EVE: On-Board Process Planning and Execution

Steve Tanner, M. Alshayeb, E. Criswell, M. Iyer, A. McDowell, M. McEniry, and K. Regner
Information Technology and Systems Center
University of Alabama in Huntsville
S345 Technology Hall
Huntsville, Alabama 35899
stanner@itsc.uah.edu
www.itsc.uah.edu

The Information Technology and Systems Center (ITSC) at The University of Alabama in Huntsville (UAH) has designed and is now developing an innovative processing framework aimed at helping science users exploit the unique constraints and characteristics of the on-board satellite data and information environment. The Environment for On-Board Processing (EVE) system will serve as a proof-of-concept of advanced information systems technology for remote sensing platforms. Because data will be processed as it's collected, such a system will produce custom data products on-board and in real-time. First, the EVE editor allows science users to build processing plans, which are compatible with the constraints of on-orbit computing environments. The EVE on-board, real-time processing infrastructure in turn, will upload, schedule, and control the execution of these plans. Operations within the plans provide capabilities focused on the areas of autonomous data mining, classification and feature extraction. These will contribute to Earth Science research applications, including natural hazard detection and prediction, fusion of multi-sensor measurements, intelligent sensor control, and the generation of customized data products for direct distribution to users. A ground-based testbed is being created to provide testing of EVE and associated Earth Science applications in a heterogeneous embedded hardware and software environment. [*]

1 INTRODUCTION

According to NASA's Earth Science Vision, on-board processing will play a significant role in the next generation of Earth Science missions, providing the opportunity for greater flexibility and versatility in measurements of the Earth's systems. Such on-board processing can contribute to many Earth Science research applications, including natural hazard detection and prediction, intelligent sensor control, and the generation of customized data products for direct distribution to users. Ideally, the availability of custom processing, feature extraction, and data mining on-board satellites can allow end users to specify their own data products through the definition of a processing plan. On-board processing will reduce the volume of delivered data since only the data specified by the processing plan is transmitted to the user. This in turn will improve the accessibility and utility of Earth Science data sets, and overcome, in part, the autonomous nature of satellite sensor data.

The Information Technology and Systems Center (ITSC) at the University of Alabama in Huntsville (UAH) has begun the second year of a three-year research and development effort. The goal of the effort is to build a prototype system which will allow research scientists to build processing plans and execute those plans on on-board processing environments. In this paper, the team presents the current efforts and progress of the project.

This work, to address on-board satellite data processing, is based on the Center's existing custom processing, feature extraction, and data mining technologies. Therefore, the Environment for On-Board Processing (EVE) benefits from the ITSC's experience with scientific data mining and knowledge discovery. Other current ITSC research projects deal with the distributed and heterogeneous nature of Earth Science data sets, as well as data integration, data fusion, and high-performance networking. Based on this background and a broad range of research affiliations, the ITSC has designed and is now developing a new breed of processing system capable of handling the unique constraints and characteristics of the on-board data and information environment. The final EVE system will be adaptable to new Earth Science measurements, and will enable new information products.

While the ITSC has made significant progress in custom processing [1,2,3,4] and data mining technology [5,6], the current architectures were not designed to be optimal for on-board processing in an autonomous environment. In particular, the on-board processing environment will include significant hardware constraints, coupled with requirements for processing real-time streams of sensor measurements. Furthermore, the need for fusion of multi-sensor inputs, both on-board a single craft, and ultimately through communication between crafts, is of great importance. Therefore, EVE requires a complete re-engineering of current well-established software, ranging from fundamental changes in basic system architecture through new implementation of processing modules for even greater efficiency. This approach will reduce the risk, cost, and time associated with development of a full suite of on-board processing functionality. Integral to this effort is the simultaneous development of a ground-based

[*] Financial and technical support for this effort is provided by NASA's Earth Science Technology Office.

testbed, which enables researchers to perform testing and certification in an environment simulating the expected on-board processing environment.

During the latter phases of this research, the ITSC hopes to take advantage of a flight of opportunity, which would allow an actual on-board test of the EVE prototype on an Uninhabited Aerial Vehicle (UAV) experimental flight. These high altitude UAVs fly meteorological instruments for extended time periods (> 18 hrs). By collaborating with NASA scientists who are planning a UAV experiment, the ITSC will define an on-board processing plan that is meaningful in the UAV context. Goals of such a flight may include some autonomous control and flight planning based upon real-time data mining of sensor input.

The ITSC is embarking on a phased approach for this research, advancing the custom processing and data mining software from NASA's Technology Readiness Level (TRL) 2 to TRL 5 over a three-year period. This means that the system will progress from a purely theoretical research endeavor into a demonstrable system.

2 EVE System Requirements

The primary goal of the EVE project is to prototype a processing framework for the on-board satellite environment that includes data mining, classification, and feature extraction capabilities in order to support multi-sensor fusion, intelligent sensor control, and real-time customized data product research applications. These capabilities and applications will be tested and refined as they are exercised in the simulated on-board environment provided by the EVE ground-based testbed. To achieve this goal, the EVE team has defined a set of requirements and documented them in a System Requirements Specification.

These include the following primary items:

- Functional requirements: Support for real-time LINUX; Use of pre-defined and user-defined operations; and Centralized repository of operations and documentation.
- Processing requirements: Execution of multiple processing plans simultaneously; Multiple communication protocols, including streaming data; Support for real-time polling and interrupts; and Initiation of operations in response to detected events.
- User Interface Requirements: Development of a user friendly interface that provides pre-mission plan editing and upload of operations; and System monitoring and control functions vital to mission success.

3 Major EVE Components

To meet the requirements, the EVE system relies on the concepts of Operations, Plans and Carts (Figure 1). Operations are discrete prewritten routines that perform a specific stand-alone function, such as an image processing algorithm, a data mining routine, or other math or sensor based function. These operations can be connected together to form a plan which, when coupled with sensor data, can perform a more abstract computation, such as cloud detection. A plan in turn can be placed into one or more carts, each representing a single real-time computing unit. These carts can then be uploaded to specific on-board systems, scheduled and executed.

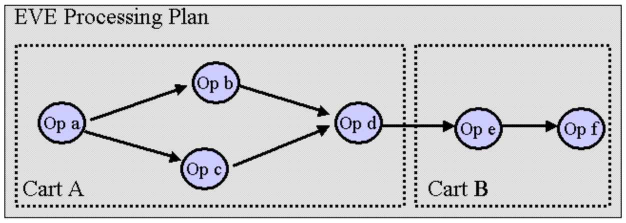

Figure 1: Operations, Carts and Plans

To accomplish the construction, testing and execution of these plans and carts, the EVE system uses three primary components: the editor; the on-board system; and the ground station (Figure 2). The editor is a graphical interface, which can be used to write and validate the plans and carts, and is the user's primary interface with the EVE system. Once a plan is written and placed into carts, it is uploaded to the on-board system, where it is scheduled for execution. The on-board system executes the plan (via the carts) in real-time, based upon both user processing goals and system constraints. The ground station is the primary communication facility between the editor and on-board system. It also maintains a list of available operations and plans in the system as well as the on-board platforms available for use. The EVE system also has system monitoring and control (SMAC) functionality, which monitors all EVE components for malfunctions and also collects performance metrics.

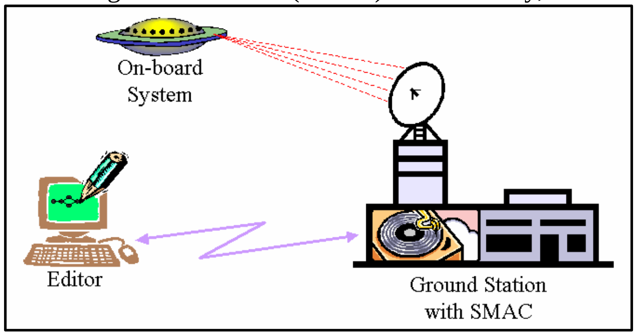

Figure 2: Major EVE Components.

3.1 EVE Editor

The EVE editor is a graphical tool to assist in building plans for upload and execution. To accomplish this, the EVE editor provides the user with a set of tools and operations and an editing workspace in which to manipulate these plans. The user can choose operations from a library of prewritten routines, each represented by an icon. By dragging and dropping these icons from the toolbar to the editing workspace, they can connect them into a workable execution plan. This build up of a plan also requires the user to specify the associated parameters for each operation. The editor will validate the plan and generate a plan file for use by the on-board platforms.

The editor will estimate the resources the plan will consume before it is uploaded. This estimation is based on the operations metadata, which is provided with each operation.

The editor will also display other information such as the actual resources the plan is using and any other information sent from the ground station and the on-board system.

3.2 EVE Ground Station

The ground station serves as the communication relay between the editor components and the on-board systems. Its main purpose is to keep track of what deployment systems are available and to relay messages between these systems and the EVE editor. However, the ground station, via its System Monitoring and Control feature, also provides system management and control functionality. The SMAC functionality includes system resource monitoring as well as the ability to issue control signals to processes executing on the on-board system. The ground station also maintains a library of available operations and their metadata.

3.3 EVE On-board Systems

The EVE on-board components provide a context for real-time execution of plans. The primary responsibilities include managing a plan's lifecycle, and coordinating with the on-board real-time operating system. This coordination includes process scheduling and control, resource allocation, and inter-process communication. In addition, the EVE on-board components must also handle the caching of operations for later execution and collect performance information for later profiling and analysis.

4 Plans and Carts

As stated earlier, the EVE editor allows a user to build a processing plan which specifies a set of operations and the data stream connections between them. These operations have been written in a general nature and are not aimed at a specific on-board platform. The concept of a cart is introduced in order to package such general operations for execution in specific real-time environments.

A cart is a subset of a plan, which holds a sequence of operations that can be executed as a single real-time unit. One cart may be used for an entire plan or a plan may be divided into several carts. Each cart contains a set of operations and provides a set of services to those operations. It holds metadata about the resources required by the operations and resource usage of the platform in which its execution is targeted. The cart also is responsible for tracking the ID tag of the plan and the other carts that comprise that plan.

4.1 Plans

Plans specify a set of operations and the connections between them. A plan specification forms a data flow diagram where the nodes are the operations themselves and the directed edges specify which operation consumes the output of another. Operations may have multiple inputs, multiple outputs, or both. A plan description must contain all of the data necessary to execute, such as:

- Information about the operations involved.
- Parameters for the operations involved.
- Connections for all of the operations' input and output data streams.
- Information about the graphical layout of the plan from the editor, to preserve the appearance when reloaded by a user at a later time.

4.2 Carts

In a real-time system, it is difficult to schedule tasks that depend upon multiple processes with a high degree of intercommunication. One method for dealing with this issue is to bundle all of the atomic operations of an entire plan into a single process image. However, this conflicts with the EVE goal of flexibility and modular plan building. An approach that offers a more flexible architecture is the construct of the cart. In effect, a cart represents several well-defined operations, bundled into a single process image. This bundling offers ease of communication, resource sharing, and task scheduling between the bundled operations within a cart. This model lets task planners worry less about blocking communication and process interruptions and focus more on constructing algorithms from the available operations.

Another interesting benefit of the cart architecture is that it allows splitting up an entire plan into multiple sub-sections, with each sub-section having its own priority and resource usage profile and even its own targeted on-board platform. In this way the scientist designing the process can have extraneous but interesting portions of the process to be done on an auxiliary basis, without losing focus on the primary processing path and goals.

5 On-board System Details

EVE's on-board components are currently based upon RTLinux (http://www.fsmlabs.com/), a hard-real-time variant of the Linux kernel. RTLinux provides a real-time microkernel that treats the Linux kernel (including all the user space processes) as a non-real-time process. This approach allows the use of POSIX real-time facilities without sacrificing the convenience of standard development for components that are not real-time-sensitive.

This approach allows EVE to perform most of its bookkeeping and overhead functions in the non-real-time environment, only using the real-time mode of operation when executing and controlling the cart components.

5.1 *Event Handling*

For some historical perspective, the first implementation of EVE plans and operations consisted of operations that were standalone executables that ran as normal UNIX processes. An operation had zero or more input streams and zero or more output streams. The operations themselves opened the associated file handles for each input and output as the first step of execution. Data was read from the inputs and manipulated to produce output. For each operation, a description file existed describing its "geometry", which consisted of the number of inputs, outputs, and parameters. A plan consisted of a list of operations with their parameter specifications, and described the connections between the input and output streams of the operations involved. The plan executor simply spawned each operation as a separate process and redirected the inputs and outputs for each operation to make the data flow in the correct directions.

In a non-real-time environment, this implementation works quite well. However, in a real-time environment operations must be implemented in a different manner. They need to make use of event handlers, with functions that handle specific events to deal with the real-time constraints and dearth of resources. Those functions must be called by an entity that manages the events. The registration of the event handlers of an operation can be done by code in the initialization of an operation, or can be done by something external to the operation itself. In the case of EVE, this registration is the responsibility of the cart.

To illustrate the difference in the non-real-time and real-time implementation of an operation, consider a very simple pixel-by-pixel threshold operation that produces a 0 if the input is less than 128 and 256 if the value is 128 or larger. The non-real-time version would simply read characters until and end-of stream was reached. For each character it would write the thresholded values onto the output stream. The real-time version would instead handle events rather than input and output. One such event would be "new value arrived". The operation would contain a function that is specifically registered to handle that event, and would be executed at that time. Another event could be "new threshold parameter", allowing the changing of the threshold value being used.

For EVE, the cart provides the event dispatching for the operations. Data streaming can be implemented as dispatching of events with the data being part of the event messages.

It may be necessary for operations themselves to produce events. For example, when an operation produces new output based upon incoming data, it can create an event specifying that output has been produced. The cart would then handle that event and pass the data to the next operation in the "data stream" by calling its "new data available" event handler.

5.2 *Cart Coordination*

The Eve on-board system is designed to facilitate the uploading, execution, and monitoring of the plans, created by the scientist. To provide this functionality, it has been divided in to 2 major components: the Coordinator, and the Conductor.

The Coordinator is responsible for initializing, tracking, and maintaining communication channels for active plans. The Conductor provides event scheduling and notification for the active plans. Other minor components fill necessary support roles such as metrics and logging facilities as well as fail-safe mechanisms.

The Coordinator component maintains a connection with the ground-station and awaits messages. When a new plan is uploaded, a session is created within the Coordinator and expected resources are marshaled. A Session ID is then sent back to the ground-station so that it can communicate with the now-ready plan instance. Once the plan (and its carts) have been instantiated, the Coordinator turns the plan over to the Conductor and awaits further instruction from either the ground-station or the plan itself.

6 Editor

The editor consists of seven major components: file manager, plan handler, plan view/model, plan validation, performance analyzer, display, and communication port.

The file manager allows the user to open, display and save an existing plan, including its graphical layout, the connections between operations, and all associated parameters.

The plan handler is responsible for generating the plan file for use by the ground station and on-board components. Written in an Eve plan syntax language, the file contains information about the plan layout, its operations and associated parameters.

The plan view, a graphical user interface, displays and allows manipulation of the operations and their connections. The plan model is the code representation of these operations and their connections. The plan validation checks for the correctness of the plan, in terms of such things as matching input and output parameters.

The performance analyzer provides an estimate of the resources a plan is expected to require. This static information comes from the metadata with each operation, and includes such things as memory and CPU usage.

The editor will also provide a display for other information, such as estimated resources usage compared to actual resources usage after a plan has been executed.

The last component is the communication port, which sends and receives information from the ground station.

7 Software Engineering on EVE

The EVE research team is guided by a flexible project management plan, a high-level milestone schedule, and a tailored set of well-defined software engineering processes and procedures.

The EVE development team wrote and baselined a System Requirements Specification toward the end of the project's first year. This document formed the basis for the design phase and will be used to develop a test plan that covers multiple phases of the development life cycle, such as testing of individual subsystems, integration testing and final product testing. The EVE Design Document addresses functional decomposition, interface definition, operational timelines, data definition, concurrency considerations, consolidation, and test procedures. It also documents the basic architecture of the system and is used by the team as a development guide. During the ongoing implementation phase, the team is using peer reviews, bi-weekly technical team meetings, and other techniques to ensure adherence to basic Quality Assurance processes and project requirements. The project management team meets weekly to monitor progress and look for signs of schedule slips, cost overruns, and technical risk; and to recommend appropriate risk abatement measures. Configuration Management is being implemented through the use of processes currently in place at ITSC.

8 Collaboration with Other Efforts

An emphasis on collaboration and reuse has been a priority of the EVE project. The use of RT Linux is based in part on work being performed by the ESTO funded Flight Linux effort [7]. In addition, the team expects to benefit from work in on-board scheduling [8] and sensor modeling currently underway [9]. The team is also in the process of incorporating techniques from other application areas [10] and research domains [11].

9 Current and Future Project Plans

The EVE system is currently under development and testing. This effort will continue through most of the second year of the project and includes implementation in the RT Linux environment, and web based editor services. During the third year, the emphasis will shift to use of EVE with a full scale scenarios and a possible “Flight of opportunity”, as well as support for sensor web and grid environments, and incorporation of external scheduling algorithms.

10 References

[1] Beaumont B., Helen Conover, and Sara Graves, 1996. “Information Systems Research at the Global Hydrology and Climate Center”, American Institute of Aeronautics and Astronautics Spaces Programs and Technologies Conference, September 24 - 26, 1996.

[2] Conover H. and Sara J. Graves, 1999. “Promoting Science Data through Innovative Information Systems”, American Geophysical Union, January 1999.

[3] Ramachandran R., H. Conover, S. Graves, K. Keiser, C. Pearson and J. Rushing, 1999. “A Next Generation Information System for Earth Science Data”, The International Symposium on Optical Science, Engineering and Instrumentation, Denver, 1999.

[4] Graves S., 1998. “Automating the Process of Information Extraction in Digital Libraries”, Panel Chair. IEEE Forum on Research and Technology Advances in Digital Libraries, April 22-24, 1998, Santa Barbara, CA.

[5] Keiser, K., J. Rushing, H. Conover and S. Graves, 1999. “Data Mining System Toolkit for Earth Science Data”, Earth Observation and Geo-Spatial Web and Internet Workshop (EOGEO)-1999, Washington, Feb 9-11.

[6] Graves S., 1998. “Creating an Infrastructure for Data Warehousing and Mining”, RCI-NASA Applications of Data Warehousing and Mining. Panel: Strategic Directions of Data Warehousing and Mining, April 20, 1998, Santa Barbara, CA.

[7] Stakem, P., “FlightLinux: A New Option for Spacecraft Embedded Computers”, 2001 Earth Science Technology Conference, Collage Park, MD, August 2001.
[SRS Reference]

[8] Frank, J., A. Jonsson, R. Morris, D. Smith, “Planning and Scheduling for Fleets of Earth Observing Satellites”, Earth Science Technology Conference, Collage Park, MD, August 2001.

[9] Marinucci, T., A. Neelamegam, B. Tjaden, L. Tong, L. Welch, B. Goldman, G. Greer, D. Kaul, B. Pfarr, “Sensor Web Adaptive Resource Manager”, Earth Science Technology Conference, Collage Park, MD, August 2001.

[10] Tanner, S., H. Conover, S. Graves, K. Keiser, A Framework for Sensor Data and Product Processing, Workshop on Multi/Hyperspectral Technology and Applications, Redstone Arsenal, Alabama, February 6-7, 2002

[11] Tanner, S., K. Keiser, H. Conover, D. Hardin, S. Graves, K. Regner, and M. Smith, EVE: An Environment for On-Orbit Data Mining, IJCAI Workshop on Knowledge Discovery from Distributed, Dynamic, Heterogeneous, Autonomous Data and Knowledge Sources, Seattle, Washington, August 4-10, 2001.