, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. Gpu cluster for high performance computing. In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, SC '04, pages 47–, Washington, DC, USA, 2004. IEEE Computer Society.

[19] Violeta Holmes and Ibad Kureshi. Developing high performance computing resources for teaching cluster and grid computing courses. *Procedia Computer Science*, 51:1714 – 1723, 2015. International Conference On Computational Science, ICCS 2015.

[20] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[21] Abhishek Gupta, Laxmikant V Kale, Filippo Gioachin, Verdi March, Chun Hui Suen, Bu Sung Lee, Paolo Faraboschi, Richard Kaufmann, and Dejan Milojicic. The who, what, why and how of high performance computing applications in the cloud. *HP Laboratories Technical Report*, (49), 8 2013.

[22] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[23] Peter M. Kasson, Daniel L. Ensign, and Vijay S. Pande. Combining molecular dynamics with bayesian analysis to predict and evaluate ligand-binding mutations in influenza hemagglutinin. *Journal of the American Chemical Society*, 131(32):11338–11340, 2009. PMID: 19637916.

[24] Zied TRIFA, Mohamed LABIDI, and Maher KHEMAKHEM. Arabic cursive characters distributed recognition using the dtw algorithm on boinc: Performance analysis. *International Journal of Advanced Computer Science and Applications*, 2(3), 2011.

[25] Erick Lavoie and Laurie Hendren. Personal volunteer computing. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, CF '19, pages 240–246, New York, NY, USA, 2019. ACM.

[26] Erick Lavoie, Laurie Hendren, Frederic Desprez, and Miguel Correia. Pando: Personal volunteer computing in browsers, 2018.

[27] Oded Nov, David Anderson, and Ofer Arazy. Volunteer computing: A model of the factors determining contribution to community-based scientific research. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 741–750, New York, NY, USA, 2010. ACM.

[28] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. The sunway taihulight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, 2016.

[29] Shurong Tian, Todd Takken, Vic Mahaney, Christopher Marroquin, Mark Schultz, Mark Hoffmeyer, Yuan Yao, Kevin O'Connell, Anil Yuksel, and Paul Coteus. Summit and sierra supercomputer cooling solutions. *IBM Journal of Research and Development*, 2019.

[30] James R Kozloski, Timothy M Lynar, Mark D Nelson, and John M Wagner. Energy efficient supercomputer job allocation, 2018. US Patent 10,025,639.

[31] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[32] Zvi Tannenbaum and Dean E Dauger. Cluster computing, 2019. US Patent 10,333,768.

[33] Minakshi Tripathy and C.R. Tripathy. On a virtual shared memory cluster system with virtual machines. *International Journal of Computer and Electrical Engineering*, 3:754–761, 01 2011.

[34] Mohammad Moghadasi, Seyed Majid Mousavi, and Gábor Fazekas. Cloud computing auditing. *International Journal of Advanced Computer Science and Applications*, 9(12), 2018.

[35] Aferdita Ibrahimi. Cloud computing: Pricing model. *International Journal of Advanced Computer Science and Applications*, 8(6), 2017.

[36] Babur Hayat Malik, Jazba Asad, Sabila Kousar, Faiza Nawaz, Zainab, Farania Hayder, Sehresh Bibi, Amina Yousaf, and Ali Raza. Cloud computing adoption in small and medium- sized enterprises (smes) of asia and africa. *International Journal of Advanced Computer Science and Applications*, 10(5), 2019.

[37] Hans-Joachim Bungartz. Sparse grids and their impact on hpc and big data. In *Kolloquiumsvortrag*, 2019.

[38] Tessema M Mengistu and Dunren Che. Survey and taxonomy of volunteer computing. *ACM Computing Surveys (CSUR)*, 52(3):1–35, 2019.

[39] Bruno Donassolo, Henri Casanova, Arnaud Legrand, and Pedro Velho. Fast and scalable simulation of volunteer computing systems using simgrid. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 605–612, New York, NY, USA, 2010. ACM.

[40] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: An experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[41] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande. Folding@home: Lessons from eight years of volunteer distributed computing. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–8, May 2009.

[42] Wei Li and William W Guo. The optimization potential of volunteer computing for compute or data intensive applications. *Journal of Communications*, 14(10), 2019.

[43] Harold E Castro. Capacity of desktop clouds for running hpc applications: A revisited analysis. In *Applied Informatics: Second International Conference, ICAI 2019, Madrid, Spain, November 7–9, 2019, Proceedings*, volume 1051, page 257. Springer Nature, 2019.

[44] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[45] Maarten Van Steen and Andrew S Tanenbaum. *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[46] Qingshan Luo and John B. Drake. A scalable parallel strassen's matrix multiplication algorithm for distributed-memory computers. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, SAC '95, pages 221–226, New York, NY, USA, 1995. ACM.

[47] Chandan Misra, Sourangshu Bhattacharya, and Soumya K. Ghosh. Stark: Fast and scalable strassen's matrix multiplication using apache spark, 2018.

[48] M. Son and K. Lee. Distributed matrix multiplication performance estimator for machine learning jobs in cloud computing. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 638–645, July 2018.

[49] Zhen Xie, Guangming Tan, Weifeng Liu, and Ninghui Sun. Ia-spgemm: An input-aware auto-tuning framework for parallel sparse matrix-matrix multiplication. In *Proceedings of the ACM International Conference on Supercomputing*, ICS '19, pages 94–105, New York, NY, USA, 2019. ACM.

[50] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[51] Travis Desell. Developing a volunteer computing project to evolve convolutional neural networks and their hyperparameters. pages 19–28, 10 2017.