JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, February 2004

Migration of an Image Classification Algorithm to an Onboard Computer for Downlink Data Reduction

Patrick Stakem[*]
QSS Group, Inc. Lanham, Maryland 20706

This paper discusses the migration of a data processing algorithm from a traditional ground processing environment to onboard. This is feasible because of the similar operating system environments provided by Linux and Java. Also, the increasing capability of onboard computational systems allows for the hosting of complex algorithms onboard. The advantage of this approach is a reduction in the downlink data set. With increasing onboard systems capability, higher-order data products can be downlinked, as opposed to raw data.

I. Introduction

This report describes the process and results of the implementation of an image classification algorithm on a prototype onboard computer architecture. The application was implemented in Java, and ran on a Java virtual machine under the Linux operating system. The target onboard hardware was the FlatSat, a prototypical onboard computer architecture provided by the Orbiting Missions as Nodes on the Internet (OMNI) Project, at the National Aeronautics and Space Administration's (NASA) Goddard Space Flight Center (GSFC). The implementation modeled a scanning multispectral instrument, connected via an onboard local area Network (LAN) to the computer. The computer ran a classifier algorithm to preprocess and reduce the raw data to selected data sets of interest. This data was then "downlinked" to a ground system via transmission control protocol/internet protocol (TCP/IP) transport mechanism, and displayed. The implementation was accomplished very rapidly due to the use of standard hardware, software, and communication protocol interfaces, with ground-space commonality. The processing throughput agreed with expected results. The potential data reduction achieved ranged from 70% to 90%, which would manifest as a reduction of downlink bandwidth, and/or reduced usage of onboard storage resources.

This demonstration demonstrated the value of using standardized software environments across similar hardware platforms, with standard communication protocols, to leverage ground-space commonality for rapid prototyping and migration.

Remote sensing satellites utilize multispectral scanners to collect information about the Earth's environment.[3] The images formed by such instruments may be seen as a set of images each corresponding to one spectral band. A multispectral image's pixel is represented by a vector of size equal to the number of bands. The combination of the multiple spectrum measurements represented by each element of the pixel vector determine a signature that correspond to a physical element. Through the observation of an image and the comparison of certain pixels to elements from known locations (in-situ measurements), a scientist is able to identify signatures and compose classes. These classes contain representations of multispectral pixels that are closely related. Several neural network schemes have been devised for the automatic classification of multispectral images.[1]

II. The Example Algorithm

The algorithm chosen for this implementation was selected on the basis of a large baseline of experience with its implementation, and the fact that a Linux-based version was available. It is not atypical of a class of image classifiers that may be used to classify data sets for subsequent processing. The Probabilistic Neural Network (PNN) classifier presented good accuracy, very short training time, robustness to weight changes, and negligible retraining time. A description of the derivation of the PNN classifier is given in Refs. 1 and 2.

Received 24 July 2003; accepted for publication 21 October 2003.
[*] Senior Staff Engineer, QSS-MEDS, 7404 Executive Place, Suite 400, pstakem@qssmeds.com.

The same algorithm and data set used in Ref. 2 were used to demonstrate the effectiveness of field programmable gate array (FPGA) based computing. The Black Hills (South Dakota, USA) data set was generated by the Landsat 2 multispectral scanner (MSS). The image's 4 spectral bands (0.5-0.6 um, 0.6-0.7 um, 0.7-0.8 um, and 0.8-1.1 um) correspond to channels 4 through 7 of the Landsat MSS sensor. There are 262,144 pixels in the image, corresponding to a 512 x 512 image size, and each pixel represents 76m x 76m on the ground. The image data set was obtained in 1973. The ground truth was provided by the United States Geological Survey (USGS). Table 1, extracted from Ref. 2, shows the distribution of the image data in the sample data set. The algorithm can classify the data into one of nine categories, as defined by the USGS. For the particular scene viewed, the number of pixels in each class is shown, and what percent of the image this represents. In some cases, there are no pixels of a particular class in this particular image. Then, the data reduction percent is shown as 100% (no data).

Table 1 Distribution of data, Black Hills data set

Class	Number of Pixels	Class name USGS	Percent image reduction
0	6676	Urban	2.5%-97.5%
1	42,432	Agricultural	16.2%-83.8%
2	16,727	Rangeland	6.4%-93.69%
3	198,868	Forested Land	75.8%-24.2%
4	0	Water	0%-100%
5	0	Wetlands	0%-100%
6	1441	Barren	0.5%-99.5%
7	0	Tundra	0%-100%
8		Snow & Ice	0%-100%

Each multispectral pixel, represented by a vector, is compared to a set of pixels belonging to a specific class. An equation is then used to derive the value that indicates the probability that the pixel fits in that class. A probability value is calculated for each class, and the highest value indicates the class with the best fit for the pixel. Based on the classification, a decision can be made to downlink the data subset. The calculations are done on the fly, ideally at the scan rate, which in this case is 32.2 milliseconds.

The algorithm has been implemented on a variety of systems, including various workstations, an FPGA-based accelerator for a workstation, and a Beowulf[*] cluster. Baseline results were enumerated for various PC-based architectures under Linux, both single and multiple processor. The application is embarrassingly parallel; scan lines may be parceled out to different processors, and are independent.

A. The Source Hardware

Originally, the algorithm was implemented on a Silicon Graphics, Inc. (SGI) machine under the SGI Unix variant Irix. We have implemented both a C language and a Java version of the algorithm under the Linux operating system. As expected, the c version runs faster, but not overwhelmingly so. The Java version enhances the client-server nature of the algorithm, which is particularly useful in a multiserver (Beowulf) environment. Given the Java Virtual machine (JVM), the application becomes platform-independent, even more so than the c language implementation. The Java version of the algorithm was developed from the c language version, verified, and implemented on a Beowulf cluster within a 3-month time frame. The Java version under Linux was chosen for the migration.

The algorithm, when run on a Beowulf cluster, scales nearly linearly with the number of processors, up to the communications bandwidth limit. Further results may be found in Ref. 10.

B. The Target Hardware

The FlatSat[4] prototype we used was a flight-like PC-104 bus system, containing a central processing unit (CPU) card with a Cyrix 6x86 GXII-233 MHz processor, 256 megabytes of random access memory (RAM), a serial link to a laptop serving as a console, and an ethernet connection. The Cyrix chip is a Pentium-II equivalent, IA-32 architecture, although with a less capable floating-point unit.

The FlatSat PC-104 model's 256 megabytes of RAM might appear to be a lot for an onboard architecture. However, the recent GSFC Microwave Anisortopy Probe (MAP) mission, using a Mongoose-V processor, has 256

[*]Data available online at http://www.scyld.com (cited 19 February 2004).

megabytes of ram, partitioned as 32 megabytes of processor memory, and 224 megabytes of solid state recorder (disk equivalent).

C. The Target Software

The FlatSat* computer implemented RedHat Linux version 6.2 (kernel 2.2.14-5). We implemented under this Java, specifically JDK Java 2-version 1.3. The application ran under the Java virtual machine. The memory footprint of the application, including the Java Virtual machine, was less than 8 megabytes. The classification algorithm ran at the rate of 4-5 seconds per scan line, which is within the expected range of the CPU of the FlatSat (233 Mhz). This is, of course, not fast enough to keep up with the scan rate of the instrument. However, since the algorithm tends to scale linearly, we can predict the required processing throughput to achieve real time functionality.[6]

D. The Migration

The algorithm was implemented on the FlatSat architecture in one day. This was feasible because the FlatSat architecture matched a previous implementation platform, and was already running Linux. The physical architecture was a remote, desktop computer running Linux, attached via LAN and the internet to the FlatSat in the OMNI Lab. TCP/IP protocols were used to link the systems. The FlatSat was specifically configured to allow remote access from the desktop system located at the QSS offices, about 1 mile from GSFC. The commercial Internet was used for the link. No specific security features beyond the allowed host were implemented at this time.

The virtual configuration of the testbed was as follows. The scanning instrument was simulated by the desktop computer, outputting scanline data from a file to the FlatSat processor via LAN. The FlatSat implementation of the PNN algorithm classified the images, and outputted them to the ground station, via TCP/IP protocols. The ground station role was also played by the desktop computer. In actual case, the onboard computer would have output framed data to the downlink telemetry system, which would have transmitted the data directly to the ground to an appropriate ground station. A router at the ground station would have stripped out the appropriate data from the downlink stream, and put it on a LAN or the internet to the principal investigator's (PI's) display terminal. Note that the roles of the scanning instrument and the PI's computer display terminal could have been implemented as separate machines, although in this case, they were the same.

In this experiment, we inferred the reduction in downlink bandwidth from the classification percentages; in fact, all of the data was sent. It is a simple matter to modify the algorithm to send only the pre-selected data of interest. In an operational scenario, the PI would uplink a data preference (tundra, water, wetlands, etc) to the onboard computer, and only that data would be downlinked. For example, with our test dataset, if the PI selects tundra, he does not get any data via this link, because the scanning instrument is not currently looking at tundra.

III. Discussion

Future Earth observing missions will extract more science information from sensed data sets, re-using or reconfiguring sensor sets to observe different phenomena. In addition, reconfiguration of sensor processing for observations of opportunity, or for science product generation not anticipated by launch time, will increase the overall science yield and efficiency of the mission.[4]

For example, NASA's Plans for Post-2002 Earth Observing Missions (Oct. 1998) discusses soil moisture and ocean salinity observing with a single instrument, as part of the global water cycle discipline. The method involves low-frequency dual-polarization passive microwave radiometry, with a large antenna to achieve a large synthetic aperture, with a spatial resolution of 10km or less. It is known that lower frequencies can penetrate dense vegetation.

When used over ground, the instrument measures wide area soil wetness for macroscale hydrological studies and precipitation anomaly prediction over continental areas. Soil moisture is a component of ground water storage, with impacts on vegetation.

When used over ocean water, the instrument can measure surface salinity to determine the water budget, with a goal of understanding the deep-water formation in the North Atlantic.

With the same sensed data, and different processing algorithms, we maximize the science return from the same sensor data stream.[5]

This technique is applicable to direct downlink scenarios with strict timeliness requirements such as tornado warning, fire detection, tracking of ash cone from volcanic eruptions, oil spill tracking, and general environmental disaster monitoring.

*Data available online at http://ipinspace.gsfc.nasa.gov/flatsat/ (cited 19 February 2004).

IV. Lessons Learned

1. Migration of a complex algorithm from a ground-based to an onboard architecture is facilitated by having the same environment in both cases. This does not require the same hardware configuration, as software can be used to abstract the hardware details. The operating environment, if implemented correctly, eliminates or mitigates implementation details. Using a Linux environment on the source and target machines and a Java (Virtual Machine) implementation allowed the re-hosting of the application discussed here in a matter of days. The use of standard hardware, software, and communications protocols between the ground and space-based computing facilities (ground-space commonality) enables rapid prototyping, and rapid implementation.

2. In a space-to-ground direct broadcast scenario, a reduction in downlink bandwidth requirements would be realized. In a traditional data downlink scenario, the reduction is seen either in the requirements for the amount of Tracking and Data Relay Satellite Service-Single Access (TDRSS-SA) service, or of onboard data storage. We assumed the use of a TCP/IP-based onboard LAN, and a TCP/IP-based uplink/downlink.

References

[1]Chettri, S. R., Cromp, R. F., and Birmingham, M., "Design of Neural Networks for Classification of Remotely Sensed Imagery," *Telematics and Informatics*, Vol. 9, No 3, 1992, pp. 145-156.

[2]Chettri, S. R., and Cromp, R. F., "Probabilistic Neural Network Architecture for High-Speed Classification of Remotely Sensed Imagery," *Telematics and Informatics*, Vol. 10, No 4, 1993, pp. 187-198.

[3]Verbyla, D. L., "Satellite Remote Sensing of Natural Resources," Lewis Publishers, 1995.

[4]Figueiredo, M. A., Stakem, P. H., Hines, T. M, and Flatley, T. P.; "Extending NASA's Data Processing to Spacecraft," *IEEE Computer*, Vol. 32, No. 6, June 1999, pp. 115-118.

[5]Stakem, P., and Figueiredo, M., "Enabling Onboard Data Assessment for Hyperspectral Data with Adaptive Computing Resources," NASA/GSFC whitepaper, July 1998.

[6]Technical Report (internal), "Identifying Algorithm Candidates for Migration Onboard," NASA/GSFC, Mar. 2000.