ABSTRACT

Distributed Object Systems provide an excellent support for evolutionary development of software for distributed applications through software reuse. Request brokers like Common Object Broker Architecture (CORBA) shield the application developers from low-level, tedious and error prone platform details. They provide platform, language, network, hardware, protocol and object location transparency by including an abstraction layer between the application programs and the networking protocols. Java Remote Method Invocation (RMI) is language dependent and Distributed Component Object Model (DCOM) is platform dependent. But CORBA is transparent to programming language and platform.

CORBA uses General Interoperable Protocol (GIOP) as its communication protocol. It fits into the application, presentation and session layers in the Open Systems Interconnection (OSI) model. Presentation conversion deals with the conversion of data from local machine representation to a common network format. This process of marshalling is carried out using static stubs generated by the Interface Definition Language (IDL) compiler or generic dynamic stubs. The stub code generated from the IDL takes care of marshalling the requests transparently to the user. Major limitations of core CORBA include lack of selective inheritance, creation/ destruction of the objects by the client and inefficient marshalling, align and check, buffering and data copying procedures. Also CORBA lacks support for generation of time-efficient and compact stub code. Other inefficiencies in CORBA include

The major objective of this thesis is to study, analyse and propose methods to enhance the performance of stub code. The research work is focused on the following issues:

- Enhancements on CORBA standard:
  1. A method to achieve selective multiple inheritance in CORBA statically.

Selective multiple inheritance cannot be directly implemented in CORBA, because function redefinition is not allowed in derived interface. A method using conditional interfaces to implement selective multiple inheritance in CORBA is proposed. This uses a container class derived from the base interface based on the condition.

2. A method for creation and destruction of CORBA objects using a base interface.

   Deactivation of the CORBA object occurs when the server terminates. This leads to under-utilization of resources. Hence a base interface with operations for reference counting, instantiation and destruction of CORBA objects is implemented.

- Identification of the common sources of overhead in marshalling in CORBA-based applications.
- Improvements in the efficiency of marshalling using the following approaches:
  1. Incorporating changes in the transmission medium.
  2. Introducing new encoding rules to take care of reduction of size of the data passed in the network and the marshalling time.
  3. Improving the security in link level communication of the marshaled data.
  4. Optimised encode/decode software to generate time-efficient and compact encoding/decoding marshalling routines.

Common sources of overhead in marshalling in CORBA-based applications have been identified. This includes inefficient align and check, multiplexing and data copying algorithms which reduces the speed of Internet Interoperable Protocol (IIOP), presence of extra padding bytes in Common Data Representation (CDR) for alignment and lack of time-efficient and compact stub code. Based on this study, changes were proposed in the transmission medium, encoding format and encoding procedures in CORBA.

IIOP is the transmission medium for CORBA based applications. Though it brings about interoperability, it is slow. Hence changes in the transmission medium to improve the performance of the stub code have been proposed. To achieve this, componentized IDL compiler is designed. It has three phases – front end, presentation generator and back end. The back end of the compiler generates stubs and skeletons for a particular transmission mechanism and message format. Conventional CORBA stubs use TCP/IP sockets over IIOP. Since it is slow, modules are incorporated in the back end to produce stubs with faster IPC mechanisms like shared memory, multiple shared memory segments, threading multiple shared memory segments, sockets over TCP/IP and multiple sockets.

The componentized design of the IDL compiler facilitates the usage of multiple IDLs, back ends and target languages. Using this model, interoperability between CORBA and RMI using RMI/IIOP is achieved. A static bridge between CORBA and COM has been implemented using this model.

Encoding mechanisms also influence the efficiency of presentation code. CORBA uses Common Data Representation (CDR) mechanism to encode data. Since CDR uses excessive alignment at the word boundaries, new encoding rules to generate efficient stub code have been proposed. This includes the following:

1) Representation of boolean data in bit format in an array.
2) Removing the alignment at the natural word boundaries to eliminate the extra padding bytes.
3) Correct allocation of the send and receive buffer space depending upon the size of the data being transmitted.
4) Reordering of parameters in the descending order of size to reduce the number of padding bits in CDR.
5) A novel link level encryption scheme to bring about security in CORBA based applications has been proposed. Encryption is performed by choosing a set of functions from a domain of functions based on

validation. The validation process is done by mapping every bit of the plain text block to a base string for a match.

Faster and compact encoding procedures by optimizing the implementation of the encode/decode software have been proposed. The techniques used to implement presentation conversion routines can be interpreted and procedure driven. Interpreted routines result in smaller code, while procedure-driven routines execute faster. So the later can be used for more frequently occurring types. An optimizer implemented in the presentation generation stage of the IDL compiler generates a hybrid stub code. It achieves a balance between code size and execution speed of the stub code. The optimizer uses single and multiple objective genetic algorithms. The suitability of Multiple Objective Cross-over Mutation, Self Reproduction Mutation (MOCM-SRM) and Genetic Local Search (GLS) methods have been studied. Parallel implementation of MOCM-SRM method has been identified to be suitable. The performance of the hybrid stub code generated using this approach has been investigated. Further, CORBA treats DII and SII as two different invocation mechanisms. SII stubs like compiled, interpreted and inlined compilation differ in their code size and execution speed. So the proposed method is extended to generate a hybridized stub code to strike a balance between these four invocation mechanisms.

Since presentation conversion is an expensive part of network communication, the following methods are implemented to improve the efficiency of presentation layer conversion:

1. Efficient IPC mechanisms like shared memory, multiple segments of shared memory, threaded shared memory segments and multiple sockets were implemented
2. Efficient hybrid stub code was generated by optimizing the encoder/decoder software considering static and dynamic frequency of the data types.
3. Changes were implemented in the encoding mechanism of CORBA. The following approaches were used to improve the efficiency of encoding in CORBA:

- Representation of the boolean arrays in bit format.
- Removal of the extra padding bits in CDR.
- Reordering the parameters in the descending order of their size to minimize the number of padding bytes.
- Implementing secure communication between CORBA objects at the link level using a novel symmetric key encryption method.