Assignment-1 FIT3142 Report

Student ID – 26356104
Name – PAARTH BHASIN

(a) Summarise the differences between the BSD Socket Inter Process Communications and the CORBA schemes.

Basic Differences:

Firstly, the BSD Socket Scheme is part of the Stream Oriented IPC whereas CORBA is part of the Remote Object Invocation scheme. BSD Sockets are based on sockets which are a peer-to-peer communication endpoint abstraction. It is a low-level communication/network-programming interface/API. On the other hand, CORBA is a middleware that allows heterogeneous client and server applications to communicate in distributed systems.

Architecture:

CORBA is a much more high-level abstraction scheme compared to the BSD Socket scheme & provides a suite of pre-built complex services that can be used by the client, which otherwise must be manually implemented in the Socket scheme.

Infrastructure and Implementation:

With CORBA, most of the repetitive & error-prone complexity of developing distributed applications are handled and taken care of by its reusable infrastructure. Marshalling in programming refers to the conversion of memory representation of an object to a suitable data format, which can be used for proper storage of the object.

In the socket scheme, marshalling and unmarshalling of complex messages containing arrays, nested structures or floating-point numbers requires a considerable amount of effort on the programmer's side. In addition to this, developers must also ensure that the client and servers never go out of sync when changes are made. Whereas with CORBA, parameters are passed in the request or response and are automatically & transparently marshalled by the ORB (Object Request Broker). This marshalling process ensures that applications and objects running on different computer architectures are correctly interworking.

Communication, Set-Up and Error Handling:

The socket connection has to be manually configured and the code for receiving and sending messages with sockets is subtl, highly complicated where the programmer has to detect and implement all error-handling. On the other hand, with CORBA, to invoke a service (like achieving a connection on BSD Sockets) an application just needs to have a reference to the service. All the other common communication infrastructure activities which are done manually by programmers in the socket scheme, are automated by the ORB. These activities can range from locating a service, if required then activating it, delivering request to it, and returning a response back to the caller.

Interfaces:

The BSD Socket Interface is defined using C functions, whereas the CORBA object interface is defined using the Interface Definition Language (IDL), which is similar to C++ but is much simpler.

Interoperability and Portability:

With BSD Socket Scheme we get a dependency on sockets within the source code. Using this code on another platform without sockets will require major changes to the source code, and hence the BSD Socket Scheme fares very poorly in terms of portability. For CORBA, portability is still an issue but does a much better job in comparison to the socket scheme. Portability of an application from ORB to ORB is limited until more conformance happens. The OMG (Object Management Group) has recently approved the IDL to C++ mapping, Naming service and ORB initialisation service. But very few ORBs provide services conforming to these standards.

(b) Summarise the differences between the CORBA scheme, and the Web Services scheme.

Basic Differences:

CORBA cannot be directly compared to Web Services/SOAP. Web Services are a protocol of the RPC mechanism which corresponds to IIOP (Internet Inter-Orb Protocol) in CORBA. Web Services are essentially a different form of CORBA with a lot of important things missing like Security & Notification.

Architecture:

CORBA has a pure object-oriented architecture, but Web Services are message based despite the fact that SOAP stands for Simple *Object* Access Protocol.

CORBA scheme is stateless, i.e. it does not remember the details of the client and server, whereas SOAP is stateful, i.e. it needs to remember client and server details.

CORBA causes the Client and Server to be tightly coupled. Therefore, they share the same interface and have to run ORB on both as well. Interaction between the client and the server is direct: client obtains handle on CORBA object and calls methods on it. However everything is de-coupled in Webservices. The client sends or receives messages, but a response does not automatically imply that access to next step is possible.

Scalability and Reliability:

In CORBA, the Portable Object Adaptor (POA) policies combine with fault-tolerance and load balancing features to provide application scalability. These features are not present in Web Services, but instead must be implemented by Application servers themselves.

Static and Runtime Checks:

CORBA uses IDL as a contract language. It is strongly-typed language, provides static guarantees like compilation time, performance, output and error handling ability, at runtime. With Web Services on the other hand, there is no Infrastructure support to do the static-checking. At runtime, only SOAP message structure is verified for correctness and the payload sent only has to be a well-formed XML message in terms of it structure. The application itself will have to verify the payload to ensure validity. Despite this, the checks involved during SOAP message validation are a lot more fine-grained and detailed than IDL based checks.

Request Semantics:

For data consistency, infrastructure has to provide at-most 1 semantic to protect from multiple executions of the same type of client request running on the server. CORBA ORBs provide 1 semantic and thus ensure data integrity. In web services, data integrity is defined by protocol underlying SOAP, ex. HTTP, which is more than 1 possibility, and hence does not ensure data integrity.

Interface Understandability:

Web Services/SOAP uses XML, which is a good tool, but a C/C++ programmer will have a lot more difficulty reading and deducing it, compared to the simple IDL language used in CORBA.

Standard Language Mappings:

Web-Services/SOAP do not have a portable standard API. You can't map a program's objects and data model to SOAP or vice-versa. Web Services don't have a language and platform independent interface like CORBA's IDL. On top of that, CORBA also has fully portable POA (Portable Object Adaptor), which makes all the client-server code portable across any other ORB, unlike Web-Services/SOAP which is not portable.

Programmer Effort:

Performing a SOAP request is complicated; programmer has to generate the message themselves, put the arguments inside and then send the message. Then they have to wait for a reply, parse the XML reply to extract the required data. CORBA is a lot more functional than SOAP/Web Services. The programmer does not have to handle call objects, set method names, extract results, check for faults etc.

(c) Summarise the differences between the cluster model and the Cloud Computing model.

Basic Differences:

Cloud computing is a technology that delivers different kinds of resources like Database, Virtual OS etc. as services (i.e. they can be used on demand), usually on the internet. On the other hand, cluster computing focuses more towards improved performance and availability of a service by inter-connecting a number of stand-alone computing machines to form a single computing resource.

Clusters are mainly implemented for load balancing and providing high availability.

Clusters are a group of computers connected by a local area network, whereas cloud is more wide-scale and can be distributed geographically, i.e. a cluster is tightly coupled whereas a cloud model is loosely coupled.

In clusters, the machines involved are made of the similar hardware configurations, whereas for cloud, the machines involved can and do have different hardware configurations, i.e. cloud can be heterogeneous or homogeneous in terms of machines, but clusters must be homogeneous.

Cloud does not provide security and privacy of user information, but clusters do. This is because with clouds, user does not know where the data is stored.

Cloud provides user friendly interface for ease of use, but clusters don’t.

Cloud is scalable, but clusters are not scalable, i.e. new hardware and software resources can be conjured at will by a user on cloud but this can’t be achieved on a cluster.

Cloud have on-demand self-service but clusters don’t. This means that a user can access the cloud computing resources themselves and do whatever they want, without contacting the cloud service provider. Clusters on the other hand don’t have this ability, and the users thus have to contact the cluster provider itself in order to be able to use it and configure it to their needs.

Cloud is inexpensive to use and operate, but clusters are not. This is because cloud is a service on the internet, with vast amounts of virtualization, which does not cost much at all. Whereas, using a cluster, you are having access to physical machines itself connected together on a high-speed LAN. The costs of the LAN, operating the machines, maintaining them etc. make it quite costly.

Cloud has both centralized and distributed resource handling, whereas clusters only have centralized resource handling.

The protocols used in cloud for communication are SOAP, TCP/IP, REST, AJAX. The protocols used on clusters for communication are MPI and Parallel Virtual.

Cloud can exploit data locality to their advantage but clusters can’t. This is because clouds are not specific to a certain location, it is a service over the internet, and hence can reach far off places. Whereas for clusters, they are limited in size and function only in a limited area, which stops them from accessing data stores not close to them.

Clouds have high switching costs, but clusters have low switching costs.

References on next page

REFERENCES:

Question 1:

[1] D. C. Schmidt, S. Vinoski, "Object Interconnections Comparing Alternative Client-side Distributed Programming Techniques", C++ Report, May 1995

[2] D. C. Schmidt, "IPC SAP: An Object-Oriented Interface to Interprocess Communication Services," C++ Report, vol. 4, November/December 1992.

[3] 2017. [Online]. Available: http://www.ecs.umass.edu/ece/andras/courses/ECE397A_S2005/distrib.pdf. [Accessed: 20- Aug- 2017].

Question 2:

[1] A. Gokhale, B. Kumar, A. Sahuguet, "Reinventing the Wheel? CORBA vs Web Services?", In *The Eleventh World Wide Web Conference, Honolulu* (2002)

[2] N.A.B. Gray, "Performance of Java Middleware - Java RMI, JAXRPC, and CORBA", Fifth Australasian Workshop on Software and System Architectures, University of Wollongong (2005)

[3] 2017. [Online]. Available: http://www.ecs.umass.edu/ece/andras/courses/ECE397A_S2005/distrib.pdf. [Accessed: 20- Aug- 2017].

Question 3:

[1] M. Sharma and S. Husain, "Analyzing the Difference of Cluster, Grid, Utility & Cloud Computing", IOSR Journal of Computer Engineering, vol. 19, no. 01, pp. 55-60, 2017.

[2] Naeem, Makhdoom & Mahar, Hidayatullah & Memon, Furqan & Siddique, Muhammad & Chohan, Abid. "Cluster computing vs Cloud computing: A Comparison and an Overview" 1. 28. 5267-5271 (2016)

[3] N. Sadashiv & S.M. Dilip Kumar, "Cluster, grid and cloud computing: A detailed comparison." 477-482. (August 2011).

[4] 2017. [Online]. Available: http://www.differencebetween.com/difference-between-cloud-computing-and-vs-cluster-computing. [Accessed: 20- Aug- 2017].