Grid computing is bringing a force of great processing power to solving problems that are in need of increased amount of processing power. It is providing a practical and improved solution to using and maintaining supercomputers to perform tasks and research. However, much like every solution there are some flaws within grid computing. Once these issues are addressed and solved grid computing will continue to expand and provide solutions to new projects.

Grid computing is a distributed architecture of large numbers of computers connected to solve complex issues. In the grid computing model, servers or personal computers run independent tasks and are loosely linked through the Internet or low-speed networks and may even connect directly or via scheduling systems (Rouse & O’Reilly, 2018). Grid computing is typically applied to projects that have no time dependency. Larger projects are also spread across many countries and continents. Cycle-scavenging, or search programs that use the idle power of computers are used and running the background for many weeks (Rouse & O’Reilly, 2018).

An example of a grid computing project is that of SETI@home with is a search for extraterrestrial intelligence based at the University of California at Berkeley (Rouse & O’Reilly, 2018). A segmented piece of radio telescope data has a search program being run on it by millions of computers. There is no specified end date for this program. Grid computing also brings computing power to the fields of genetics research, drug-candidate matching, and even the search for Genghis Khan’s tomb (Rouse & O’Reilly, 2018).

Grid computing also has some setback and drawbacks, like that of the infrastructure. Middleware is used to create and control grids easier. SETI@home uses BOINC open source grid computing software (Rouse & O’Reilly, 2018). This allows the projects to build on the tools through the addition of user-friendly GUIs and a way to distribute raw data as well as receive and store results (Rouse & O’Reilly, 2018). The distributed model however, only works well for a narrow range of applications. The application of grid computing, ensure redundancy and robust failure recovery are built into the model, seeing as it is very likely that many of the compute nodes will disconnect or fail (Rouse & O’Reilly, 2018). In other words, the typical grid computing infrastructure and set up is not very reliable. Therefore, there has been much work done to seek out more reliable set ups that also include increase scale, heterogeneity, and dynamism (Rouse & O’Reilly, 2018).

There has been a lot of effort directed toward ensuring the reliability of computing resources that make up grid systems. Grid resources include processor clusters, supercomputers, storage devices, and related hardware, all together with an operating system and the software for managing these resources (Rouse & O’Reilly, 2018). Software used for data mining is also included in these resources along with data stores used in data and multimedia grids. Fault tolerance is a focus of a developing method to ensure the reliability of grid computing. Fault tolerance consists of detecting faults and failure in grid resources and recovery to allow computations to keep being performed (Dabrowski, 2009). Researchers has investigated scalable fault detection methods for grid systems environments as well as fault isolation and diagnostic techniques to determine what kind of fault occurred. Hard-to-detect faults are expected to occur in large scale grids, however, most failure detection method so far remain experimental (Dabrowski, 2009).

There are limitations for fault detection on large scale grids. One reason is that available network monitoring protocols and tools , like those based on SNMP, rely on detailed knowledge of network infrastructure (Dabrowski, 2009). This knowledge is not always readily available in large-scale, dynamic environments. Failure detection in grid systems can also be compromised through asynchronous distributed systems in which management functions are decentralized and subject to failure (Dabrowski, 2009). Scientists tried to combat this problem using heartbeat techniques in which resources regularly sent messages or “heartbeats” to other members of a heartbeat group. If the beats were absent, group members then would use a consensus procedure to determine which members had failed (Dabrowski, 2009). However, this method degraded performance of the systems through bottle necking and did not account for network load scenarios, thus leading to more research and methods.

Horita proposed a scalable, self-organizing fault detection system based on earlier work on using group membership protocols for fault detection (Dabrowski, 2009). In this method, each process was monitored by a small group of randomly selected processes on remote nodes (Dabrowski, 2009). Thus, creating a virtual monitoring subnetwork with a grid that consisted of heterogeneous resource types. Through which notifications of connection failure could be propagated (Dabrowski, 2009). Experiments showed scalability for a system of three node grid sites that contained 300 resources (Dabrowski, 2009). Resources were organized into separate domains in which they emitted heartbeat to a domain monitor. There, the monitoring domains were structured hierarchically for scalability (Dabrowski, 2009).

Detecting different types of faults was also an important aspect of seeking completeness and accuracy. Early work on fault-handling framework could distinguish between different kinds of failure during simulated grid operations (Dabrowski, 2009). It could also initiate recovery action to remedy different fault types. Jisumoto developed a detector that differentiated between hardware, process, and transmission faults. Users were allowed to preselect a recovery procedure to be invoked in response to occurrence of a particular fault type (Dabrowski, 2009). This method tested well in a 32-node cluster testbed (Dabrowski, 2009). Later an adaptive scheme was used to periodically store the state of replicated processes that executed in parallel (Dabrowski, 2009). The stored states were compared with discover erroneous companions affected by transient faults. This procedure was computationally intense, the checkpoint interval was dynamically varied in response to the observed frequency of faults to improve efficiency (Dabrowski, 2009). Jin developed a hierarchical grid failure detector and failure handler then adapted to changing user requirements and system conditions. Tests showed that this scaled to up to 1000 components at two sites (Dabrowski, 2009).

Within some grid environments some types of faults may be difficult to find. “Silent” fault types do not typically indicate their presence immediately after they occur (Dabrowski, 2009). Another kind of fault, known as Byzantine faults cannot easily be traced to failed links, processes, and messages(Dabrowski, 2009). They are created when equipment periodically or randomly malfunctions due to aging, outage, or external damage, and even electromagnetic interference. Faults can propose cascading failures over a grid system and methods for isolating faults is still being studied (Dabrowski, 2009). Faults lower user confidence, thus continuing efforts to develop scalable method for isolation and detection of types of faults are still very important today.

Security is another area where grid computing is in need of improvement and solutions. Grid computing has distinct security issues when compared with traditional computing systems, stand-alone, or networked architecture (Geetha & Ramyachitra, 2013). These security problems include impact on local host, vulnerable hosts, interception, and packet losses.

With grid computing alien code is running in the host system. Thus, code can interfere jobs that are running locally and compromise local data security, thus impacting the host (Geetha & Ramyachitra, 2013). Clients using the grid remain in danger from the local hosts as well. Local hosts shutting down resulting in denial of service, viruses, or other malware in the local host affecting an entire process, and local hosts compromising client data integrity and confidentiality are all major vulnerabilities (Geetha & Ramyachitra, 2013).

Another security risk is that of an attacker intercepting the resources and data in the grid. These attacks can take many forms such as a distributed denial -of-service attack and others like that (Geetha & Ramyachitra, 2013). Packet loss is also a concern and security risk. The interruption of nodes during the routing process to get packets from source to the destination decreases total packet delivery and loss or corruption of data. However, it is important to keep in mind that the security risks also depend on the intellectual property put in the hosted environment of a grid system (Geetha & Ramyachitra, 2013).

Solutions to security risks are important to research and develop. One solution includes a monitoring agency with the role of monitoring resource usage, institutionalizing a trust management system to, manage, and negotiate trust among different units within the grid, and establish an authorization system to grant user access specific resources (Geetha & Ramyachitra, 2013). The user that is running an application on a remote machine in the grid network requires assurance of the machine retaining its integrity, thus ensuring that proprietary application remains safe (Geetha & Ramyachitra, 2013). The local host requires a similar assurance in regard to client data and processes that are being run on the host. The safeguards of a traditional system aim at protecting the system and data from its users, the security orientation of grid systems need to be a step ahead (Geetha & Ramyachitra, 2013). They should also protect application and data from the system where the computation takes place.

Strong Authentication and restrictions on local execution from remote systems is required for security. Some ways to include this are using a public key cryptography to secure grid communication, mutual authentication of the participant, single sign on or deletion capacity, firewalls, SSL layers, filtering and auditing data, erasure of data after use, and hardware based Virtual Private Network (Geetha & Ramyachitra, 2013). As security problems with grid computing find increased resolution with the advancement of technology, it is sure to become more and more widely used.

Grid computing is a concept that not only speeds up computing but causes a new era o begin within computing. However, some challenges still stand in the way of grid computing, like that of infrastructure and security. However, research is being conducted and is providing new solutions for these issues. As this technology advances there will be an increase in beneficial solutions to the problems faced now in grid computing.

Work Cited

Abbas, Ahmar. *Grid Computing: a Practical Guide to Technology and Applications*. Charles River Media, 2004.

Dabrowski, Christopher. “Reliability in Grid Computing Systems.” *Concurrency and Computation: Practice and Experience*, vol. 21, no. 8, 2009, pp. 927–959., doi:10.1002/cpe.1410.

Geetha, and Ramyachitra. “Security Issues in Grid Computing.” *International Journal of Computer Applications®* , 2013, doi:(IJCA) (0975 – 8887).

Nayab. “What Are the Security Problems with Grid Computing?” *Bright Hub*, 7 Nov. 2010, www.brighthub.com/environment/green-computing/articles/94587.aspx.

Rouse, Margaret, and Jim O'Reilly . “What Is Grid Computing? - Definition from WhatIs.com.” *SearchDataCenter*, 2018, searchdatacenter.techtarget.com/definition/grid-computing.