Benchmarking Porting Costs of the SKYNET High Performance Signal Processing Middleware

Michael J. Linnig
Engineering Fellow
Space and Airborne Systems
Raytheon
Dallas, Texas, USA
Linnig@Raytheon.com

Gary R. Suder
Engineering Fellow
Space and Airborne Systems
Raytheon
Dallas, Texas, USA
Gary_R_Suder@Raytheon.com

***Abstract*— This paper gives an overview of the portable high performance signal processing middleware nicknamed SKYNET (with a nod to James Cameron). The paper will discuss the costs of porting SKYNET based radar modes to six different parallel processor architectures.**

The SKYNET Open Middleware has been used in multiple embodiments for more than 10 years on U.S. surveillance radars. Seven organizations, including Raytheon, Northrop Grumman and two Federally Funded Research and Development Centers make use of the SKYNET Middleware for radar mode development. The middleware defines a supercomputer class open architecture for massively parallel computing, allowing hundreds of individual COTS processors to be efficiently netted together to solve the real-time signal processing needs of surveillance radars.

Keywords— Open Architecture; High Performance Computing; Middleware; Radar; Third Party Mode Development; Linux

I. INTRODUCTION

Surveillance Radar applications require significant computing resources to process their sizeable data rates in real time. Surveillance Radars have high data rates because they have a large number of simultaneous receive beams, high bandwidth, and large swath widths.

High bandwidth radars generate prodigious quantities of data. Each receive channel's analog-to-digital converter can produce hundreds of millions of samples per second if high range resolution is needed. If digital beamforming is used, there may be multiple simultaneous receive beams. The total receive data rate may be on the order of tens of gigabytes per second or more and may require many tens of teraflop's of processing power. This is more compute power than is available in most Von Neumann computers. As sketched out in Figure 1, a parallel processing solution is called for.

A. "Embarrassingly Parallel Problem"

Signal processing for radars is often an "Embarrassingly Parallel Problem" that is solvable by decomposing the computations and allocating the processing to a large number of identical processors running in parallel, each working on a subset of the data. These processors need not share memory. In the taxonomy of parallel processing techniques, there is an approach called Single Program Multiple Data (SPMD) [1] programming. In SPMD programming, a copy of a single application executes on multiple processors such that each copy operates on a portion of the data. While working to obtain a solution, such a distributed application may have to share some intermediate results between tasks.

The SKYNET computing infrastructure uses the proven SPMD parallel processing abstraction. The SKYNET Middleware makes SPMD programming easier and it coordinates sharing of data between processors. Programmers develop parallel applications targeted to a rectangular grid of processors, shown in Figure 1. This approach allows efficient algorithm execution spanning many processors. The resulting radar mode software can be largely independent of the number and the kind of processors.

B. Current User Community and History

Third party developers use the SKYNET middleware to create radar modes. There is a robust development community based around SKYNET. Seven organizations, including Raytheon, Northrop Grumman and two Federally Funded Research and Development Centers employ the middleware.

The non-proprietary SKYNET API leverages heavily from the Spectron SPOX Operating System produced by Texas Instruments in 1998. Raytheon chose the SPOX API for

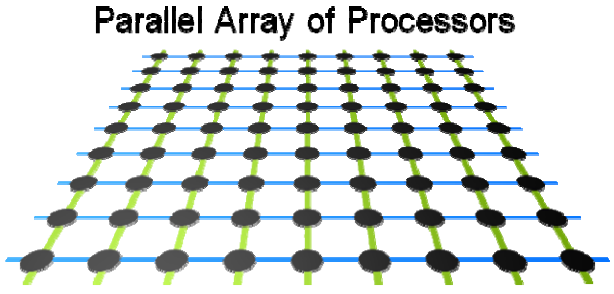

Figure 1: The Surveillance Radar Problem Needs a Parallel Processing Solution

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

SKYNET to minimize the amount of software rework to port existing signal processing implementations (built using SPOX O/S) to new architectures not supported by SPOX.

As is, the SPOX operating system did not provide support for distributed applications on multiple processors. Raytheon developed the Array Operating System (AOS) portion of the SKYNET middleware to add inter-processor communication between applications. The combination of SPOX and AOS APIs form the core SKYNET Middleware API. The original SKYNET radar modes and the middleware ran on an array of hundreds of COTS Digital Signal Processors (DSPs).

In 2003, Northrop Grumman Corporation released a version of the SKYNET middleware that used the Message Passing Interface (MPI) protocol [2] internally to support inter-processor communications.

Since 2003, Raytheon has refined and improved the SKYNET Middleware to keep pace with evolving processor technologies. Raytheon easily ported the existing radar modes to an array of Linux based COTS high performance processors by using the SKYNET Middleware.

C. Open Systems Approach to Signal Processing Middleware

Our approach to signal processing middleware, shown in Figure 4, decomposes the middleware into two components:

(1) SKYNET MetaWare – A thin, non-proprietary, Government owned API that decouples the radar applications from hardware and operating system dependency; and

(2) Open Middleware – An open compute environment, based on Linux and open standards, which can be easily targeted to new hardware platforms.

We decomposed the middleware this way because it provides an open systems core while preserving the high-performance processing needed for Surveillance Radar systems.

The foundation of the SKYNET MetaWare is the Linux based open middleware. This foundation includes open standards such as the Message Passing Interface (MPI) [3], Data Distribution Service (DDS) [4] and the Linux Operating System itself. Linux and these open standards are a powerful toolbox and provide parallel computing building blocks, but for high-performance computing, a well defined parallel computing infrastructure such as SKYNET is needed. The Linux base of SKYNET allows it to be ported to almost any modern processor easily, breaking processor vendor lock.

> "Each component will share a common component interface to the infrastructure through a thin "applications isolation" layer. It is desired to eliminate direct component-to-component communication links. Instead, all component-to-component communications will be mediated by the infrastructure to allow and enable component modularity through abstraction and encapsulation."
>
> **--DoD Radar Open System Architecture Defense Support Team**

Figure 2: Motivation for Isolation Layers

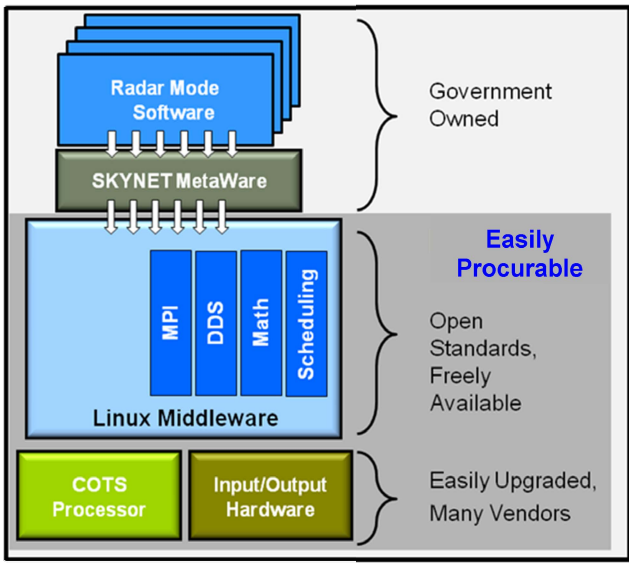

Figure 4: Portable Signal Processing Middleware Approach

The SKYNET MetaWare layer provides isolation from any changes to the operating system and also meets the desires of the DoD Radar Open System Architecture Defense Support Team [1], shown in Figure 2. The MetaWare layer is a thin "applications isolation" layer. The radar modes are not directly dependent on Linux; Future developers can port modes to non-Linux processors by only changing the MetaWare.

The SKYNET Middleware includes both a Sensor Manager

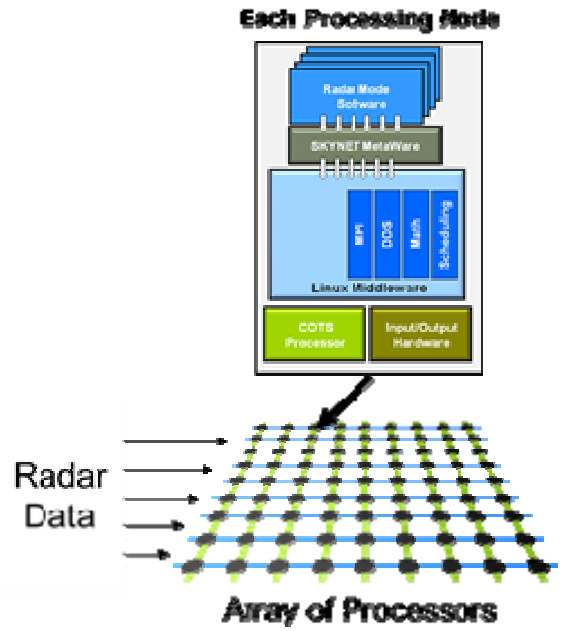

Figure 3: Approach scales up or down to hundreds of processors or cores

(radar scheduler) and a signal-processing component. This paper only discusses the signal-processing component.

II. SKYNET MIDDLEWARE ABSTRACTIONS

The SKYNET computing infrastructure makes use of Single Program, Multiple Data parallel processing. As shown in Figure 3, programmers develop parallel applications targeted to a rectangular grid of processors. This approach allows efficient algorithms spanning hundreds of processors. The SKYNET Middleware is the signal processing infrastructure that allows the radar mode code to be scalable to a variable number of processors or arrays of processors with hundreds of cores. The resulting radar mode software is largely independent of the number and the kind of processors.

A. The Virtual Node Abstraction

To support the development of SPMD applications, the middleware provides an abstraction called a Virtual Node, that is layered upon the real-time OS abstraction and the Message Passing Interface abstraction. A Virtual Node is a task that is defined to have four NEWS neighbors (North, East, West and South, hence NEWS.). As shown in Figure 5, the neighbors are Virtual Node tasks.

The Virtual Nodes logically form a grid superimposed on processors, as shown in Figure 6. The programmer writes the application from a perspective of needing data from one of four possible neighbors, in the North, East, South or West directions (hence the NEWS moniker) as shown in Figure 6. A SKYNET Middleware developer can easily configure the number of virtual nodes per processor to accommodate multiple core processors or for algorithmic convenience.

Multiple processors are arranged to form grids of Virtual Nodes. The middleware internally characterizes transfers between Virtual Nodes as either off-chip or on-chip depending if the neighbor Virtual Nodes resides on the same processor. In the current implementation, Off-chip transfers are handled via MPI while on-chip transfers are simply memory-to-memory copies. The SKYNET AOS middleware library provides the API to establish these large grids and provides communication between the Virtual Nodes. The application developer does not need to know if the neighbor is an off chip or on chip neighbor, the middleware handles the details.

B. The Array OS Abstraction

The Array OS (AOS) Abstraction provides operations to the signal processing application that do operations on a rectangular connection topology across processor boundaries.

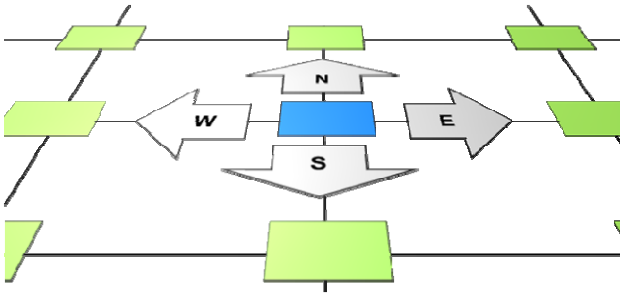

Figure 5: NEWS -- Cartesian Communication Optimized

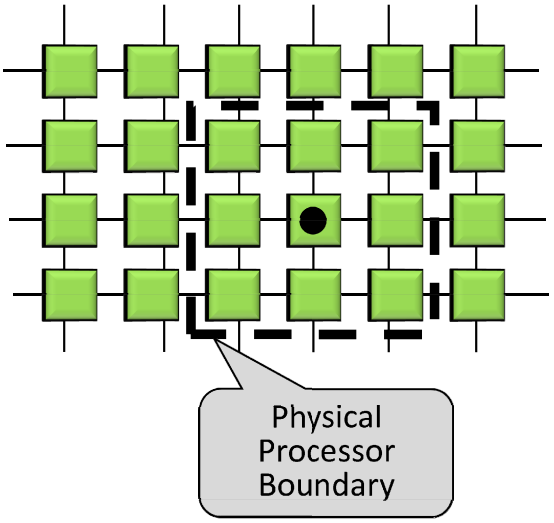

Figure 6: Virtual Nodes are Multi-core Friendly

It provides Barrier Synchronization primitives allowing simple multiple processor operations.

C. North-East-South-West Communications

NEWS communications may be performed when the application needs to share data between neighboring Virtual Nodes. For example, if the data being processed is spatial in nature, then a Virtual Node may need its neighbor's data to perform seamless calculations (i.e. calculations with a seam at the Virtual Node boundaries. This data sharing is performed in multiple steps. First the data is sent east (received from the west) and then it is sent west (received from the east). The complexity of this communication is hidden from the application programmer by the Array OS portion of the middleware.

Very large arrays of data can be processed by distributing the storage across many Virtual Nodes (and their processors) and doing most of the processing locally. NEWS based math libraries execute distributed array operations that can span all processors in the array. The necessary communication for distributed processing operations is provided efficiently by the SKYNET AOS middleware without the application program having to have knowledge of the details.

D. The Operating System Abstraction

The SKYNET Middleware has an Operating System Abstraction API that isolates the application program from the details of the underlying Operating System. Functionality in this API includes task management, semaphores, mailboxes, memory management, queues and periodic alarms and timers. The Operating System Abstraction allows the underlying operating system to be changed without impacting the application. Even a non-POSIX based Operating Systems can be accommodated without changing radar mode source code.

E. Other Abstractions

The Middleware defines all the APIs needed for radar mode development. This includes high performance math

libraries, processor affinity control, stream I/O and instrumentation.

III. Portability and Performance Study

As part of an extensive processor selection trade study, Raytheon ported the SKYNET Middleware to six different hardware architectures, including large arrays of COTS DSPs, COTS Power and Intel processors.

Originally, the SKYNET radar modes and the Middleware were developed for arrays of COTS Digital Signal Processors (DSPs). It was time for a processor refresh. The trade study was designed to find candidate processor architectures for the next generation system. Two of the key discriminators for this study were (1) radar mode portability cost since portability drives the cost of future processor technology refreshes and (2) processor efficiency since efficiency drives Size, Weight, and Power (SWaP).

During the study, our engineers ported two fielded radar applications onto homogenous arrays of processors (arrays of identical processors). The study evaluated the costs of porting the radar mode to each architecture, noting what fraction of the mode software had to change to work on the processor being examined.

The portability percentages shown in Figure 7 are the fraction of radar application source lines of code (SLOC) changed to port to the new processor (definition from the High Performance Embedded Computing Software Initiative).

$$Metric = \frac{\Delta\ SLOC}{Baseline\ \ SLOC}$$

As shown in Figure 7, we found that that for most architectures, little or no changes were needed to the radar mode software. For many architectures, only the middleware had to change to accommodate a new processor. A notable exception is the Mercury Cell Processor. The very constrained limitations of that processor required significant changes to the radar mode software to allow it to run on the Cell Processor. Note (*) that the team was not able to run Radar App 2 on the TigerSharc processor array.

Raytheon engineers developed performance-monitoring tools that allowed the team to tune the middleware to achieve optimal throughput from each processor being evaluated. We found that the middleware is an important contributor to ultimate radar mode performance. A well designed middleware does not require signal processing application changes when radar modes are migrated to a new processor.

Middleware Allowed Radar Modes to be Ported to Most Processor Architectures with Few Changes

Vendor	Platform	Support Libraries	Radar App1	Radar App 2
Bittware	TigerSharc	1%	7%	*
Sky	PPC	0%	0%	0%
Mercury	PPC	0%	0%	0%
CSPI	PPC	0%	0%	0%
Dell	Xeon	0%	0%	0%
Mercury	Cell	19%	46%	24%
DRS	Octopus	0%	0%	0%
IBM	POWER Series	0%	0%	0%
Generic	Optimizations	15%	6%	8%

Figure 7: Radar Application Percent Change Needed in Port

The team also measured the efficiency of the processors and discovered that the peak throughput quoted by the vendor were often unachievable in practice. Efficiency here is defined as ratio of Total Algorithm GFLOPS divided by Peak GFLOPS (sometimes called marketing FLOPS).

The study found that the measured processor efficiency was dramatically uneven from family to family. We are reluctant to quote exact efficiency numbers as it is application dependent and will vary as technology changes. Note that, for our applications, many processors delivered well below 5 percent of the advertised marketing FLOPs. We believe that processor efficiency should be evaluated when choosing a processor; we found some of the more powerful processors in theory had poor efficiencies in practice.

IV. SUMMARY

We have described the SKYNET Middleware and its long history in high performance real-time signal processing. The SKYNET Middleware is in use in a robust community of developers producing signal processing applications. Because the interfaces in the SKYNET Middleware are U.S. Government owned, applications built using the middleware are free of vendor lock. There is a thriving third party development culture in place.

We have described how the SKYNET Middleware is based on an Open Architecture approach using Linux. We have shown how having a thin middleware layer over Linux provides a very portable solution. That portability was demonstrated when fielded radar applications were ported to many different processor architectures with little or no change.

V. REFERENCES

[1] DoD's Perspective on Radar Open Architectures, June 2010, Scott Lucero1 et al.

[2] Algorithms and Theory of Computation Handbook, 1999 by CRC Press LLC.

[3] Using MPI, 2nd Edition: Portable Parallel Programming with the Message Passing Interface. Cambridge, MA, USA: MIT Press Scientific And Engineering Computation Series, 1999, By Gropp, William; Lusk, Ewing; Skjellum, Anthony

[4] Data Distribution Service for Real-time Systems, Version 1.2, 2007, Object Management Group