A Distributed Parallel Processing System for Command and Control Imagery

Dr. Scott E. Spetka[1][2], Dr. George O. Ramseyer[3], Dennis Fitzgerald[1] and Dr. Richard E. Linderman[3]

[1] ITT Industries Advanced Engineering and Sciences
775 Daedalian Drive
Rome NY 13441-4909

[2] SUNY Institute of Technology
P.O. Box 3050
Utica NY 13501-3050
scott@cs.sunyit.edu

[3]Air Force Research Laboratory
Information Directorate
26 Electronics Parkway
Rome NY 13441-4514
Phone: (315)330-3492
George.Ramseyer@rl.af.mil

Abstract

Many modern command and control applications are driven by imagery, and imagery, in general, may originate from diverse geographically distributed sources. A rapid increase in image sizes and processing requirements has resulted from improved hardware for collection and display, and techniques for parallel processing are keeping pace with requirements by harnessing powerful parallel processors to facilitate the analysis of images. In addition, tools for distributed processing are under development that can be used to coordinate the processing effort across a network of parallel processing systems. This paper describes a system that can be used to request image analysis over distributed parallel processing facilities and to integrate the results into command and control systems.

Introduction

Effective Command and Control begins with battlespace monitoring. Sensor data collected from relevant aspects of the battlefield can be transformed into databases of imagery products, and tools for data collection and processing for all types of images are under development. Many of these imagery systems are unique, and the integration of those systems with other systems has in many cases been challenging.

Photographic imagery can be used to identify and extract characterizing attributes. Densities of trees or availability of water can be gleaned from photographs. Visual analysis can fall short in identifying some of the characteristics in a photograph. For example, trees and leaves may be identified, but the type of tree may not be visually identifiable. New technologies allow additional information content to be gleaned from images through automated processes. Image processing algorithms help extend human recognition through the application of pattern recognition. Although algorithms have been development that can process very large images, a system is needed to exploit these technologies. The system must be capable of performing the required imagery analysis and must also be defined with sufficient network capacity to deliver imagery to points of service and provide for collaboration. Due to the intense processing requirements related to very large image algorithms, high-performance parallel computing systems are essential.

The FrameWork Architecture is designed to support algorithms hosted on tightly coupled architectures, such as Sky and Mercury high performance computing systems and modern cluster Linux Beowulf computing systems. In addition to intense processing requirements, high-performance network support will also be needed to transfer very large images to high-performance computing centers where algorithms may be supported.

Sharing Software and HPCs - Open Architecture

Very large image processing algorithms and high-performance computers are integrated into a flexible architecture that provides an opportunity for expansion while offering the performance and flexibility required to meet the needs of Command and Control. A range of algorithms has been developed to serve different purposes, and is available to those with access and who are familiar with their functionality. The architecture allows new algorithms to be installed without disrupting access to existing algorithms. The Architecture depicted in Figure 1 supports sharing software across available high performance computers through a high-speed network.

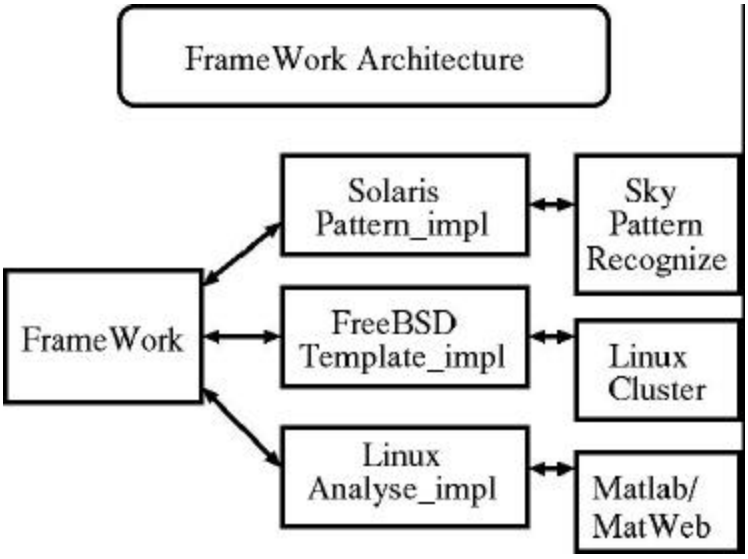

Figure 1 - The FrameWork Architecture

Figure 1 shows very large image processing algorithms executing on HPCs and accessed through host computers using the Solaris and Linux operating systems. Raw images can be processed for pattern recognition through software executing on a Sky computer with the implementation hosted on a Solaris system. The resulting image can be processed on a Linux system hosting a Beowulf cluster with MATLAB (1) installed on the cluster computers. The MATLAB implementation allows querying the characteristics of an image that may be significant, but which are not readily visibly discernible. A "Template" object implementation is used to design an algorithm's interface to the FrameWork client.

Image characteristics recognized as important, based on very large image analysis, must be shared with others to be useful in a decision-making process. This requires the translation of imagery products into reports that can be readily understood by those in Command and Control. Very large image data semantics often cannot be inferred from a visual examination of an image display, and systematic analysis by pattern recognition algorithms is necessary.

The FrameWork is designed to interface to systems for sharing images. The Broadsword (2) interface allows images to be cataloged into a variety of heterogeneous database systems, including the National Imagery and Mapping Agency's (NIMA's) Image Product Library (IPL). Currently under development is the FrameWork interface to the Joint Battlespace Infosphere (JBI) (3). When completed, this FrameWork/JBI system will allow raw images and processed images to be published for possible downloading by interested parties, including those from the Command and Control community. Figure 2 shows how very large imagery data would be exploited with the FrameWork.

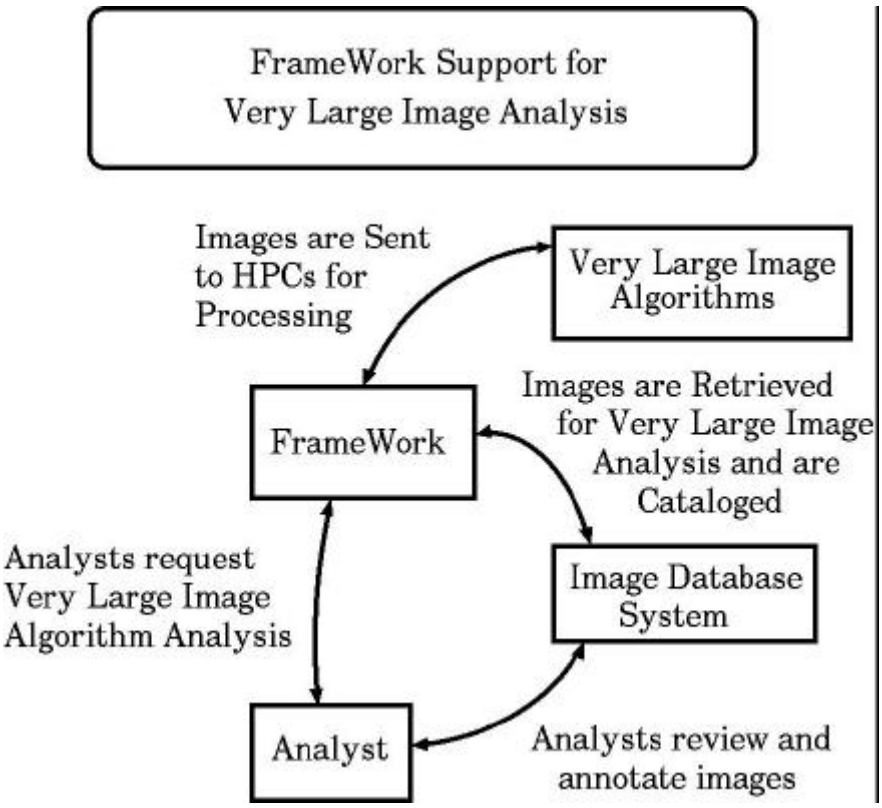

Figure 2 - Using the FrameWork to Share Very Large Images

Imagery data is requested utilizing Broadsword or in the future an information system such as the Joint Battlespace Infosphere. After the imagery data is processed to form an imagery product, the product may be annotated, noting for instance features that are recognized. Then the imagery product may be cataloged into large databases using Broadsword or, when development is complete, re-published through the JBI. If the user is authorized to query other databases, additional imagery products or reports may be requested.

FrameWork Implementation

The FrameWork, an overview of which is presented in Figure 3, is built using technologies that allow for flexibility and reliability. The system is driven by CGI-BIN Web interface software (4) that is installed along with a web server. The Web-based implementation allows access to any authorized user who has a Web browser and an Internet connection. The implementation of new algorithms can be incorporated into the system without disrupting normal system operation. Common Object Request Broker Architecture (CORBA) software (5) is used to allow new algorithms to be added to the system. Each image processing object implementation is accessed by the FrameWork client via the Common Object Services (COS) Naming Server (6). Servers that implement very large image processing algorithms register themselves with the COS Naming Server when they become available.

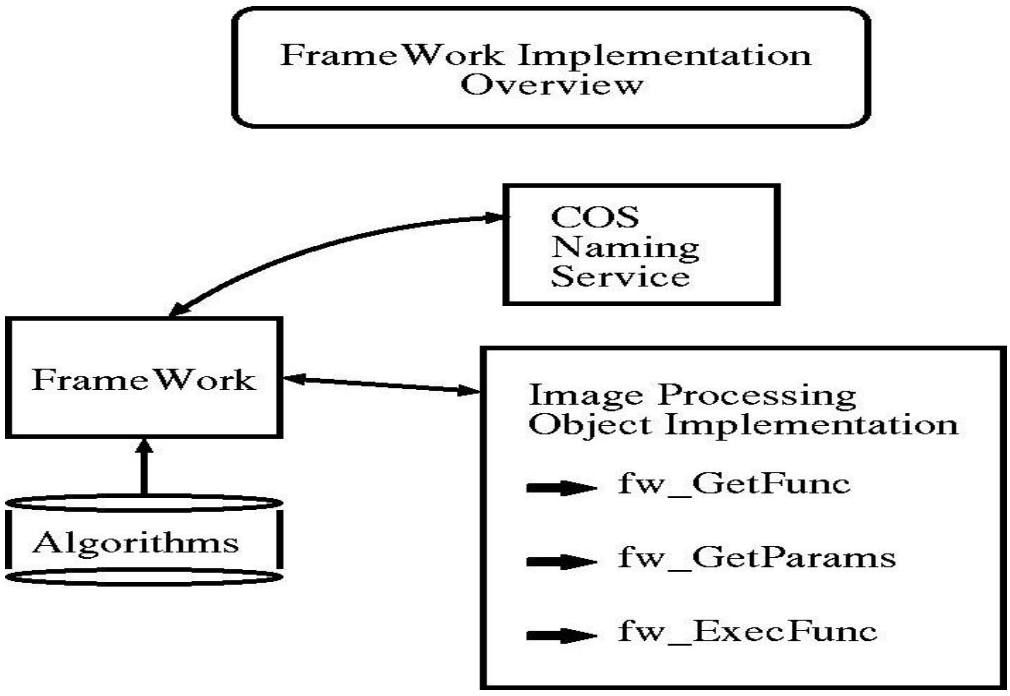

Figure 3 - FrameWork Implementation Overview

The FrameWork uses the COS naming service to connect to the algorithm server implementation. An administrative function is performed for the Framework to configure the names of available algorithms. The names are displayed for the user to select through the Framework client. There are three main functions implemented by each algorithm that is accessible through the FrameWork client.

Figure 4 presents the FrameWork implementation in further detail. Initially, when an object which implements an algorithm becomes available to the FrameWork client, it registers itself with the COS naming service. If the FrameWork client has that algorithm configured for access, it can retrieve an object reference that can be used to invoke the algorithm from the COS naming service. In the initial implementation of the FrameWork, the available algorithms are configured into an object that is capable of displaying them for the Framework client.

The client uses the object reference to retrieve a list of functions from the algorithm object that allows the user to select one of the functions implemented by the algorithm. The Framework then sends a request to the algorithm implementation to determine the parameters that are needed by the function. The response from the object must be formatted to describe the input structures required by the algorithm. Inputs can be files, selected from specified directories (or folders) or input by the user along with a directory location. A list of files in a selected output directory is displayed for the convenience of the users. An algorithm may also request inputs via select boxes, check boxes and radio buttons when limited inputs are allowed. Text inputs where users may type any input they wish are also available to the algorithm designer.

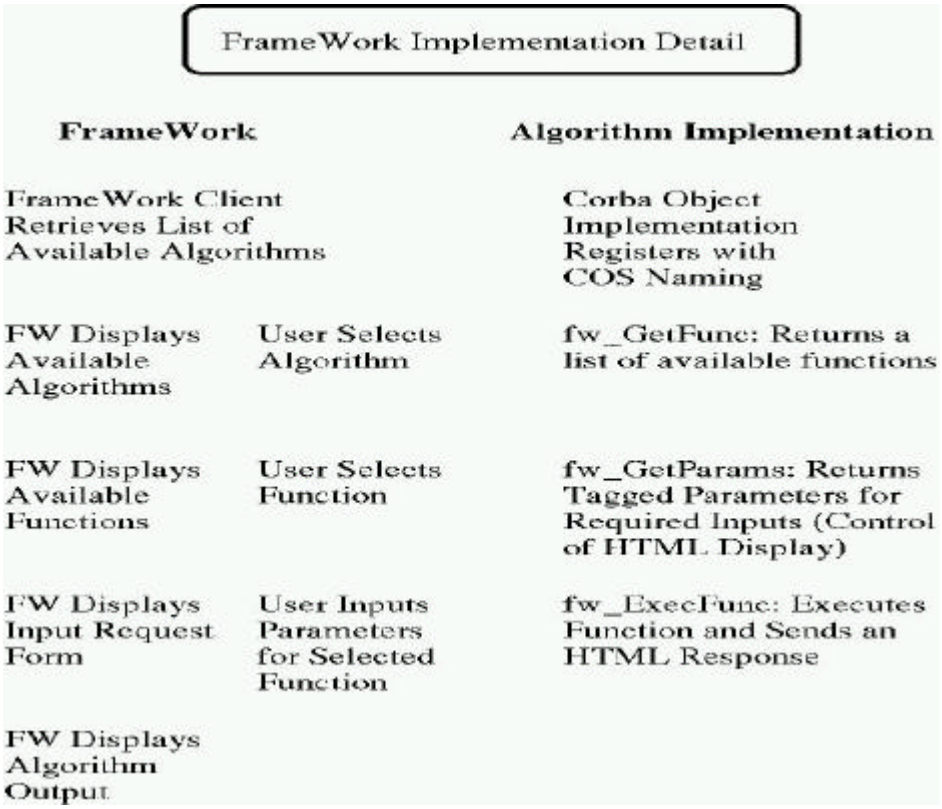

Figure 4 - FrameWork Implementation Details

The parameters are passed to the algorithm for execution after all inputs are specified. The object executes the function and constructs a response string that is displayed by the FrameWork client. The response string may contain the HTML necessary to display a button or hypertext link that can be used to view a resulting image. Output tables are another of the unlimited options available through the FrameWork's open architecture.

Conclusion

This paper describes an open systems architecture that provides a foundation for very large image exploitation. High-speed communications allows the processing of imagery data on remote parallel processing computers. The resultant imagery products are then efficiently disseminated to decision makers. Thus, with an open systems architecture the FrameWork enables processing resources for very large image exploitation to be fully utilized in information-based Command and Control systems.

References

1) http://www.mathworks.com/ accessed 4 April 2002.

2) http://extranet.if.afrl.af.mil/bsword/ accessed 1 April 2002.

3) http://spock.deepthought.rl.af.mil/programs/jbi/ /accessed 1 April 2002.

4) http://www.execpc.com/~keithp/bdlogcgi.htm / accessed 4 April 2002.

5) http://www.omg.org/ / accessed 1 April 2002.

6) http://java.sun.com/products/jdk/1.2/docs/guide/idl/jidlNaming.html /accessed 1 April 2002.

Acknowledgement

This work was sponsored in part by the Department of Defense High Performance Computing Modernization Program's Common High Performance Computing Software Support Initiative (CHSSI). This work was previously presented, in part, at the 2002 Command and Control Research and Technology Symposium.