

A Beowulf-Style Cluster Processing Concept for Large Volume Imaging Spectroscopy Data

Jason BRAZILE, Michael SCHAEPMAN, Daniel SCHLÄPFER, Johannes KAISER, and Klaus ITTEN
Remote Sensing Laboratories, Dept. of Geography, University of Zurich
CH-8057 Zurich, Switzerland

ABSTRACT

The commodity availability of inexpensive computers, large capacity hard drives, and local area network infrastructure has increased the viability of using cluster computing (e.g. Beowulf-style Linux clusters) for even small to medium data processing projects. As a consequence, it is becoming increasingly prominent in numerous earth observing applications [1]. However, there are many factors to be considered affecting how such a processing cluster can be built and used efficiently. Foremost, a fundamental understanding of the interplay between computation and I/O activities of the application is required [2]. It is also important to be familiar with available state-of-the-art software components which can be used for the well-understood portions of application processing (e.g. FFTW [3] for convolutions). And when the application requires novel algorithms, an understanding of modern computer architectural concepts such as superscalar pipelining and cache management can make a significant difference in processing efficiency [4]. In addition to technical requirements, other factors such as the inhomogeneity of developer skills and resources, and deployment and future maintenance/upgrade expectations should be addressed. Here, we address each of these factors and discuss how they were analyzed to model a clustering solution for the calibration and processing of large volume imaging spectroscopy data for the European Space Agency's Airborne Prism Experiment (APEX).

Keywords: Hyperspectral, Beowulf, Multi-Domain Iterative Development

1. INTRODUCTION

An airborne pushbroom imaging spectrometer is in development for the European Space Agency's APEX (Airborne Prism Experiment) project. The intent of the APEX project is to provide a broad range of well-calibrated imaging spectroscopy data suitable for supporting research in many different earth observation applications at a local and regional scale [5], [6].

The project started in 1997 by performing a feasibility study on the design of an imaging spectrometer [7] and

Specified Parameter	Value
Field of View (FOV)	$\pm 14^\circ$
Instantaneous Field of View (IFOV)	0.48 mrad
Flight Altitude	4,000 - 10,000 m.a.s.l.
Spectral channels	VNIR: approx. 140 SWIR: approx. 145
Spectral Range	400 - 2500 nm
Spectral Sampling Interval	400 - 1050 nm: < 5 nm 1050 - 2500 nm: < 10 nm
Spectral Sampling Width	< 1.5 * spectral sampling interval
Scanning Mechanism	Pushbroom
Dynamic Range	12 . . . 16 bit
Positional Knowledge	20% of the ground sampling distance
Attitude Knowledge	20% of IFOV
Navigation system, flight line repeatability	$\pm 5\%$ of FOV

Table 1. Selected APEX Specifications

resulted in a first performance definition and a subsequent design phase. Currently, various parts of APEX are being finalized in design, breadboarding, and performance analysis. The subsequent construction of the instrument is planned to be final in early 2005.

2. THE APEX PROJECT

Technically, APEX is designed to be a dispersive pushbroom imaging spectrometer operating in the solar reflected wavelength range between 400 and 2500 nm. The spectral resolution is designed to be better than 10 nm in the SWIR and 5 nm in the VIS/NIR range of the spectrum. The total FOV is on the order of $\pm 14^\circ$ recording 1000 pixels across track with a swath width of 2.5 - 5 km depending on flight altitude and with a maximum of 300 spectral bands simultaneously [8], [9]. Selected additional specifications are shown in Table 1.

It is expected that individual campaigns will collect data on the order of 100s of GB that need to undergo a chain of data correction processes based on previously acquired and in-flight calibration parameters. It has been determined that even with the large data volumes involved,

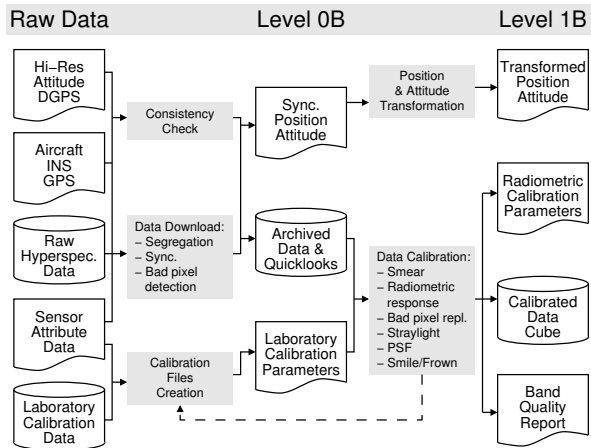


Fig. 1. Simplified overview of APEX level 1 processing

basic level 1 radiometric processing (i.e. conversion of raw data values into SI units) could be performed within the allowed time constraints without the need for a processing cluster. However, higher level application processing, such as correction of at-sensor radiance values to ortho-rectified ground reflectance considering atmospheric and geometric effects, will be bounded above by however much processing power is available.

Proposals for calibration [10], processing and archiving [11] have been designed, presented and accepted [12] and the project has proceeded to the implementation phase.

In the conceptual phase, the data processing chain was well-specified in terms of the calibration and characterization pre-processing steps needed as well as the correction post-processing steps (see Fig. 1 for a simplified view). However these conceptual plans need to be mapped to specific algorithms that perform these steps, software that implements those algorithms and a hardware platform capable of running the software. The software development is constrained by the time given for implementation and the development capacity of a small team of scientists. The hardware platform is constrained by the budget and the expected processing workload.

3. PROCESSING WORKLOAD

When a workload is “embarrassingly parallel”, then it is enough to decompose the problem into independent pieces that can be computed on the nodes asynchronously and completely independent from work being done by the other nodes. [13] An example of this would be frame-by-frame data conversion from one format to another.

However, if the workload is such that sub-problems have dependencies on each other, then there are typically 2 alternatives for solving this depending on the degree of

inter-dependency. If they are highly interdependent, then they are typically decomposed into tasks that run together synchronously on each node, exchanging intermediate data with each other as the algorithm progresses. For example, many atmospheric modeling problems that take this approach.

In the case where subproblems are not highly interdependent, the decomposition of the task might involve organizing a pipeline where processors work (maybe in subgroups) on different stages of the pipeline asynchronously but that the results of one stage of the pipeline are passed on as input to later stages. One class of problems that uses this processing model might involve transforming data from one domain to another using a Fourier transform stage feeding to a filtering stage in that domain before finally feeding back to an inverse transform stage. If the various stages are unbalanced with respect to running time, then it is often possible to vary the number of instances of processes handling the various stages.

In each of these cases, the configuration of the computing cluster can greatly affect the efficiency (or even feasibility) of the solution. For example, if a mostly “embarrassingly parallel” workload is expected, then it might make the most sense to maximize the cluster in terms of CPU power - for example, acquire the fastest processors and settle with the inexpensive standard per-node 10Mbit ethernet.

However, if mostly synchronous highly interdependent workloads are expected, then it might make sense to maximize the cluster in terms of internode communication latency - for example, replace standard 10 Mbit ethernet with something like Myrinet which might give 10 times better latency but maybe at 200 times the cost.

In the last case, one might maximize the cluster for I/O bandwidth - for example, adding faster SCSI disks or gigabit ethernet - assuming that the I/O between the various stages in the pipeline would be the bottleneck.

4. (IN)HOMOGENEITY OF DEVELOPER SKILLS AND RESOURCES

Another factor in the decision of how to map these conceptual processing steps into algorithms and programs is assessing how to maximize the strengths of the development team.

As with “embarrassingly parallel” computing, the best case would be if all software development team members were equally able to implement all tasks independently and asynchronously. Additionally, it would be ideal if all team members were equally knowledgeable in different development paradigms and the most appropriate paradigm could be arbitrarily chosen depending on the problem.

More common, however, is that each team member has

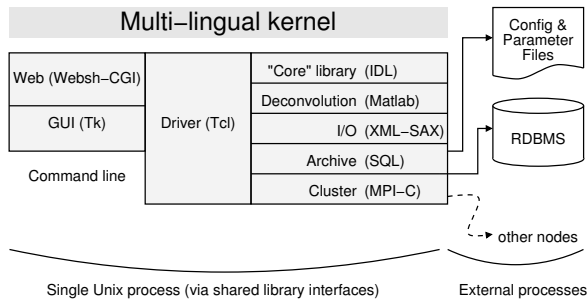


Fig. 2. Breakdown of processing kernel

unique strengths and weaknesses and knowledge of only a limited number of development paradigms. Also common is that the decomposed subproblems have an interdependence on each other that requires scheduling, communication, and synchronization among the developers. One way to mitigate problems in this area and to ensure the coherency of the overall design is to adopt a prototype-based iterative development model [14]. The first iteration consists of simulating program flow using a high level prototyping paradigm and subsequent iterations involve refining the simulated steps by gradually replacing them with more realistic pieces - more realistic first in terms of data size and shape and then in terms of processing resource requirements.

Development efficiency, which is orthogonal to runtime efficiency, can be further improved by allowing continued use of multiple development environments throughout the development process and attempting to arrange automatic inter-operability between these paradigms. An example might be allowing a team member to use Matlab to implement highly precise but perhaps non-performance critical portions of the code, or another team member to use a scripting language for the GUI interface, and then attempting to automatically allow linkage into a complete system. If all the interesting development paradigms in the project could somehow automatically be made to inter-operate, the development flexibility is maximized at all stages of the project.

In the case of the APEX data processor, this is determined to be possible because all the relevant systems have a lowest common denominator C callable interface (see Figure 2).

Although providing an inter-operable runtime environment in this way incurs some development overhead, an offsetting benefit of a highly “multi-lingual” development environment is the ability to enable the composition of domain specific tools to help increase development efficiency [15].

These factors affect the overall design of the cluster because some of the budget may be needed for software li-

censes and/or specific hardware that might need to be dedicated to specific functions (e.g. the GUI front-end, the license server, etc.).

5. DEPLOYMENT, MAINTENANCE, AND UPGRADE INTENTIONS

The final factor discussed in the modeling of the cluster involves the expected future of the cluster. In some cases the cluster will be dedicated to a particular problem and deployed continuously in solving only that problem. However, it is often more common for a cluster to have extra capacity allowing it to be used for additional purposes. In this case, it is worthwhile investigating workload characteristics of other possible problems that might be of interest to determine if the original hardware maximization decision (processing vs. I/O) needs to be updated.

Another important factor in the development of the cluster is planning of cluster support tools. Tools that show the performance of the cluster will be essential if only to help justify its existence. However other practical tools will also be immediately needed such as debugging support, profiling, and job queue management.

When further considering the future of the cluster it is important to plan for hardware failures and upgrades. Some clusters are designed to minimize reliance upon commonly failing components such as per node hard drives [16]. Other clusters merely assume that nodes will fail and plan to take them out of service as they do, perhaps reducing the overall size of the cluster over time.

If the cluster is highly successful, it may justify additional funding for either upgrades or extensions. It is worthwhile considering how this might be done and if the initial acquisition should support growth in this way (e.g. extra capacity in network switching hardware, or not fully populating racks in a rack-mounted system).

6. OUTLOOK

The APEX instrument is currently undergoing ESA’s Preliminary Design Review (PDR) and all aspects of the hardware design are scheduled to be frozen in Q3 2003. The implementation of the APEX offline data processor is currently in the initial prototyping phase as described in section 4. The shared library “glue” for the various development environments shown in Figure 2 have been implemented and/or integrated, a database schema has been designed and implemented, high-level processing modules are being prototyped in IDL, and a few low level algorithms are being developed in Matlab. Regular internal software releases based on the iterative prototyping model are scheduled to start in Q3 2003. Cluster hardware acquisition for the processor is scheduled to start in Q4 2003.

The end of the development and implementation phase is scheduled for Q3 2005.

7. REFERENCES

- [1] D. Komatitsch, J. Ritsema, and J. Tromp, "The spectral-element method, beowulf computing, and global seismology," **Science**, 2002.
- [2] E. Rosti, G. Serazzi, E. Smirni, and M.S. Squillante, "Models of parallel applications with large computation and I/O requirements," **IEEE Transactions on Software Engineering**, vol. 28(3), pp. 286–307, 2002.
- [3] M. Frigo and S. Johnson, "Fftw: An adaptive software architecture for the fft," **Proceedings ICASSP**, vol. 3, pp. 1381–1384, 1988.
- [4] S. Manegold, P. Boncz, and M. Kersten, "Optimizing main-memory join on modern hardware," **IEEE Transactions on Knowledge and Data Engineering**, vol. 14(4), pp. 709–730, 2002.
- [5] M. Schaepman, L. De Vos, and K.I. Itten, "APEX - airborne PRISM experiment: hyperspectral radiometric performance analysis for the simulation of the future ESA land surface processes earth explorer mission," in **SPIE Imaging Spectrometry IV**, S. Schen and M. Descour, Eds., 1998, vol. 3438, pp. 253–262.
- [6] M. Schaepman and K.I. Itten, "APEX-airborne PRISM experiment: An airborne imaging spectrometer serving as a precursor instrument of the future ESA land surface processes and interactions mission," in **ISPRS Commission VII Symposium on Resource and Environmental Monitoring**, Budapest, 1998, vol. 22(7), pp. 31–37.
- [7] K. I. Itten, M. Schaepman, L. De Vos, L. Hermans, D. Schläpfer, and F. Droz, "APEX - airborne prism experiment: A new concept for an airborne imaging spectrometer," in **3rd Intl. Airborne Remote Sensing Conference and Exhibition**. ERIM, 1997, vol. 1, pp. 181–188.
- [8] M. Schaepman, D. Schläpfer, and A. Müller, "Performance requirements for airborne imaging spectrometers," in **SPIE Imaging Spectrometry VII**, 2001, vol. 4480, pp. 23–31.
- [9] D. Schläpfer and M. Schaepman, "Modeling the noise equivalent radiance requirements of imaging spectrometers based on scientific applications," **OSA Journal of Applied Optics**, 2002.
- [10] M.E. Schaepman, D. Schläpfer, and K.I. Itten, "APEX - a new pushbroom imaging spectrometer for imaging spectroscopy applications: Current design and status," in **IGARSS 2000**, Hawaii, 2000, vol. Vol. VII, pp. 828–830.
- [11] M. Schaepman, D. Schläpfer, J. Brazile, and S. Bojinski, "Processing of large-volume airborne imaging spectrometer data: the APEX approach," in **SPIE Imaging Spectrometry VIII**, Bellingham, Washington, USA, 2002, vol. 4816, pp. 72–79.
- [12] M. Schaepman, D. Schläpfer, F. Bucher, and S. Bojinski, "APEX phase B final report," Tech. Rep., Remote Sensing Laboratories, University of Zürich, Switzerland (available from ESA), 2000.
- [13] G. Fox, R. Williams, and P. Messina, **Parallel Computing Works**, Morgan Kaufmann Publishers, 1994.
- [14] B. Boehm, "A spiral model of software development and enhancement," **IEEE Computer**, vol. 21(5), pp. 61–72, 1988.
- [15] A. Lédeczi, A. Bakay, M. Marióti, P. Völgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, "Composing domain-specific design environments," **IEEE Computer**, vol. 34(11), pp. 44–51, 2001.
- [16] R. Minnich, "Pink vital statistics," <http://www.lanl.gov/projects/pink/vitals/index.html>, Visited Jan 2003.