Persistent Surveillance Supercomputing in a Can

Jeremy Kepner, William Arcand, Chansup Byun, Bill Bergeron, Matthew Hubbell,
Andrew McCabe, Peter Michaleas, Julie Mullen & Albert Reuther
{kepner,warcand,cbyun,bbergeron,mhubbell,amccabe,pmichaleas,jsm,reuther}@ll.mit.edu
MIT Lincoln Laboratory, Lexington, MA 02420

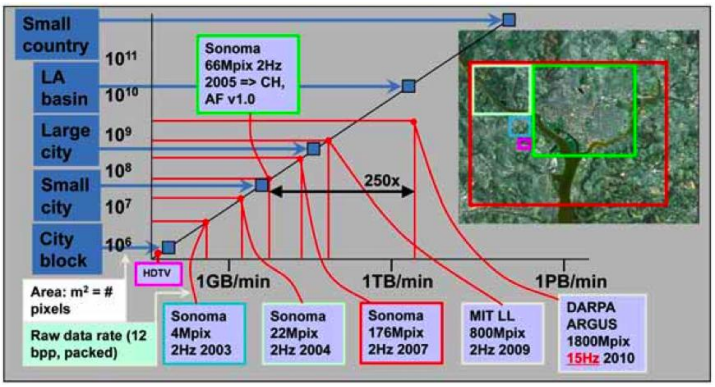


Figure 1: Current and projected data rates for EO persistent surveillance sensors [JASONS 2008]**.** The MIT LL persistent surveillance system generates ~1 TB/hour (compressed).

Abstract

Sensor processing in the Cold War focused on detecting fast moving "stealthy" targets. The Global War On Terror (GWOT) targets are visible in plain sight over a period of days or even months, but only their *behavior* (e.g., movements, destinations, associations, …) distinguishes them from non-targets. Observing these behaviors requires Persistent Surveillance: the continuous monitoring of wide areas using multiple sensor modalities (EO, IR, SAR, GMTI, SigInt, HumInt, …). Persistent Surveillance sensors generate terabytes of data per day, only a fraction of which can be exploited by human analysts. Nevertheless, the power of Persistent Surveillance to detect threats has been clearly demonstrated and is pushing the rapid expansion of Persistent Surveillance sensor capabilities (see Figure 1). The projected 10x-100x increase in data is driving the development of automated detection, tracking, and exploitation algorithms in order to reduce the burden on the human analysts. Lincoln is extensively involved with a variety of fielded Persistent Surveillance assets from sensor design to algorithm development to user displays. Persistent surveillance algorithm development requires enormous computation. To assist the algorithm analyst Lincoln is using its extensive experience with Grid, Cloud, P2P and Graph computing to develop the next generation Persistent Surveillance processing pipeline (see Table 1 and Figure 2). In addition, to providing core computing capabilities, our next generation system will also use containerized computing technology. Once the system has been validated, both the algorithms and the hardware can be replicated for deployment in theatre. [Note: the definition of cloud computing is domain specific. In this context we only imply the use of a particular "cloud" software stack (e.g., Hadoop/MapReduce/HBase) on a remotely deployable cluster.]. This talk will present an overview of the various technical tradeoffs that are involved in employing a containerized approach to support Persistent Surveillance Supercomputing.

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government

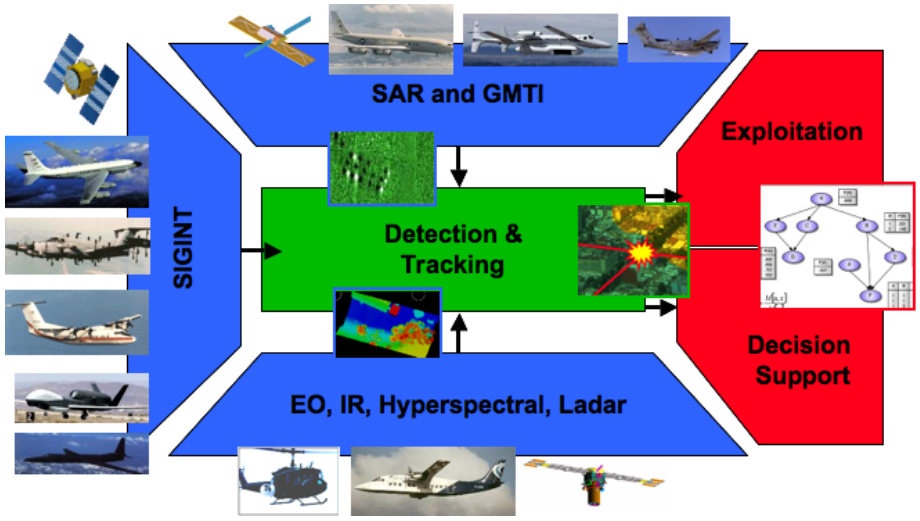
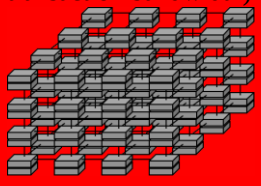


Figure 2: Next Generation Persistent Surveillance Processing Pipeline. The goal is to use supercomputing to develop the required processing algorithms and processing architectures for rapidly processing persistent surveillance data through calibration & registration, detection & tracking, and exploitation & decisions support. SAR=Synthetic Aperture Radar, GMTI=Ground Moving Target Indicator, EO=Electro-Optical, IR=Infra-Red.

Table 1: Next Generation Persistent Surveillance Processing Pipeline. The type of algorithms, input data, and compute kernels vary significantly across pipeline stages. Our system will emulate, prototype, or simulate the required processing architectures for each pipeline stage. These processing architectures will then be duplicated in the field. FFT=Fast Fourier Transform, FIR=Finite Impulse Response, SVD=Singular Value Decompostion, MHT=Multiple Hypothesis Tracker, FAT=Feature Aided Tracker, BFS=Breadth First Search, DFS=Depth First Search, SSSP= Single Source Shortest Path

Stage	**Calibration & registration**	**Detection & tracking**	**Exploitation & decision support**
Required algorithms to be developed	Front end signal processing	Back end signal processing	Graph/database analysis
Input data	Sensor input	Images/dense array	Graphs/sparse arrays
Algorithm kernels	FFT, FIR, SVD, …	Kalman, MHT, FAT…	BFS, DFS, SSSP, …
Next generation computer architecture	Embedded (high sensor bandwidth)  **- Emulate -**	Cluster/cloud container (high disk bandwidth)  **- Prototype -**	Graph processor (high transaction bandwidth)  **- Simulate -**