**K.L.N.COLLEGE OF INFORMATION TECHNOLOGY**
**POTTAPALAYAM, SIVAGANGA (Dt). Pin: 630 612**

**CLASS TEST KEY – 1**

**Branch / Section: B.E IT A&B** **Year / Semester: IV / 7**
**Course Code / Name: CS6703/Grid and Cloud Computing** **Date: 20.07.2017**

**1. Compare parallel and distributed computing paradigms.(K2)**

| Parallel | Distributed |
|---|---|
| The Parallel system are **tightly coupled**. | The Distributed systems are **loosely coupled**. |
| All processors may have access to a **shared memory** to exchange information between processors. | Each processor has its **own private memory**. Information is exchanged by passing messages between the processors. |
| Large problems are divided into smaller ones, which are then solved **concurrent**ly. | An important goal and challenge of distributed systems is **location transparency**. |
| System time not depent on global clock | System time depents on global clock. |

**2. What are the four major technologies that cloud converge on?(K2)**

Cloud computing is enabled by the convergence of technologies in four areas:

1. Hardware virtualization and Multi-core chips
2. Utility and Grid computing
3. SOA, Web 2.0, and WS mashup
4. Atonomic computing and Data center automation.

**3. Explain which parallelism technology is best in current era.(K2)**

The JLP (Job Level Parallelism) technique is the best one. Because of the Instruction Level Parallelism (ILP) and Task Level Parallelism(TLP) are **performance improvement for generating programming and execution of parallel processing**. But the JLP (Job Level Parallelism) is the new technology **improved from parallel processing in to distributed processing**. The coarse-grain parallelism is built on top of fine-grain parallelism. For example Grid and Cloud.

**4. Show how the cloud computing resolves the traditional systems bottlenecks.(K1)**

The traditional distributed computing network systems bottlenecks are constant system maintenance, poor utilization, and increasing costs associated with hardware / software upgrades.

Cloud computin as an On-demand computing paradigms resolves these problems by its three cloud service models.

1. Infrastructures as a Service (IaaS): server, storage, networks, and data center. User does not manage or control it.
2. Platform as a Service (PaaS): Middleware, databases, development tool, run time support Web2.0 and Java. User to deploy to built application onto a virtualized platform and free from to manage infrastructures.
3. Software as a Service (SaaS): Browser initiated application software over thousands of paid cloud customers. Business process, ERP, Industry applications, Consumer relationship manangement (CRM), Human Resource (HR). Customer freed from to invest in Server and Software License.

**5. How SOA communication established from low-level to high-level?(K1)**

In SOA low-level considered as entity level interfaces, which are the Web Services Description Launguage (WSDL), Java Method, and CORBA interface definition language (IDL) specifications. These interfaces are linked with customized, high-level communication systems as SOAP, RMI, and IIOP.

These communication system support features including particular message patterns such as RPC, fault recovery, and specialized routing.

**PART – B** **1 × 15 = 15 Marks**

**6. (a) 1. Explain the Grid Computing Infrastructures. (8)(K2)**

**Introduction** **(2)**

Grid computing infrastructure comes from the growth of Internet to Web and Grid computing services. Internet services (Telnet) command enables a local computer to connect to a remote computer. A web service (HTTP) enables remote access of remote web pages. Grid computing is to allow close interaction among application running on distant computers simultaneously.

The Grid computing infrastructure provides two kind of computing environments.

1. Computational / Data Grid
2. Peer-to-Peer Grid

**Computational Grid** **(2)**

The grid infrastructure couples computers, software / middleware, special instruments, and people and sensors together. The gird is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale.They can also be viewed as virtual platforms to support virtual organizations. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system.

The resource sites offer complementary computing resources, including workstations, large servers, a mesh of processors, and Linux clusters to satisfy a chain of computational needs. The grid is built across various IP broadband networks including LANs and WANs already used by enterprises or organizations over the Internet. The grid is presented to users as an integrated resource pool .

Computational grid or data grid providing computing utility, data, and information services through resource sharing and cooperation among participating organizations. Special instruments may be involved such as using the radio telescope in SETI@Home search of life in the galaxy and the austrophysics@Swineburne for pulsars. At the server end, the grid is a network. At the client end, we see wired or wireless terminal devices.The grid integrates the computing, communication, contents, and transactions as rented services. Enterprises and consumers form the user base, which then defines the usage trends and service characteristics. Many national and international grids will be reported, the NSF TeraGrid in US, EGEE in Europe, and ChinaGrid in China for various distributed scientific grid applications.

**Grid Families** **(2)**

Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platform development by IBM, Microsoft, Sun, HP, Dell,Cisco, EMC, Platform Computing, and others. New grid service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades. In grid systems are classified in essentially two categories: computational or data grids and P2P grids. Computing or data grids are built primarily at the national level.

**peer-to-peer network families** **(2)**

An example of a well-established distributed system is the client-server architecture. In this scenario, client machines (PCs and workstations) are connected to a central server for compute, e-mail, file access, and database applications. The P2P architecture offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented. In this section, P2P systems are introduced at the physical level and overlay networks at the logical level.

**P2P Systems**

In a P2P system, every node acts as both a client and a server, providing part of the system resources. Peer machines are simply client computers connected to the Internet. All client machines act autonomously to join or leave the system freely. This implies that no master-slave relationship exists among the peers. No central coordination or central database is needed. In other words, no peer machine has a global view of the entire P2P system. The system is self- organizing with distributed control. The architecture of a P2P network at two abstraction levels. Initially, the peers are totally unrelated. Each peer machine joins or leaves the P2P network voluntarily. Only the participating peers form the physical network at any time. Unlike the cluster or grid, a P2P network does not use a dedicated interconnection network. The physical network is simply an ad hoc network formed at various Internet domains randomly using the TCP/IP and NAI protocols. Thus, the physical network varies in size and topology dynamically due to the free membership in the P2P network.

**2. Show that increasing cluster size alone may not result in a good speedup stated by Amdahl's laws sequential bottleneck. (7)(K2)**

Supppose if a fraction of code in a program must be executed sequentially, then this sequential operation is called the sequential bottleneck.

Consider a program execute on a uniprocessor workstation with a total execution time of T minutes.

The fraction $(\alpha)$ executed sequentially, therefore $(1- \alpha)$ of the code can be compiled for parallel execution by 'n' processors.

The total execution time of the program is calculated by $\alpha T + (1- \alpha)T/n$

where $\alpha T$ is the sequential execution time on a single processor

$(1- \alpha)T/n$ is the parallel execution time on 'n' processing node.

By the Amdahl's Law, speedup factor of using the 'n' processor system over the use of a single processor is expressed by:

$$\text{Speedup } (S) = T/ [\alpha T + (1- \alpha)T/n] = 1 / [\alpha + (1- \alpha) / n]$$

If $\alpha$=0; the cluster size is large $(n->\infty)$, then Speedup $(S) = 1 / \alpha$ an upper bound. This upper bound is independent of the cluster size 'n'.

For example:

The maximum speedup achieved is 4, if $\alpha$=0.25 or $1- \alpha = 0.75$ even if one uses hundreds of processors.

If n=4; then S = 1 / [0.25 + (0.75 / 4)] = 1 / 0.4375 = 2.28

If n=100; then S = 1 / [0.25 + (0.75 / 100)] = 1 / 0.2575 = 3.88

If n=256, then S = 1 / [0.25 + (0.75 / 256)] = 1 / 0.2529 = 3.9

Increasing the cluster size from 100 and 256 processors, the speedup is not much varied. Therefore increasing the cluster size alone may not result in a good speedup.

In Amdahl's law, assumed the same amount of workload for both sequential and parallel execution of the program with a fixed problem size or data set called ***fixed workload speedup***. The system efficiency is

$$E=S/n = 1 / [ \alpha n + 1- \alpha ]$$

To execute a fixed workload on a Cluster with n=256 nodes,

$$E = 1 / [0.25x256 + 0.75] = 1 / [64+0.75] = 1 / 64.75 = 0.0154 = 1.5 \%$$

When the cluster size is very large the system efficiency is low. This is because only a few processors are kept busy, while the majority of the nodes are left idling.

**(OR)**

**(b) 1. Explain about the Service-Oriented Architecture (SOA) (8)(K2)**

**Introduction (2)**

In grids/web services, Java, and CORBA, an entity is, respectively, a service, a Java object, and a CORBA distributed object in a variety of languages. These architectures build on the traditional seven Open Systems Interconnection (OSI) layers that provide the base networking abstractions.

On top of this we have a base software environment, which would be .NET or Apache Axis for web services, the Java Virtual Machine for Java, and a broker network for CORBA.

## Layered Architecture for Web Services and Grids (6)

| | | |
|---|---|---|
| Higher Level Services | Higher Level Environment for special feature of Dist.Computing Environment (High Level Communication) | SOAP, RMI, IIOP, XML, APACHE AXIS, XML over HTTP (REST) |
| Service Context | Service Management | 1.Service State<br>2.Life Time Support<br>Jini Life Time model<br>EJB model |

| | | |
|---|---|---|
| | | CORBA Life Cycle<br>Suit of Web Services |
| | Service Discovery and Information | JNDI – Jini and Java Naming and Directory Interface<br>CORBA Trading Service<br>UDDI – Universal Description and Discovery Interface<br>LDPA – Lightweight Directory Access Protocol<br>ebXML – Electronic Business using eXtensible Markup Language. |
| Service Internet | Service Internet Transport --><br>Protocol Service Interface<br>(Low Level Communication) | WSDL ---> SOAP<br>IDL ---> IIOP<br>Java Method ---> RMI |
| Base Hosting Environment | Base Software Environment | WS – NET, Apache Axis<br>Java – JVM<br>CORBA – Broker Network<br>commonly: WebSphere MQ<br>Java Message Service |
| | | |
| OSI Layers (7) | Appln: HTTP, FTP, DNS,etc.<br>Pres: XDR, etc.<br>Sess: SSH, etc.<br>Trans: TCP, UDP, etc.<br>N/W: IP, etc.<br>Data Link / Physical | Bit Level Internet |

**2. Show that how Gustafson's law solve the scaled problems. (7)(K2)**

To achieve higher efficiency when using large cluster, we must consider scaling the problem size to match the cluster capability. John Gustafson referred ***scaled – workload speedup*.**
Let W be the workload in a given program. When using an 'n' processor system, the user scales the workload to $W' = \alpha W + (1-\alpha)nW$.
Note: Only the parallelizable portion of the workload is scaled 'n' times in the second term. This scaled workload W' is essentially the sequential execution time on a single processor. The parallel execution time of a scaled workload W' on 'n' processors is defined by a scaled-workload speedup as

$$S' = W' / W = [\alpha W + (1-\alpha)nW] / W = \alpha + (1- \alpha) n$$

The parallel execution time at level W, the efficiency expression is

$$E'=S'/n = \alpha / n + [1- \alpha ]$$

We can improve the efficiency of using a 256-node cluster to

$$E' = 0.25 / 256 + 0.75 = 0.751 = \mathbf{75.1\%}$$

In a fixed workload on a Cluster with n=256 nodes,

$$E = 1 / [0.25x256 + 0.75] = 1 / [64+0.75] = 1 / 64.75 = 0.0154 = \mathbf{1.5\ \%}$$

Comparing the fixed and scaled workload efficiency, usind Gustafson's law increased the efficiency using 256 nodes.