Project Summary and Context

Daniel Beatty

April 27, 2005

The purpose of this document is to outline the context of the SDSS XGrid project and its specific objectives to be achieved for the Advanced Topics in Numerical Methods II. The SDSS XGrid project is part of a larger context of providing a Scientific Knowledge Base and Computational Service. For this class, there are few parts of the project which are deemed reasonable for demonstration in a Grid context. The choices of these parts include:

- A Wavelet transform in distributed form.
- Bonus items:
  - A template for NSPorts distributed objects with Bonjour
  - A template for distributable objects

Every context of computing, including super-computing, can be described in term of the Turing Machine. The Turing Machine is a model of the simplest computing machine possible. Note that a Turing machine is a five tuple $T = \{K, \Sigma, \delta, s, h\}$ where

- K are the states of the Turing machine
- Σ is the alphabet of the Turing machine
- δ are transition functions on the Turing machine
- s are the starting states
- h are the halting states.

Also, a Turing machine can be viewed as a control unit with its collective of states. The transition functions tell the control unit to move from one set of states to another. A collective of states can be defined as one collective state for simplicity. In the distributed model, grid communications represents the I/O transmitting transition functions of the form

$$(K - H) \times \Sigma \rightarrow K \times (\Sigma \cup \{\leftarrow, \rightarrow\})$$

The nodes in a distributed system (super-computer, cluster, or grid) are control units which contain a collection of states. Further paradigms such as object oriented computing entail that the object house the transition functions and states, and the process or thread house the control unit. Distributed, Grid and Service Oriented computing focus mostly on the control unit. The particulars of these transition functions, states, and control units are the subject of the Scientific Knowledge Representation and Computation Service which could very well be dissertation matter.

1 Using Parallel Turing Machines

There are ways use and exploit benefits in parallelizing computing. Most of the effort focus in on the input and output mechanisms connecting the control units, and the collective control of the control units. Facilities that help in generating such a collective of computing machines include:

- **Library Frameworks:** are a set of objects, protocols and modules which are contained collectively.
- **Interconnection Library (Framework)**: are library frameworks designed connect and migrate other frameworks.

In general, clusters require middle-ware such as MPI, BEEP, NSPorts, XML-RPC or other mechanism to provide the communications for their input and output. MPI, BEEP and NSPorts each form libraries which are part of a library framework. The costs of these middle-wares contribute to the engining of specific Turing machines with an emphasis on performance. During this project, BEEP was explored and discovered to be useful for the overall SDSS Knowledge Base and Computational Service. However, BEEP is not necessarily this project. MPI and NSPorts are comparable methods of distribution and are useful for this project.

1.1 BEEP Overview

BEEP serves as a form of glue for XGrid (zilla and zillion). It is the inter-communication framework. On OSX it is one of three frameworks for XGrid. BEEP itself stands for Block Extensible Exchange Protocol. BEEP consists of the following:

- sessions
- channels
- exchanges
- messages

[2] A session is a peer-wise, full duplex pipe. Two kinds of roles peers play are:

- Listener or Initiator
- Client / Server

BEEP connections can have multi-relations (client, server, or peer).

A channel is a full duplex pipe, and the application protocol designer specifies rules who can initiate message exchange along with a stream of type messages.

Basic features that most implementations of BEEP claim the following:

- Portability
- Object Oriented Design
- Robustness and stability
- High Performance
- Multi-threaded

In essence BEEP is another message passing, and general communication protocol at the . BEEP is a layer that can be used to implement XML-RPC, NSPorts, SOAP, MPI and other service libraries. Like other web type protocols, TLS/SSL security is available. However, the XGrid implementation makes no mention of it except in its API framework.

On OSX, the primary BEEP framework header imports the following headers:

- BEEPSession.h
- BEEPSessionAcceptor.h
- BEEPSessionConnector.h
- BEEPChannel.h
- BEEPMessage.h
- BEEPError.h
- BEEPSSLContext.h

1.1.1 Beep Channels

“The channel number. The zero channel is the private BEEPSession management channel. Even-numbered channels are created by the listener session. Odd-numbered channels are created by the initiator session.” [3]

1.1.2 Beep Session Connector

This object controls session connections, and provides methods initialize connections. It provides a method to cancel the connection. Also provided are delegates for *did connect* and *failed with error*.

1.1.3 Beep Session Acceptor

Probably one of the unique features of the XGrid (OSX) implementation of BEEP is the use of Rendezvous. This is useful for service discovery with zero configuration DNS. The methods to set the values of the Rendezvous service must be set before letting the session acceptor begin. This applies to both the controller and the agents. This may apply to the clients as well only when service is being requested.

> This method asynchronously opens a socket and listens for connections.
>
> When a connection is accepted the delegate is notified via the appropriate delegate method. The accepted session must be opened before it can be used to create new channels and send messages. Note that if the accepted session is not opened soon after the notification is posted the remote peer's session may time out. Also note that the acceptor will continue to run asynchronously and notify the delegate when additional connections are accepted.
>
> If the BEEPSessionAcceptor can not listen on the port specified or another error occurs it will cancel the asynchronous operation and notify the delegate with the appropriate method.

[3]

1.1.4 BEEP Message

This is object is the core of the message is itself, and seems rather simple. For OSX is it is not clear as to whether marshalling is handled by the "archiving" feature of Cocoa (Objective-C) or some other feature.

1.2 NSPorts

NSPorts is a classic method that has been available since the early days of NeXT. It is a remote object mechanism that:

- Provides objects via proxy
- Publishes services via discovery mechanisms such as Bonjour (Rendezvous)
- Provides distributed objects via marshalling and un-marshalling methods (archiving and unarchiving by Cocoa terminology).

An NSConnection object has two instances of NSPort: one receives data and the other sends data. An NSPort is a superclass to all other ports. NSMachPort uses Mach messaging and is typically used solely on the machine itself. NSSocketPorts use socket to go between machines.

There are addition identifier/ modifier types applied to distributed objects: functions, methods and members alike. These key words are as follows:

- oneway void
  (client does not wait for a response.)
- in
  (A receiver is going to read the value but not change it.)
- out
  (A value is changed by the receiver by not read)
- inout
  (receiver is to both read and write the value).
- bycopy
  (argument is archived before sent and de-archived
  in the receiver's process space)
- byref
  (the argument is represented by proxy).

Each connection can have a delegate. Each time the connection spawns a new "child" connection, the "child" will have its delegate outlet set to point to its parent delegate. The connection monitor is a class for logging delegates and their connections.

Another key to distributed objects is Objective-C's "archiving" mechanisms. These mechanisms are analogous to Java's serialization. The object stores its members into a byte array of arbitrary size. This size of this array and the array itself is what is sent over when an object is asked for by proxy. It is an effective means of marshalling and unmarshalling data types within the Cocoa environment (NeXTStep).

2 Basic Science

For this exercise, NSPorts and MPI are favored to demonstrate the rudimentary transition functions and input-output operations of this distributed Turing Machine. Furthermore, simple SDSS image compression and knowledge base work are the problems being solved as applications.

Also for this exercise, the wavelet transform in both ψ^n and multi-resolution expansion are used to demonstrate computation in this distributed environment. The ψ^n is preferred for computation. However, multi-resolution expansion has useful storage, retrieval, and spacial analysis. Details of the wavelet transform are defined in Applications of Wavelets to Image Processing and Matrix Multiplication [1].

For the both transforms, parallelism is found in the row and column transforms. In particular, the transforms of each row are independent of each other. Likewise, the column transforms are independent of each other. The dependencies are that the second transform of the pair depends on the first transform. In other words, if row transforms are performed first, then the column transform depends on the completion of the row transform. In the case of the column transform being performed first, then order of dependency is the reverse of the row transform.

The column and row transforms are called repetitively in the ψ^n wavelet transform, and parts-wise repetitive for multi-resolution expansion. For columns or rows of sufficient number, parallelism becomes practical and simple. The choices is call for transforms on specific rows and columns. One objective of this paper is to determine optimal vector size and distribution for efficient parallel computation.

For this exercise, the ψ^n wavelet transform has been translated from its C++ form to Objective-C. In this form, the code is much cleaner, easier to read, and uncluttered from previous trials on the wavelet transform. Addition methods added include archiving (in the Cocoa sense).

A variant of this object is made to work with MPI. The variety of MPI implementation is immaterial so long as it works with the clustering middleware. In cases of genuine super-computers, the OS provides the clustering mechanisms for processors. For ethernet connected clusters such as Rocks, Sun's Grid Engine, Apple's XGrid (and other Zilla decendents), Condor, and other batch submission middleware the key is for a driving program to be initiated on each worker node. Once initiated, the MPI calls act as transition functions for the Turing Machine Nodes computing the wavelet transform.

A variant of this object is made for a NSPorts Distributed Object (NSPDO). In the case of NSPDO, the clustering mechanism is similar to that of MPI. In the case of super-computers, Mach-ports tend to be mechanism for NSPorts to be implements, and as a result more flexibility is available. In the case of clusters and grids, NSPorts are simply calls over XML-RPC, BEEP, SocketPorts, or some of each. These calls are made by object proxies, which provide a transparency these libraries. The two cases of implementation require separate descriptions for library initialization, but the calls within object itself are the same.

In the case of Mach-Ports, there is no need for Bonjour publishing, since instansiation of these objects will inherently produce either a process or thread on a separate processors. This instansiation naturally provides a links between threads or processors for inter-processor (thread) communication. The reduced overhead is handy for method calls and object migration.

In the case of cluster or grid type environment Mach-Ports are generally out of the question. Instead, NSPorts provides distributed object proxies for these objects and communication costs between instances of these distributed objects becomes more pronounced. One feature that does assist NSPorts in the cluster environment is the use of Bonjour publishing (previously known as Rendezvous publishing). In this case, an initial program is run on each node to publish the ψ^n object. When the user calls for an ψ^n transform, the transform calls the NSPDO version of column and row transforms. This version may call one object for each division of the column/row transform. The returned result is a matrix containing values for the columns/rows transformed and zeros everywhere else. The sum of the results happens to be the complete transform.

The trick in the case of cluster and grid environments is the discovery process of the available objects. Obviously invocation of these objects can not be made if the instances of the objects are not confirmed to exist. Once discovered, the objects proxies can be contained in an array of object proxies and invoked in sequence. The special invocation includes a range of rows or columns to ensure consistent computation.

An MPI implementation and NSPorts implementation have similar mathematics to represent computation conducted in this instance. The MPI calls and NSPDO invocations both serve as transition functions. In the case of temporary objects, both mechanisms incur a cost of instansiating a set of listening "objects". NSPDO has one advantage for long term use. Namely, the listening objects can be left active. As such, the libraries can be invoked by many processes (including processes that had nothing to do with the starting of the listening objects). In the case of residual listening NSPDO, the cost of use includes discovery of the NSPDO and the proxy call. In case of MPI and non-residual NSPDO, the cost includes discovery of the proxy, proxy call, and instansiation of the objects.

XGrid and other cluster middleware are the mechanisms for instansating these objects. Classic means of MPI instanciation is to submit a batch job with a number of programs each containing the calls desired. In the case of NSPDO, the middle-ware can be setup seperately from the initial program or can be called by the program itself.

What would be nice is an XGrid version of NSPorts which used BEEP to broker location and provisions such as ports. Discovery would furthermore be handled by the XGrid server in the immediate vicinity. Whether services in additional realms where visible may be considered if statistically and computationally sound.

In addition to the XGrid version of NSPorts, a fiber channel version of NSPorts or NSPorts over BEEP may be a good choice. In the case of NSPorts over BEEP where flavors of BEEP exist for both ethernet and fiber channel would have a portability advantage. Knowledge of the flavor would then only be useful for heuristic-al information for library optimization.

2.1 Bonjour Example

The Bonjour example shows a practical implementation of the ψ^n wavelet transform with distributed objects. Bonjour provides the discovery process for available objects. Note that all objects are of the same type. They differ only by name to retain node information. Additional information such as dual processor may also be included to determine how many processes/ threads may implemented on each node.

XGrid serves the purpose of activating each library. Again an XGrid version of the NSPort would be useful in abstracting the discovery and setup process. Also, such an API may gain performance by reduced discovery time and performance heuristics.

There are four basic branches for all service libraries:

- The system resource manager launches publishing programs if permanent services do not already exist, or if more are possible and needed.
- The publishing program uses its associated frameworks, and publishes those objects as service objects.
- Migratory objects use marshalling/ unmarshalling methods transform themselves in reference or copy objects. These are either function or return arguments for service objects.
- Programs using service objects require discovery services and or launch services to use these services. The node from which the program itself is running may be a candidate for hosting the service libraries.

Parallel algorithms are passed a list of proxies to the other objects and protocols. One note is that parallel algorithms tend have a recursive nature to them. Thus a point of convergence is generally necessary to determine when the job stops dividing work, and performs it. Note a service that manages the launcher and coordinates the parallel algorithms would be handy.

3 Future Work

An MPI over BEEP library would be nice. BEEP is an another XML communications layer standard with basic layer supports. Some nice features include the ability to publish itself via Rendezvous, and be secured with SSL. As such, BEEP would simply be a medium for MPI to communicate over.

The nice idea for MPI over BEEP would be that once job begins, it can publish its MPI channel. Thus the discovery of any MPI world with its communicator is a simple matter of locating the Rendezvous service.

Some of these demonstration choices inherently require services from some of the others. The services on the FITS objects as mentioned can be and should be SDSS operations, and those operations should be optimized to use the Vector Libraries, and distribute where appropriate. A view requires both services (FITS objects and SDSS Services). The Wavelet transform is more or less independent, but can be made to contribute as a service in the SDSS suite of services.

The easiest of these parts to make happen would be a distributed wavelet transform. The reason for this assessment is the fact that the wavelet transform object itself exists. What separates a distributed wavelet transform versus a serial wavelet transform is the decision of where and when the convolutions occur.

A FITS object could come in handy to demonstrate the use of wavelets on FITS data. The key element of a FITS dataset is the image. The other data are meta-data about the image. Some meta-data are derived. Others are required data about how the data was acquired. For example, the image pixels represents a set of angles, from a set position, with a set conditions on the instrument and the instruments characteristics. Whether or not this is feasible by the end of the course remains to be seen.

References

[1] Daniel D. Beatty, Eric Sinzinger, Alan Sill, Noe Benitez, Phil Smith *Applications of Wavelets to Image Processing and Matrix Multiplication* March 2004 Texas Tech University, http://venus.cs.ttu.edu/wavelet/wavelet.pdf

[2] Marshall T. Rose *BEEP: the Definite Guide* copyright 2002 O'Reilly and Associates, Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472

[3] Apple Computer *XGrid API Documentation and Source Code* copyright Apple Computer.