Georgia Tech Research Institute

Problem. Solved.

GPU-Based Space-Time Adaptive Processing for Radar

Thomas M. Benson, GTRI

Ryan K. Hersey, GTRI

Edwin Culpepper, AFRL

What is STAP?

- Moving radar platform → clutter spread in Doppler
- Detecting targets with speeds similar to background clutter requires clutter suppression
- STAP applies an adaptive 2D filter to suppress clutter and other sources of interference
- Adaptively optimal solutions are currently computationally impractical, but families of more efficient STAP algorithms have been developed
- We focus here on the extended factored algorithm (EFA)

STAP Overview

- Space and slow-time adaptivity enables simultaneous clutter and noise jammer suppression
- Detection of weak and/or slow-moving targets

Covariance Matrix Estimation

$$\hat{\mathbf{R}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k \mathbf{x}_k^H$$

↓

Adaptive Weight Calculation

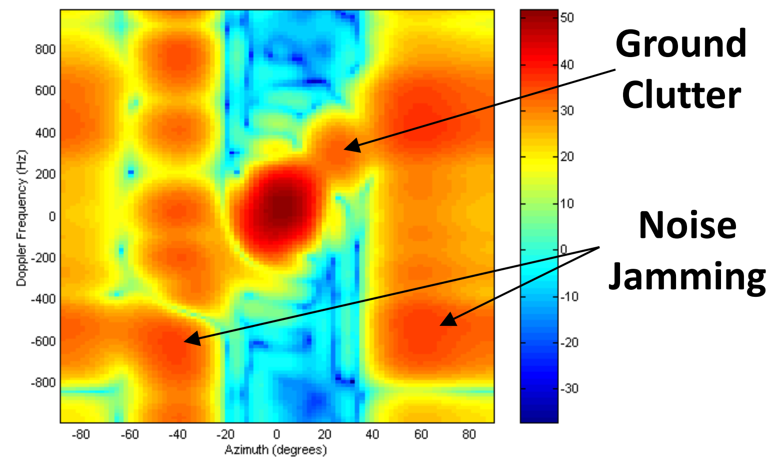
$$\mathbf{w} = \hat{\mathbf{R}}^{-1}\mathbf{v}$$

↓

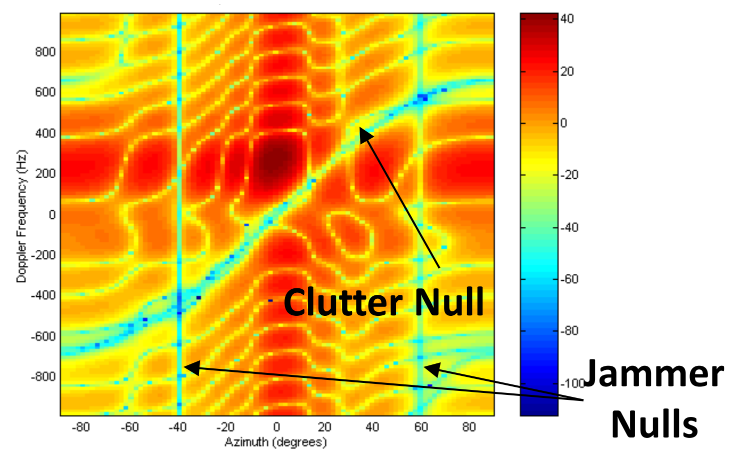
Adaptive Weight Application

$$y = \frac{|\mathbf{w}^H \mathbf{x}|^2}{\mathbf{v}^H \hat{\mathbf{R}}^{-1} \mathbf{v}}$$

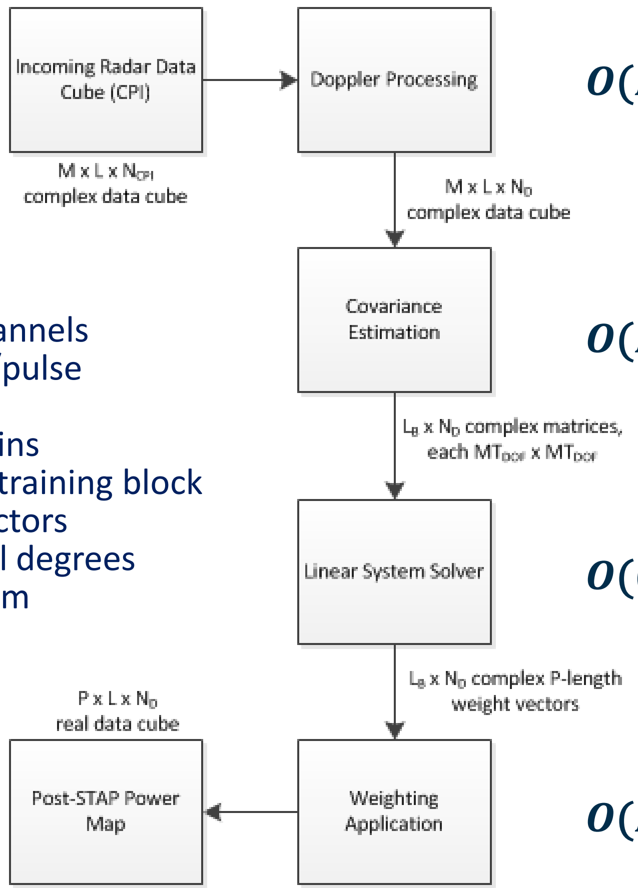
Power Spectral Density



STAP Filter Response



Notional EFA Data Flow



Complexity

$$O(M \cdot L \cdot N_D \cdot \log N_D)$$

$$O(L \cdot N_D \cdot (M \cdot T_{DOF})^2)$$

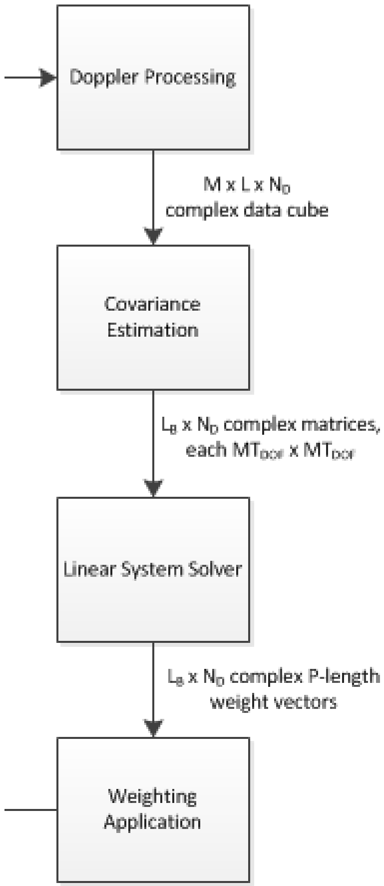
$$O((L/B) \cdot N_D \cdot ((M \cdot T_{DOF})^3 + P \cdot (M \cdot T_{DOF})^2))$$

$$O(L \cdot N_D \cdot P \cdot (M \cdot T_{DOF}))$$

Legend
M – # spatial channels
L – # range bins/pulse
N_{CPI} - # pulses
N_D - # Doppler bins
B - # range bins/training block
P - # steering vectors
T_{DOF} - # temporal degrees of freedom

EFA Complexity Analysis

Eliminating common terms…



$$\frac{\log N_D}{T_{DOF}}$$

$$M \cdot T_{DOF}$$

$$\frac{(M \cdot T_{DOF})^2 + P \cdot (M \cdot T_{DOF})}{B}$$

$$P$$

Typically, $B > M \cdot T_{DOF}$.

For the values of B, M, and T_{DOF} presented later, the weighting step exceeds the system solver for $P \geq 8$, with the caveat that we have ignored constants.

Legend

M – # spatial channels
L – # range bins/pulse
N_{CPI} - # pulses
N_D - # Doppler bins
B - # range bins/training block
P - # steering vectors
T_{DOF} - # temporal degrees of freedom

Summary: Doppler Processing

- Applies windowing + FFT along the pulse (N) dimension
- Can utilize efficient FFT libraries (e.g., CUFFT), but requires a corner turn for FFTs over contiguous arrays
- May involve zero-padding prior to the FFT

Summary: Covariance Estimation

- Goal: Estimate the background covariance for each range-Doppler pair
- As a computational savings, range is split into blocks of B range bins with one covariance estimate for all B bins
- Covariance is then estimated as the mean of two neighboring blocks in range (local block is a guard)
- The estimate for each range block is the sum of B outer products, $\mathbf{x}\mathbf{x}^H$, where $\mathbf{x}$ is an $M \times T_{DOF}$ length vector
- Equivalently, each covariance entry can be viewed as a B-length inner product

Summary: Linear System Solver

- Covariance matrices are Hermitian and positive semi-definite by construction
- Given sufficient training, linear independence due to noise, and potential diagonal loading, positive definiteness is typical and assumed for this work
- Many small linear systems to solve (i.e. batch mode)
- Can utilize Cholesky factorization, Gauss-Jordan elimination, QR decomposition, etc.

Summary: Weight Application

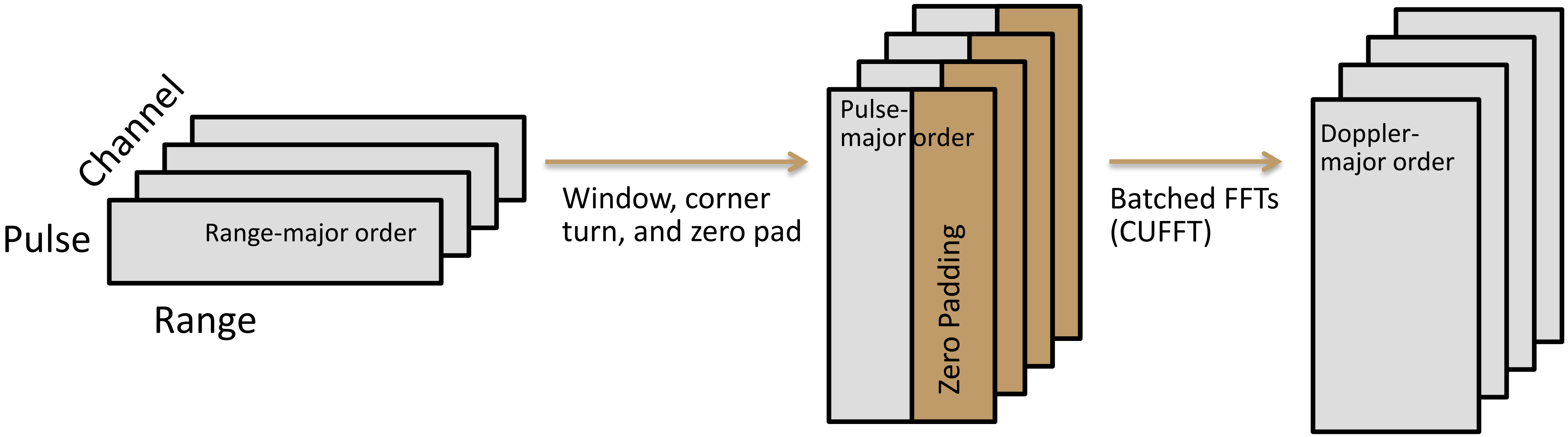
- Applies the generated adaptive weights to Doppler-processed data cube to obtain an output power map as a function of Doppler, range, steering vector
  - Weights applied to same M x T_{DOF} snapshots used for outer products in covariance estimation
- Every output point requires a M x T_{DOF} length complex inner product and normalization
- Adaptive weights are re-used for all *B* range bins in a range block
- Workload scales ~linearly with number of steering vectors

Data Set Parameters

Parameter	Variable	Value
Spatial channels	M	4
Pulses per CPI	N_{CPI}	128
Doppler bins	N_D	256
Range bins	L	512
Training block size	B	32
# Training blocks	L_B	16
Temporal DoF	T_{DOF}	3
Steering Vectors	P	32

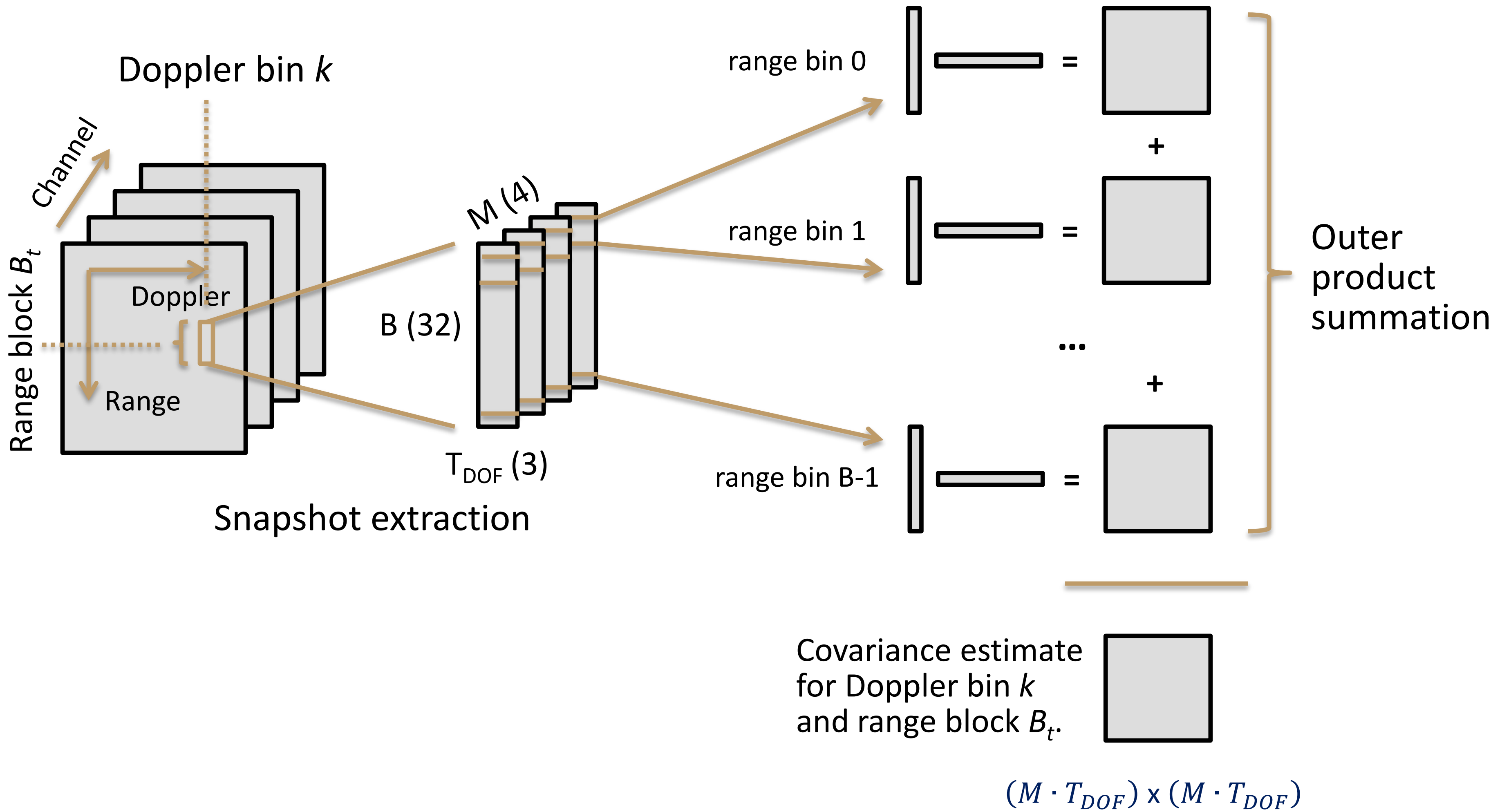
Implementation: Doppler Processing

Georgia Tech Research Institute



- Threads map to pulse indices with a block for each range bin and channel pair
- No smem usage currently; could likely improve corner turn performance using smem as a staging area, but that kernel's performance is not a bottleneck
- FFTs performed via CUFFT

Implementation: Covariance Estimation



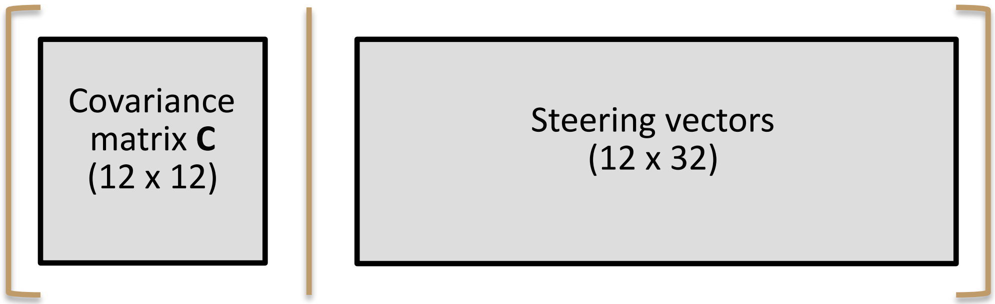
Implementation: Linear System Solver (1)

- We have Cholesky and Gauss-Jordan implementations; G-J is ~30% faster for our parameter set

Augmented system matrix

- Applying G-J to the augmented system matrix yields the identity matrix in place of **C** and the adaptive weights in place of the steering vectors

Implementation: Linear System Solver (2)

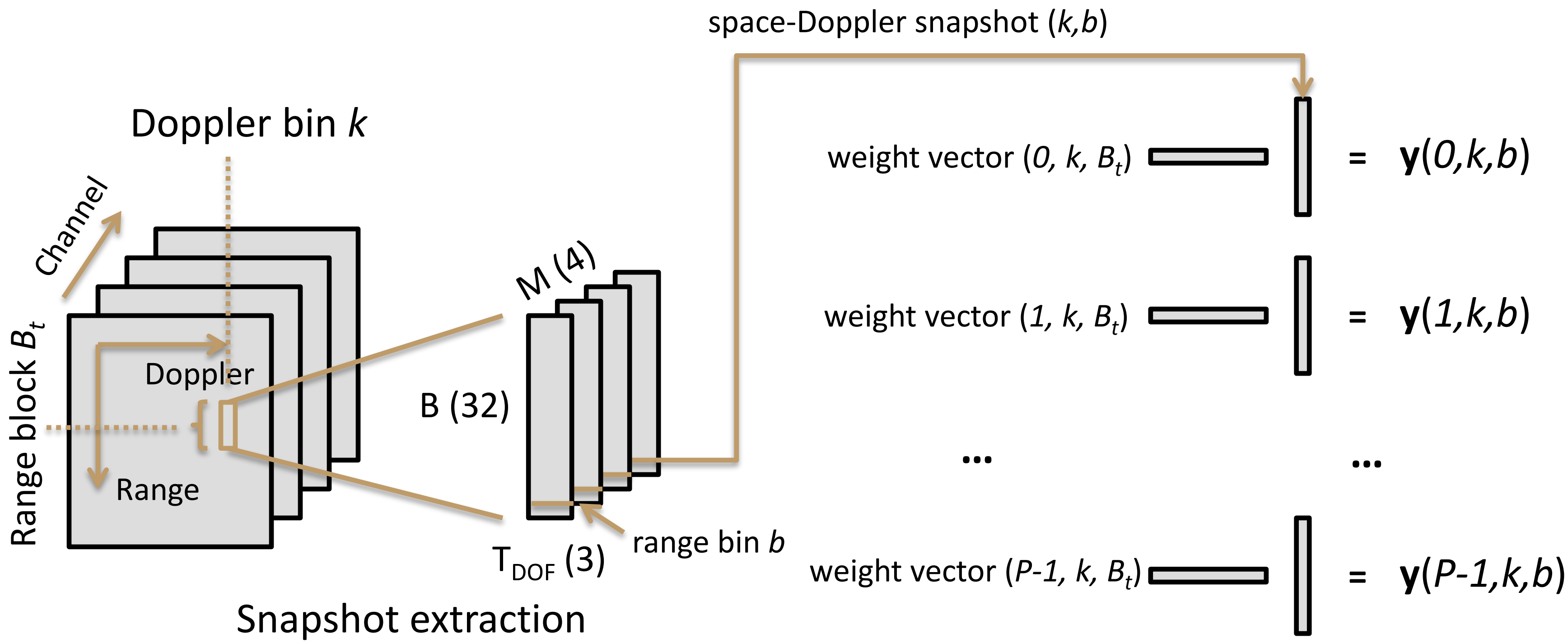


Augmented system matrix

- Load augmented system matrix into shared memory (4224 bytes)
- Each thread block notionally assigns one thread per element (528), but we add a blocking factor to manage multiple elements per thread
  - Optimal blocking factor determined empirically (3 in this case)
- No pivoting needed, so applying G-J elimination is straightforward
- Workload imbalance: lower diagonal entries in **C** become zero

Implementation: Weight Application

Georgia Tech Research Institute



- Each block includes *B* threads
- Steering vectors stored in shared memory; each thread applies all steering vectors to the same space-Doppler snapshot (producing *P* output values)
- *B* is a small block size, but enables storing the space-Doppler snapshot vector in registers

GPU Test Platforms

GPU Model	Peak FP32 GFLOPS	Peak Memory BW	TDP	Peak GFLOPS/W	Compute Capability
Tesla M2090	1331	155* GB/s	250W	5.32	2.0
Tesla K20c	3519	182* GB/s	225 W	15.64	2.1
Quadro Q3000M	432	80 GB/s	75 W	5.76	3.5

* ECC enabled, which reduces peak memory BW.

- All tests utilize driver version 310.44 with CUDA 5.0 on Linux
- Code is generated for the highest supported compute capability
- Timings are averaged over 32 data sets
- All code was originally tuned for the M2090 with no specific re-tuning for the K20c or Q3000M

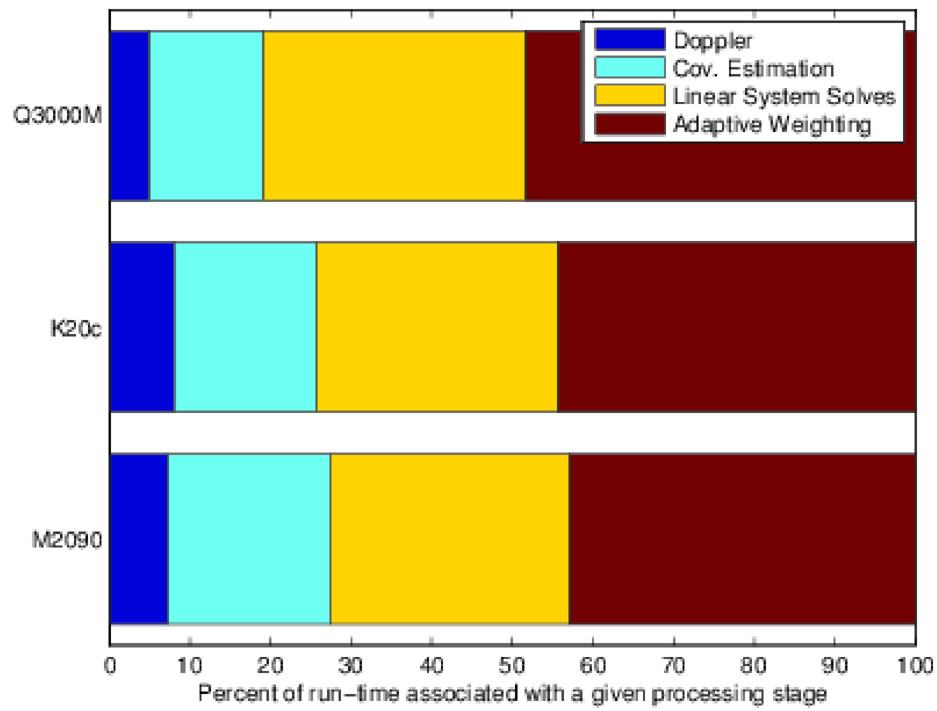
Absolute Performance Results

	M2090	K20c	Q3000M
Doppler Processing	0.30 ms	0.24 ms	0.80 ms
Covariance Estimation	0.82 ms	0.52 ms	2.24 ms
Linear System Solves	1.21 ms	0.88 ms	5.20 ms
Adaptive Weighting	1.75 ms	1.31 ms	7.69 ms
Total	4.07 ms	2.95 ms	15.93 ms
Relative Perf	0.7x	1.0x	0.2x

Absolute timing performance on the GPU test platforms.

Relative Performance Results

Georgia Tech Research Institute



The linear system solves and adaptive weighting are relatively more expensive on the Q3000M than the M2090/K20c.

Relative Power Efficiency Results

Georgia Tech Research Institute

- To estimate relative power efficiency, we use the thermal design power (TDP) as a surrogate for power consumption and compute data sets processed per second per Watt

	M2090	K20c	Q3000M
Data sets / second / W	0.98	1.51	0.84

The Kepler-generation hardware (K20c) offers ~1.5x better power efficiency than Fermi for this particular application.

Theoretical peak power efficiency for the K20c relative to the M2090 is 3x higher: 15.64 GFLOPS/W versus 5.32 GFLOPS/W.

Conclusions & Summary

Georgia Tech Research Institute

- Modern GPUs offer a compelling platform for STAP and are available in rugged form factors
- Shared memory utilization and our thread mapping strategies sensitize the linear system solver and adaptive weighting implementations to parameter changes
  - Such optimizations challenge cross-architecture perf portability
- The Kepler-generation hardware exhibited ~1.5x improved power efficiency relative to Fermi