Survey On Distributed Computing Platform

Ajit Sul¹, Vishal Razdan², Satyendra Tiwari ³, Rupali Pashte⁴

¹B.E Student, Dept. of Computer Engineering, PVPP College Of Engineering, Maharashtra, India
²B.E Student, Dept. of Computer Engineering, PVPP College Of Engineering, Maharashtra, India
³B.E Student, Dept. of Computer Engineering, PVPP College Of Engineering, Maharashtra, India
⁴Asst. Professor, Dept. of Computer Engineering, PVPP College Of Engineering, Maharashtra, India

Abstract - The main goal of this paper is to extend our learning and application on already available systems for creating Distributed Computing Applications. Distributed Computing Platforms give programmers an opportunity to create applications that have overall execution time less than their counterparts. There are various systems present in the market that enable us to create applications running on distributed system such as BOINC, WCG (WORLD COMMUNITY GRID). These systems have certain limitations when used by small projects which can be overcome. Limitations like platform dependency, architecture complexity, and knowledge about the platform are very challenging for the programmers. These limitations don’t play a very crucial role if dealing with the big projects but comes into picture if a project is small or medium sized. Hence throughout the paper we will discuss these systems and their limitations.

Key Words: platform, distributed, computing, application, small, projects, limitations

1. INTRODUCTION

Distributed Computing is an environment in which a group of independent and geographically dispersed computer systems take part to solve a complex problem, each by solving a part of solution and then combining the result from all computers. These systems are loosely coupled systems coordinately working for a common goal. The main aim of distributed computing was to reduce the execution time of the program without affecting the complexity of the program. As the name suggests “Distributed Computing” provides a platform through which different modules of the program can be distributed over a network to different processors thereby simultaneously executing all the modules on the network. It can be defined as

1. A computing system in which services are provided by a pool of computers collaborating over a network.
2. A computing environment that may involve computers of differing architecture and data representation formats that share data and system resources.

Key design issues that must be considered while designing any distributed computing systems are

- **Transparency**
  Overall system should look like a single coherent system to its user.
- **Scalability**
  Distributed system must be able to cope up with increase in number of nodes.
- **Heterogeneity**
  Distributed system should work with heterogeneous system i.e. it should be platform independent.
- **Fault – tolerance**
  Failure of individual node must be taken care of.
- **Task scheduling**
  Policy for distribution of task to individual node should be selected so as to maximize the performance.
- **Security**
  Due to involvement of networks, network security is one of the measure issues. Also, important data and other confidential things should not get revealed to the user of individual node.

Any Distributed system should take the above design issues in consideration.

2. EXISTING SYSTEMS

The concept of Distributed computing was introduced long ago and there are several systems that use this concept to provide a platform to create applications that will run on independent volunteer nodes. Some very powerful ones are BOINC and WCG.

2.1 BOINC (Berkeley Open Infrastructure for Network Computing)

The Berkeley Open Infrastructure for Network Computing (BOINC), an open-source middleware system, supports volunteer grid computing. BOINC (Berkeley Open Infrastructure for Network Computing) is a software...
system that makes it easy for scientists to create and operate public-resource computing projects. It supports diverse applications, including those with large storage or communication requirements. PC owners can participate in multiple BOINC projects, and can specify how their resources are allocated among these projects.[2]

Goals of BOINC:

- Reduce the barriers of entry to public-resource computing: In order to provide a platform for the programmers with the help of which publicly available resources can be used by the program.[4]
- Share resources among autonomous projects: Willing resources can be assigned to the projects.
- Support diverse applications: BOINC accommodates a wide range of applications; it provides flexible and scalable mechanism for distributing data, and its scheduling algorithms intelligently match requirements with resources.[5]
- Reward participants: The BOINC platform rewards its clients in order to attract participants. It assigns each participant with a score - credit in order to show how much they have contributed in the project.

Steps Followed By BOINC

- Subdividing long-running variable-length analyses into short, fixed-length BOINC work units: We describe a scheme for subdividing long-running, variable-length analyses into short, and fixed-length BOINC work units using phylogenetic analyses as an example. Fixed-length work units decrease variance in analysis runtime, improve overall system throughput, and make boinc a more useful resource for analyses that require a relatively fast turnaround time, such as the phylogenetic analyses submitted by users of the garli web service at molecularevolution.org[1] (This web site provides web services for software commonly used in molecular evolutionary analyses, including GARLI and GSI. You can run these programs on our grid resources using the web services available on this site).

- Distributing The BOINC work units to different clients:

Distributing Boinc work units to clients

1. Client PC gets a set of tasks from the project’s scheduling server. The tasks depend on client PC: for example, the server won’t give it tasks that require more RAM than client has. Projects can support several applications, and the server may send Client tasks from any of them.
2. Client PC downloads executable and input files from the project’s data server. If the project releases new versions of its applications, the executable files are downloaded automatically to Client PC.
3. Client PC runs the application programs, producing output files.
4. Client PC uploads the output files to the data server.
5. Later (up to several days later, depending on preferences) Client PC reports the completed tasks to the scheduling server, and gets new tasks. This cycle is repeated indefinitely. BOINC does this all automatically; clients don’t have to do anything.

- All the result is merged at the server as per the merging logic and the output required from the project. Once result of all the modules is submitted to the servers, the output can be displayed to the end user or to the stakeholders as per their interests.

Limitations of BOINC

- Platform Dependent Architecture: BOINC has a platform dependent architecture. As the servers set up by BOINC for any project can run only on LINUX servers. The BOINC client has been ported to several platforms, but the BOINC server can only be executed on Linux-based operating systems.[3] It requires researchers to have experience with Linux system administration in order to create a new BOINC project.

- Complexity: The architecture of BOINC is very Complex. The researchers creating BOINC projects must learn the BOINC programming API and be proficient in Linux system administration, MySQL15 relational database administration, the Extensible
Mark-up Language (XML), and the C++
programming language.[4]

- Lack of Documentation: This is the biggest barrier
  faced by the programmers for developing BOINC
  project. There are also very few tools to facilitate
  the creation of new projects, resulting in a long,
  manual process.

- Not Suitable for small and medium sized projects:
  Due to large overhead task of understanding the
  BOINC architecture which makes it unsuitable to be
  used for small and medium size projects. The
  complexities of BOINC can be prohibitive factors for
  researchers interested in creating small to medium
  size Public Resource Computing projects.

- BOINC provides a Grid Computing platform which is
  not helpful in case of small and medium sized
  projects. As Grid Computing consumes resources
  which are not necessarily required for solving such
  small and medium sized problems.

2.2 WCG (World Community Grid)

World Community Grid (WCG) is an effort to create the
world’s largest public computing grid to tackle scientific
research projects that benefit humanity. Launched on
November 16, 2004, it is co-ordinate by IBM with client
software currently available for Windows, Linux, Mac OS X,
and Android operating systems. World Community Grid
enables anyone with a computer, smart-phone or tablet to
donate their unused computing power to advance cutting-
edge scientific research on topics related to health, poverty
and sustainability. Through the contributions of over 650,000
individuals and 460 organizations, World Community Grid
has supported 26 research projects to date, including
searches for more effective treatments for cancer, HIV/AIDS
and neglected tropical diseases. Other projects are looking for
low-cost water filtration systems and new materials for
capturing solar energy efficiently. Through World Community
Grid, many volunteers from all over the world provide
computing power to advance leading research. Researchers are
using world community grid for majorly tackling health
problems. The computing power can be donated by all in the
world. Even the smart phone like devices can donate its
computing power for the projects. Using the idle time of
computers around the world, World Community Grid’s
research projects have analyzed aspects the human
genoem, HIV, dengue, muscular dystrophy, cancer, influenza,
Ebola, virtual screening, rice crop yields, and clean energy. As
of October 2014, the organization has partnered with 466
other companies and organizations to assist in its work, and
has over 55,000 active registered users.[6]

The overall working of WGC is beyond the scope of this
paper.

Limitations of WCG:
WCG also has similar limitations as that of the BOINC, except
the platform dependency.

3. PROPOSED SYSTEM TO OVERCOME THE
LIMITATIONS

System can be made for overcoming the limitations of BOINC.
Following are the things that are needs to be taken care of while
designing distributed computing platform:

- For Overcoming platform dependent architecture:
  - A system can be developed that can be executed on any
    platform. A system made in Java (platform
    independent) makes it possible to execute client/server
    programs on any platform thus making the system
    platform independent.

- For Overcoming Complexity: A system can be
  developed in which the server is always in a free state
  whereas the state of clients is maintained at client side
  only. This makes the architecture of the server very
  simple as the server does not have to remember
  anything about client except from maintaining the list
  of active clients present in the system. Due to which
  the whole architecture becomes simple.

- Making System Suitable for small and medium
  sized projects: Distributing the modules over local or
  metropolitan area network and not on wide area
  network makes it easy for the programmer. As the
  programmers have not to understand the architecture of
  the system and which makes easy for the programmers
  to use the system.

- No Documentation Required: As the architecture of
  the system is very simple, it is not necessary for the
  programmers to refer to the documentation of the
  designed system. Only the code needed for the
  formation of modules is needed to be written in the
  program.

4. CONCLUSIONS

Making server platform independent will facilitate us to
provide platform independent distributed computing
platform. Moreover platform dependency could be avoided
making a good use of resources and providing a reliable
platform for the programmers interested in small and
medium size projects.
REFERENCES


