CASE STUDIES IN HIGH THROUGHPUT COMPUTING (HTC) USING CONDOR FOR PHYSICS-BASED REMOTE SENSING ALGORITHM DEVELOPMENT AND APPLICATION

Rolando Raqueno and Robert Krzaczek
Rochester Institute of Technology

ABSTRACT
The use of physics-based numerical models has been an emerging paradigm in the analysis of remote sensing data and the development of exploitation algorithms. Understanding the key parameters that affect the complex imaging chain to exploit a scene and associated phenomenology is facilitated by these numerical models. As computing power to perform these calculations improve, the application of algorithms become operationally viable. This gain, however, is offset by the need to address complex scenarios through higher fidelity modeling. This involves computing additional model factors at many more levels.

While the iterative process of executing these computations is conceptually simple, the implementation to make them operational requires special methods. Unfortunately, the mechanics and infrastructure to the solution of realizing these large number of calculations are often glossed over in the remote sensing literature. This leaves investigators without the technical details to address these class of problems. We present our experiences in addressing these types of problems through the use of the Condor High Throughput Computing (HTC) system. We present several remote sensing research case studies conducted by the Digital Imaging and Remote Sensing Laboratory at RIT over the past decade and highlight the computational gains realized by these HTC frameworks.

Index Terms— Condor, MODTRAN, HYDROLIGHT, ENVI, IDL, MATLAB, Hyperspectral, LIDAR, High Throughput Computing, Sparse Aperture, Target Detection

1. INTRODUCTION
The formation of remotely sensed imagery is governed by many complex physical processes that influence the propagation of radiance fields, reflected or emitted, from targets and backgrounds to a sensor [1]. The dynamic nature of the atmosphere and these complex interactions in real scene result in phenomenology that can only be described with high fidelity using numerical models. Models for atmospheric propagation [2] and radiative transfer into the natural waters [3], as examples, have been used to estimate numerous instances of the light field for varying physical, environmental, and geometrical parameters. These spectral libraries model the sensor reaching radiances for a selected suite of parameters resulting database often referred to as Look Up Tables (LUTs). The calculated information can be used to solve the inverse problem by matching remote sensing observations to these estimates using an optimization procedure to extract a desired parameter in the form of target reflectance and atmospheric parameters[4], or water constituents [5]. Other applications use these precomputed solution spaces as an initial starting point for a dimensionality reduction process for detecting target spectra from background clutter [6]. Modeling applications also uses these type of computations to generate synthetic spectral scenes emulating the spectral statistics of real imagery to test exploitation algorithm performance in a controlled virtual environment.[7].

In all these cases, numerous computations were needed to produce these databases for input into the different exploitation schemes. This cost of computation is often overlooked as a mundane “number crunching” task without regard to how it is accomplished. The concept that a single fast workstation can address this computational requirement is an unrealistic one and the omission of how to complete these computations hinders other investigators from testing and building upon these techniques as well as assessing the operational viability of the algorithm. We present several remote sensing applications that have been addressed using the Condor High Throughput Computing (HTC) framework. This infrastructure not only helps finds solutions, but arrives at these solutions in a timely manner allowing in-depth analysis of each problem domain that would otherwise not have been possible.

2. CONDOR
A computing infrastructure that utilizes unused compute cycles from common workstations was developed by the University of Wisconsin-Madison. This system called Condor [8] has been instrumental in the development and testing of physics-based algorithms requiring massive computations. Details of the Condor environment are well documented (http://www.cs.wisc.edu/condor/), but we highlight some of the main points to understand the value of the infrastructure.
1. OPPORTUNISTIC SCHEDULING: Only seeks idle workstations

2. DISTRIBUTION OF JOBS ACROSS FLOCKS (Workstation Group): Assignment of jobs to other idle workstations.

3. JOB MIGRATION: Includes transfer of in-progress jobs (preempted due to user activity) from one workstation to another.

4. EXTENSIVE LOGS AND TOOLS FOR MONITORING JOB PROGRESS

Priority of processing can be defined at the system administrator as well as the workstation owner level allowing flexible definitions of priority and scheduling requirements. Additionally, Condor can also be configured to suspend and even migrate its processes to other systems, to avoid interfering with a user’s interactive use of a workstation. Users within a flock (workstation group) would normally hold local accounts within these networks and would submit jobs from these machines in these groups. One of the attractive features of Condor is its ability to migrate jobs across to other flocks where a user does not necessarily have interactive access to the machines. As an example, jobs submitted by a user in one department can be serviced by machines in another department and vice-versa. This “shared” arrangement effectively enhances compute resources using existing equipment that would otherwise be underutilized. In cases of heterogeneous architectures, binary executables can be compiled and configured to make processing architecture transparent to the user.

Another significant advantage of this computing framework is that it can also schedule and distribute both commercial software packages (based on architecture and license limitations) as well as packages compiled from source code. While high-end workstations provide a large contribution to the progress of these computations, the impact of older and lower end systems have been shown to be considerable by virtue of their numbers. Several of these jobs have also been processed using Condor on Symmetric Multiprocessing architecture (SMP) machines. This type of hardware is typically associated with High Performance Compute (HPC) jobs using the Message Passing Interface (MPI) or Parallel Virtual Machine (PVM) schemes for parallel-computation processing. Our experience has shown that both Condor and MPI jobs were seamlessly scheduled under these architectures. In cases where MPI jobs do not utilize the full complement of CPUs of cluster, Condor jobs will execute on the remaining CPUs. This is significant because it makes available a class of systems that is often underutilized because of the significant effort necessary in customizing software packages (where source code is available) for this architecture. We will give a synopsis on each of the remote sensing problems areas and highlight processing points unique to these cases.

The class of problems we will describe are termed “independently parallel” meaning that different computational cases are independent of previous or subsequent calculations. In the case of processing an image, we can process subimages independent of other subimages. In scenarios of a parameter sweep for a numerical model, it is possible to run these cases in a piecewise manner for each subset of parameters. The idea is to isolate these units into subdirectories and distribute these independent computational blocks to separate processing units to be scheduled and distributed by Condor. The use of subdirectories with a common file naming convention under each directory facilitates the creation, debugging, and management of this case “tree”. Depending on the specific problem, the scripting tools to establish this structure can be involved and often requires specific references to input parameters inherent to the numerical model in use.

3. MODTRAN CASE STUDY

The initial uses of the Condor was to produce atmospheric databases through a parameter sweep of key inputs into the MODTRAN atmospheric propagation code. These databases were initially used as part of an atmospheric compensation algorithm optimizing for atmospheric visibility, water vapor content, and target surface elevation served in a model-predicted vs. real sensor reaching radiance comparison. These selected atmospheric inputs were used as the basis for converting a sensor reaching radiance scene to surface spectral reflectances. The same type of database has been used as part of a target detection scheme where a target spectrum is propagated through a range of atmospheric parameters. This results in a set of estimates of how a sensor would image this target through various atmospheric conditions which generates a subspace on which target detection schemes can be applied. Because it is possible to finely sample the solution space of numerical models such as MODTRAN, statistical descriptions of these atmospheric conditions can be devised as a more compact description of an entire LUT [9].

This case study represents a model that can be compiled from source code. This allows the final executable to take advantage of Condor specific libraries that allows checkpointing and migration of jobs within a given architecture. This insures that a partially completed process can be restarted on another idle machine when it is evicted from another machine that is currently being used. As an example of the increased throughput statistics, a single workstation run of 2200 MODTRAN DISORT (8-stream) runs in the reflective region (0.4-2.5 microns @ 0.001 micron increment) is estimated at 15 days of run time. Condor, with 200 workstations, completes a comparable run in 2 hours.
4. LIDAR-HYPERSPECTRAL CASE STUDY

A proof of concept for improving the target detection performance on hyperspectral data set using LIDAR data was made possible using the Condor system. In this example, LIDAR data point clouds provide geometric information to target detection techniques for hyperspectral data sets [10]. This additional information provides sensor-target-illumination geometry that influences the radiometry of the sensor reaching radiance. In these cases, an analysis of the LIDAR data point clouds can provide information regarding the occlusion of each LIDAR point by other points in the data cloud. Direct illumination can be determined by tracing a ray to the sun and a diffuse hemispherical illumination can be estimated by the fraction of the sky that is not occluded by other LIDAR points. These direct and diffuse illumination terms refine the radiometric estimates for the corresponding hyperspectral pixel which in turn improves the target detection performance.

The implementation of this algorithm used the IDL programming environment and presented some challenges in running it the Condor system. This programming environment is used heavily in the remote sensing because of its ability to rapidly prototyping algorithms along with extensive tools for data visualization. It is also a cross-platform environment that has a version for all major operating systems and the development language of the ENVI software package, a hyperspectral analysis package. In cases where a common file system is accessible to all the workstations in the Condor flock, the software is typically installed in a central server and the startup scripts and licensing information is readily accessible to these local machines. The main challenge comes when it becomes necessary to run these jobs on machines outside of a flock where the resources do not necessarily have a given software package installed or a user does not have an account on these systems. In this situation, some effort was necessary to take advantage of these resources since the software may not necessarily be installed on these machines. In order to run these packages under Condor, it is necessary to isolate the specific binaries, license files, and associated libraries for a specific architecture and operating system. In addition to the binaries any IDL programs that are part of the resource library needed to be enumerated in a transfer list that is copied to the executing machines. These details are hidden by start-up scripts when invoking IDL interactively, but is necessary to properly package a stand-alone payload that can be sent to the different workstations in the flock. The advantage of this capability is that a significant number of CPUs (limited by the number of licenses) become available to process the jobs.

The Condor runs for the same number LIDAR points has been logged at about 3 hours of runtime for the direct illumination calculation and approximately 35 hours for the diffuse illumination calculation for a flock of approximately 50 workstations. An estimate of using a single workstation to process the entire LIDAR data set for both cases was not made due to memory limitations of typical workstations. A rough extrapolation for a single workstation run time would be 5 days for the direct illumination case and about 50 days for the diffuse illumination case.

5. WATER CONSTITUENT CONCENTRATION RETRIEVAL CASE STUDY

This remote sensing problem combines a physics model-based estimation of atmospheric and hydrologic radiative transfer predictions. It is essentially an atmospheric compensation of hyperspectral imagery over coastal waters that attempts to simultaneously retrieve constituent concentration of chlorophyll, colored-dissolved organic matter, and suspended sediments. It uses spectral predictions from both MODTRAN and HYDROLIGHT to find a weighted spectral match against the sensor reaching radiance of an airborne spectrometer.

The implementation of this algorithm takes advantage of the ENVI software package and its associated file formats and structures to decompose an AVIRIS image into individual image lines for processing by Condor. The methods used to run the ENVI process under Condor are similar to the IDL process. The AVIRIS subimages are then processed against an ENVI spectral library of over 100,000 entries using a downhill simplex optimization method. The estimated single workstation run time was 20 days which was reduced to about 20 hours under Condor system using over 100 machines.

6. SPARSE APERTURE ANALYSIS CASE STUDY

The sparse aperture analysis problem delves into the effects of design and configurations on spectral image quality. These systems use smaller optical elements as a compromise to the traditional imaging system using monolithic optical elements. This is attractive for space-based systems where the traditional design becomes prohibitive in terms of cost and launch weight.

The modeling analysis under study for these imaging systems configurations quantified spectral effects of the various systematic errors inherent to these type of designs. Each of the optical elements in the system will have numerous combinations of geometrical errors that will manifest itself as deviations in image quality compared to the theoretical image quality derived from the monolithic systems. The computational modeling involves the simulation of images as it is formed through these systems with different types and levels of optical element errors [11]. From a computational standpoint, this problem presented an interesting mix of packages that combined both IDL and MATLAB tools. The IDL processing was handled in the same manner as the LIDAR case.
study, while the MATLAB processing required the generation of compiled MATLAB code that was targeted at a specific architecture. To simplify the MATLAB process, a flock using a common filesystem was used on an SMP computing cluster. This demonstrated Condor’s seamless integration in a high performance computing environment. The estimate for a single workstation processing time was 15 days while the Condor processing time brought reduced to about 25 hours.

7. SUMMARY AND FUTURE WORK

The utility of the Condor system has consistently proven itself through the speedup of the processes by an order of magnitude. This is a result of the growing number of CPUs that have been brought online to service the many remote sensing jobs that are necessary for physics-based approaches. The ability for Condor to take several problems implemented in a variety of environments and provide answers overnight rather than several weeks is a major enabling technology for algorithm development and remote sensing data analysis. The computations of these numerical models also opens possibilities in investigating the dimensionality and solution subspaces for domains addressed by these models.

This is not to say that process of configuring one’s jobs to execute under Condor is a trivial process. Creative scripting to establish a directory structure to decompose the problem into a distributable form is necessary along with a set of routines to assemble the answers in a context of the larger problem. Our experience has shown that there are practical solutions for both codes that have an open source base as well as commercially available packages such as ENVI/IDL and MATLAB. The latter cases do require additional configurations to target specific architectures and isolate any dependencies to interactive processing, but in most cases the core code that drives the computations require little to no modifications to run under this environment. We have collected links to the details in adapting the different software tools for Condor at http://wiki.cis.rit.edu/DIRS/CondorNotes.

The case studies presented in this paper each had scripts and data structures that were specific to each problem. It is apparent, however, that a generalized framework of specifying parameters, common and unique inputs, and output formats is needed. Plans are underway to produce a Web-based interface to aid investigators in producing the various Condor submit files and monitoring tools necessary to insulate the user from the Condor specific details and focus on the computational problems at hand.

8. REFERENCES