CERSe
A Component-based Environment for Remote Sensing

Nathan A. DeBardeleben, William M. Jones and Walter B. Ligon III

Parallel Architecture Research Laboratory
Clemson University
{ndebard, wjones, walt}@parl.clemson.edu

1 Introduction

There are a growing number of people who want to use remotely sensed data and GIS data. The different applications that they want to run require increasing amounts of spatial, temporal, and spectral resolution. Some users, for instance, are satisfied with a single image a day, while others require many images an hour. This expanding amount of data presents two problems: how to store the data, and how to process the data.

Generally, centers have configurations to receive satellite data like that depicted in Fig. 1. Data comes to the center via either a satellite link or some other high-speed network and is placed into mass storage. Users can then process the data through some form of interface. However, with the ever-increasing volume of data and the complexity of the processing to be done to the data, previous systems are not adequate.

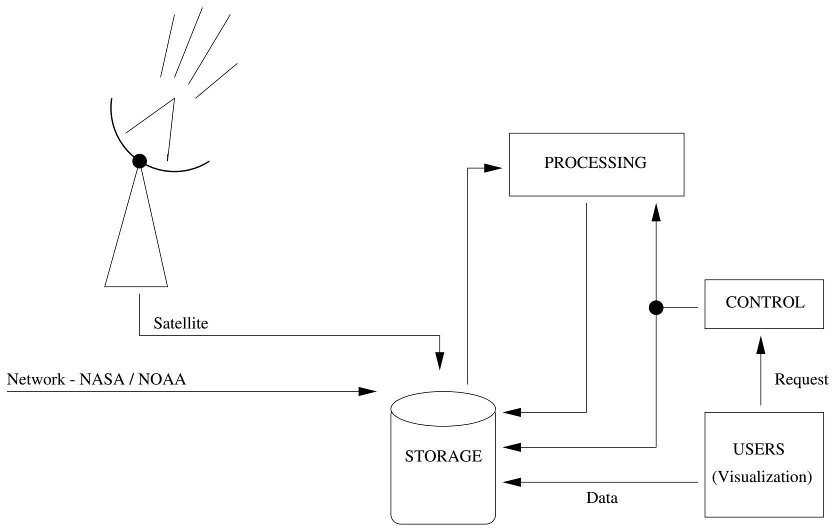

Fig. 1. Satellite Data Processing Center

What is needed is a large scale processing and storage system that provides high bandwidth at low cost. Scalable distributed memory systems and massively parallel processors generally do not fit the latter criterion. A cluster of parallel computers is a viable option due to its impressive cost-to-performance ratio. A cluster also has the potential to provide balanced I/O, unlike many MPPs, since each node can have its own disk. Parallel file systems for clusters can stripe data among the nodes of a cluster so that potential bandwidth is increased and network bottlenecks are minimized [1]. Clusters, however, come at a price. They are generally harder to program than the more traditional computer. An environment for remote sensing and telemetry application development on a cluster would provide a mechanism for the scientist or engineer to utilize high-performance computer systems without requiring extensive programming knowledge.

Those who want to use this satellite data generally fall into two categories: scientific researchers such as biologists, physicists, or meteorologists; and non-scientists such as farmers, business people, and off-shore oil drillers. The first group consists of users who typically have been writing their own code while the second group tend not to be programmers at all. Both groups require a system that caters to their needs and facilitates the running of their applications on supercomputers.

In the past, systems have been designed by or for the parties directly involved in having satellites built and launched. For instance, researchers who determined that they needed a certain type of radar data acquired funding, designed, built, and launched a new satellite. They would also design a specific system for processing the data produced by this new satellite. If, after the fact, others were interested in using the satellite data, that was acceptable but there were only a limited number of processing systems.

As the user base continues to diversify, multi-million dollar investments are no longer feasible and a general system is needed for processing satellite data that is not bound to a specific user group or its goals. NASA's RAC (Regional Application Center) program aims at identifying a set of technologies that provides the required capabilities of the remote sensing user community at minimum cost. An RAC-based remote sensing data center is comprised of a capability for intelligent information management, hyperspectral data acquisition and processing, and ground truth acquisition along with satellite direct readout and parallel computing. The RAC program aims at meeting the growing needs of the remote sensing community by providing not only advanced technologies but also ways to process data efficiently and quickly. This is therefore the context in which we develop this project.

2 Component-based Environment

We present the design of a system for development of high-performance code named the Component-based Environment for Remote Sensing (CERSe). CERSe is designed with a component-based approach where components are highly cohesive and exhibit a low degree of internal dependence among other components. Components operate on data provided to them by the I/O Manager and implement the algorithm supplied by the user via the Controller. Components can relay data back to the I/O Manager for storage or between other components. Fig. 2 depicts this interaction.

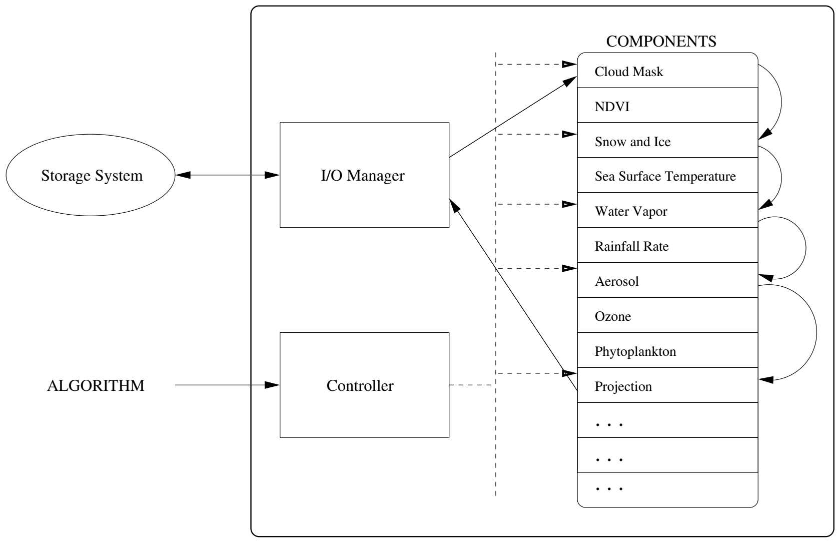

Fig. 2. CERSe System Organization

2.1 Components

A component is a building-block that does not lend itself to further modularization. Each component performs useful operations and has a well-defined set of inputs and outputs with types. The component-based approach lends itself well to parallelism in that it is easy to extract the parallel nature of code on a small scale and then combine the parallel components on a larger scale. The fact that many remote sensing and telemetry application operations are naturally parallel increases the ability to exploit this. Components in CERSe have a well-defined framework and interface which allows easily for design and connectivity of new components. This framework comes directly from previous research into the design of problem solving environments [2]. We provide ways for users to design new components as well as a standard set of components useful to processing satellite data.

Part of this interface is a well-defined set of inputs and outputs from components. These inputs and outputs specify what types of components can plug into each other.

The data typing of inputs and outputs of components lends itself to automatic type checking of a design. In much the same way as high-level languages have types and compilers which check for compatibility between types, CERSe can easily check for type agreement between components and make automatic conversions. Types also allow for *planning*, a term used in artificial intelligence. With planning, an input data type and a desired data type output could be specified and the system could determine the series of components required to satisfy this operation.

2.2 Out of Core Computation

CERSe also takes advantage of out of core computation. With this approach, data can be broken into smaller chunks and operated on in turn. This allows for more efficient use of memory. As previously stated, data volume is increasing rapidly. It is not uncommon for channels from satellite data to take up several tens or hundreds of megabytes of memory. Also, since many important algorithms that process this data use many different channels, a potential memory problem exists. If the datasets use up more memory than the computer has, then the data will be swapped to disk, which is detrimental to performance. CERSe handles this in two ways. The first way is with out of core computation. With OOC datasets can be broken into smaller pieces and each processed individually, thereby preserving memory usage. Use of out of core computation is similar to virtual memory except that with OOC the program instead of the operating system specifies what to load in and out of memory and where to load it. In this way, the I/O Manager can make intelligent decisions about how much data to load and how to distribute it to the components. This also opens up the ability to do prefetching of data that will be needed in the near future. The second way in which CERSe intelligently uses memory is that CERSe is targeted at cluster computer systems where the workload is distributed among the nodes. On a cluster, the more nodes that are available, the smaller the portion of the dataset that is processed – thereby also preserving memory usage.

2.3 Satellite Data Design Language

Components cannot exist on their own. There must be a way to describe the interaction between components. We are developing SDDL - Satellite Data Design Language. With SDDL the scientist or engineer can specify how components hook together. SDDL code can be written by hand, or generated automatically. The automatic generation of SDDL would be the task of a user interface.

3 Comparisons

Khoral Research's Cantata visual programming environment for image processing (depicted in Fig. 3) is a component-based environment with similarities to CERSe. However, Cantata has several major differences from CERSe.

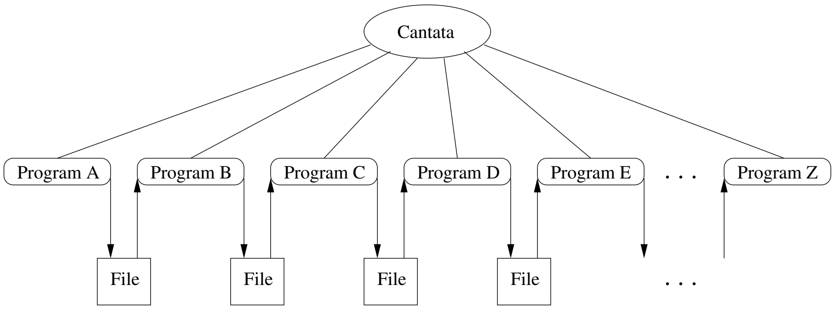

Fig. 3. Cantata System Organization

Cantata is suitable for prototyping since it provides a way to take small operators and wrap them in Cantata code to make a working program. CERSe, on the other hand, is aimed at building production code by providing ways for highly cohesive components to be connected. Cantata components use files for input and output. For instance, a component reads data from a file, operates on that data, writes data to a file and then passes this file on to the next components.

With CERSe, all operations are bound into a single process which has access to the data objects in memory. The I/O Manager is charged with the details of distributing data to the components to process. This allows for memory and cache optimizations since the I/O Manager is conscious of the architecture it runs on.

On a distributed system, writes to the file system are costly. In the instance of there being a single storage system, all the compute nodes would have to send data to that system, which creates a large communication problem. This can be alleviated to some extent by using a distributed file system where each node maintains a portion of the file system locally. In this second system, writes would likely go to the local disk. Regardless, memory is considerably faster than disk and therefore it is advantageous to minimize disk interaction. CERSe handles this by passing data between components in memory instead of by disk. This limited file system interaction is key to the performance of CERSe.

Cantata loads entire datasets into memory before processing begins. As previously stated, this can be extremely detrimental to performance if the data is larger than the available memory. CERSe handles this through out of core computation, thereby not affecting performance in these cases.

4 Conclusion

As the growing volume of satellite data increases with the growing number of users who want to process the data, there is a need to move away from the traditional computer to more powerful supercomputers. The cost, however, of these computers generally places a constraint on the types of users. Clusters of parallel computers provide a good ratio of cost-to-performance and it is within this framework that we design CERSe. CERSe is aimed at making it easy to process large amounts of satellite data quickly. This is accomplished through an environment tailored towards design of parallel component-based software which can easily be connected and the language describing the connections. We provide a standard set of parallel components important to those processing satellite data, as well as a mechanism to produce new components easily with a well defined framework and interface.

References

1. W. B. Ligon III and R. B. Ross, "Implementation and Performance of a Parallel File System for High Performance Distributed Applications" in *Proceedings of the Fifth IEEE Symposium of High Performance Distributed Computing*, 1996, pp. 471-480.
2. K. Hazelwood et al., "Creating Applications in RCADE" in *Proceedings of the IEEE Aerospace Conference*, 1999.