Basic Concepts for Service Oriented Computing Applied to Scientific Computing Part 1

Daniel Beatty

March 23, 2005

1 Introduction

The term service oriented computing has been used in many papers to be a form of distributed computing. The term grid has also been used in the distributed computing community to describe a distributed operating system, of sorts. What may help is set of perspectives that encompass core components of computer science. The obvious questions are:

- How can distributed systems implement the Turing Machine?
- What features of a classic operating system can be applied to a distributed system?
- What degrees of failure limit the features of the computing system?
- What impact does this have on scientific computing?

Many computing paradigms have developed as computing science has spread to a larger crew participants. The paradigms have included procedural, object-oriented, list oriented and logic oriented type paradigms. Some these paradigms have incorporated how-to concepts which are described as algorithmic. Others have been coined as descriptive for there ability define a problem by a precise enough description.

Aiding these paradigms are concepts of processing these paradigms such single instruction single data, to parallel schemes. Currently multiple data or multiple instructions tend be the exclusive mechanisms of performing parallel computation. Further aiding these paradigms software to manage processing agents.

The big issue for distributed computing currently is what concepts apply in the distributed forms? There are many system types that could be considered to be distributed. What concept is held in common for distributed computing is that each node in the distributed system have independent processing and memory systems. Storage at large, and degrees of processes management may or may not be shared. This description allows for a range of systems from a tightly coupled super computer and clusters to globally separated systems.

The level of separation of these systems contributes communications issues of these systems. These communications issues lead levels of access, and coordination. These levels provide limitations as what degree of parallelism can be applied to these systems both at large and in particular groups.

In the worst case, simple batch system load sharing and basic data sharing is about all that can be expect of these distributed systems. In other cases, a sophisticated sharing of memory, storage, processing, and other resources can produce efficient means of combining computing power. These varying degrees combining computing power demands paradigms to describe how to use this power effectively. Thus if we are to call a grid a collection of computing devices connected by a network to reduce overhead, accelerate productivity and maximize computing power, then we need a paradigm to describe how this can be done. If the how-to is defined, then the descriptive paradigms benefit as well by having means to answer their queries. The paradigm associated with this grid can be as a Service Oriented Paradigm. This service paradigm covers both the middle-ware that services a management scheme to this grid, and how programs are designed implementation of the Turing Machine.

The focus of this series of studies is to determine how to define the varying levels sharing, how to connect these levels in a productive manner, and how this effect scientific computing and the paradigms to make scientific computing effective. This first part of the series describes the concepts in terms of classic process management, data management, memory management, service calls, user services, and application examples.

2 Process Management

Process management is a fundamental part of every modern operating system. Any service oriented paradigm must include management schemes to coordinate these process managers effectively. In some cases, the coordination is limited by the lack of sophistication of the host OS. In other cases, more sophistication can be applied. The sophistication process management can be applied in cases of OS's with sharing micro-kernels, or schedulers which can either altered or interfaced with to share management information.

Even if the OS process manager support shared management, the infrastructure connecting the shared system also provides limitation. These limits are results of communication limits and precision on which the system clocks can be coordinated. In general, designs for distributed system assume that the communications are unreliable and time can not be coordinated. However, this statement fails to account for degrees of reliability and how precise time can be cooridinated. Anywhere on earth, time services such as WWV supplied by National Institute of Standards and Technology. [?] The maximum precision available from NIST is down to 30 ps, but that time is not transmitted at intervals of 55ms. The variance at which such a signal reaches a location versus another defines the precision for which time is reliable for these location. For shared process management, the purpose of coordinated time is for determining order events in these systems. Of course ties are still possible, but the goal is to ensure the same reliability as a single processor system.

Time on a supercomputer is very precisely coordinated. Each processor could be running exclusively of the same clock. A cluster tends to be made of machines with independent clocks. However, if the communications are quick in that cluster, and the latency is small then precision can be made on the order of 10^{-9} to 10^{-6} seconds. As a latency and bandwidth decreases the degree of precision decreases dramatically. In cases where these properties can be calculated or measured, a limit of clock coordinated. For example, many ping counts in university settings are recorded on the order of microseconds.

As stated, the issue of time coordination is a matter of establishing order of events. In the case of a supercomputers, ties occur between processor working in parallel, but those ties are based on the same clock. For a cluster, the order of messages from threads and processes in not as precise. A gray area exists in the precision limit and any message received in that window, must be declared as a tie. A set of time windows can be used to separate groups of messages as occurring at the same time.

Another issue is the generation of processes and the distribution of threads. The purpose of multi-processes is to allow for multiple jobs to run simultaneously. The source of these jobs may be application (the jobs themselves) or spawned copies of those jobs. In other words, some process may be copies of the same program or different program.

The coordination of these processes include a process space which an operating system uses to encapsulate all operations and properties to manage that job. Threads are considered to be sub-jobs which share their space within a process. Their purpose is to allow parallel operation without the overhead associated with multiple process spaces. Threads are limited to system where memory can be considered to be shared.

In some cases, virtual shared memory can be employed. In cases where virtual shared memory is impractical, then migrating a thread from processor to processor means generating a process on the other processor which communicates to a thread on the original processor by proxy.

2.1 Example: GRAM

2.2 Example: Nimrod-G

2.3 Example: Avaki/ Legion

2.4 XGrid

3 Data Management

3.1 Example: GridFTP

3.2 Example: SAM

3.3 Example: AFS

3.4 Example: Avaki Data Grid

3.5 SORCER: Data Management

4 Services: A SOC Operating System Call

4.1 Examples

4.1.1 Darwin and Distributed Objects

OSX is a product of NeXTStep with the Mach micro-kernel. As such it also has NSPorts. One feature that is also present is another service registration scheme called Rendezvous. Rendezvous is Apple's implementation of Zeroconf DNS which allows services to declare the name, type, port, etc.

OSX uses NSPorts to provide distributed objects (DO(s)) and uses the run loop and or thread to achieve a non-blocking solution. Such DO are called via normal message passing routines associated with Objective C and NS Objects. This mechanism provides a sort of proxy for which there are two classes" NSDistributedObject and NSProxy.

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.

4.1.2 Distributed Tasks

Of course, there is nothing wrong with calling distributed tasks either. An example was provided by O'Reilly's articles and written by Drew McCormack May 11, 2004 [?]. This analysis examines the crucial parts.

Apply Filters is the method that calls Distributed Task. There are many nuggets of value in addition to the calls for:

- Add Sub Task with Identifier . This call includes
    1. The identifier
    2. Launch path
    3. Working Directory
    4. Output Directory
    5. Standard Input
    6. Standard Output
- Launch

The methods of how "Photo Industry" provides these values are somewhat interesting.

- The first section of Apply Filters acquires the time.
- Next initiates local instances of the file manager.
- The output directory is fed into Apply Filters and is not interesting.

- The temporary directory segment is interesting.

  1. It uses the processes own information (supplied by NS (OSX) which identifies the process in all of its details. The way this is used to access programs is with in the application itself.
  2. The temporary directory of functions which acquires the temporary directory as specified by the OS. (Any where NS applies).

- The next section claims to produce standard input for the filters which are actually programs and the parameters to those programs. The means for this is the typical array/ dictionary scheme of Objective-C.
- The next segment produces input and temporary directories for the input data (the photos). Features of these production(s) is the production of directories for the sub-tasks. Thus a scheme for dividing the work judiciously is being applied.

The question of the thread oriented submission becomes an issue.

Also, the feeding of data structures becomes an issue for the parent application:

- The manner the sub-tasks are divided up as a list of files (input). Items copied into these directories into these directories are the data (photos) and the programs to work on them.
- These structures include message forwarding which is the purpose of a NeXTStep delegate.
- "A delegate is an object directed to carry out an action by another object." page 456 [**?**]
- Once the sub-tasks and its data are determined, the sub job is copied out of the bundle (app), and the executable (script), then the sub-task queue is loaded.

- The rest of the methods are delegate methods.

4.1.3 Devise xGrid client from Source

One thing to be said about the distributed tasks devised by Drew McCormack is that it is a client of a client. Of course, xGrid has three basic components by design: client, agent and controller. Since xGrid's agent and client are open source, a good clean examination of these components may be in order to devise an xGrid API that makes any application simply a client of the xGrid system. Some of these clients could broker xGrid's services to Federated systems like SORCER, Condor, Globus, and the like.

5 Services: User Libraries

6 Application Example: Sloan Digital Sky Survey

Suffice to say, the Sloan Digital Sky Survey has collected an enormous amount of visual and spectral data of the night sky. The basic scheme for representing the SDSS repository of data is to store its images in a data store of some kind with a meta-data catalog containing information on the images themselves as well as pre-computed data. Having said that, there are some issues to bring this

- Generate an object to represent a FITS file
- Generate an astrotools object to manipulate the FITS file
- Generate objects that represent derived attributes of the FITS file.

- Include optimizations to the FITS object such as wavelet representation.

Having said this the FITS object should include be able to understood by both native and Java code. The astrotools should represent a set of services should be both mobile and efficient.

The Objective-C version of this object most likely will be a wrapper around the C version. The FITS object will most likely have its own reference to the fitsfile, status bit, number of elements, bit pixels, number of axes, size of axes, and null value discriminator. Operations include:

- Open Filename: mode:
- Open Table name: mode:
- Open Data file name: mode:
- Open Image Name: mode:
- Create Filename:
- Close file
- Hidden methods for copying the FITS file to memory
  - Get Num of HDUs:
  - Get HDU Move To HDU
- Image I/O
  - Get Image Type
  - Get Image Dim
  - Get Image Size

    - Get Image Parameters
    - Create Image
    - Write Pix
    - Write Null Pix
    - Read Pix
    - Write Subset
    - Read subset

Note that image types may use:

- Byte size
- Short, int, or long
- Floats or doubles

Tables may also be used in the FITS object. The goal of the table object is to store dictionary type data, meta-data, and may be relational/ knowledge base information.

6.1 SDSS Wavelet Application

This feasibility study uses the problems encountered by the Sloan Digital Sky Survey to show the need of this scheme. How does this scheme derive the use of wavelets? How does this scheme work? In fact, the application of wavelets to the SDSS data releases could be a topic in and of itself. However, there is a common need.

All of the SDSS data can be reduced to a collection of FITS files and the meta-data derived from it. These FITS files can be viewed in terms of objects where the data comes from the files, and the operations contained in the object make it compatible with conventional and wavelet numerical methods. Such an object can act as a proxy, and deliver the information as the transportable object itself.

What is the point of using wavelets in these FITS objects? First a matrix in a wavelet domain tends to be more sparse and easier to compact. The wavelet domain also allows for feature detection, image enhancement, and noise reduction a much simpler operation. Can these operations be handled by these objects? Can these operations be handled concurrently? What message handling is in order? These are questions for the feasibility study.

The first set of tools are for measuring outputs of the images themselves. Each image consists of a sub-image for each image color filter.

The original SDSS used a few pipelines for its image processing. These pipelines included:

- Astrometric Pipeline: which performed astrometric calibration.
- Postage Stamp Pipeline: which characterizes the behavior of the point spread function as a function of time and location in the focal plane.
- Frames pipeline: finds, deblends, and measures the properties of objects
- Final calibration pipeline applies to photometric calibration to the objects
- Monitor telescope pipeline provides calibration data for the psp, and frames pipeline.

These pipelines arranged data into the following categories:

- Image Properties
- Spectroscopic parameters
- Color Images
- FITS images (corrected images)
- Spectra:

The image category includes image parameters, the images themselves, the corrected images, mask frames, atlas image (a listing of which pixels were part of the object), color images.

It is the intended objective of this project to provide a set of libraries that provide the data that was one precomputed, and use the more fundamental data to produce a powerful knowledge base. This knowledge base can therefore use these libraries to discover other facts of the SDSS data.

The original database servers included a catalog archive server, data archive, sky server. The sky server was for outreach. The data archive server provides detailed data such as corrected frames, images, or spectra available. The catalog server provided searches on the magnitudes of the objects based on the five filters.

Each great circle coordinate system was defined for each stripe. The coordinates that a pixel was to be corrected for empirically derived optical distortion terms, and provide corrected row and column coordinates to these pixels. Some of these terms were derived for USNO CCD Astrograph Catalog values of known stars. These mappings result in an affine transformation relating to corrected pixel positions to celestial coordinates.

From a computer vision point of view, the astronomic coordinate system simply represents a standardized spherical coordinate system. Each image can further more mapped if there are sufficient number of points in the image that map to precise coordinates in the spherical coordinate system. Each point in the image represents an angle of the sky by the CCD camera. The general angle is known from the mechanisms aiming the telescope.

7 Auto Linear Fits

This section analyzes the atLinearFits module of the astrotools.

7.1 Auto Vector Liner Equation

Description Solve a set of linear equations:

$$AX = B$$

, where

- A is a matrix of independent coefficients,
- X is a vector of unknown
- B is a vector of dependent variables.

The independent vectors A are passed as a list of vectors.

The primary tool for working this equation is known as dgefs. This function is part of LinPack, LAPACK, and BLAS. It solves an $n \times n$ matrix, A for X given A and condition vector B.

7.2 Auto Vector Linear Fits on Bivariate Correlated Errors and Intrinsic Scatter

BCES (Bivariate Correlated Errors and intrinsic Scatter) is a linear regression algorithm that allows for:

- measurement errors on both variables
- measurement errors that depend on the measurement
- correlated measurement errors
- scatter intrinsic to the relation being fitted, beyond measument error

The routine performs four fits: y regressed on x, x regressed on y, the bisector, and orthogonal errors. Which answer is the "right" one depends on the situation. (A simplified guide would be: if you wish to predict a y given an x, use y regressed on x. If you wish to learn about the relationship of two quantities, use the bisector. See Feigelson and Babu, ApJ 397, 55 1992 for details.)

The algorithm and the base fortran code are from Akritas and Bershady, ApJ 470, ? 1996.

- Also returned are the results of a bootstrap ananlysis.
- The "slopeErr" and "slopeErrBoot" lists have two extra elements on them. These are variences for the bisector and orthogonal slopes calculated using a technique of wider applicability than the usual one (which assumes that the residuals in Y about a line are independant of the value of X; see Isobe, Feigelson, Akritas, and Babu, ApJ 364, 104 1990)
- The covarience vector may be all zeros.

- James Annis, June 14, 1996

8 Auto Sla-LIB Package

This library is for translating structs used in Astrotools to ones that the SLALIB package can understand. Future versions of this probably should use standard LAPACK. Most of these values are common astronomy values and may be useful if only the conversion of these astrotools.

9 AT Survey Geometry

"Converts Equatorial coordinates to Great Circle coordinates" , and vise-versa. Converts equatorial to galactic coordinates, and vise versa. "Converts Equatorial to Survey coordinates," and vise-versa. Converts Great Circle coordinates to Survey coordinates and vise-versa.

> Based on an experimental version of atSurveyToAzelpa, this gets the parallactic angle wrt scanning direction correctly- 5 March 1998
>
> This version converts (LMST,lat) to (LAST,lat), which are zenith coordinates referred to true equator and equinox of date, then applies precession/nutation to convert to zenith coordinates referred to mean eqtr & eqnx of J2000. From that, can convert to GC coordinates, and determine difference between zenith direction and direction of scanning.
>
> This could all be done as well in survey coordinates, but still need to know the node and inclination of GC path. The survey coordinates of a star do not tell you what

great circle is being scanned, which is what is needed to get mu & nu components of refraction.

Survey coordinates are (lambda, eta). Lines of constant lambda are parallels, and lines of constant eta are meridians which go through the survey poles. The center of a great circle scan will be a line of constant eta.

Great circle coordinates (mu, nu) are defined so that the line down the center of a stripe (which is a meridian in survey coordinates) is the parallel nu=0. So, lines of constant mu are meridians and lines of constant nu are parallels. Great circle coordinates are specific to a survey stripe.

To convert to and from Great Circle coordinates, you must input the node and inclination of the reference great circle. For ”normal” drift scan great circles, use ¡code¿node=at_surveyCenterRa - 90¡/code¿degrees, and ¡code¿inc=survey latitude + at_surveyCenterDec¡/code¿.

The survey latitudes for SDSS stripes are ¡code¿ +/- n*at_stripeSeparation ¡/code¿.

The limits on these coordinates are:

- 0 ¡= (ra, glong, mu) ¡ 360.0
- -180.0 ¡= lambda ¡ 180.0
- -90 ¡= (dec, glat, nu, eta) ¡ 90.0

The survey center is defined with the external const double values

- ¡code¿ at_surveyCenterRa = 185.0¡/code¿
- ¡code¿ at_surveyCenterDec = 32.5¡/code¿

This (ra,dec) transforms to:

- ¡bf¿galactic¡/bf¿ gLong=172.79876542 gLat=81.32406952

- ¡bf¿great circle¡/bf¿(with node=95.0, inclination=32.5) mu=185.0 nu=0.0
- ¡bf¿survey¡/bf¿ lambda=0.0 eta=0.0

10 AT Conversions

More conversion libraries.

11 AT Air Mass

C routines for calculating air mass for equatorial, meridian, and zenith positions.

12 AT Galaxies, AT Objects

Calculates number of objects either in our galaxy or with in a specified range.

Description

13 Application Example: Grass GIS 6 (Service Oriented GIS)

References

[1] Jason Denton and Dan Beatty *Lecture notes from Operating Systems* 2003-2004 Texas Tech University

[2] Simple - XGrid project

[3]

[4]

[5]

[6]

[7] Michael Lombardi *Computer Time Synchronization* http://www.boulder.nist.gov/timefreq/service/time-computer.html