Introduction

SINGLE-CORE PROCESSOR TO MOBILE CLOUD COMPUTING: EVOLUTION AND FUTURE

The evolutions of computing devices, communication networks and processor-level architectures have witnessed a rapid growth during last couple of decades. The semiconductor devices are increasingly reduced in size, which have given the birth of mobile computing devices such as, smart-phones, PDAs and laptop computers. However, the mobile computing devices have an inherent limitation related to available capacity of battery power. The limitation of electrical energy source has resulted in the limitation of functionalities of mobile computing devices and their computing lifetime. However, the platforms in traditional computing domain involving computers have gone through evolution too. The single-core CPU (Central Processing Unit) architecture has grown into multi-core CPU architecture. In addition, the main memory capacity has increased enormously during last couple of decades. In the domain of data communication networks, the network architectures have evolved from LAN (Local Area Network) to WAN (Wide Area Network) and lastly the Internet, which is a geographically distributed network. The recent addition to computer networking is the mobile communication network integrated with the Internet. The integration of such heterogeneous network architectures is essential for realizations of Internet-equipped smart-phones and location-transparent mobile computing paradigm. The evolutionary trajectory of distributed computing systems involving multiple computing devices can be partitioned into several stages such as, distributed systems, cluster computing, grid computing, cloud computing and, lastly the mobile cloud computing.

Distributed Systems

The traditional distributed systems are comprised of multiple computing nodes or processors interconnected to have data communication pathways. The early model of multi-computing is categorized as a tightly-coupled system, where the address-space is single. Thus, a single system-wide primary memory is shared by multiple processors (nodes). The main limitations of this model are bus-contention and, process-wait cycles due to shared-memory at the level of system architecture. The other end of the spectrum is loosely-coupled systems, where architecture-level shared memory is replaced by distributed shared memory. In loosely-coupled multi-processor systems, the nodes have own local address-spaces and, a part of primary memory can be shared between nodes. The distributed systems are the result of integration of computer architectures and the communication networks. The architectural models of distributed systems are basically comprised of individual computing nodes connected by computer networks.

Theoretically, this can be viewed as a graph of computing nodes. The main goal was to enhance speed of computation by executing relatively independent processes in virtual-parallel model. The distributed processes communicate to share data and to synchronize using communication networks. The models of traditional distributed computing systems can be classified into five broad classes namely, minicomputer model, workstation model, workstation-server model, processor-pool model and, hybrid model. These models are based on multiple computers, which can be categorized as minicomputers, workstations and servers. The communication medium is reliable and high-bandwidth wired network.

Cluster Computing

In the traditional models of distributed systems, the concept of clustering of nodes to perform some dedicated computations is absent. The high-performance computations often require a dedicated set of nodes connected by very high-speed wired network to accept, distribute and schedule computational jobs (processes) with user-transparency. The traditional distributed computing systems have evolved into cluster computing to achieve this goal. However, the resource management and task-assignments to nodes are the two main technical challenges of cluster computing systems. The goal is to maximize resource utilizations at each node to achieve optimal overall resource utilization globally in a cluster at any point of time. Thus, there are two issues to look at:

1. Global task-assignment to the nodes from the batch of jobs, and
2. Optimal local scheduling of tasks or processes at a node.

In general, the designing an optimal global task assignment algorithm is a computationally hard problem. A cluster comprised of heterogeneous nodes makes it difficult to design a globally optimal task assignment algorithm independent of time. The cluster computing systems often recognize the batch jobs into two such as, sequential tasks and, parallel tasks. The sequential tasks are highly data-interdependent computational processes requiring sequential computing. However, parallel tasks are comprised of data-parallel sub-tasks, where sub-tasks can execute in parallel in time at different nodes requiring data communications in intervals. This indicates, the resource management and load-balancing algorithms of cluster computing systems should consider the task categories in order to decide near optimal task assignments to nodes. However, prior determination of degree of data-dependency and data-parallelism within a large task is highly complex.

Grid Computing

The cluster computing platforms are constrained to localization of resources for speedy access. However, the advent of Internet protocol induced another change in the distributed computing paradigm. The Internet protocol allows standardized and transparent data communication between any two geographically distributed nodes. Additionally, the protocol is capable to integrate heterogeneous computing resources over the network. Thus, the grid computing paradigm is born, where the computing resources are geo-distributed and heterogeneous nodes are connected by Internet. However, the grid computing software is designed to provide user-transparency and location-transparency. The grid software architecture is designed by employing layered and component-based software architecture model. Each layer of the grid software architecture implements a set of capabilities and functions based on the software layer below it.

In other words, if one moves from lowest layer to highest layer of stratified layered grid software model, then one can see that capabilities and functions are getting expanded through augmentation. The lowest layer of grid software architecture represents the resource fabric of the grid. The resource discovery and sharing is implemented by specific grid communication protocol between software components. The Internet protocol provides a basic data-carrying infrastructure to be used by grid protocol for inter-component communications. The resource allocation, metering and, authentication functions are realized at different vertical layers of grid software. The inter-layer communication is implemented by standardized interface definitions and protocols. Thus, the grid computing systems provides a geo-distributed, user-transparent, location-transparent and, reliable distributed computing infrastructure comprised of large array of heterogeneous computing resources.

Cloud Computing

The grid computing paradigm is further evolved into service-oriented computing architecture. In absolute technical view, arguably cloud computing is a traditional form of distributed systems with a new commercial approach. In cloud computing paradigm, the user applications are executed on distributed systems as a service and the required system software modules provide the cloud platform. In other words, a cloud platform is a cluster of distributed systems comprised of heterogeneous hardware and software offered to users for a commercial price as a service to execute user applications. If the users are general public in society needing computational facilities, then the cloud platforms providing such services for a price are called public-cloud. The other spectrum is private-cloud, where the underlying distributed systems and data centers are not accessible to common users but to specific organizations. The main technical challenge of public-cloud platforms is scalability of the systems. The employment of efficient resource virtualization mechanism is another technical area needing attention in cloud computing paradigm. However, virtual machine models are generally complex to design and, computationally expensive wasting available resources.

Hybridization I: Virtualized Cluster

The cluster computing paradigm was initially centered on homogeneity of computing software platforms in order to reduce complexity. However, the proliferations of computing applications at different domains have resulted in heterogeneous applications needing heterogeneous computing resources. The mechanism to isolate data and applications of different users residing in a cluster was not well addressed in the initial designs of cluster platforms. The users from different organizations often share a cluster computing platform, which resembles the grid computing paradigm. Thus, the need to isolate and protect data and executing applications from different users in a shared cluster becomes evident. Lastly, the performance requirements or Quality-of-Service (QoS) of different user applications are different. The requirement to maintain differential management of QoS of applications having specific QoS requirements appears to be necessary in a cluster if the respective cluster is intended to execute a wide array of heterogeneous applications concurrently.

Evidently, cluster computing platforms started to evolve another direction by hybridizing with virtualization techniques of computing resources in order to address the additional set of requirements. The employment of virtualization of computing resources in a cluster computing platform is aimed to decoupling computational services from the physical hardware devices or platforms . In this hybrid model,

the users have the traditional interface to cluster computing platforms; however the back-end is split into two sections. One section on back-end is comprised of virtual work-nodes connected by virtual network and, the other section on back-end is comprised of physical infrastructure. The interface between the two sections is a software layer called Distributed Virtualizer (DV). The hybrid model of cluster computing paradigm incorporating virtual machine model can support the new set of requirements arising from shared cluster platforms executing heterogeneous user applications offering QoS and data isolation.

Hybridization II: Mobile Cloud Computing

The cloud computing paradigm can be categorized into two namely, general cloud platforms and, mobile cloud platforms. In the case of general cloud platforms, the nodes (following client/server model) are static and underlying network is wired network. However, the advent of mobile computing devices has created a different environment, where the clients are resource-thin and mobile in nature. These characteristics of clients have raised two major research questions such as,

1. How to partition application and system software stacks to distributed computation between mobile clients and cloud servers, and
2. How to manage mobility of clients within the a cloud platform.

The issues related to research question (2) can be further subdivided into followings: how to manage hopping and network connectivity to multiple servers by mobile clients in a cloud, how to authenticate a mobile client in a cloud and, how to handle data migration due to mobility of clients in a cloud. One approach to address these research questions is to view the mobile cloud computing paradigm as a hybridization of general cloud computing paradigm and mobile computing paradigm.

Following this hybrid view, the challenges associated to mobile cloud computing paradigm are fine grained in taxonomy such as, issues related to end users, issues related to mobile cloud service providers, issues related to service-level agreements, issues related to security, authentication and protection, issues related to context of mobility of clients and, issues related to efficient data management. As mentioned earlier, the mobile clients are resource-thin having very limited computational capabilities and more importantly, mobile clients are effectively the users of services provided by cloud platforms. Often, the continuity of access to main cloud platform by a mobile user is impossible to guarantee due to involvement of heterogeneous networks comprised of back-end static wired networks and wireless/mobile communication networks on mobile user side. Thus, the concept of cloudlet is formed, where the intermediate servers are used between a mobile user (client) and a cloud platform. These intermediate compute and communication servers are called cloudlet. Hence, a mobile user (mobile client), cloudlet and a cloud platform forms a chain comprised of heterogeneous networks between different stages.

It is important to note that, offloading of computations and migration of virtual machines (computation and data) are two important research issues in general cloud as well as mobile cloud computing environments.

CONCLUSION

The traditional distributed computing systems have evolved a long way. The fundamental theoretical models and algorithmic designs of distributed computing remain unchanged. However, the evolution of user applications executing on distributed systems has resulted in a set of new requirements in designing the distributed systems. The traditional model of static-client/static-server model of distributed systems has given way to mobile-client/static-server model of distributed systems. The virtualization of computing resources becomes necessary to execute heterogeneous user applications fulfilling individual QoS requirements of different applications. However, the overall performance degradation and resource wastage due to employment of virtual machine models are major concerns.

Susmit Bagchi
Gyeongsang National University, South Korea

ADDITIONAL READING

Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems Journal, 29*.

Lin, X., Mamat, A., Lu, Y., Deogun, J., & Goddard, S. (2010). Real-time scheduling of divisible loads in cluster computing environments. *Journal of Parallel and Distributed Computing*, 70.

Montero, R. S., Vozmediano, R. M., & Llorente, I. M. (2011). An elasticity model for high throughput computing clusters. *Journal of Parallel and Distributed Computing*, 71.

Sinha, P. K. (1997). *Distributed operating systems: Concepts and design*. IEEE Computer Society Press.

Yang, C. T., Cheng, K. W., & Shih, W. C. (2007). On development of an efficient parallel loop self-scheduling for grid computing environments. *Parallel Computing Journal, 33*.