Algorithm Architecture for Synthetic Aperture Radar (SAR) Ground Processing

Gary A. Mastin, Ph.D.
Lockheed Martin Management & Data Systems
Intelligence, Surveillance, and Reconnaissance Systems
Litchfield Park, Arizona
Overview

- SAR Processors - History
- Driving Algorithm Functions - Review
- Algorithm Architecture vs. Computer Architecture
- Discussion Question
In The Beginning…

GEMS Precision Optical Correlator
The Advent of Digital Electronics

HIRSADAP
• The advent of **Digital Image Processing** technology

• The problem:
  – We needed pictures of the moon’s surface for selecting landing sites
  – If the imaging spacecraft couldn’t return to earth, image capture by film wasn’t possible
  – Late 1960’s television technology was power hungry, heavy, and bulky, but the pictures were pretty good. (Yes, black & white images are good!!!!)
  – Size, power, and weight constraints limited what we could launch
  – We didn’t have the communications bandwidth to broadcast live video to the earth from the spacecraft
The solution:

– Send lower-quality cameras into space to meet the size, power, and weight constraints
– Characterize the camera deficiencies prior to launch
– Turn the video image into a grid of numbers representing intensity onboard the spacecraft. Buffer the data on board, then dump it over the communications link as fast as possible … preferably before crashing into the moon’s surface!
– Treat images like large mathematical matrices! Use computers on the ground to correct the camera deficiencies after data receipt.
– While we are at it, let’s also correct for contrast … and for motion blurs … and for perspective … and, hey, this is pretty powerful stuff!!!

Other applications

– Medicine
– Defense ➔ Synthetic Aperture Radar
Advent of the “Mini-Computer”

- The Digital Equipment Corporation (DEC) PDP-series made computing affordable
  - PDP 8, PDP 10
  - PDP 11/45 → Big step forward
    - ~256 KB of core memory
    - Video terminal for input instead of cards or paper tape
    - Attached disk, 10s of MB per disk pack (multiple platters)
    - 800 bpi 9-track tape for archive
    - RSX 11M operating system supported multiple tasks
    - Efficient DEC Fortran compiler, assembler, editor, linker, loader…
    - Interface to peripherals
      - Video monitors with disk buffers or even core. Dedicated image display functions!
      - Fixed-point and floating-point FFT hardware

- For $250,000 to $750,000, a department or a small company could have its own image processing system.
DEC VAX 11/780 – The Workhorse of 1980s
Systems like the VAX 11/780 were augmented with peripherals for SAR data input, algorithm processing, and display.
With the flexibility of programming in compiled languages came algorithm innovation → “Simple” Matter of Programming

**Nomenclature**

- **Forward**
- **Inverse**
- **Fourier Transforms**
- **DFC** Data Format Conversion
- **OFR** Output Format
- **Data I/O**
- **IPF** In-Plane Filter
- **CPF** Cross-Plane Filter
- **CPF** Complex Array Filter
- **Interpolation Filters**
- **RFG_CC** Reference Function Generators
- **RFG_CR**
- **CTM** Corner Turn
- **DET** Magnitude Detect
Modern Spotlight SAR Algorithm

Multiple Ingest Flow: Vector Subset Loop

Vector Subset Loop (continued)

Range Subset Loop (continued)

Remap Loop

Synthetic Target Phase History Generator

Syntarg

11 Synthetic Target Phase History Generator

Modern Spotlight SAR Algorithm

Multiple Ingest Flow: Vector Subset Loop

Vector Subset Loop (continued)

Range Subset Loop (continued)

Remap Loop

Synthetic Target Phase History Generator

Syntarg

Multiple Ingest Flow: Vector Subset Loop

Vector Subset Loop (continued)

Range Subset Loop (continued)

Remap Loop

Synthetic Target Phase History Generator

Syntarg
Key SAR Processing Functions

- Dechirp and Range Deskew

Key SAR Processing Functions

- **Radial Position Of Annulus Determined by Radar Center Frequency**
- **Collection Surface (Slant Plane)**
- **Annular Extent Of Data Annulus Determined by Collection Time**
- **Length of Annulus Determined by Radar Bandwidth**
- **Radar Depression Angle That Determines the Slant Plane**

Adapted from *Spotlight Synthetic Aperture Radar: A Signal Processing Approach* by Jakowitz, Wahl, Eichel, Ghiglia, and Thompson
Key SAR Processing Functions

- Polar Format Processing (Polar Reformatting)
Key SAR Processing Functions

- 2-D FFT

Contiguous Addresses

"N" samples/vector

Direction of 1-D FFT

"M" Vectors

Corner Turn (Transpose)

Contiguous Addresses

"M" samples/new vector

Direction of 1-D FFT

"N" New Vectors

Time
Key SAR Processing Functions

- Detect and Intensity Remap

\[ \sqrt{(R^2 + I^2)} \]
Algorithm vs. Computer Architecture

- The algorithm processing requirements USUALLY define the computer
  - Project/Program Requirements
    - Time to solution (throughput)
    - Data acquisition geometries & modes ➞ Range of data set sizes
    - Processing options in the baseline algorithm
  - Derived Requirements that Define the HW Architecture
    - Sustained/Peak FLOPS (floating point operations per second)
    - Main memory size
    - Processor to memory bandwidth
    - Memory to memory bandwidth
    - Disk I/O bandwidth
    - Processed and Unprocessed data archive size
Algorithm vs. Computer Architecture

Cost and Technology issues force compromises

- Can’t store the input and output data totally in main memory
  - Implies a multiple-ingest approach
  - Large data management implications
- Computation-bound. One CPU can’t handle the load.
  - Implies parallel processing, special purpose processors, or both
  - Perhaps exploit mathematical separability to improve efficiency
  - Further data management implications
- I/O-bound
  - Overlapped processing and I/O?
  - Parallel I/O streams?
  - Even greater data management implications
- Memory bandwidth-bound. Large corner turns are too slow.
  - Hardware architecture implications
  - Again, data management implications
Data management for the computer architecture is a significant algorithm complexity factor!

I can probably architect a dedicated system for SAR ground processing, but…

I don’t want to have different algorithms for different computer architectures

Is it possible to architect an algorithm for maximum portability?
- Lets explore the data management issues, then decide
Algorithm vs. Computer Architecture

- Multiple Ingest
  - First scenario (brute force)
  - Second (sequential) & Third (parallel) scenarios
ASU MAT 591: Opportunities in Industry!

Algorithm vs. Computer Architecture

Multiple Ingest Flow: Vector Subset Loop

Multiple Ingest Flow: Vector Subset Loop (continued)

Vector Subset Loop (continued)

Range Subset Loop (continued)

Range Subset Loop

Remap Loop

# For multiple ingest, this becomes an out-of-memory transpose.
@ Global. Based upon all data.
• Computation - Bound
  – Mathematical Separability
    ▪ Some 2-D tasks are performed more efficiently as separable 1-D tasks

Range Frequency Interpolation  Azimuth Frequency Interpolation
Algorithm vs. Computer Architecture

- **I/O – Bound**
  - Multiple options for overlapping Input, Processing, and Output
    - Sequential Buffering with Processing

- Overlapped I/O and Processing

```
<table>
<thead>
<tr>
<th>Input Memory Buffer</th>
<th>Algorithm Function</th>
<th>Output Memory Buffer</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>Input Memory Buffer A</th>
<th>Algorithm Function</th>
<th>Output Memory Buffer A</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>Input Memory Buffer B</th>
<th>Algorithm Function</th>
<th>Output Memory Buffer B</th>
</tr>
</thead>
</table>
```
- Memory Bandwidth – Bound
  - Distributed Memory Message Passing

**Exchange Algorithm**

Perform a local corner
Turn on each block

Do $I = 1, np-1$

\[ \text{myswap} = \text{XOR}(\text{me}, I) \]

Send block “myswap” on
PE “me” to PE “myswap”

Receive block “myswap” on
PE “me” from PE “myswap”

END DO
Discussion Question

- If I want to execute mathematically the same algorithm on the Network Computers that is executed on the Production Computer….
- And if I want to minimize the number of software implementations of the algorithm for cost savings…
- Then how should I design my algorithm architecture?
Discussion Question

- Lets consider the problem in pieces
  - How will I use memory efficiently if one computer has a native word length of 64 bits and the others have a native word length of 32 bits?
  - What are the data management issues when the entire input and output data will not fit into main memory?
    - Consider non-square input phase history
    - Remember that we are performing mixed-radix 1-D FFTs (2,3,5,7)
    - What impact does implementation on a distributed-memory message-passing architecture have on memory management?
  - How will we perform an out-of-core transpose on a shared memory computer …
    - If the computer has one processor?
    - If the computer has multiple processors working simultaneously on different parts of the data set (multiple ingest)?
Conclusion

- Hopefully, you can see that creating an architecture-independent transportable algorithm is a daunting challenge.
- Hopefully, you understand that addressing this problem early can cost a lot of money, but over time could save large amounts of money in software development and maintenance.
- Solving this problem can build customer confidence that your software produces exactly the same result regardless of the computing platform.