Plume Tracker: Interactive Mapping of Atmospheric Plumes via GPU-based Volumetric Ray Casting

Alexander (*Lex*) *Berk, Chona S. Guiang,*
Rosemary G. Kennett and Tim C. Perkins
Spectral Sciences, Inc. (SSI), *Burlington, MA*
lex@spectral.com (www.spectral.com)

Vincent J. Realmuto
Jet Propulsion Laboratory (JPL), *Pasadena, California*
vincent.j.realmuto@jpl.nasa.gov

Jet Propulsion Laboratory
California Institute of Technology

Technical Point of Contact: *Nikunj C. Oza*
NASA Ames Research Center (ARC), *Moffett Field, California*
nikunj.c.oza@nasa.gov

Earth Science Technology Office

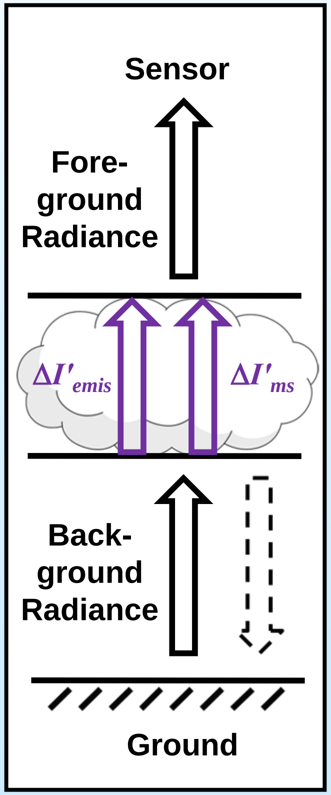
2012 HyspIRI Science Workshop
Washington, DC
16-18 October 2012

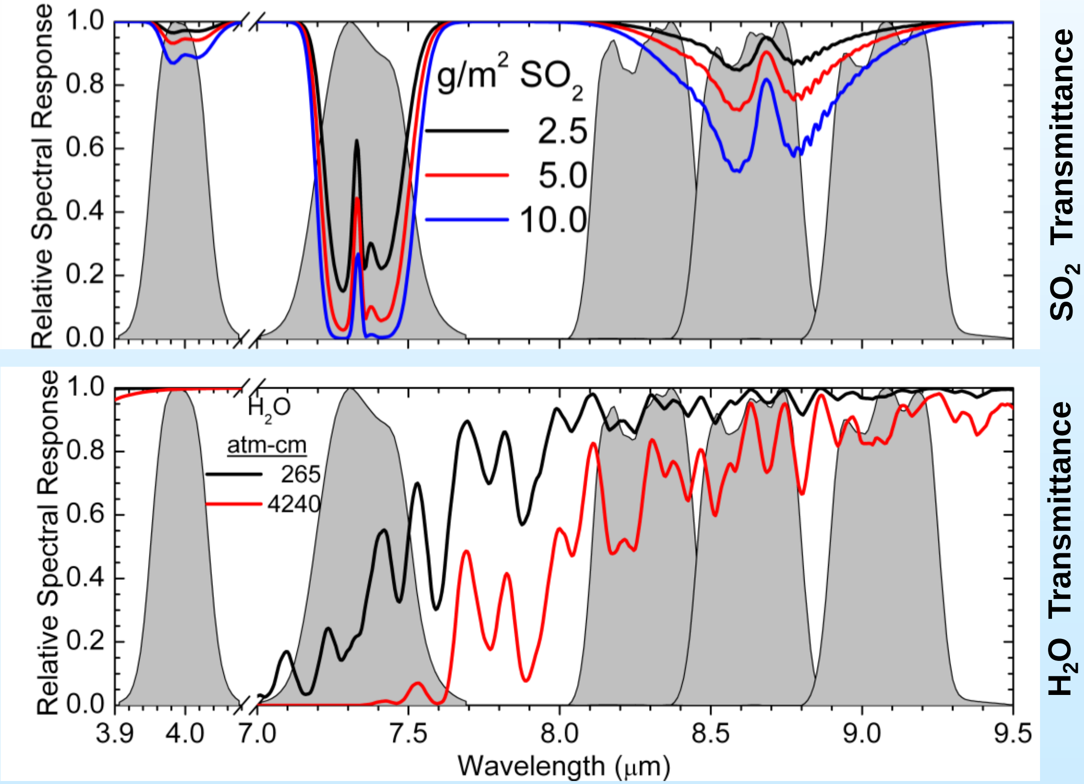
Presentation Outline

- Background and Problem
- Program Objectives and Approach
  - Plume Tracker / MODTRAN® Overview
- Upgrading Plume Tracker with Current MODTRAN
- MODTRAN Profiling and GPU Strategy
- Summary

Background: 2010 HyspIRI Workshop Presentation

- Described a new capability to model local chemical plumes within the MODTRAN atmospheric radiative transfer model
- Discussed applications to Volcanic SO_2
- Led to an ESTO Advanced Information Systems Technology Effort





The Problem

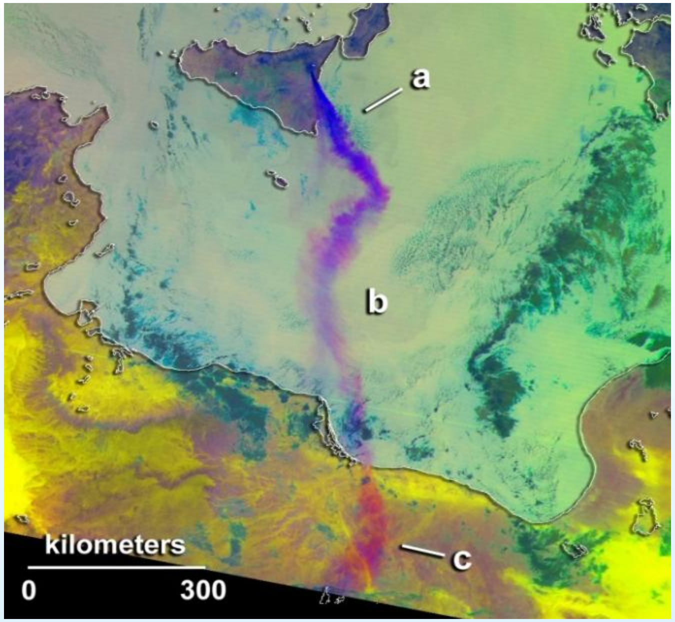
- Timely quantification of volcanic releases is important for
  - Signaling an **impending eruption** via SO_2 monitoring,
  - Assessing impact on **climate forcin**g from volcanic aerosols, and
  - Insuring safe and efficient commercial **aviation** in the presence of ash clouds.
- The JPL Plume Tracker (formerly MAP_SO2) toolkit accurate characterizes volcanic effluents from satellite-based Thermal Infrared (TIR) spectral imagery data
- The bottleneck in Plume Tracker processing is the computationally intensive but radiometrically accurate MODTRAN calculations
  - Plume Tracker uses an old version of MODTRAN (Mod3.5, circa 1997)
  - The current version of MODTRAN (Mod5.3) is more accurate, but slower!
    - Correlated-*k* algorithm added to improve accuracy of scattering calculations
    - A 0.1 cm^{-1} spectral resolution added for narrow band spectrometers
    - First-principles Voigt spectral-bin transmittance required for 0.1 cm^{-1} option
    - Upgrade from 2 to 4 parameter band model for 9.6 μm O_3 band
    - Local Chemical Plume capability

Program Objectives

3 Year Effort – 01april2012 Start Date

Jet Propulsion Laboratory
California Institute of Technology

- Timely quantification of volcanic releases from analysis of satellite-based TIR spectral imagery data
- Develop a GPU implementation of MODTRAN's TIR radiance algorithms
- Integrate the MODTRAN accelerated modules into Plume Tracker for retrieving and mapping the 3-D composition of atmospheric plumes using JPL established retrieval algorithms

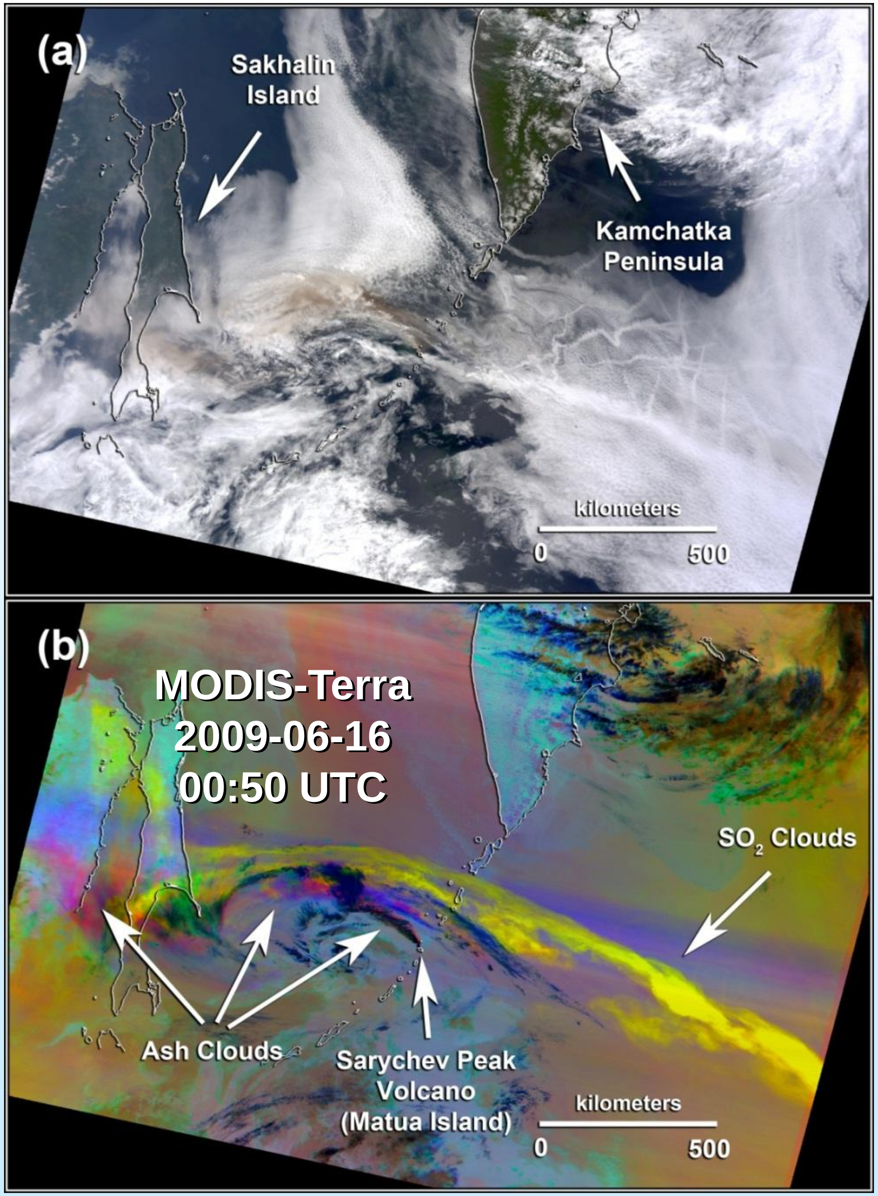


- Performance goals are 100-fold speed up of radiative transfer calculations vs. JPL's current model
- Plume Tracker will permit near real-time visualization of the impact of changes in model parameters for scenes like the MODIS-Aqua image of the Mt. Etna Eruption Plume

Plume Tracker: *Interactive Mapping of Volcanic Emissions with Radiative Transfer Modeling*

Jet Propulsion Laboratory
California Institute of Technology

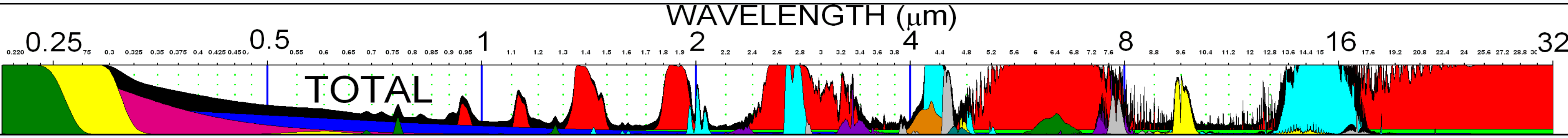
- Analysis of Data from Multiple Instruments, with Ancillary Data from Multiple Sources
- Graphic User Interface
  - Import Image and Ancillary Data
  - Specify Parameters for RT Modeling
  - Visualize Input Data & Retrieval Results
- MODTRAN3.5 serves as the Radiative Transfer Model Workhorse
  - Optimized for TIR Modeling
  - Hash Table Minimizes Calculations
  - Portable Component Architecture
- Retrieval Procedures
  - Surface Temperature and Emissivity
  - Total Column SO_2, H_2O Vapor, and O_3
  - Optimized for 2-Component Retrievals



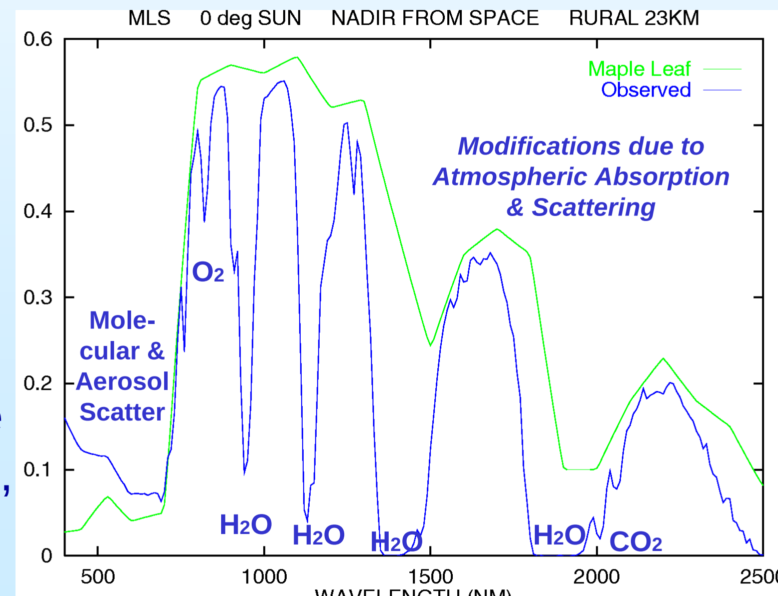
MODTRAN5 Overview

*Now **Free** to US Govt and their Contractors*

- IR / Vis / UV Transmittances, Radiances and Fluxes
  - Up to 0.2 cm^{-1} Spectral Resolution from 0.1 cm^{-1} Band Model



- Stratified Molecular / Aerosol / Cloud Atmosphere
  - Built-in, Auxiliary & User-Specified Molecules and Particulates
- 2-Stream and DISORT Solar and Thermal Scattering
  - Diffuse Transmittance and Spherical Albedo for Atmosphere Correction →

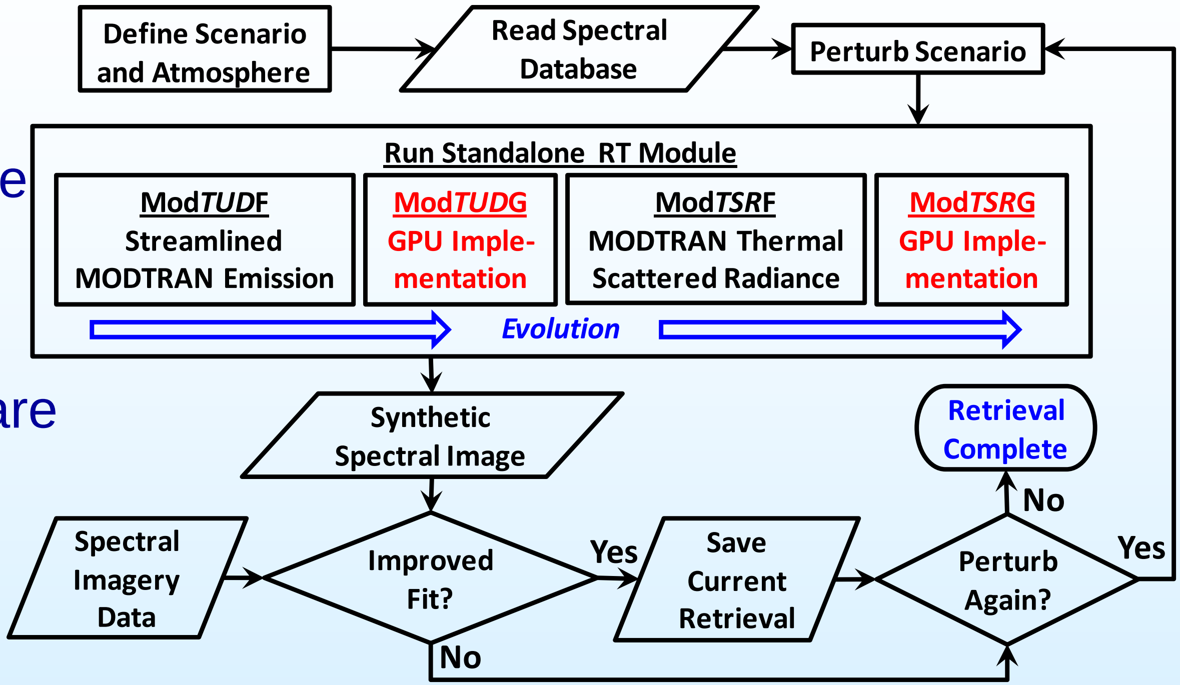


- Spherical Refractive Geometry
- Spectral Convolution & Filtering
- Many Applications Pertinent to HyspIRI
  - Atmospheric Correction/Retrieval, Scene Simulation, Measurement/Data Analyses, Sensor Design, Algorithm Development, Localized chemical clouds modeling, …

Approach

Jet Propulsion Laboratory
California Institute of Technology

1. Design architecture for GPU implementation of MODTRAN TIR radiance
2. Update Plume Tracker GUI for interfacing Mod53 with GPU software
3. Streamline MODTRAN TIR emission algorithm
4. Implement emission algorithm on a GPU with Plume Tracker
5. Streamline MODTRAN TIR scattering algorithm
6. Implement scattering algorithm on a GPU with Plume Tracker
7. Verify GPU MODTRAN/Plume Tracker against serial models (SSI)
8. Validate GPU Plume Tracker against Data (JPL)



Effort Requirements and Desired Enhancements

Jet Propulsion Laboratory
California Institute of Technology

- MODTRAN special function routines provide >13 place accuracy
  - ➢ Faster and/or GPU-tailored routines of radiometric accuracy
- Chemical (SO_2) releases modeled as infinite layer (spherical shell)
  - ➢ Implement local chemical option
- Plume Tracker is based on MODTRAN**3.5** (circa 1997)
  - ➢ Transition from Mod35 ⇒ Mod53

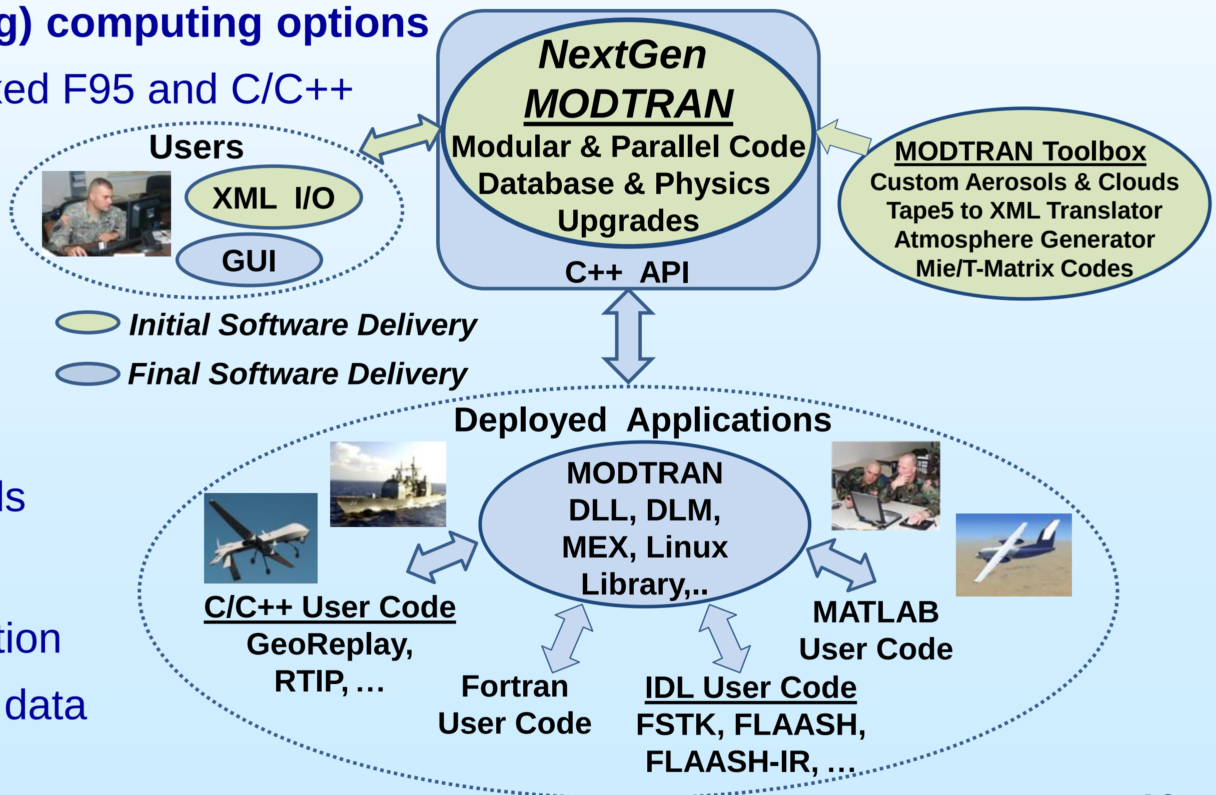
Requirements

  - ➢ Transition from Mod53 ⇒ *NextGen MODTRAN*
- MODTRAN is run multiple times per pixel (does use hash table)
  - ➢ Further reduce number of runs
- GPU version of 2-Stream Thermal Scatter
  - ➢ GPU version of DISORT-Thermal
- Band Model Radiative Transfer
  - ➢ Correlated-k Radiative Transfer

Desired Enhancements

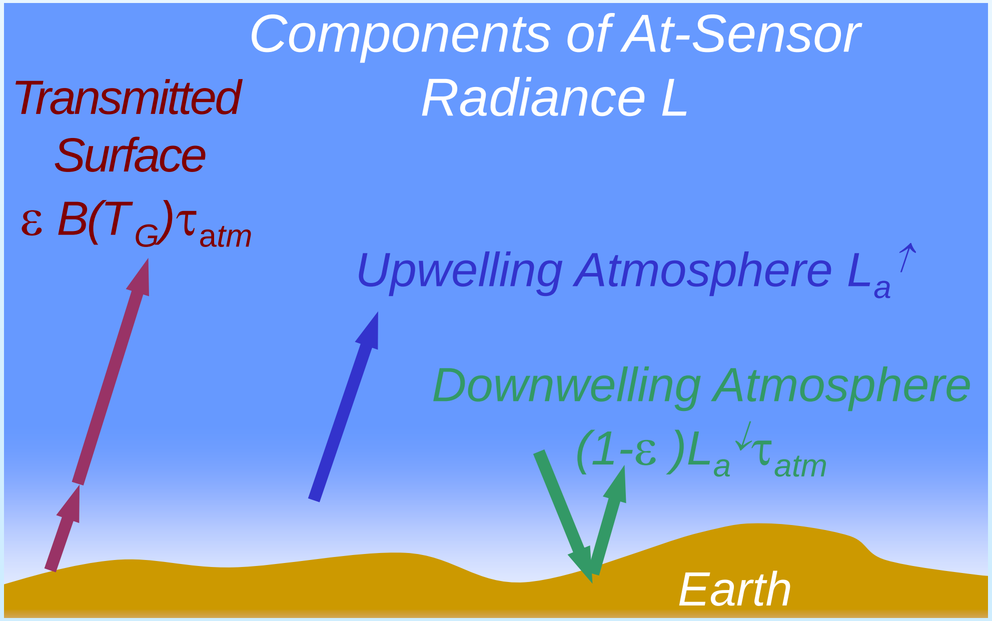
Leveraging AFRL Rapid Innovation Funding (RIF) *Program: NextGen MODTRAN* (01 Oct 2012 Start Date)

- Primary Focus: **Modernization of MODTRAN Software**
  - Retain the full functionality of MODTRAN
  - Introduce modern architecture to insure accessibility to scientists / engineers
  - Modularize MODTRAN components
  - C/C++ I/O and interface with Graphical User Interface (GUI
  - **Parallel (multi-threading) computing options**
  - Upgrade from F77 to mixed F95 and C/C++
  - Software Documentation
- MODTRAN Toolbox
  - Atmosphere generator
  - Tape5 to XML translator
  - Scattering models
  - Custom aerosols & clouds
- Physics upgrades
  - Line-by-line (LBL) RT option
  - Updated HITRAN-based data



PlumeTracker Interface to MODTRAN

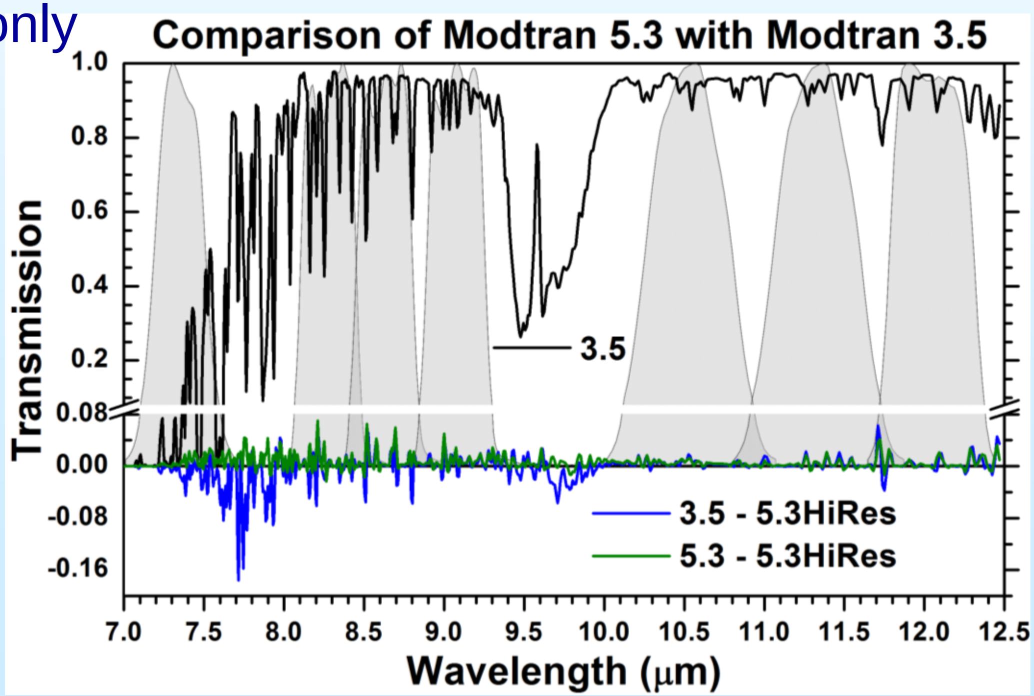
- PlumeTracker (Map_SO2) uses MODTRAN**3.5** (circa 1997)
  - JPL created MODTRAN**3.5** API interfaced through a Mod**35**.dll file
  - I/O passed in argument list including inputs not standard in 1997
    - Molecular profiles scaling factors
    - Surface spectral albedo
  - Spectral ***TUD*** outputs stored in dynamically-allocated arrays
    - ***T***ransmitted surface emission
    - ***U***p-welling path radiance
    - ***D***own-welling flux reflected by surface, transmitted to sensor
- Includes IDL front end GUI
  - Drives retrieval algorithm
  - C/C++ interfaces Mod**35**.dll
  - Multiple MODTRAN runs for each image pixel
  - Hash table



Updating PlumeTracker to Modtran5.3

Jet Propulsion Laboratory
California Institute of Technology

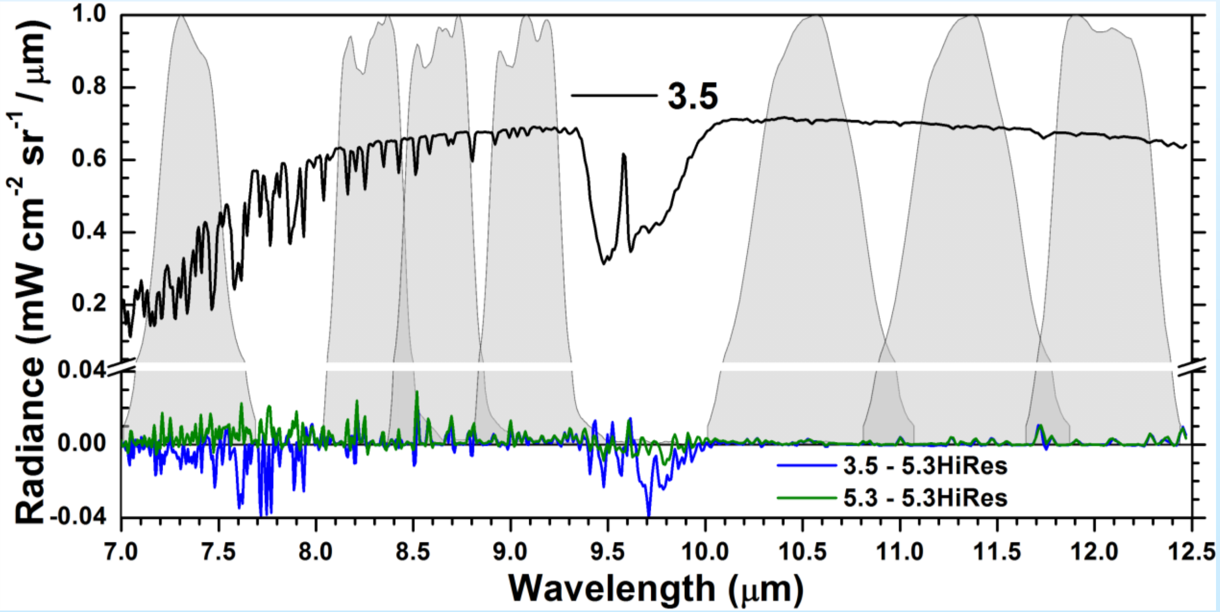
- I/O arguments were left unchanged in initial update
- Replacement Mod**53**.dll file produced and being tested
  - Stand-alone and integrated results compared to Mod**35**.dll
  - More testing required
- Entry routine arguments to be pared down
  - Current or potential inputs only
  - Will ease maintenance and simplify documentation
- Plan transitioning to NextGen MODTRAN
  - Standardize API
  - Multi-threading capability
  - Enhanced performance



3.5 vs 5.3 Radiances

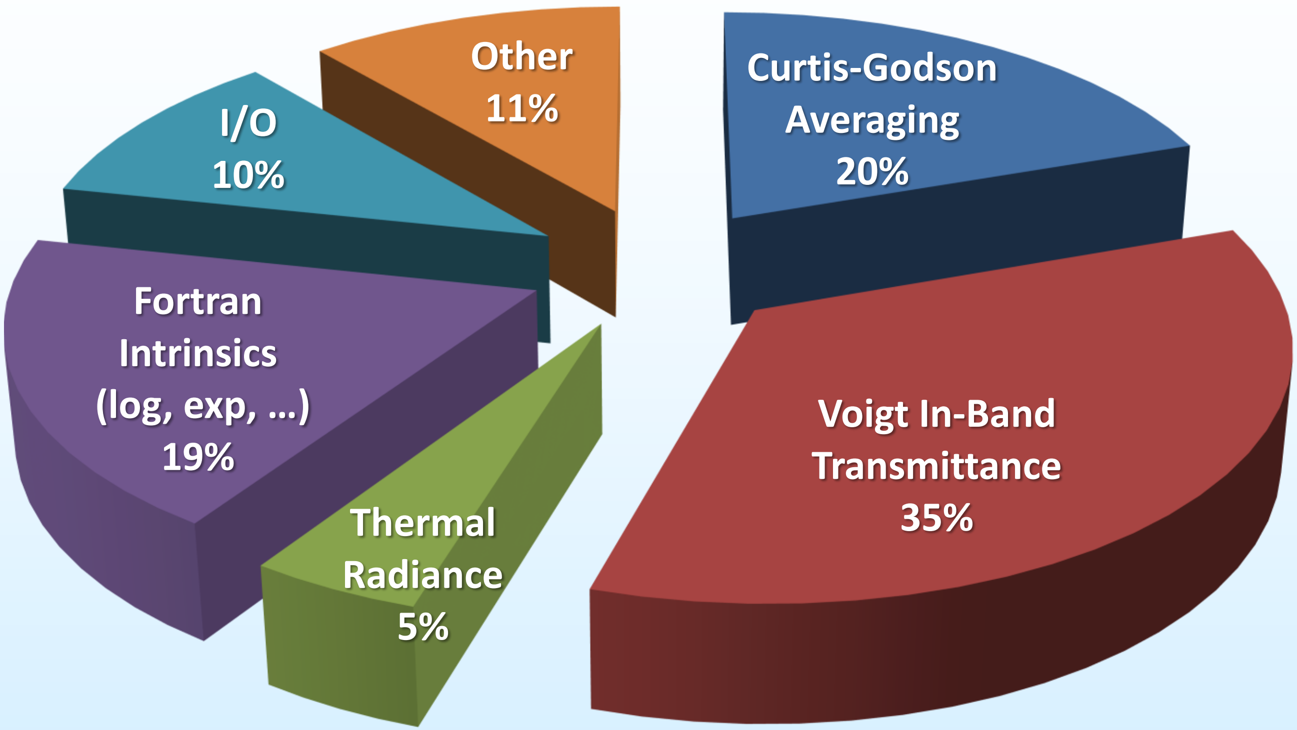
Jet Propulsion Laboratory
California Institute of Technology

- Scenario
  - 45º off-nadir from 20.4 km
  - Pacific in April radiosonde
  - 380 ppmV CO_2, 80 km visibility
  - 2 km altitude ground, 0.9 emissivity
- Updated Model Elements
  - HITRAN 2009 Band Model (BM) Data
  - 4 parameter band model, Padé fit tails
  - Voigt in-band transmittance
- Upgraded Thermal Radiance Method
  - DISORT 8-Stream scattering
  - Statistical Correlated-*k* Algorithm
  - 0.1 cm^{-1} band model
- Results
  - 0.5 sec (5.3), 58 sec (5.3 HiRes)
  - 4-parameter BM improves 9.6μm O_3
  - Updated transmittance algorithm and 1997 ⇒ 2009 data reduce residuals



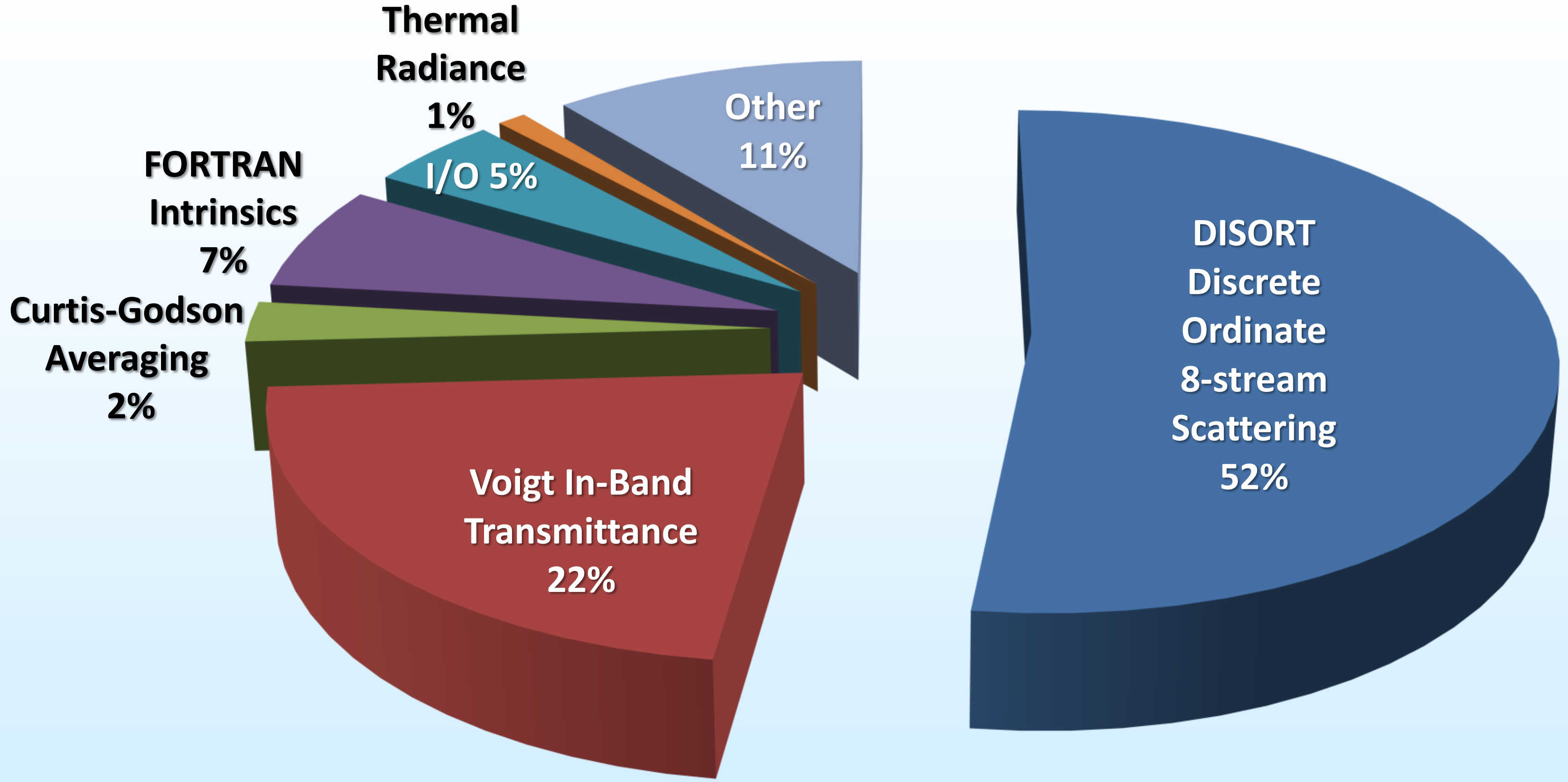
Profiling of MODTRAN Thermal Emission and **2-Stream** Scattering

Jet Propulsion Laboratory
California Institute of Technology



- Curtis-Godson Averaging is a precursor to transmittance
  - Curtis-Godson not used with Correlated-k algorithm
- Current transmittance calculations bogged down by Bessel function and Voigt line shape evaluations

*Profiling of MODTRAN Thermal Emission and **DISORT** Scattering*

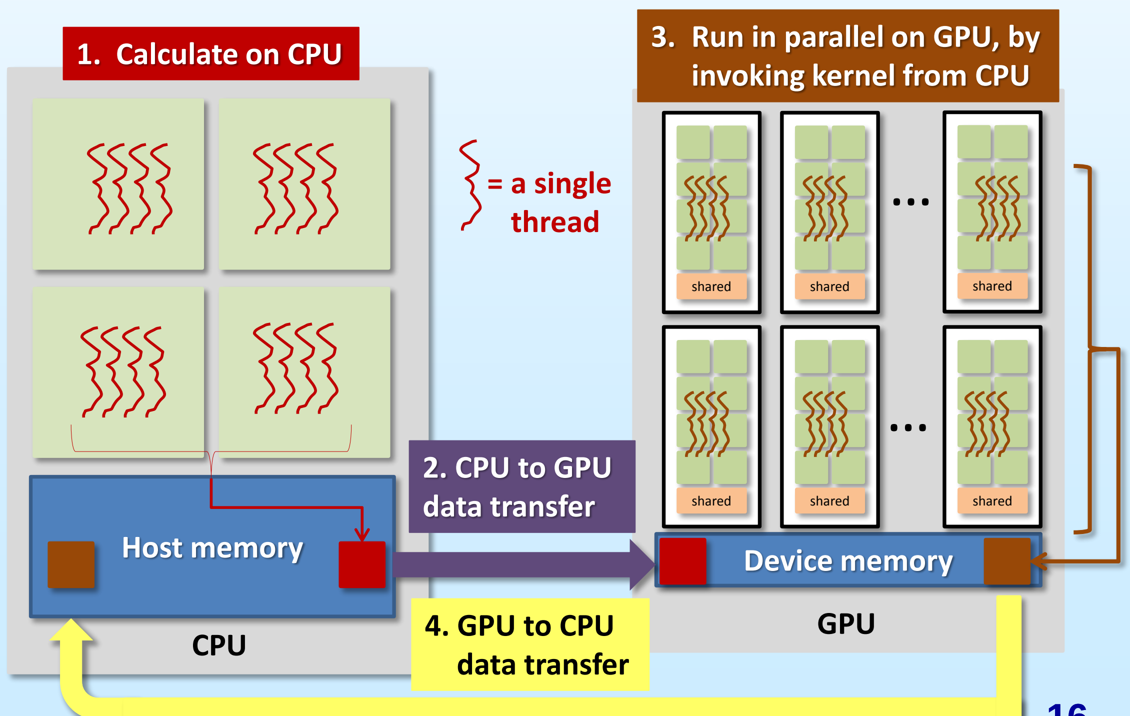


DISORT is bottleneck when used for thermal multiple scatter

- Discrete ordinate method is a matrix formulation
- GPU linear algebra libraries are readily available

Parallel RT Architecture

- Convert RT code functions to maximize use of GPU parallel resources
  - GPU hardware favors independent calculation pathways & sequential memory access
  - Trends in GPU hardware moving towards higher floating-point throughput per thread
  - Fewer threads, more computation per thread
- Re-organize sequential steps of numerical routines into independent vectorized calculations
  - Focus on "loop"-based operations spanning a frequency/molecular grid
- Architecture changes will occur internal to MODTRAN, localized to the most computationally expensive routines
  - Matrix transforms, Voigt line shape, Bessel functions, etc.
- Parallel vector operations are implemented with GPU programming libraries (NVIDIA CUDA-based)

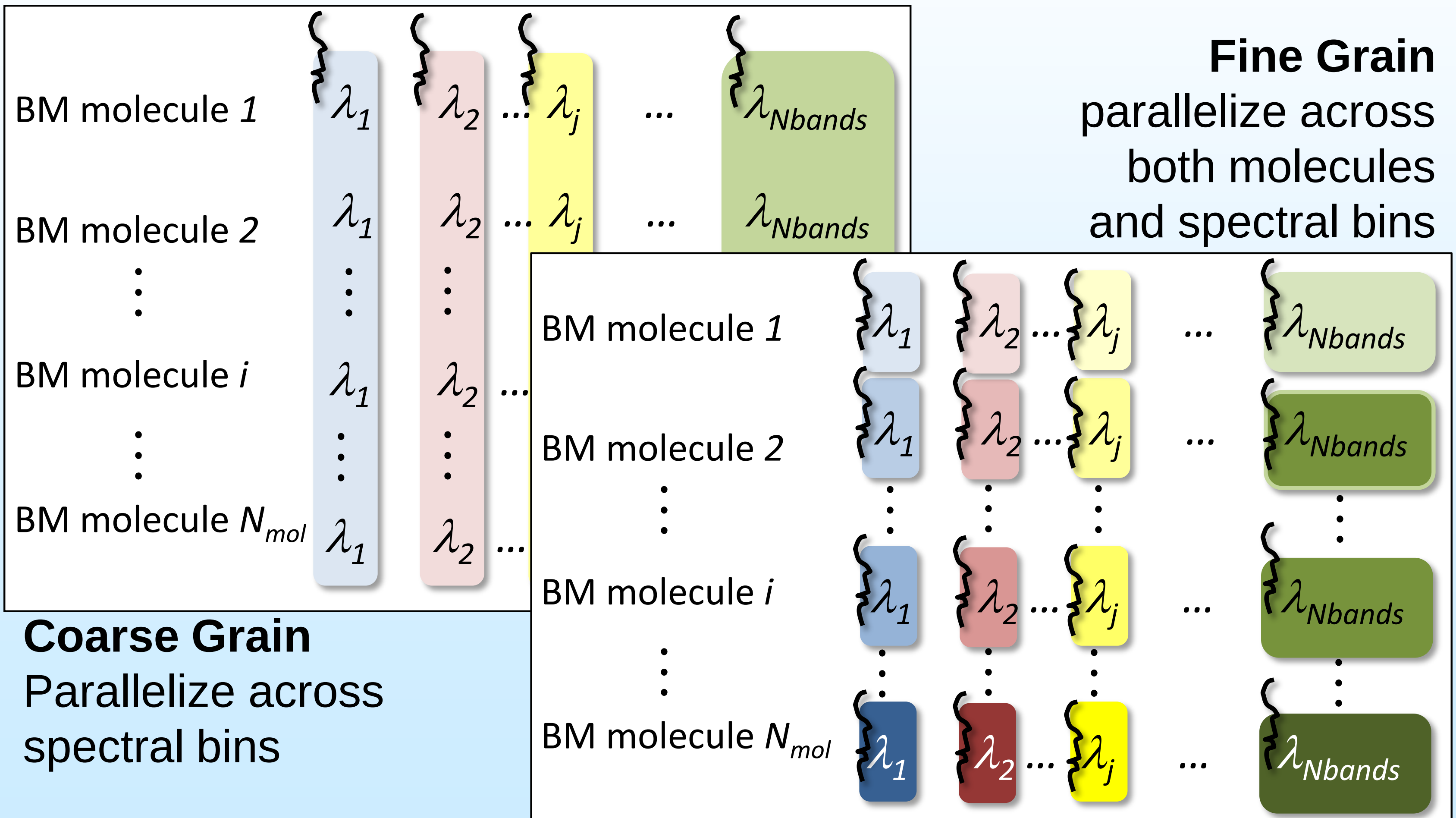


Preliminary GPU Parallelization

Jet Propulsion Laboratory
California Institute of Technology

- Initial GPU thread workload: Voigt and Bessel functions
  - Future GPU thread workload: Voigt In-Band Transmittance

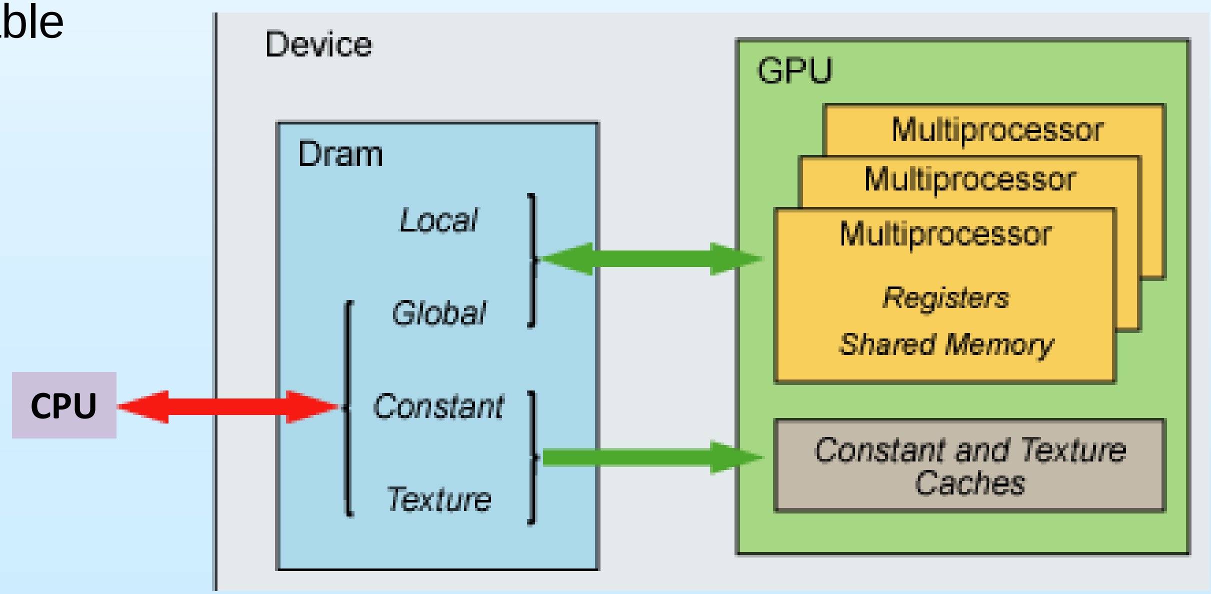
Fine Grain
parallelize across both molecules and spectral bins



Coarse Grain
Parallelize across spectral bins

Preliminary GPU porting strategies

- Restructure code and data layout being optimized for GPU
- Data storage on GPU
  - Band model data stored in (read-only) constant memory
    - 64K constant memory per GPU
    - *< 1K band model data per active-molecule/spectral-bin*
    - Ensure adjacent threads within block access adjacent memory
    - If necessary, texture memory also available
  - Registers used for Voigt in-band transmittance
  - Transmittance arrays saved in shared memory



Summary: Progress to Date

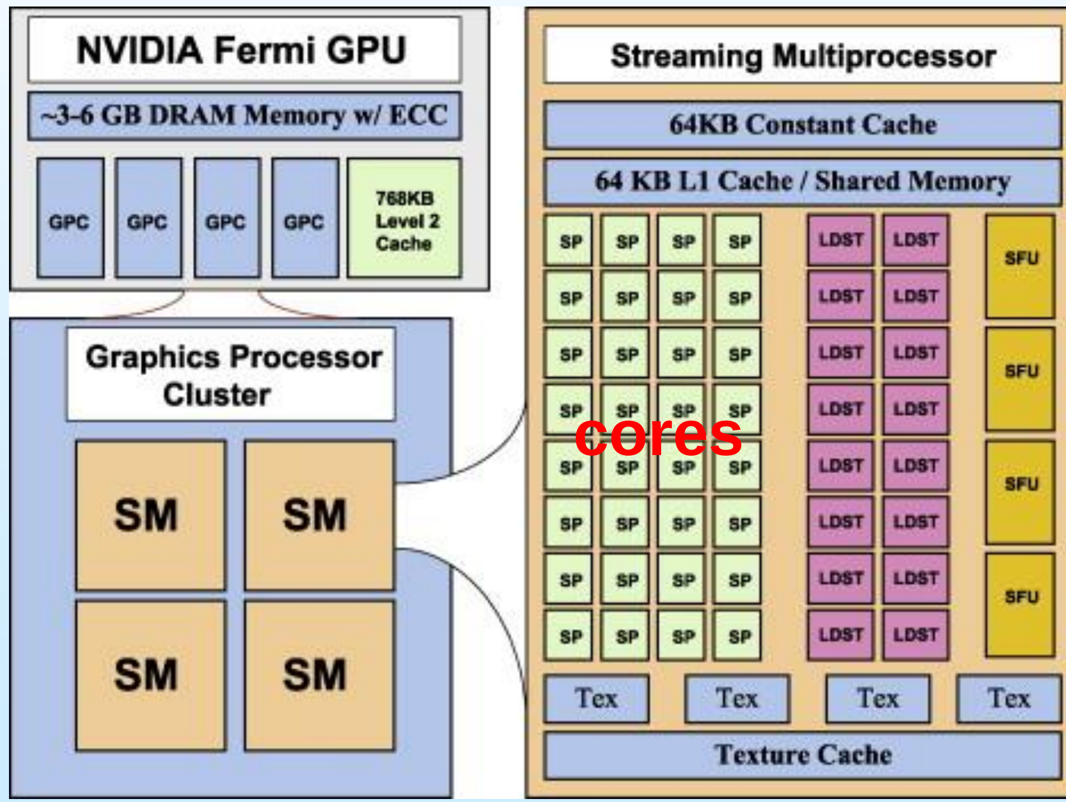
- Documentation
  - Tutorial and Algorithm Theoretical Basis Document (ATBD)
  - Voigt In-Band Transmittance Paper (submitted to JQSRT)
  - Details of MODTRAN / DISORT Integration
- Integration of MODTRAN5.3 in Plume Tracer
  - Initial integration complete
  - Undergoing testing
- GPU implementation
  - Time Profiled Plume Tracker runs
  - Identified Bottlenecks
  - Initiated testing of GPU tailored functions
- Please stay tuned!
  - **www.modtran5.com or lex@spectral.com**

Back-Ups Follow

Jet Propulsion Laboratory
California Institute of Technology

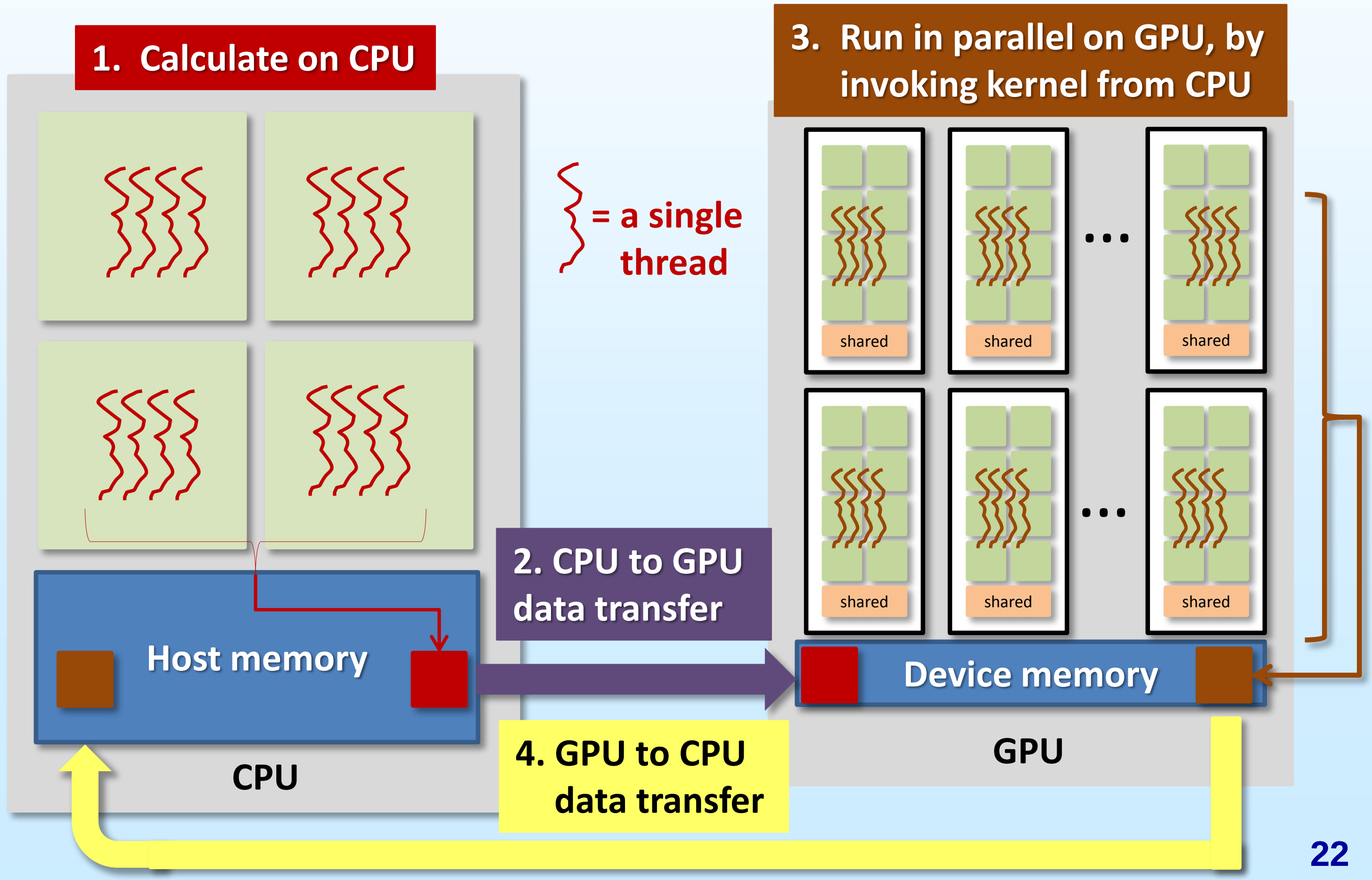
GPU Hardware

- A GPU consists of
  - Multiple Streaming-Multiprocessors (SM's), and
  - 3-6 GB Dynamic RAM global (or device) memory visible to all SMs
- Each SM consists of multiple cores that have access to
  - a common pool of shared memory and cache (in newer GPUs)
  - very fast registers with limited memory
- Constant and texture memory provide fast, cached storage for the SM.



CUDA Process Flow

"CUDA is a parallel computing platform and programming model that enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU)"



General GPU performance considerations

- Memory
  - Minimize data transfers between the CPU and device
  - Access latency hierarchy: *global memory > shared memory > registers*
  - Coalesce global memory access across threads in a block to maximize bandwidth
  - Maximize FLOPS-to-memory access ratio
- Multi-threaded execution
  - Avoid divergent execution by grouping identical tasks in a thread block
  - Ensure load-balanced workload
  - Overlap thread data transfer and execution by using asynchronous data transfer calls