Volunteer cloud computing research framework

Chiranjeevi Ankamreddy, Ioan Raicu, **Illinois Institute of Technology, Chicago**

Abstract

Over the past ten years, popularity of Cloud computing is increasing exponentially. Many companies are funding cloud computing infrastructure for research etc. The proposed work aims to donate Volunteer Resources for research areas like high-energy physics, molecular biology, medicine, astrophysics, climate study, and other areas since Volunteer computing is a type of distributed computing in which computer owners or users donate their computing resources like Storage, CPU, GPU and Network Bandwidth etc. to one or more projects.

In this framework, every Volunteer connects to a manager server when they are ready to donate resources. Manager server manages all the volunteer connections, stores the volunteer information like number of cores, Ram Size, Ram speed, Disk size, Disk Speed, Network speed etc.. securely and process the client requests. Clients requests the manager server for required resources, based on the request, Manager Server then performs the search (nearest) with all the volunteers which are connected during that time and communicates with the selected Volunteer to donate resources. When Volunteer is ready to donate resources, then Manager Server gives the volunteer details to the client. Now, Clients directly communicate with volunteer and use their resources. Manager Server monitors every volunteer who donates their resources and measures the CPU time, disk I/O, Network Bandwidth, and gives credits to the Volunteer based on the performance.

Introduction

We know that Volunteer computing is a type of distributed computing in which computer owners or users donate their computing resources like Storage, CPU, GPU and Network Bandwidth etc. to one or more projects. In the current world, there are huge number of PCs, Tablets, Mobile Phones (More than 3 billion), volunteer computing can provide more computing power to science than does any other type of computing. This computing power enables scientific research that could not be done otherwise.

Previous work

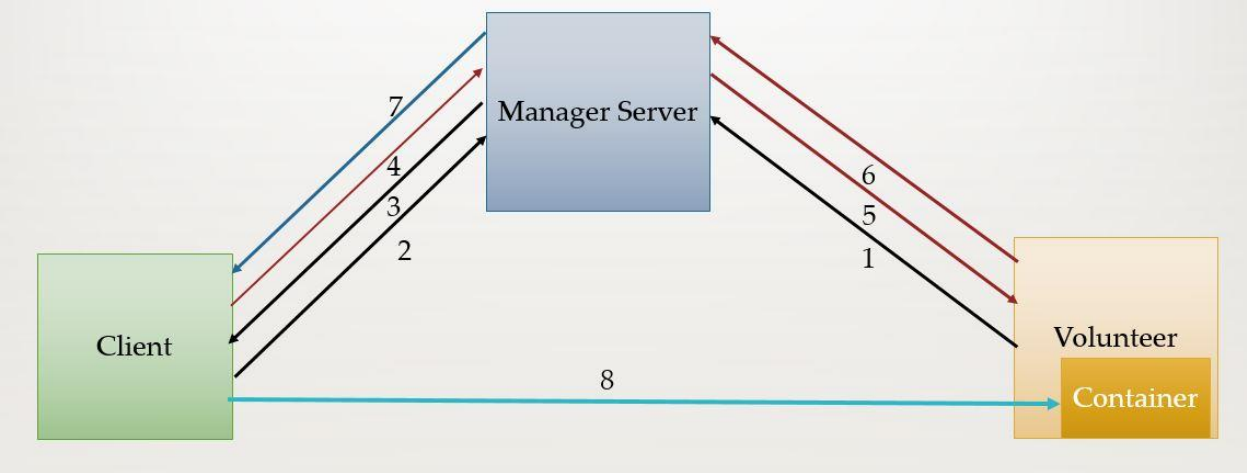
In 2004, BOINC started working on Volunteer Computing. Volunteer computing projects use a task server to manage work. Clients periodically communicate with the server to report completed tasks and get new tasks. The rate at which the server can dispatch tasks may limit the computing power available to the project. Bionic paper discusses the design of the task server in BOINC, a middleware system for volunteer computing. They presented measurements of the CPU time and disk I/O used by a BOINC server, and showed that a server consisting of a single

inexpensive computer can distribute on the order of 8.8 million tasks per day. With two additional computers, this increases to 23.6 million tasks per day.

VERTICO

The main aim of VERTICO is to donate Volunteer Resources for research areas like high-energy physics, molecular biology, medicine, astrophysics, climate study, and other areas. In vertico, each volunteer has the computer resources range : core count (1-10000 cores),core speed (0-5ghz),core brand (intel, NVidia),Ram size(0-1024 GB), Ram speed (0-100GB/s) ,Disk (0-10TB),Disk type(SSD/SATA,SSD/PCIE,HDD), Disk Speed (0-1000 Mb/s), Network speed (0-100Gb).

VERTICO Architecture:



The above architecture describes the VERTICO computing model.

1.In this framework, every Volunteer connects to a manager server when they are ready to donate resources. Manager server manages all the volunteer connections, stores the volunteer information securely and process the client requests.

2. Clients requests the manager server for required resources, sends the all the requirements to server.

3. Manager Server then performs the search with all the volunteers which are connected during that time. If search results any volunteer, then server sends the acknowledgement to client.

4.If any volunteer founds, then it asks the server about volunteer details.

5. Now, Manager server request the volunteer to start container.

6. After launching the container, volunteer sends the container details to server.

7.Manager server sends the volunteer details to client.

8.Finally, client can directly connect to container on volunteer system by using SSH.

In this project, I have used container over virtual machines because there are many benefits of container over Virtual machines.

1. Faster startup time: A containerized application usually starts in a couple of seconds. Virtual machines could take a couple of minutes.
2. Better resource distribution: Containers use up only as many system resources as they need at a given time. Virtual machines usually require some resources to be permanently allocated before the virtual machine starts. For this reason, virtual machines tie up resources on the host, even if they are not actually using them. Containers allow host resources to be distributed in an optimal way.
3. Direct hardware access: Applications running inside virtual machines generally cannot access hardware like graphics cards on the host to in order to speed processing. Containerized applications can.
4. Less redundancy: With virtual machines, you have to install an entire guest operating system, which duplicates a lot of the components already running on your host server. Containers don't require this.

In this framework, manager server need to store all the resource details and volunteer details. Initially I try to store using MySQL, but it is inefficient to search all the details of the volunteer. I tried with K-d tree which is to find nearest match. We need exact match but it doesn't lead optimal. This can be achieved by using containers if there is a match the requirements above with the client requirements. (example : If client needs 4 GB ram, no volunteer have same ram ,then we need to match the ram >4 only if it should match the other requirements.) Using container, we can restrict the resources on volunteer system.

Future work

We aim to run scalability and performance experiments on more than 100 volunteers, where each volunteer requires significant number of cores (1-10000), memory (0-1024 GB), storage (0-10 TB), and network bandwidth (0-100Gb).