

# A KASSPER Real-Time Signal Processor Testbed

Glenn Schrader  
MIT Lincoln Laboratory  
244 Wood St.  
Lexington MA, 02420  
Phone: (781)981-2579 Fax: (781)981-5255  
gschrad@ll.mit.edu

The Knowledge Aided Sensor Signal Processing and Expert Reasoning (KASSPER) Program is a Defense Advanced Research Projects Agency (DARPA) program which has the goal of improving the performance of Ground Moving Target Indicator (GMTI) radar systems by incorporating external sources of knowledge into the signal processing chain. The KASSPER Real-Time Signal Processor Testbed and its associated Signal Processing Architecture is a prototype radar system scheduling and signal processing framework that has been developed at Massachusetts Institute of Technology Lincoln Laboratory (MIT LL).

A typical scenario in which KASSPER processing could be useful is depicted in Fig. 1. Note that the GMTI clutter environment is heterogeneous (trees, open areas, an urban area, etc). By taking advantage of prior knowledge about this environment (i.e. locations of roads, terrain contours, types of ground cover, etc.), processing algorithms can have an opportunity to improve their performance by avoiding invalid assumptions about the environment.

A top level diagram of the KASSPER architecture is shown in Fig. 2. The components of this architecture that are different from a conventional radar signal processor are the knowledge database, knowledge cache, and the look-ahead scheduler.



**Fig. 1. A representative KASSPER problem.**

**This work is sponsored by the Defense Advanced Research Projects Agency, under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.**

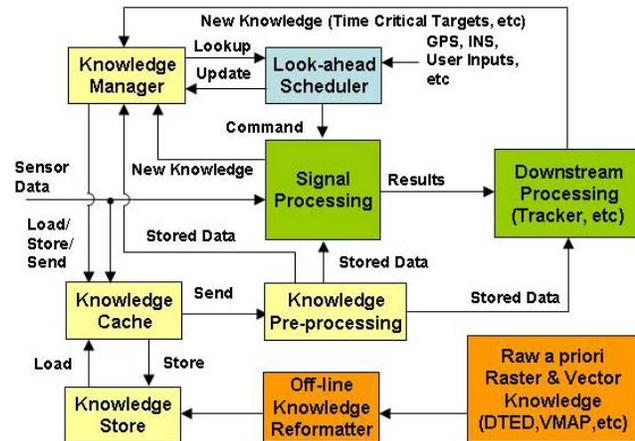
# Report Documentation Page

Form Approved  
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE <b>01 FEB 2005</b>		2. REPORT TYPE <b>N/A</b>		3. DATES COVERED <b>-</b>	
4. TITLE AND SUBTITLE <b>A KASSPERReal-Time Embedded Signal Processor Testbed</b>				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Massachusetts Institute of TechnologyLincoln Laboratory</b>				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release, distribution unlimited</b>					
13. SUPPLEMENTARY NOTES <b>See also ADM001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004. , The original document contains color images.</b>					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

The knowledge database stores the 'knowledge' that both the signal processing chain and the scheduler need to perform 'intelligent' processing. The knowledge database reformats the knowledge data into a form that is usable by the processing chain during a pre-mission data load. During system operation, the knowledge cache holds the portion of the knowledge store that is within the radar's field of view. The knowledge pre-processing step performs real-time operations such as transformations between geographic coordinates and radar system coordinates.



**Fig. 2 KASSPER Architecture Block Diagram**

An important factor in the performance of knowledge aided signal processing algorithms is that the application of knowledge must be performed locally. This tends to cause the signal processing to be done at a finer grain than is comfortable for achieving good computational efficiency. As an aid to making processor scaling choices for knowledge aided algorithms, signal processing and knowledge database benchmarks have been developed in order to quantify the efficiencies that will actually be achievable.

This talk will discuss the design of the testbed, its architecture; the testbed's software to hardware mapping, knowledge aided processing algorithms, as well as computational performance data that have been collected for important knowledge-aided processing kernels.

A number of topics (such as the look-ahead scheduler) are currently areas of ongoing research. An overview of the potential benefits as well as a vision of how the architecture will incorporate knowledge aided scheduling will be presented.



# **A KASSPER Real-Time Embedded Signal Processor Testbed**

**Glenn Schrader**

**Massachusetts Institute of Technology  
Lincoln Laboratory**

**28 September 2004**

This work is sponsored by the Defense Advanced Research Projects Agency, under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

**MIT Lincoln Laboratory**



# Outline



## **KASSPER Program Overview**

- **KASSPER Processor Testbed**
- **Computational Kernel Benchmarks**
  - **STAP Weight Application**
  - **Knowledge Database Pre-processing**
- **Summary**

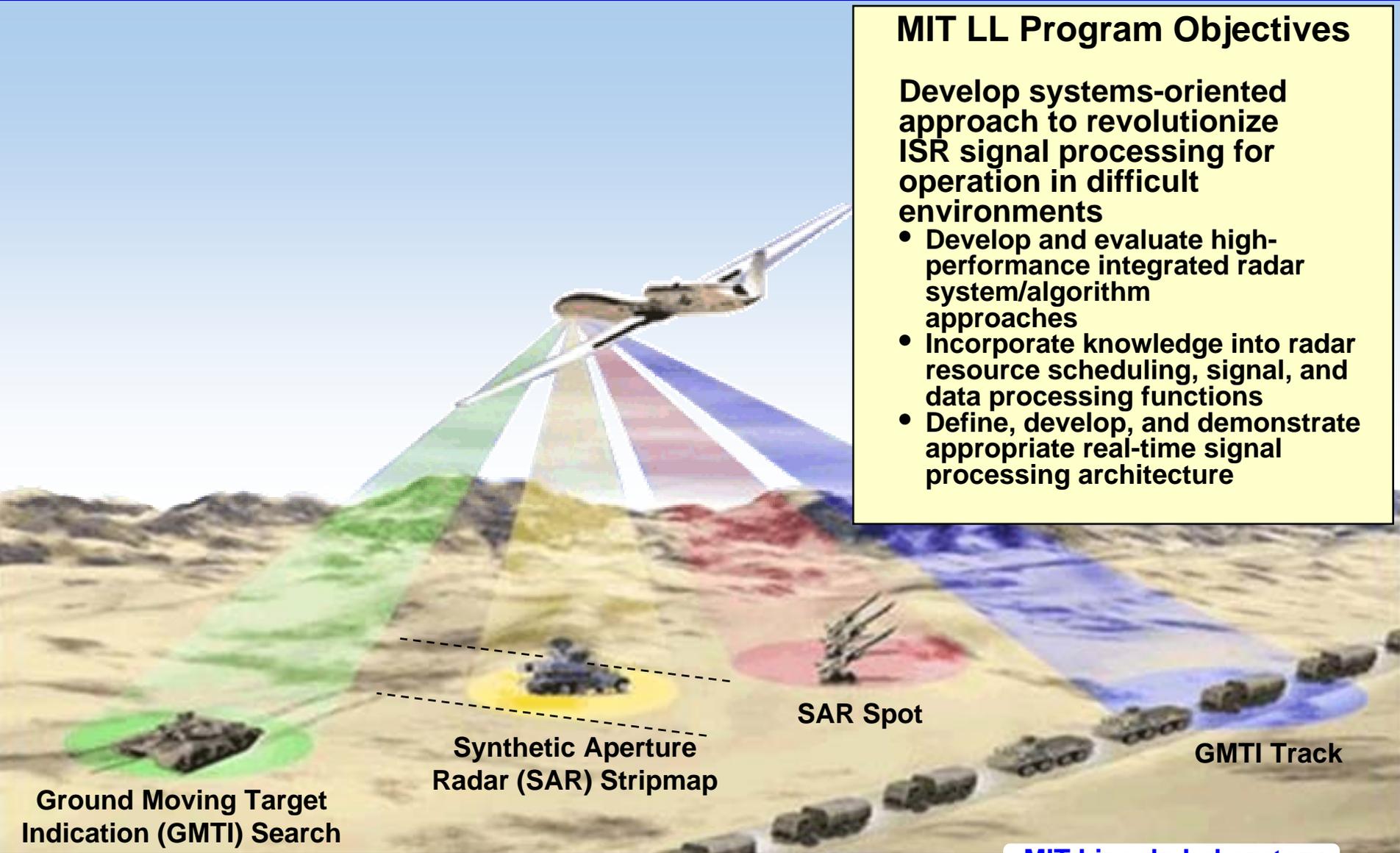


# Knowledge-Aided Sensor Signal Processing and Expert Reasoning (KASSPER)

## MIT LL Program Objectives

Develop systems-oriented approach to revolutionize ISR signal processing for operation in difficult environments

- Develop and evaluate high-performance integrated radar system/algorithm approaches
- Incorporate knowledge into radar resource scheduling, signal, and data processing functions
- Define, develop, and demonstrate appropriate real-time signal processing architecture

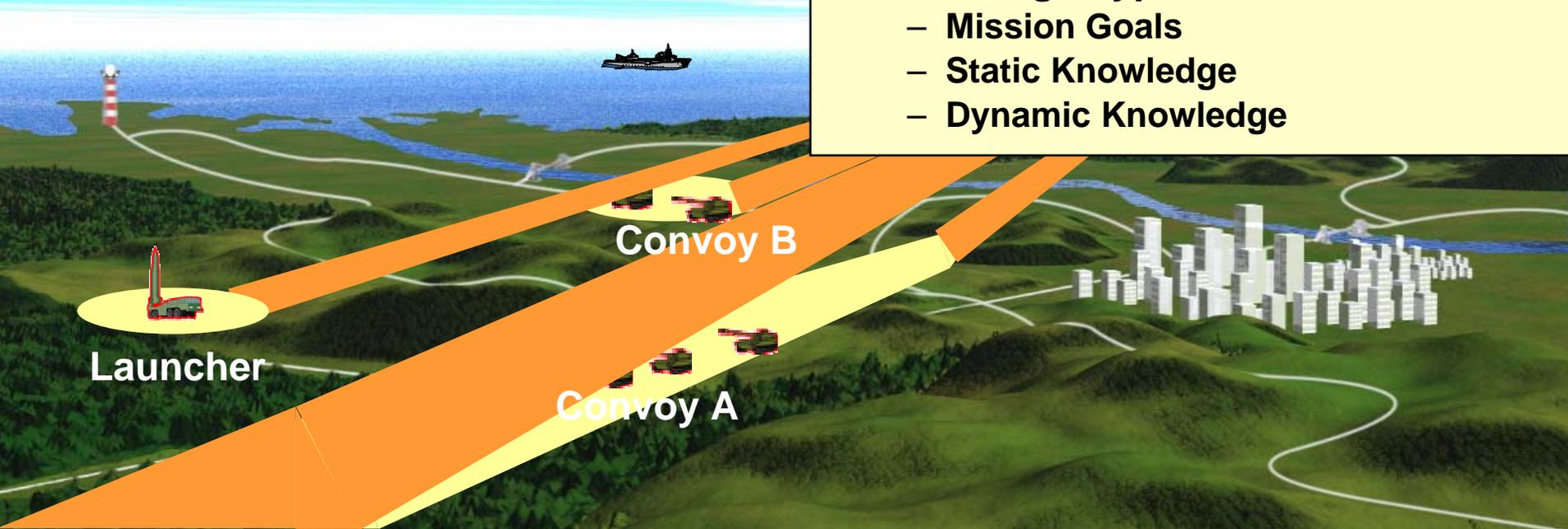




# Representative KASSPER Problem

- **Radar System Challenges**
  - Heterogenous clutter environments
  - Clutter discretets
  - Dense target environments
- **Results in decreased system performance (e.g. higher Minimum Detectable Velocity, Probability of Detection vs. False Alarm Rate)**

- **KASSPER System Characteristics**
  - Knowledge Processing
    - Knowledge-enabled Algorithms
    - Knowledge Repository
  - Multi-function Radar
    - Multiple Waveforms + Algorithms
    - Intelligent Scheduling
- **Knowledge Types**
  - Mission Goals
  - Static Knowledge
  - Dynamic Knowledge





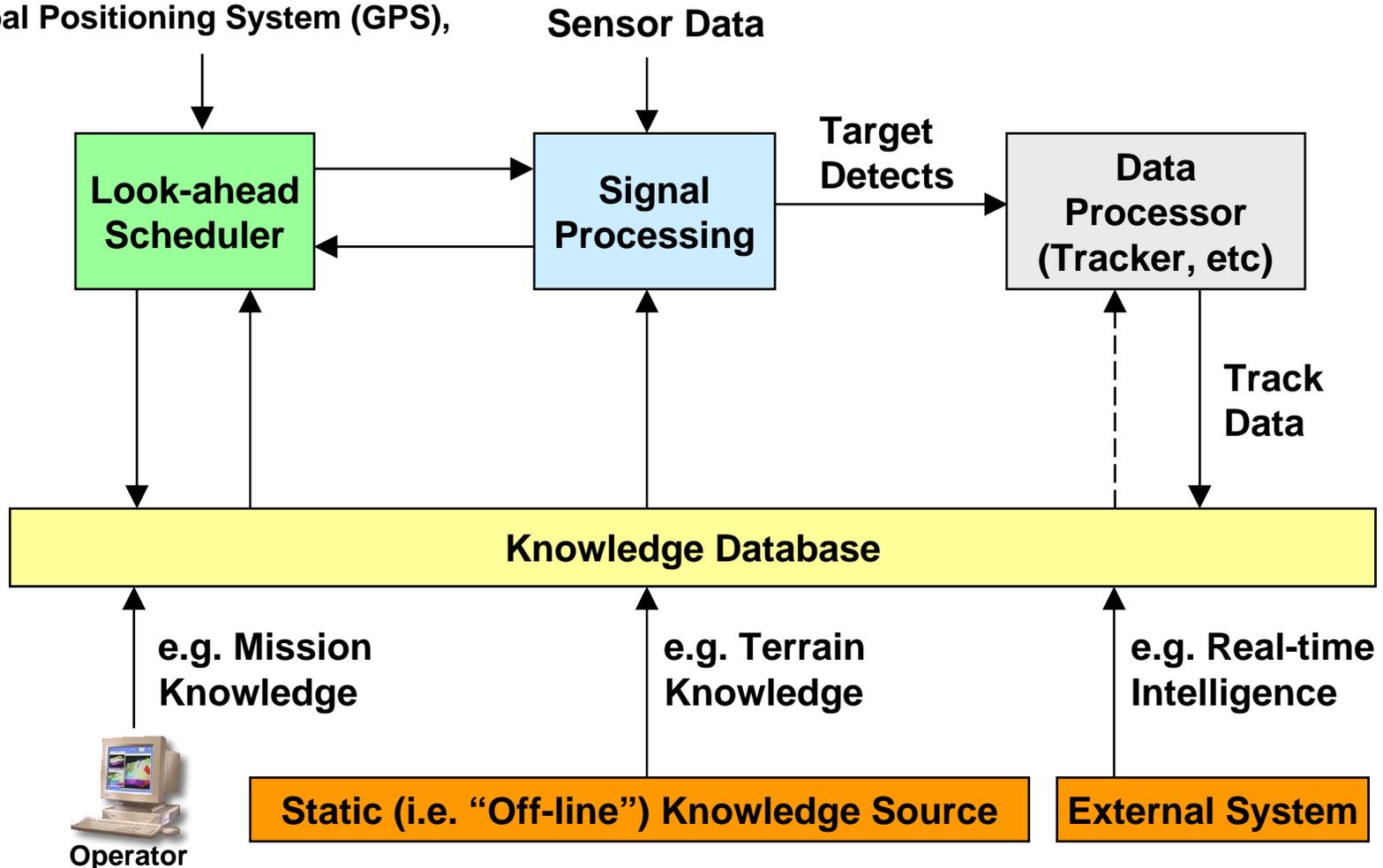
# Outline

- **KASSPER Program Overview**
- ➔ **KASSPER Processor Testbed**
- **Computational Kernel Benchmarks**
  - **STAP Weight Application**
  - **Knowledge Database Pre-processing**
- **Summary**



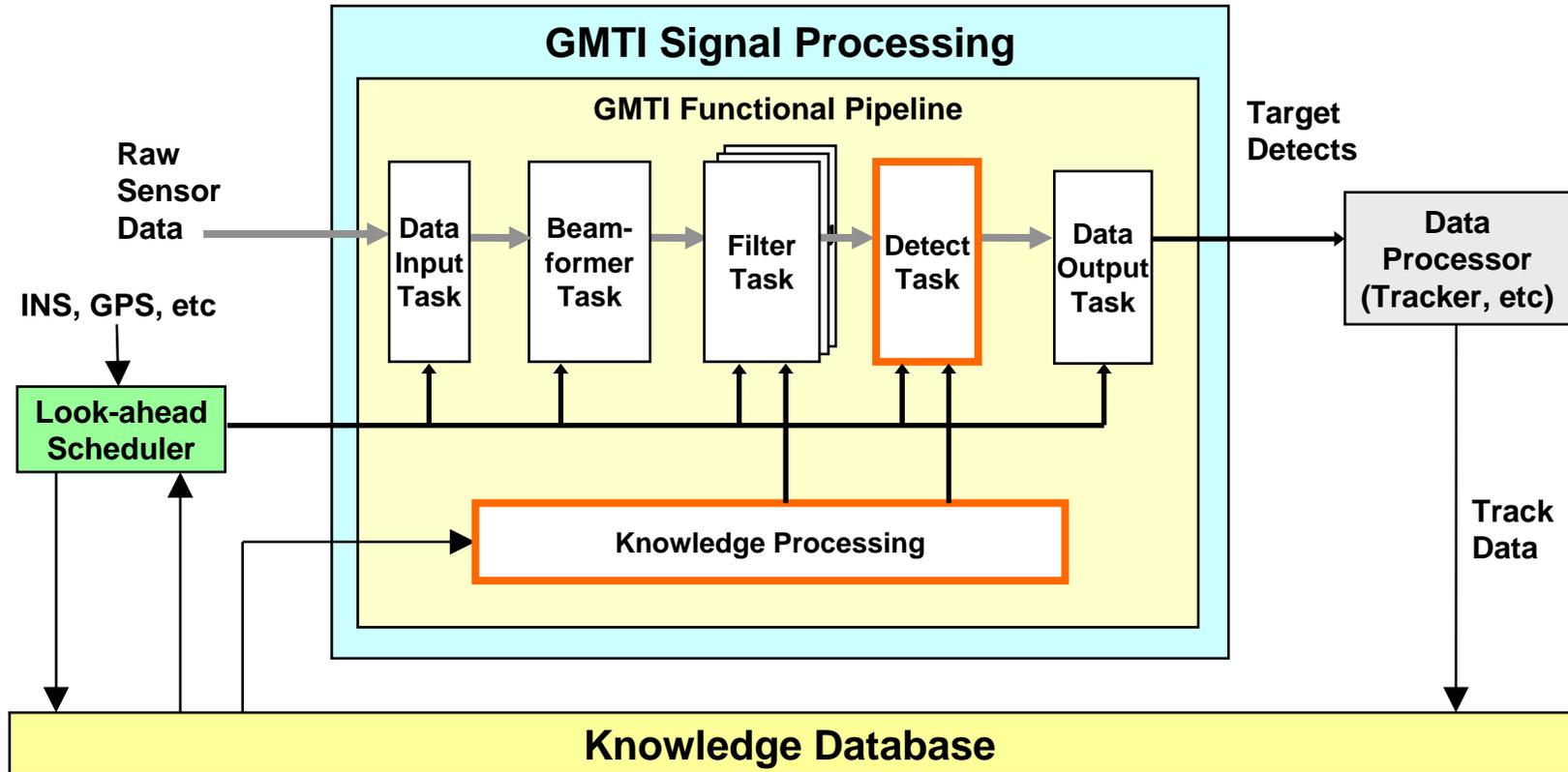
# High Level KASSPER Architecture

Inertial Navigation System (INS),  
Global Positioning System (GPS),  
etc.





# Baseline Signal Processing Chain

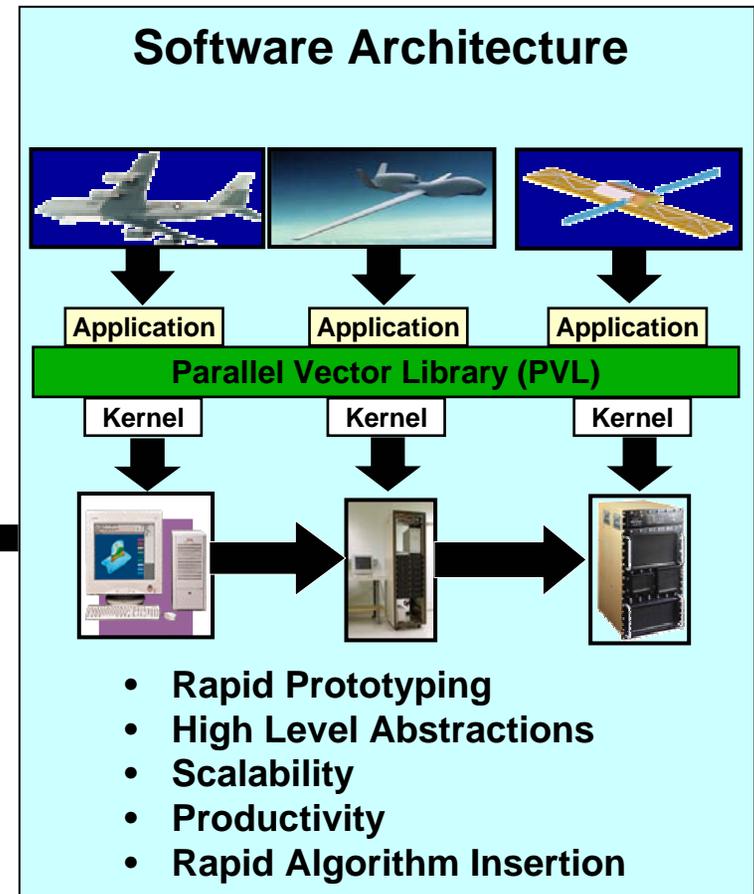
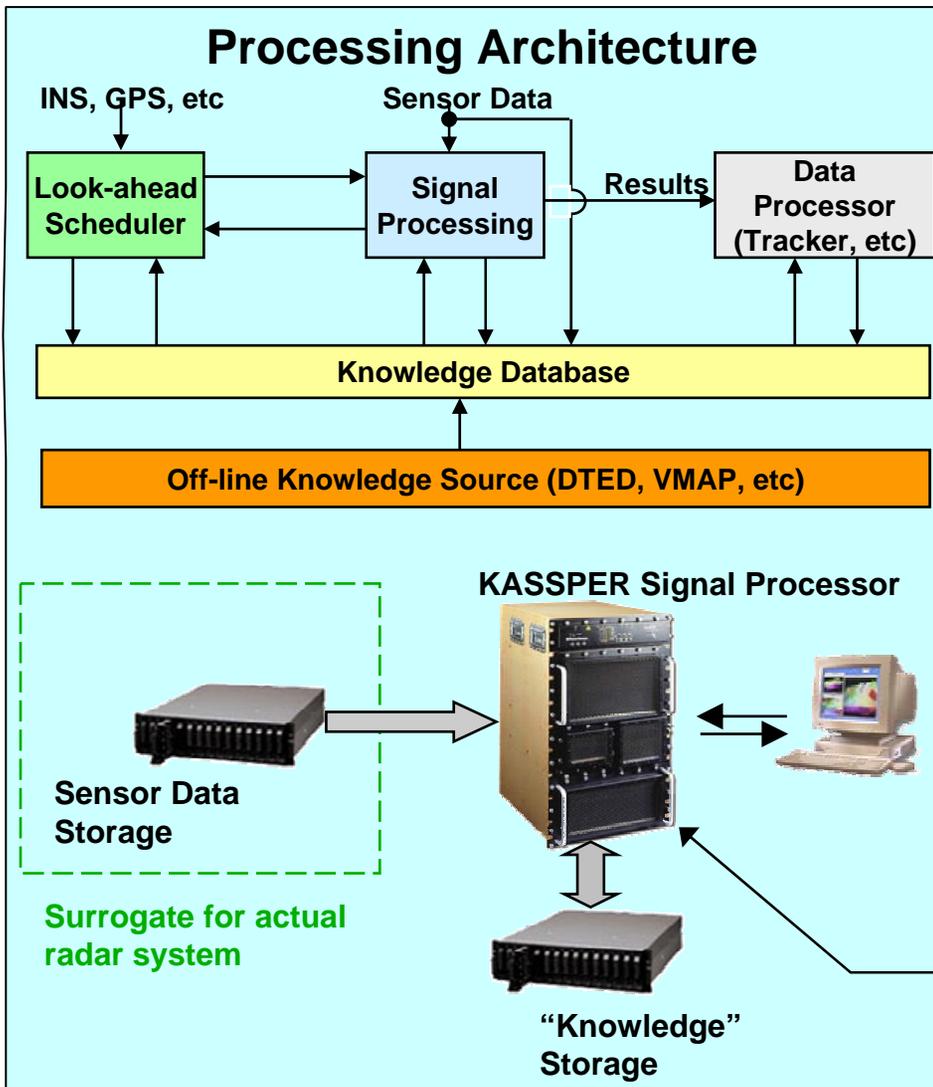


## Baseline data cube sizes:

- 3 channels x 131 pulses x 1676 range cells
- 12 channels x 35 pulses x 1666 range cells



# KASSPER Real-Time Processor Testbed

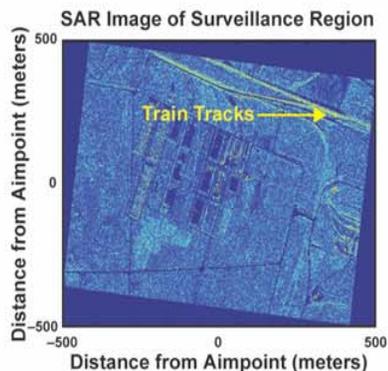
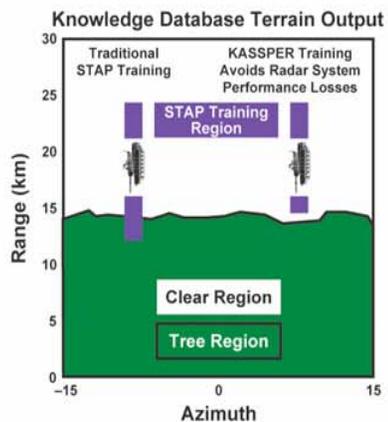


120 PPC G4 CPUs @ 500 MHZ  
4 GFlop/Sec/CPU = 480 GFlop/Sec

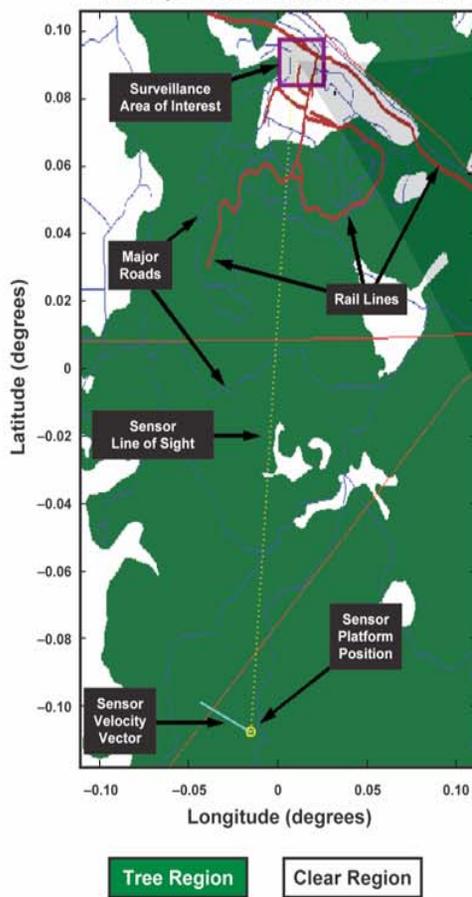


# KASSPER Knowledge-Aided Processing Benefits

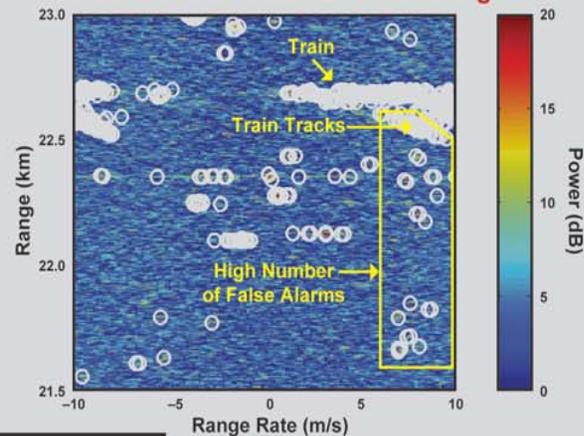
## Additional Knowledge Sources for Use in KASSPER Processing



## Example Surveillance Scenario

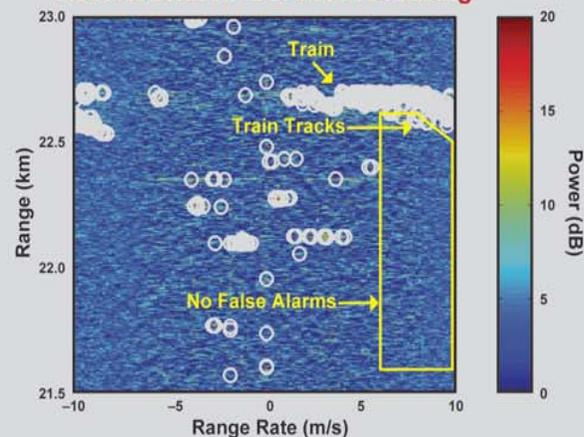


## Results Without KASSPER Processing



O Detected Targets

## Results With KASSPER Processing



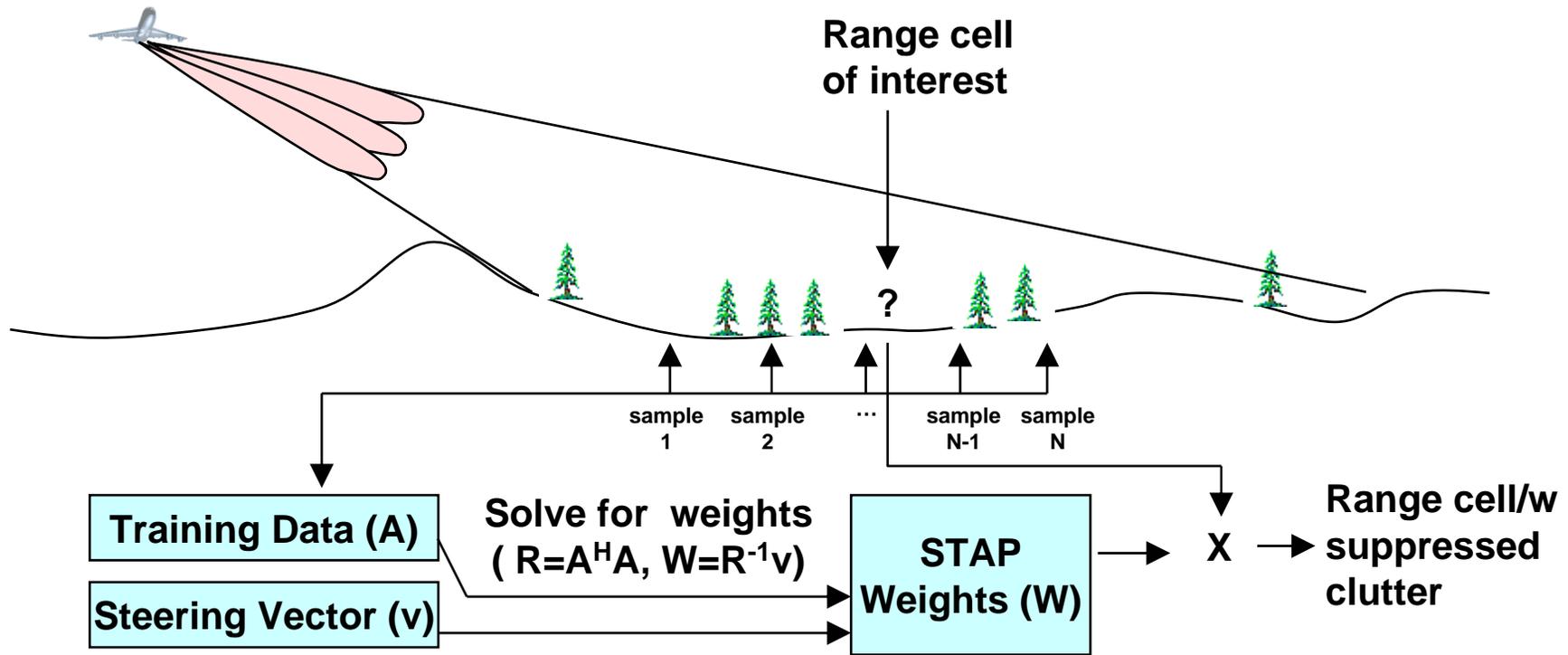


# Outline

- **KASSPER Program Overview**
- **KASSPER Processor Testbed**
- **Computational Kernel Benchmarks**
  - ➔ **– STAP Weight Application**
  - Knowledge Database Pre-processing**
- **Summary**



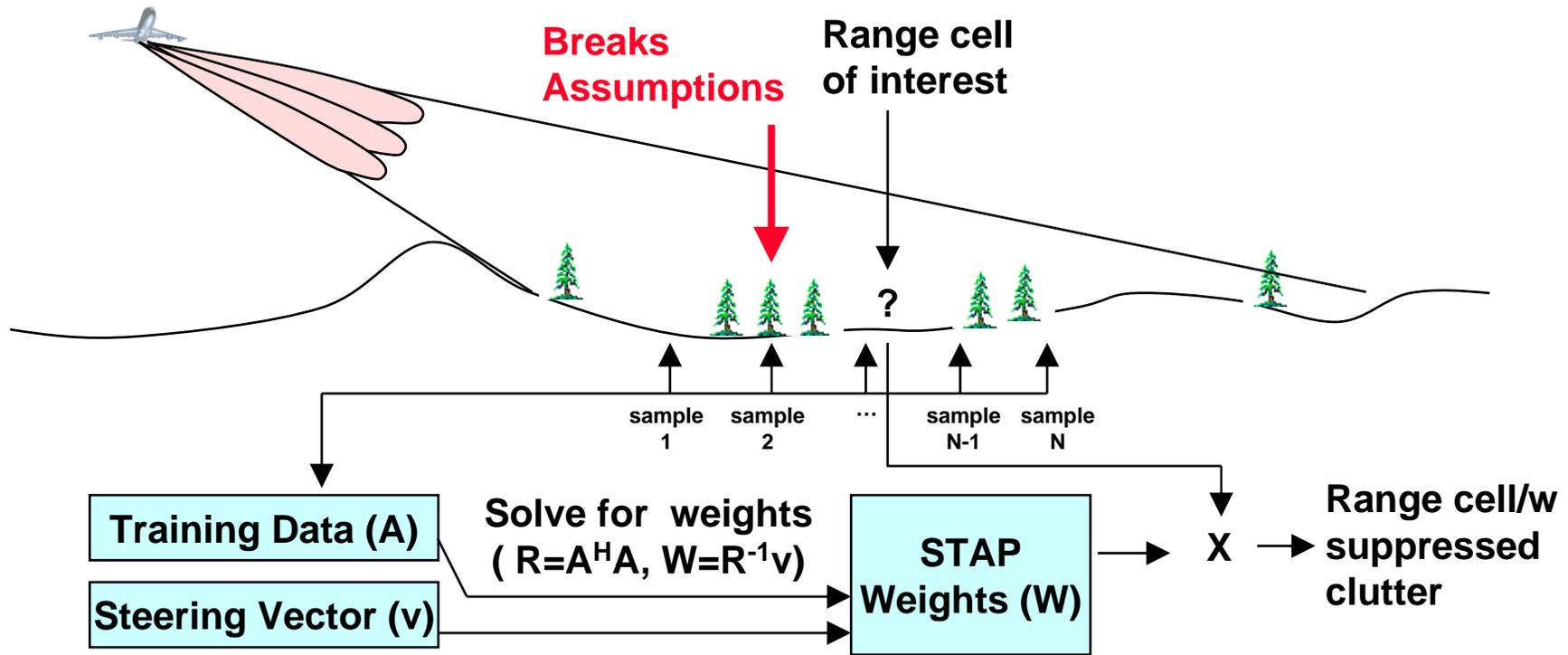
# Space Time Adaptive Processing (STAP)



- Training samples are chosen to form an estimate of the background
- Multiplying the STAP Weights by the data at a range cell suppresses features that are similar to the features that were in the training data



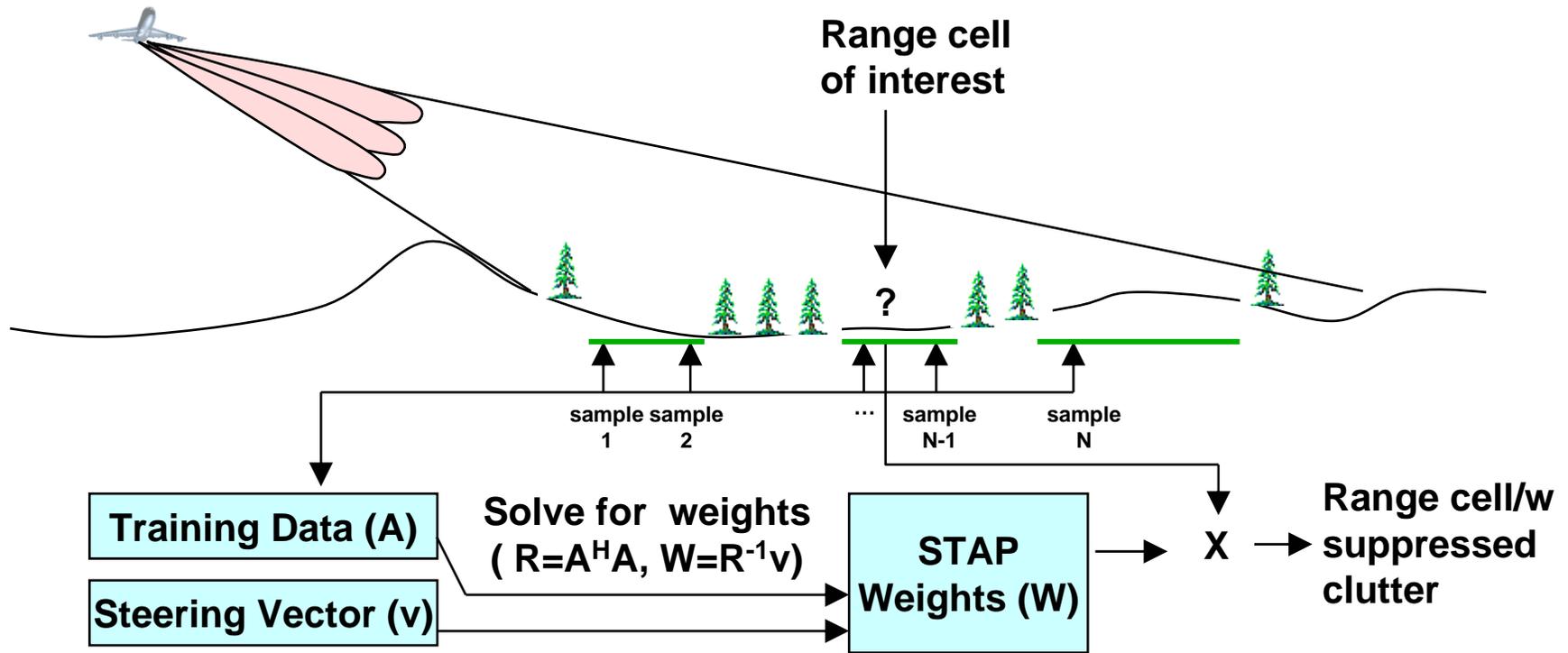
# Space Time Adaptive Processing (STAP)



- Simple training selection approaches may lead to degraded performance since the clutter in training set may not match the clutter in the range cell of interest



# Space Time Adaptive Processing (STAP)

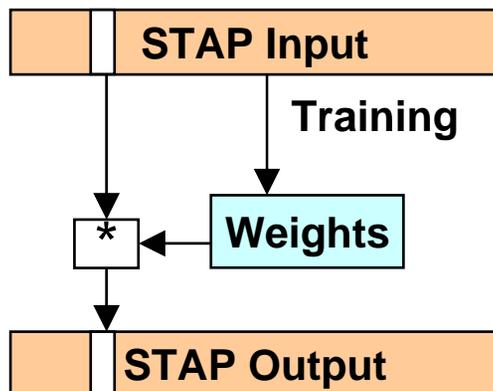


- Use of terrain knowledge to select training samples can mitigate the degradation

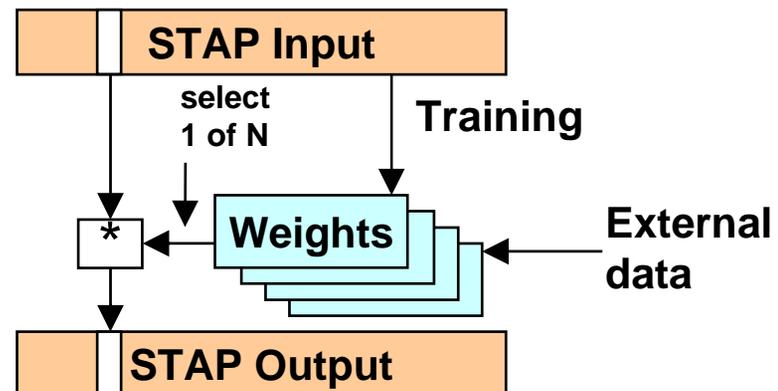


# STAP Weights and Knowledge

- **STAP weights computed from training samples within a training region**
- **To improve processor efficiency, many designs apply weights across a swath of range (i.e. matrix multiply)**
- **With knowledge enhanced STAP techniques, MUST assume that adjacent range cells will not use the same weights (i.e. weight selection/computation is data dependant)**



**Equivalent to Matrix Multiply  
since same weights  
are used for all columns**



**Cannot use Matrix Multiply  
since the weights applied to  
each column are data dependant**

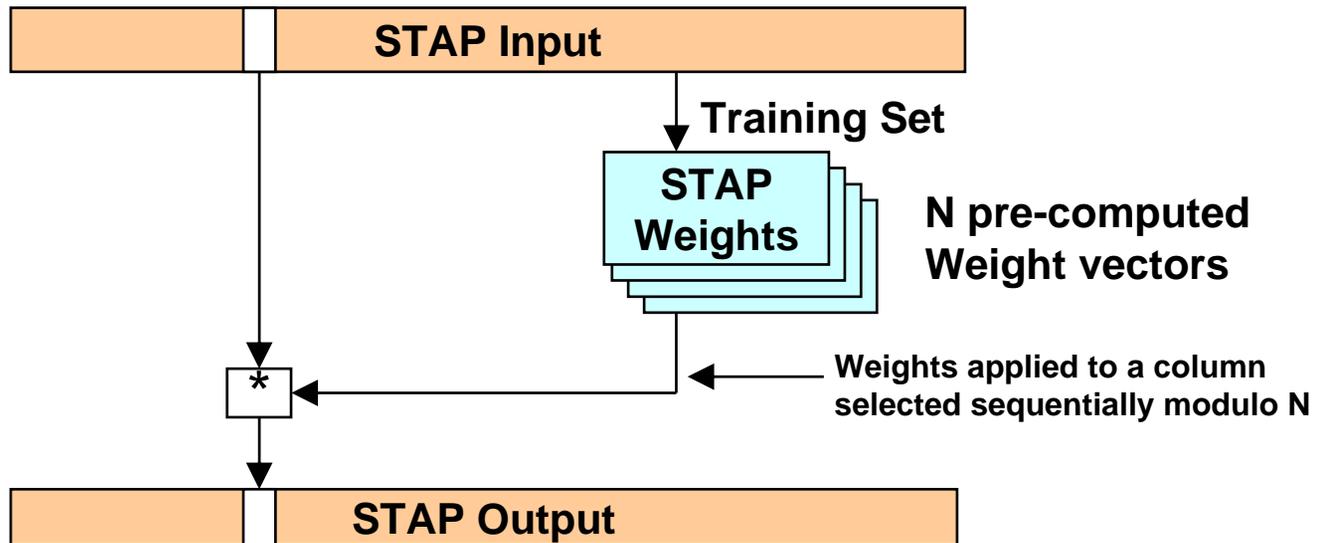


# Benchmark Algorithm Overview

## Benchmark Reference Algorithm (Single Weight Multiply)

$$\text{STAP Output} = \text{Weights} \times \text{STAP Input}$$

## Benchmark Data Dependant Algorithm (Multiple Weight Multiply)

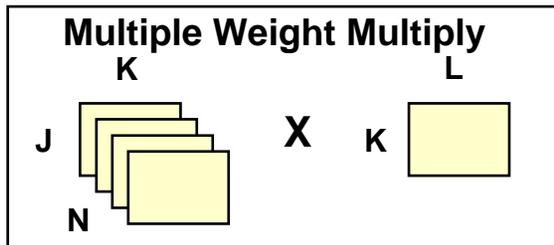
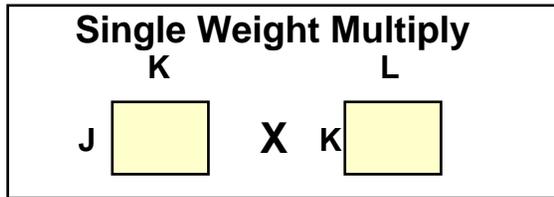


Benchmark's goal is to determine the achievable performance compared to the well-known matrix multiply algorithm.

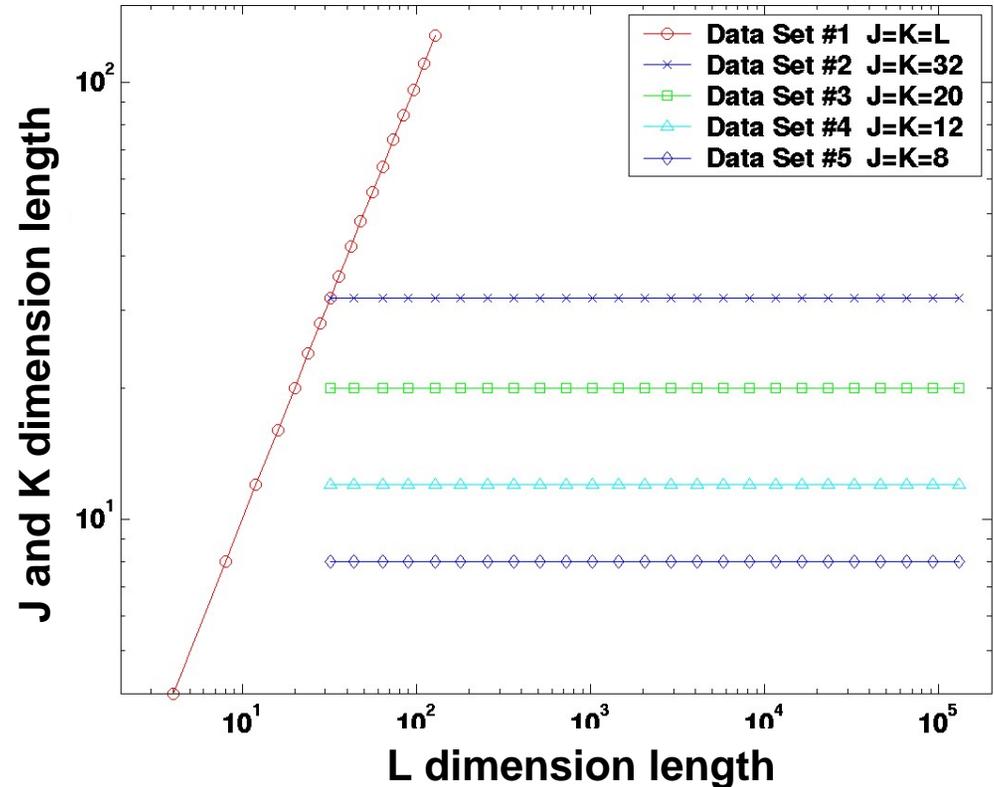


# Test Case Overview

Data Set J x K x L



N = # of weight matrices = 10

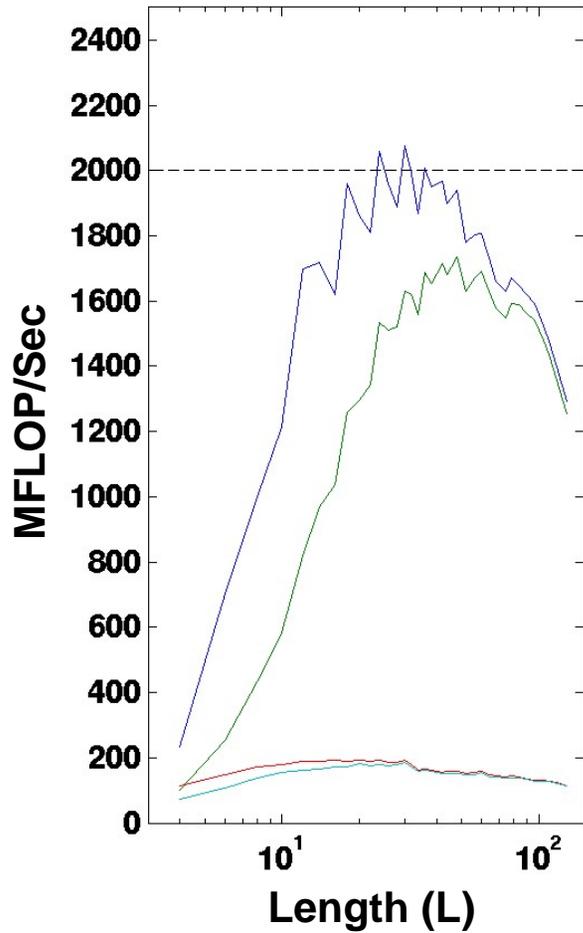


- **Complex Data**
  - Single precision interleaved
- **Two test architectures**
  - PowerPC G4 (500Mhz)
  - Pentium 4 (2.8Ghz)
- **Four test cases**
  - Single weight multiply/wo cache flush
  - Single weight multiply/w cache flush
  - Multiple weight multiply/wo cache flush
  - Multiple weight multiply/w cache flush

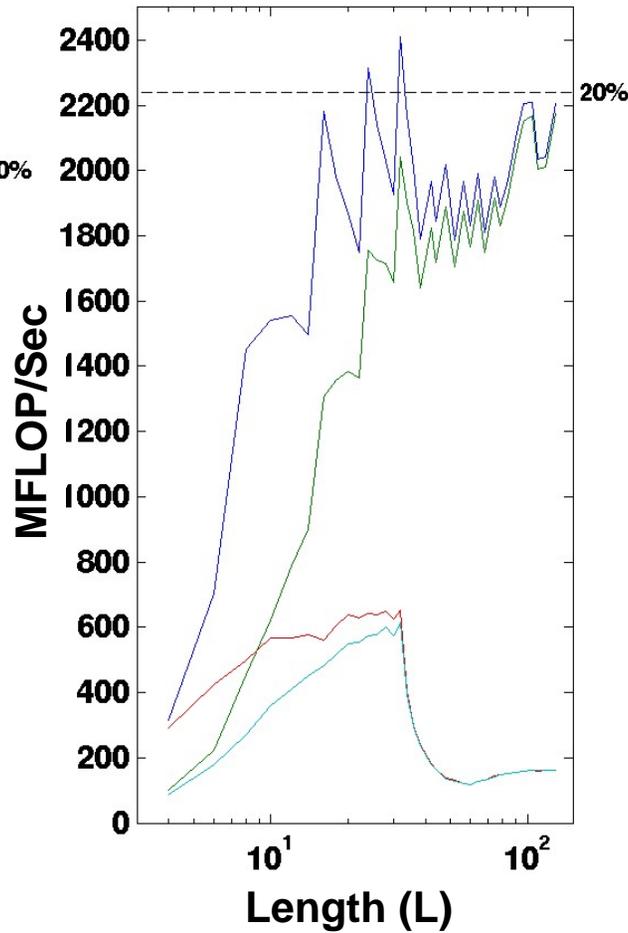


# Data Set #1 - LxLxL

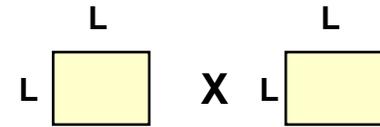
(G4 500Mhz)



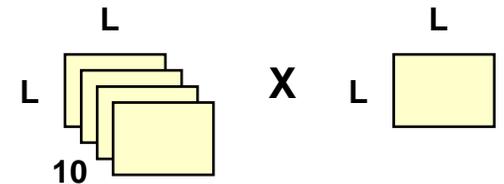
(P4 2.8Ghz)



Single Weight Multiply



Multiple Weight Multiply

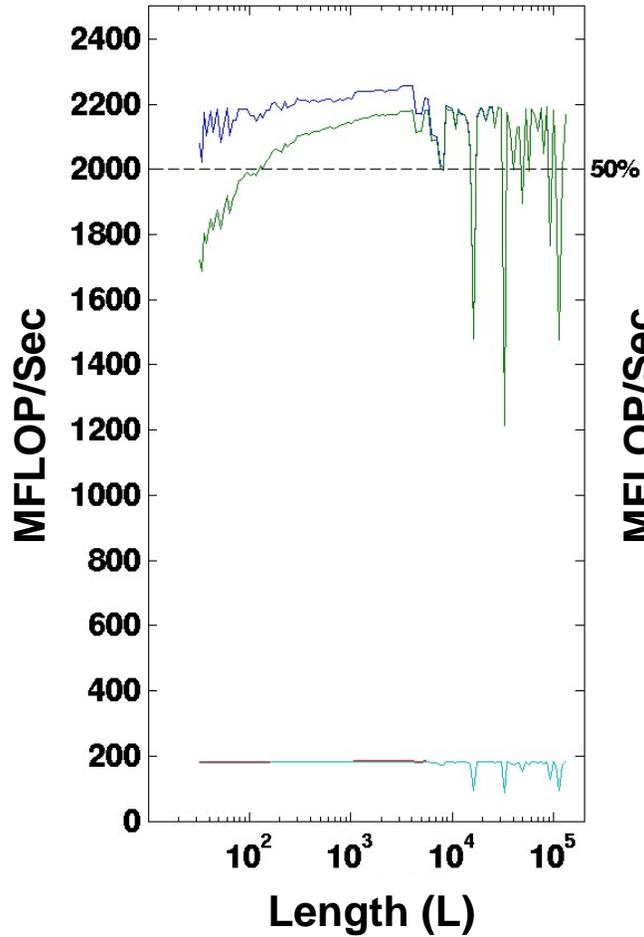


- Single/wo flush —
- Single/w flush —
- Multiple/wo flush —
- Multiple/w flush —

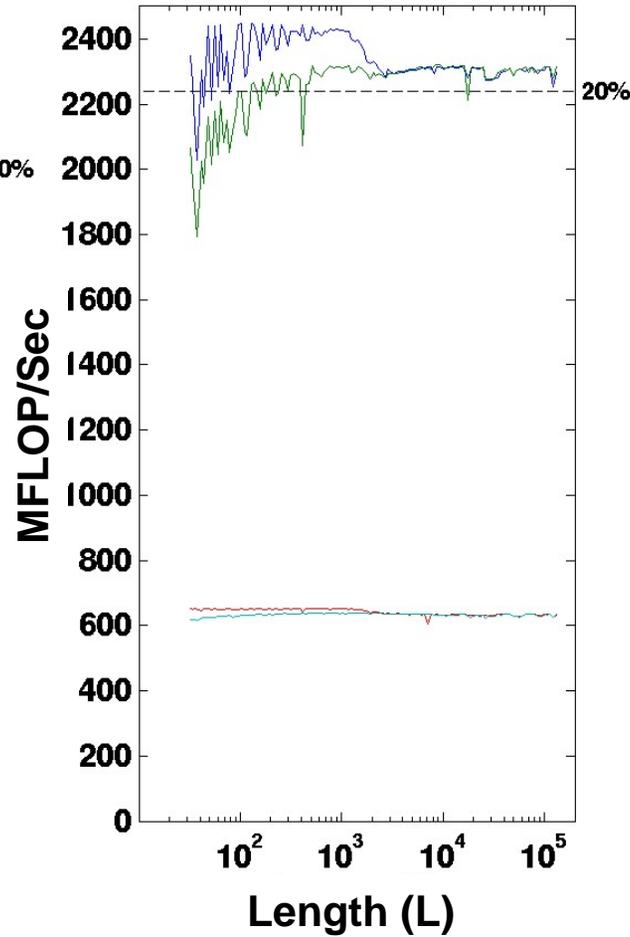


# Data Set #2 – 32x32xL

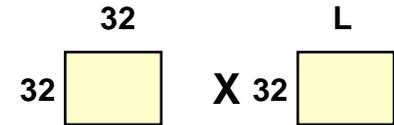
(G4 500Mhz)



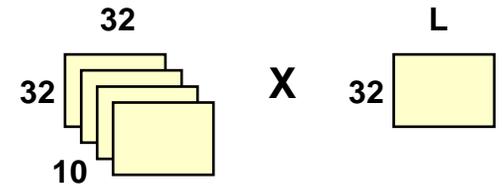
(P4 2.8Ghz)



## Single Weight Multiply



## Multiple Weight Multiply

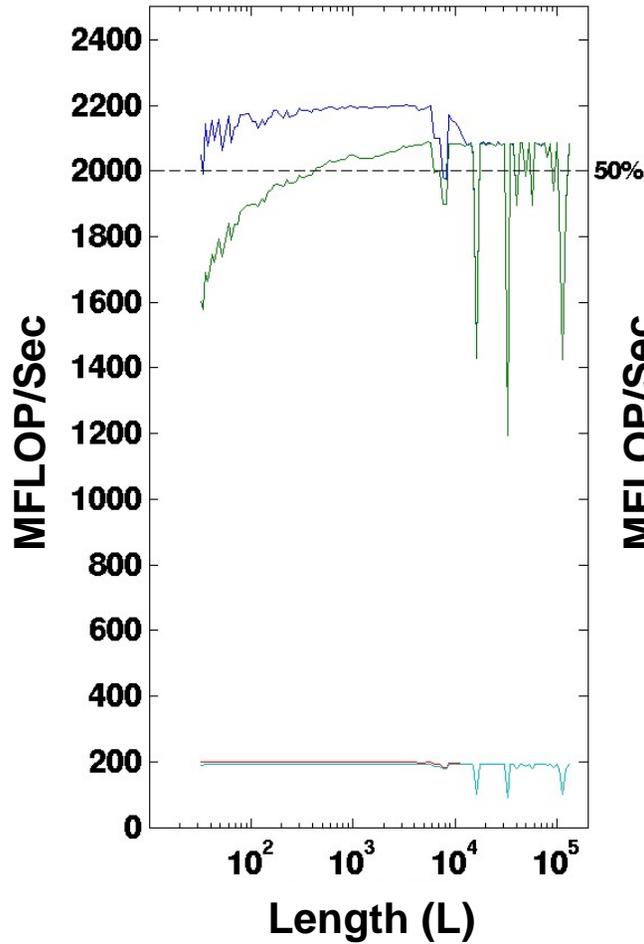


- Single/wo flush 
- Single/w flush 
- Multiple/wo flush 
- Multiple/w flush 

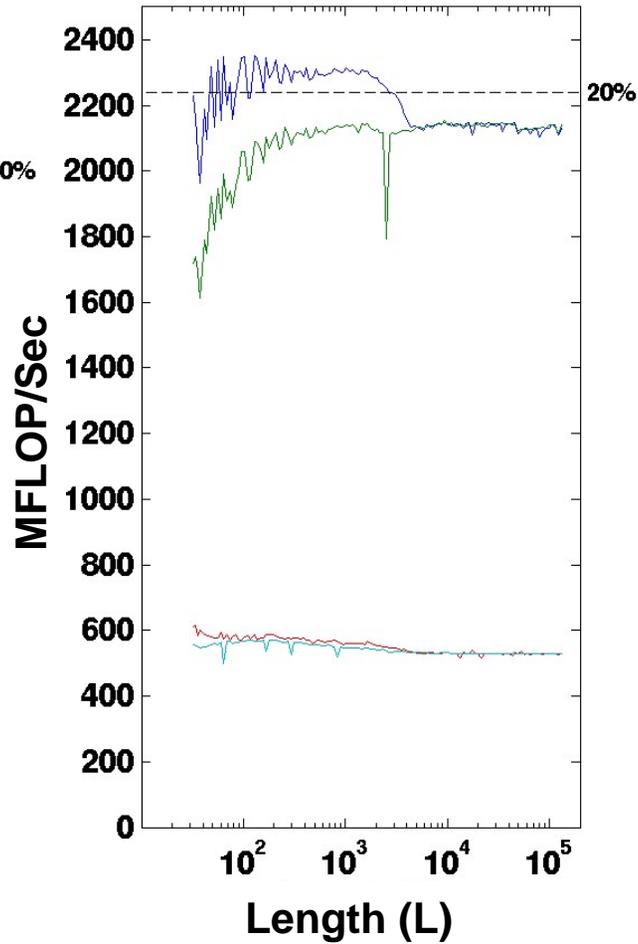


# Data Set #3 – 20x20xL

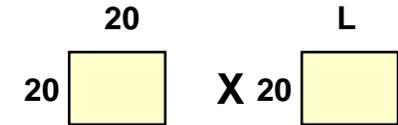
(G4 500Mhz)



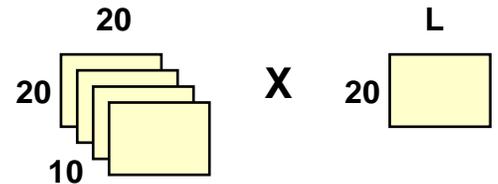
(P4 2.8Ghz)



**Single Weight Multiply**



**Multiple Weight Multiply**

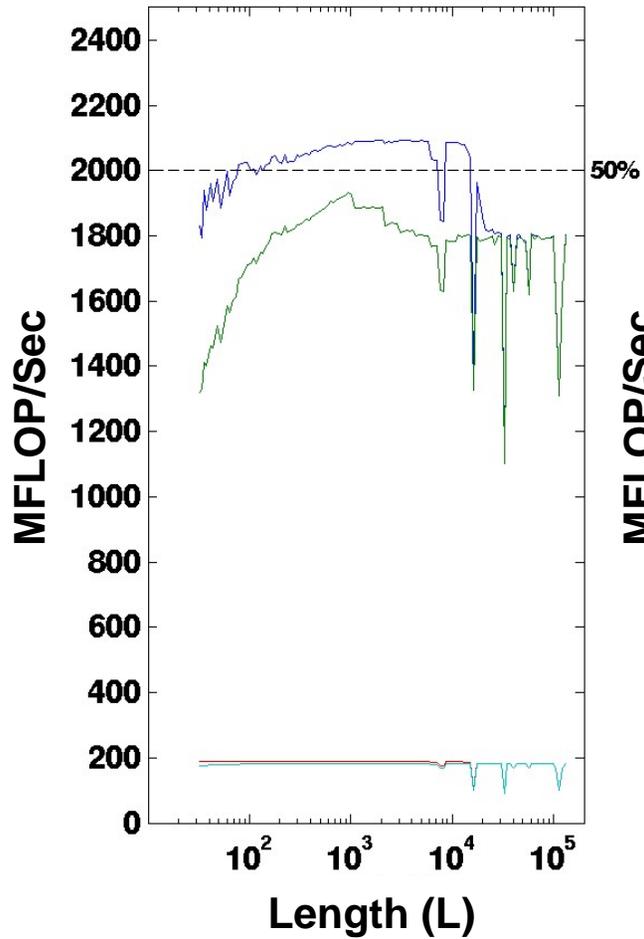


- Single/wo flush —
- Single/w flush —
- Multiple/wo flush —
- Multiple/w flush —

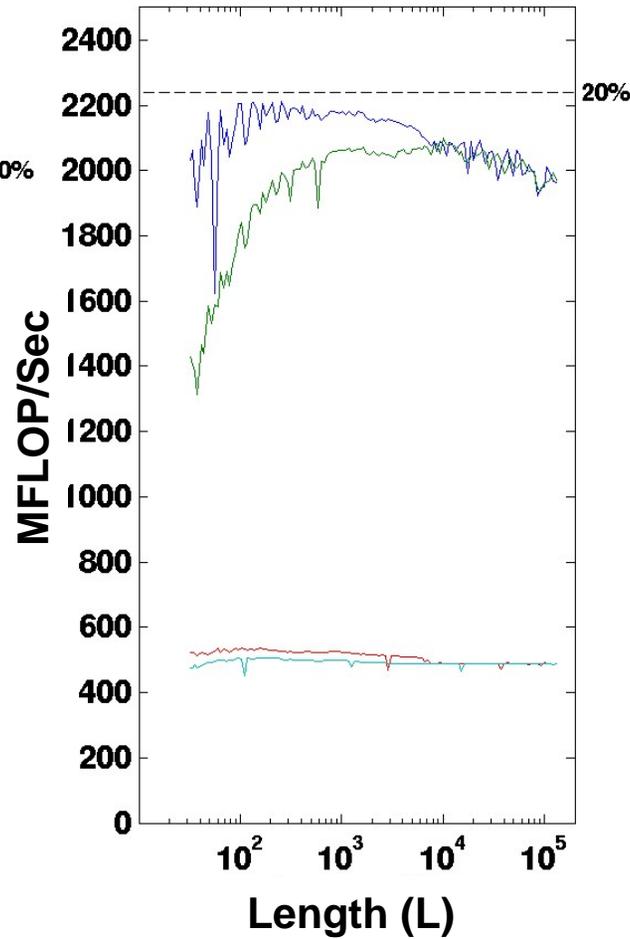


# Data Set #4 – 12x12xL

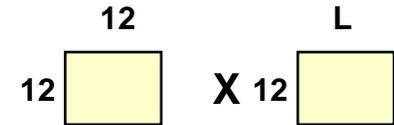
(G4 500Mhz)



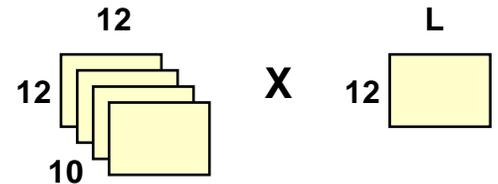
(P4 2.8Ghz)



Single Weight Multiply



Multiple Weight Multiply

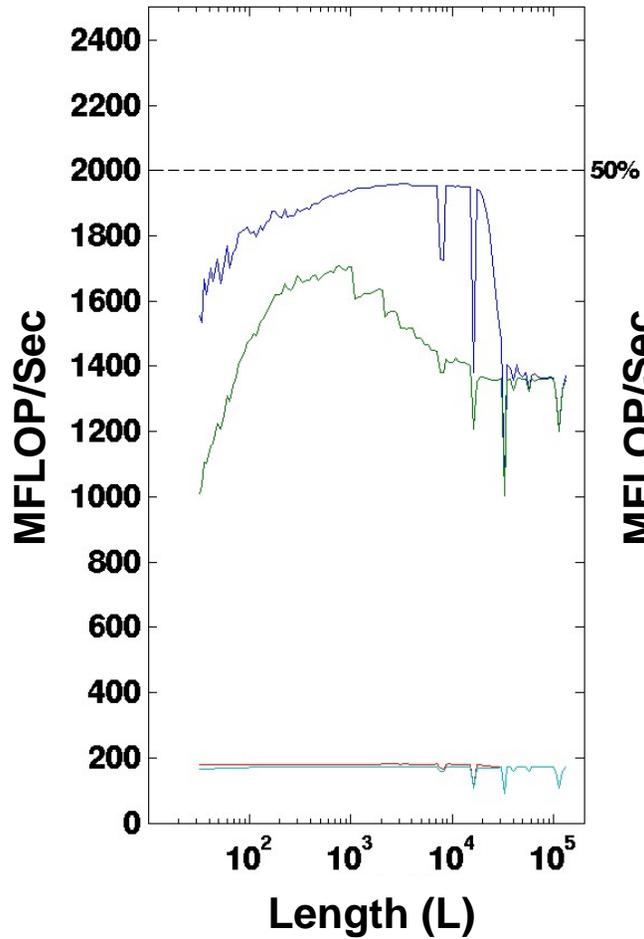


- Single/wo flush —
- Single/w flush —
- Multiple/wo flush —
- Multiple/w flush —

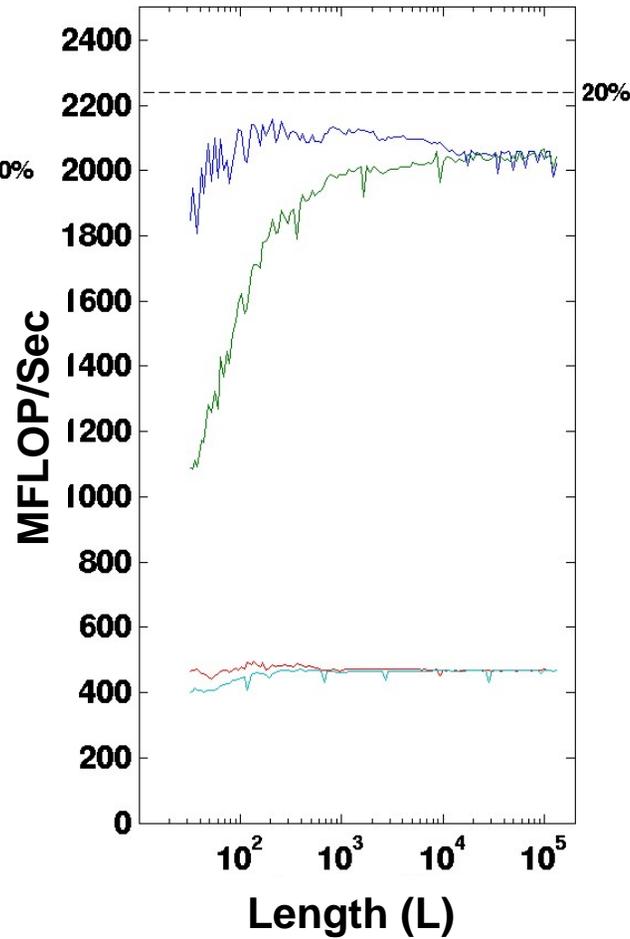


# Data Set #5 – 8x8xL

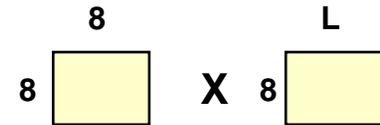
(G4 500Mhz)



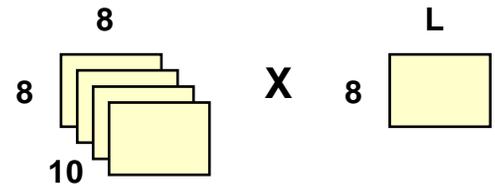
(P4 2.8Ghz)



Single Weight Multiply



Multiple Weight Multiply



Single/wo flush —  
Single/w flush —  
Multiple/wo flush —  
Multiple/w flush —



# Performance Observations

- **Data dependent processing cannot be optimized as well**
  - Branching prevents compilers from “unrolling” loops to improve pipelining
- **Data dependent processing does not allow efficient “re-use” of already loaded data**
  - Algorithms cannot make simplifying assumptions about already having the data that is needed.
- **Data dependent processing does not vectorize**
  - Using either AltiVec or SSE is very difficult since data movement patterns become data dependant
- **Higher memory bandwidth reduces cache impact**
- **Actual performance scales roughly with clock rate rather than theoretical peak performance**

**Approx 3x to 10x performance degradation,  
Processing efficiency of 2-5% depending on CPU architecture**



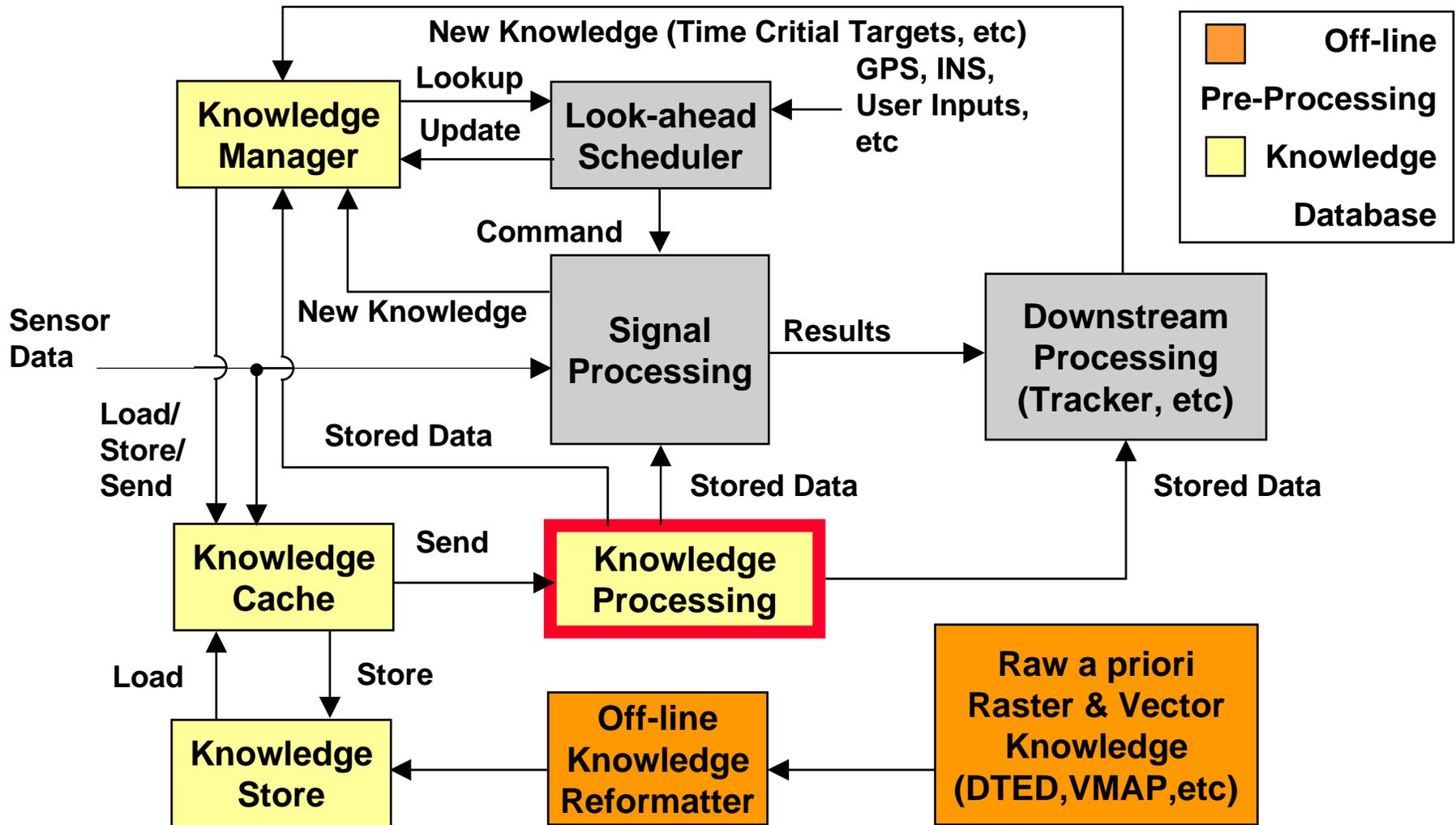
# Outline

- **KASSPER Program Overview**
- **KASSPER Processor Testbed**
- **Computational Kernel Benchmarks**
  - **STAP Weight Application**
  - **Knowledge Database Pre-processing**
- **Summary**





# Knowledge Database Architecture

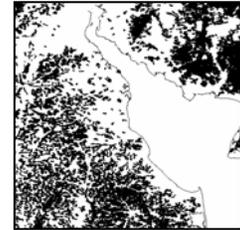




# Static Knowledge

- **Vector Data**

- Each feature represented by a set of point locations:
  - Points - longitude and latitude (i.e. towers, etc)
  - Lines - list of points, first and last points are not connected (i.e. roads, rail, etc)
  - Areas - list of points, first and last point are connected (i.e. forest, urban, etc)
- **Standard Vector Formats**
  - Vector Smart Map (VMAP)



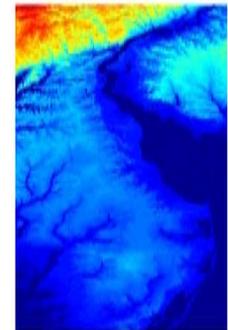
Trees (area)



Roads (line)

- **Matrix Data**

- Rectangular arrays of evenly spaced data points
- **Standard Raster Formats**
  - Digital Terrain Elevation Data (DTED)

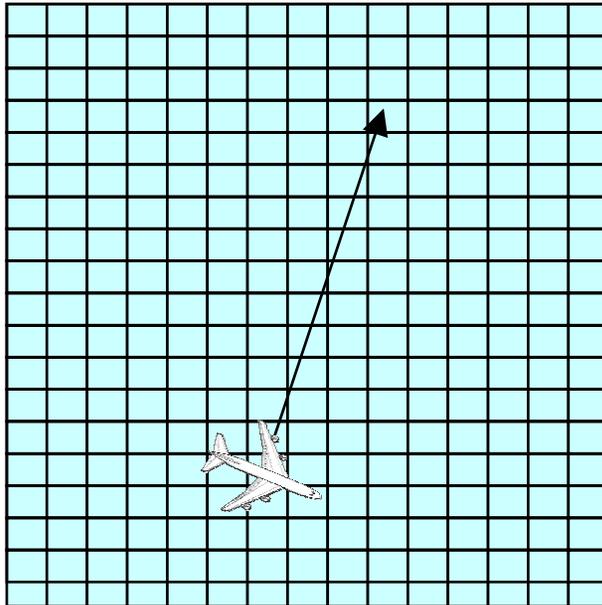


DTED

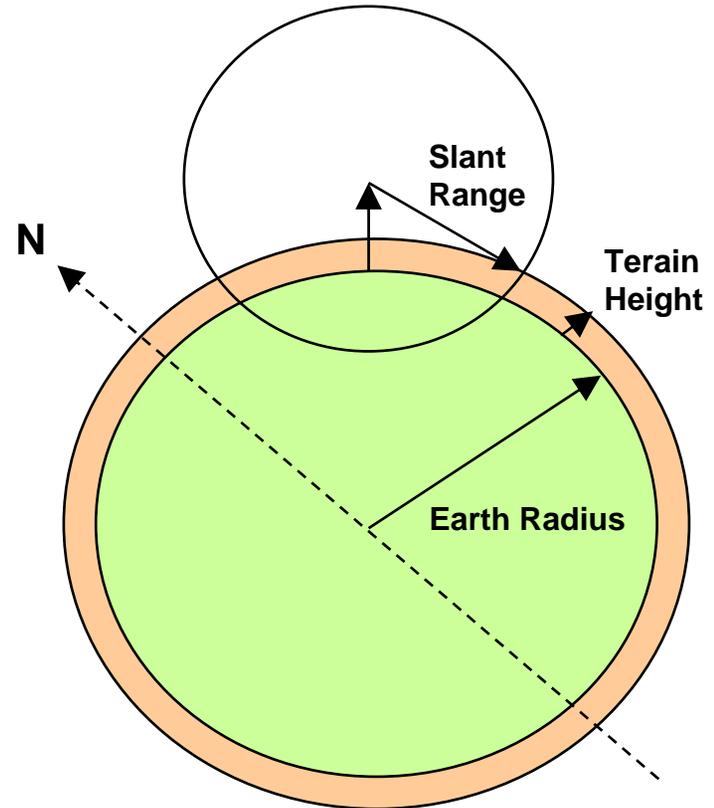


# Terrain Interpolation

**Sampled Terrain Height  
& Type data**



**Geographic Position**



**Converting from radar to geographic coordinates  
requires iterative refinement of estimate**



# Terrain Interpolation Performance Results

<u>CPU Type</u>	<u>Time per interpolation</u>	<u>Processing Rate</u>
P4 2.8	1.27uSec/interpolation	144MFlop/sec (1.2%)
G4 500	3.97uSec/interpolation	46MFlop/sec (2.3%)

- **Data dependent processing does not vectorize**
  - Using either AltiVec or SSE is very difficult
- **Data dependent processing cannot be optimized as well**
  - Compilers cannot “unroll” loops to improve pipelining
- **Data dependent processing does not allow efficient “re-use” of already loaded data**
  - Algorithms cannot make simplifying assumptions about already having the data that is needed

**Processing efficiency of 1-3% depending on CPU architecture**



# Outline

- **KASSPER Program Overview**
- **KASSPER Processor Testbed**
- **Computational Kernel Benchmarks**
  - **STAP Weight Application**
  - **Knowledge Database Pre-processing**

 **Summary**



# Summary

- **Observations**
  - The use of knowledge processing provides a significant system performance improvement
  - Compared to traditional signal processing algorithms, the implementation of knowledge enabled algorithms can result in a factor of 4 or 5 lower CPU efficiency (as low as 1%-5%)
  - Next-generation systems that employ cognitive algorithms are likely to have similar performance issues
- **Important Research Directions**
  - Heterogeneous processors (i.e. a mix of COTS CPUs, FPGAs, GPUs, etc) can improve efficiency by better matching hardware to the individual problems being solved. For what class of systems is the current technology adequate?
  - Are emerging architectures for cognitive information processing needed (e.g. Polymorphous Computing Architecture – PCA, Application Specific Instruction set Processors - ASIPs)?