Very Large Dataset Access and Manipulation: Active Data Repository (ADR) and DataCutter

Joel Saltz

Alan Sussman

Tahsin Kurc

University of Maryland, College Park

and

Johns Hopkins Medical Institutions

http://www.cs.umd.edu/projects/adr

Research Group

- **University of Maryland**
  - Charlie Chang
  - Renato Ferreira
  - Mike Beynon
  - Henrique Andrade
- **Johns Hopkins Medical Institutions**
  - Umit Catalyurek

Irregular Multi-dimensional Datasets

- **Spatial/multi-dimensional multi-scale, multi-resolution datasets**
- **Applications select portions of one or more datasets**
- **Selection of data subset makes use of spatial index (e.g., R-tree, quad-tree, etc.)**
- **Data not used “as-is”, generally preprocessing is needed - often to reduce data volumes**

Querying Irregular Multi-dimensional Datasets

- **Irregular datasets**
  - Think of disk-based unstructured meshes, data structures used in adaptive multiple grid calculations, sensor data
    - indexed by spatial location (e.g., position on earth, position of microscope stage)
  - Spatial query used to specify iterator
    - computation on data obtained from spatial query
    - computation aggregates data - resulting data product size significantly smaller than results of range query

Application Scenarios

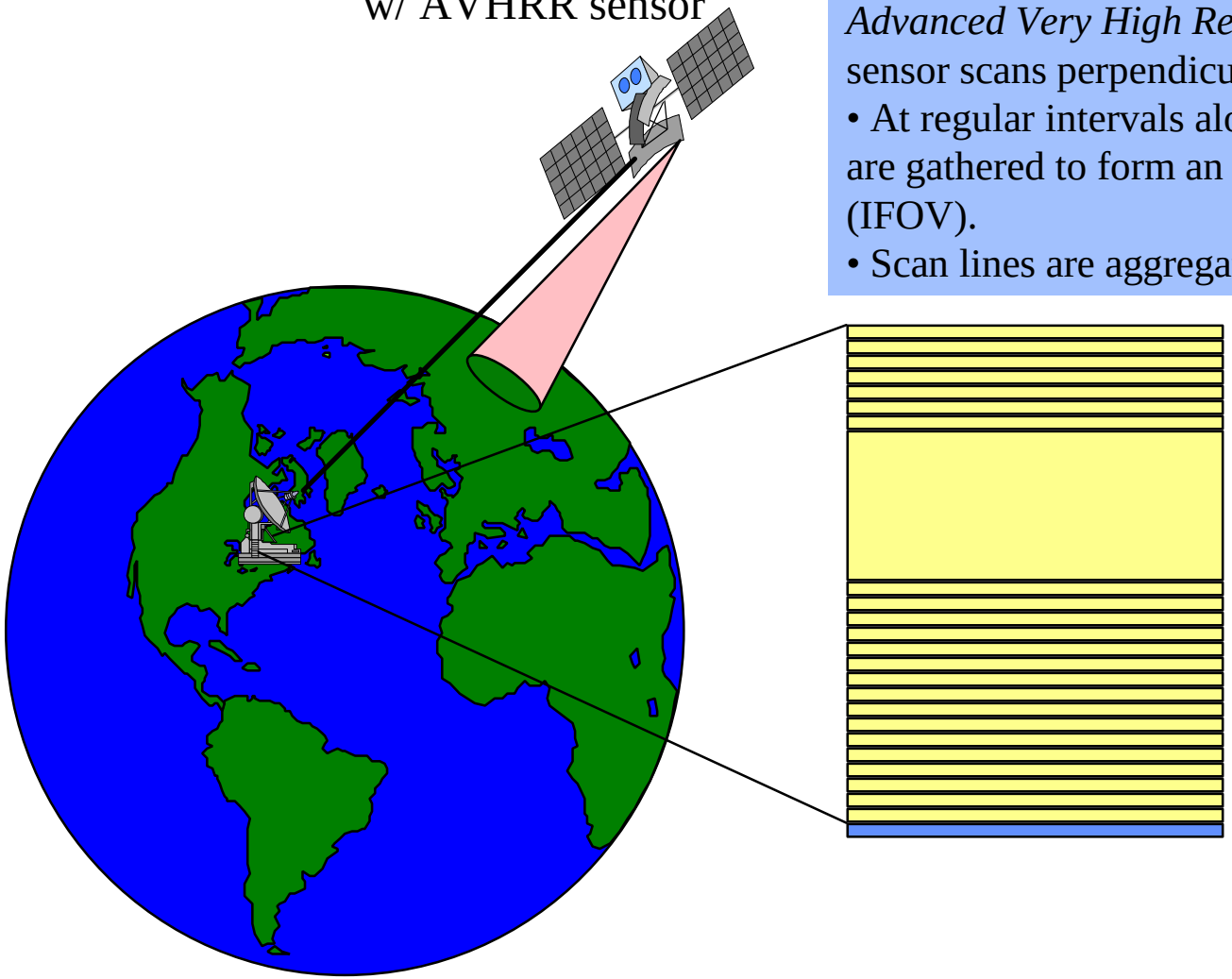
- **Ad-hoc queries, data products from satellite sensor data**
- **Sensor data, fluid dynamics and chemistry codes to predict condition of waterways (e.g. Chesapeake bay simulation) and to carry out petroleum reservoir simulation**
- **Predict materials properties using electron microscope computerized tomography sensor data**

Application Scenarios (cont.)

- **Browse or analyze (multi-resolution) digitized slides from high power light or electron microscopy**
  - 1-50 GBytes per digitized slide - 1000’s of slides per day per hospital
- **Post-processing, analysis and visualization of data generated by large scientific simulations**

Processing Remotely Sensed Data

NOAA Tiros-N
w/ AVHRR sensor



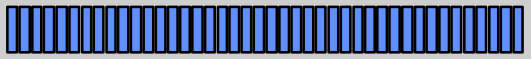
AVHRR Level 1 Data

- As the TIROS-N satellite orbits, the *Advanced Very High Resolution Radiometer* (AVHRR) sensor scans perpendicular to the satellite's track.
- At regular intervals along a scan line measurements are gathered to form an *instantaneous field of view* (IFOV).
- Scan lines are aggregated into Level 1 data sets.

A single file of *Global Area Coverage* (GAC) data represents:

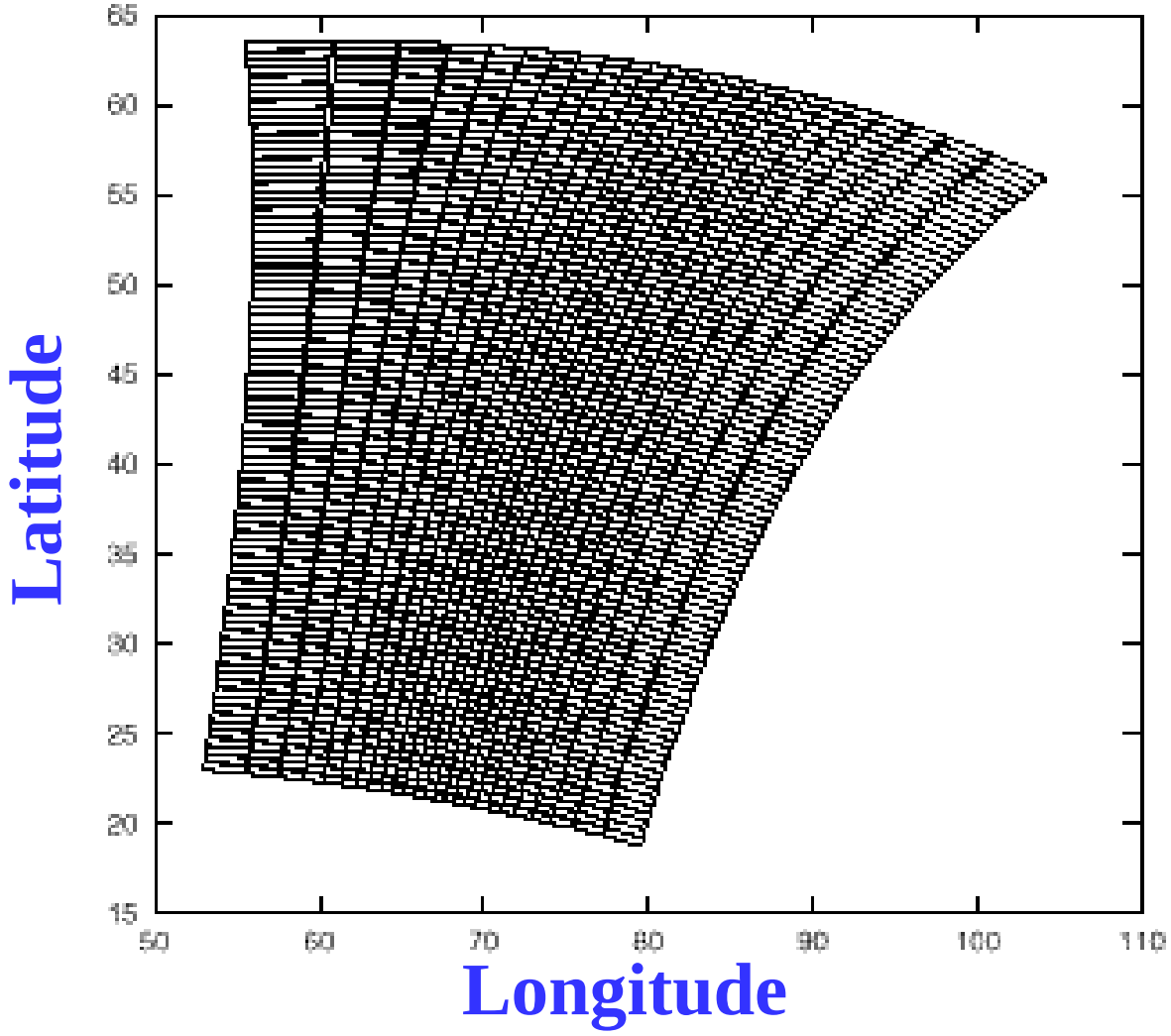
- ~one full earth orbit.
- ~110 minutes.
- ~40 megabytes.
- ~15,000 scan lines.

One scan line is 409 IFOV's



Spatial Irregularity

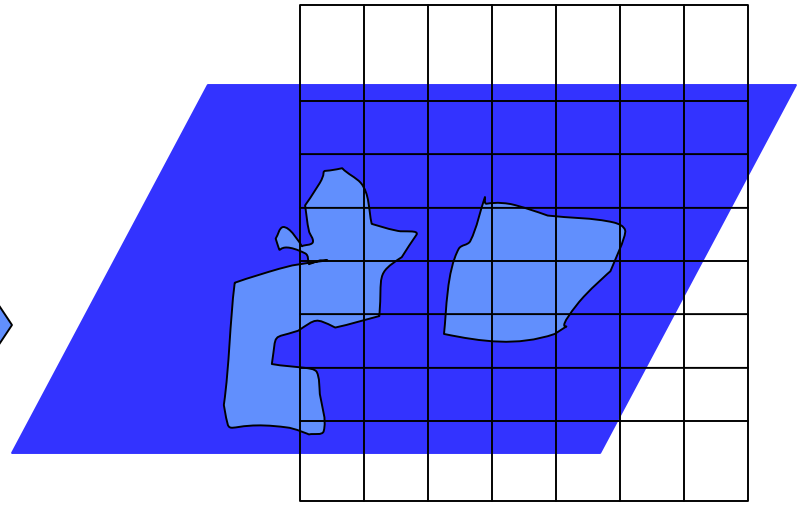
AVHRR Level 1B NOAA-7 Satellite 16x16 IFOV blocks.



Typical Query

Output grid onto which a projection is carried out

Specify portion of raw sensor data corresponding to some search criterion



Application Processing Loop

O ← Output dataset, I ← Input dataset

A ← Accumulator (intermediate results)

$[S_I, S_O]$ ← Intersect(I, O, R_{query})

foreach o_e in S_O do

    read o_e

    a_e ← Initialize(o_e)

foreach i_e in S_I do

    read i_e

    S_A ← Map(i_e) ∩ S_O

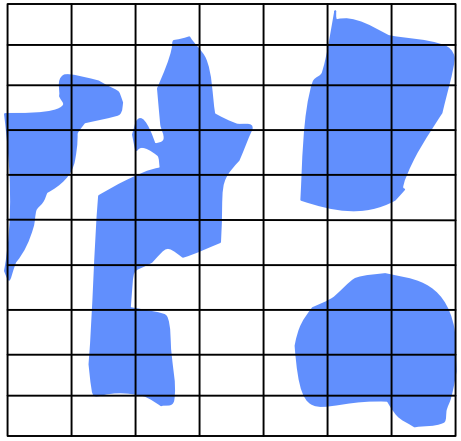
    foreach a_e in S_A do

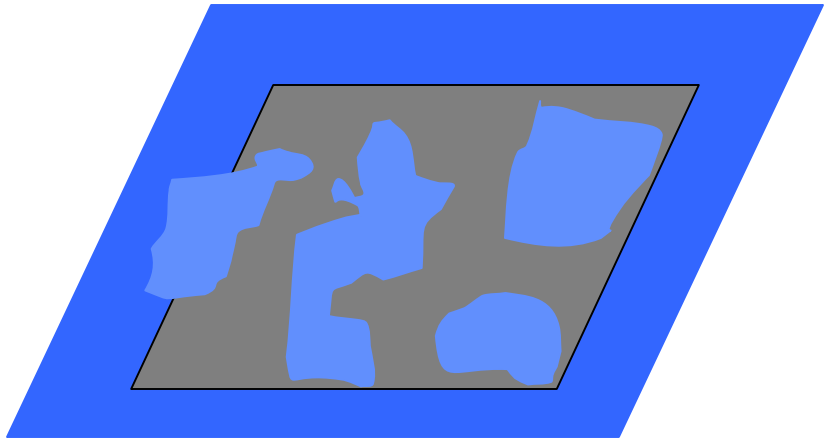
        a_e ← Aggregate(i_e, a_e)

foreach a_e in S_O do

    o_e ← Output(a_e)

    write o_e





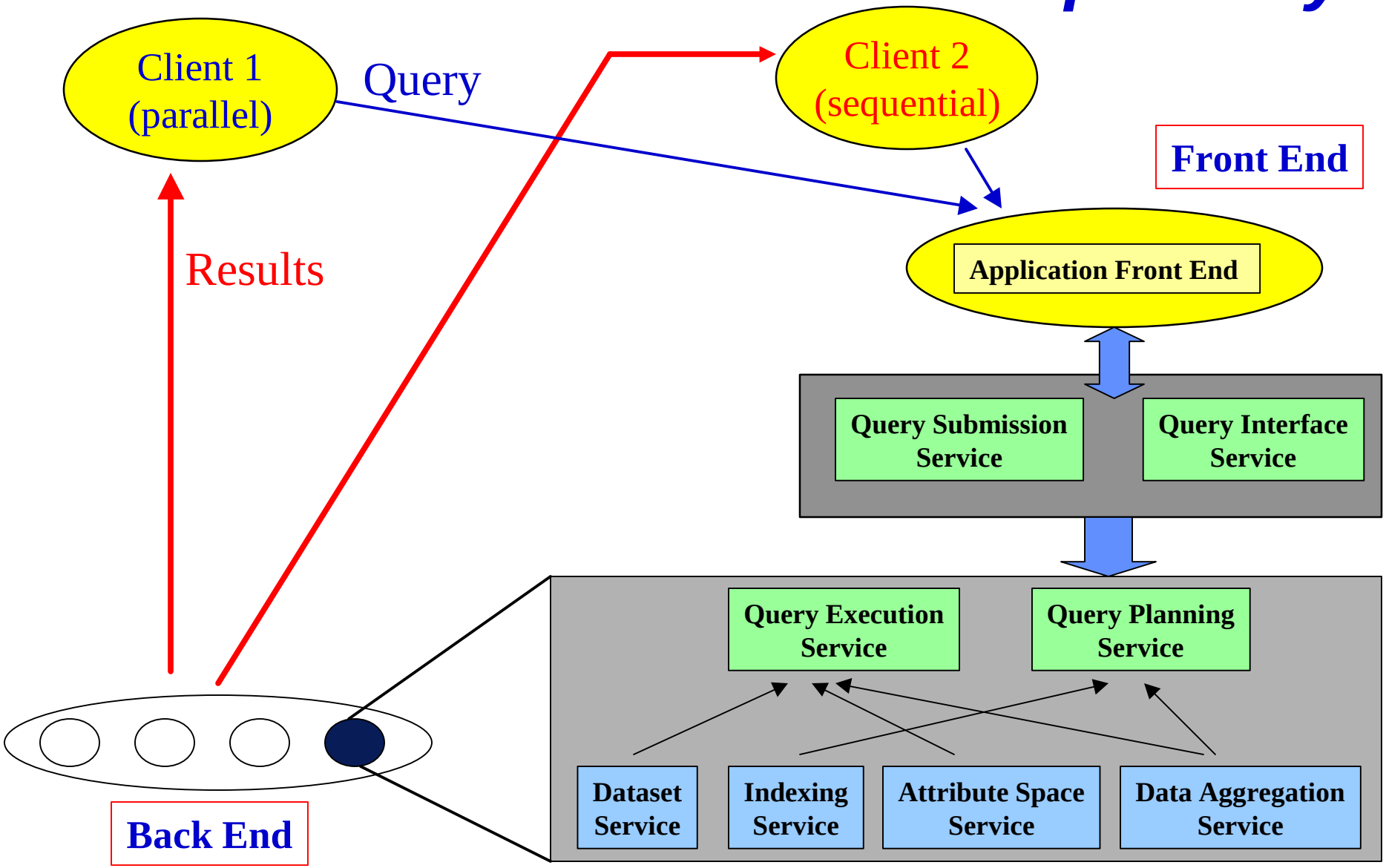
Active Data Repository (ADR)

- **Set of services for building parallel databases of multi-dimensional datasets**
  - enables integration of storage, retrieval and processing of multi-dimensional datasets on parallel machines.
  - can maintain and jointly process multiple datasets.
  - provides support and runtime system for common operations such as
    - data retrieval,
    - memory management,
    - scheduling of processing across a parallel machine.
  - customizable for various application specific processing.

Active Data Repository

- **Front-end:** the interface between clients and back-end. Provides services:
  - for clients to connect to ADR,
  - to query ADR to get information about already registered datasets and user-defined methods,
  - to create ADR queries and submit them.
- **Back-end:** data storage, retrieval, and processing.
  - Distributed memory parallel machine, with multiple disks attached to each node
  - Customizable services for application-specific processing
  - Internal services for data retrieval, resource management

Architecture of Active Data Repository



ADR Internal Services

- **Query interface service**
  - receives queries from clients and validates a query
- **Query submission service**
  - forwards validated queries to back end
- **Query planning service**
  - determines a query plan to efficiently execute a set of queries based on available system resources
- **Query execution service**
  - manages system resources and executes the query plan generated.
- **Handling Output**
  - Write to disk, or send to the client using Unix sockets, or Meta-Chaos (for parallel clients).

ADR Customizable Services

- **Developed as a set of modular services in C++**
  - customization via inheritance and virtual functions
- **Attribute space service**
  - manages registration and use of multi-dimensional attribute spaces, and mapping functions
- **Dataset service**
  - manages datasets loaded into ADR and user-defined functions that iterate through data items
- **Indexing service**
  - manages various indices for datasets loaded into ADR
- **Data aggregation service**
  - manages user-defined functions to be used in aggregation operations

Datasets in Active Data Repository

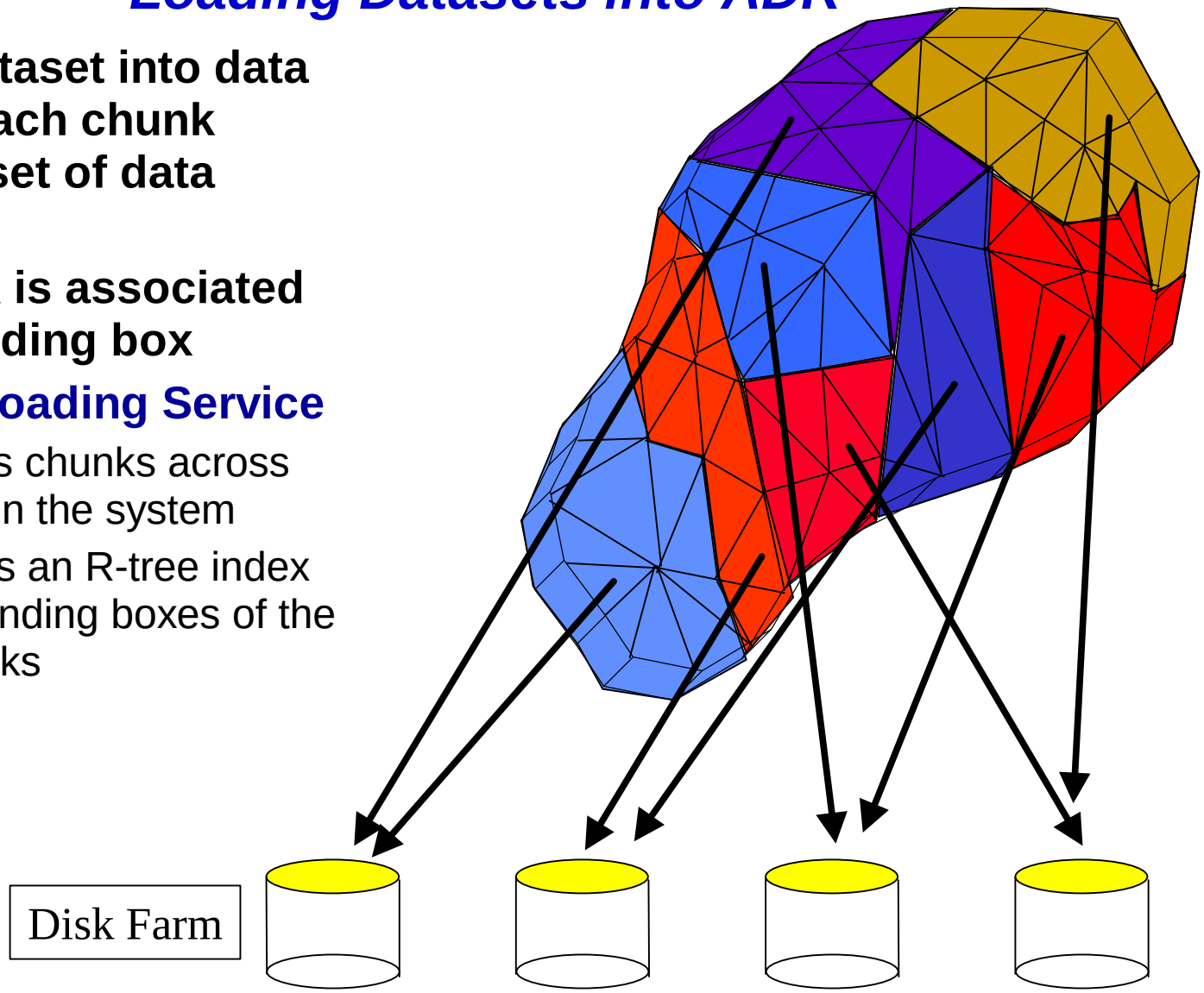
- **ADR expects the input datasets to be partitioned into data chunks.**
- **A data chunk, unit of I/O and communication,**
  - contains a subset of input data values (and associated points in input space)
  - is associated with a *minimum bounding rectangle*, which covers all the points in the chunk.
- **Data chunks are distributed across all the disks in the system.**
- **An index has to be built on minimum bounding rectangles of chunks**

Loading Datasets into ADR

- **A user**
  - should partition dataset into data chunks
  - can distribute chunks across the disks, and provide an index for accessing them
- **ADR, given data chunks and associated minimum bounding rectangles in a set of files**
  - can distribute data chunks across the disks using a Hilbert-curve based declustering algorithm,
  - can create an R-tree based index on the dataset.

Loading Datasets into ADR

- **Partition dataset into data chunks -- each chunk contains a set of data elements**
- **Each chunk is associated with a bounding box**
- **ADR Data Loading Service**
  - Distributes chunks across the disks in the system
  - Constructs an R-tree index using bounding boxes of the data chunks



Active Data Repository -- Customization

- **Indexing Service:**
  - *Index lookup functions* that return data chunks given a range query.
  - ADR provides an R-tree index as default.
- **Dataset Service:**
  - *Iterator functions* that return input elements (data value and associated point in input space) from a retrieved data chunk
- **Attribute Space Service:**
  - *Projection functions* that map a point in input space to a region in output space

Active Data Repository -- Customization

- **Data Aggregation Service:**
  - *Accumulator Functions* to create and tile the *accumulator* to hold intermediate results
  - *Aggregation functions* to aggregate input elements that map to the same output element.
  - *Output functions* to generate output from intermediate results.

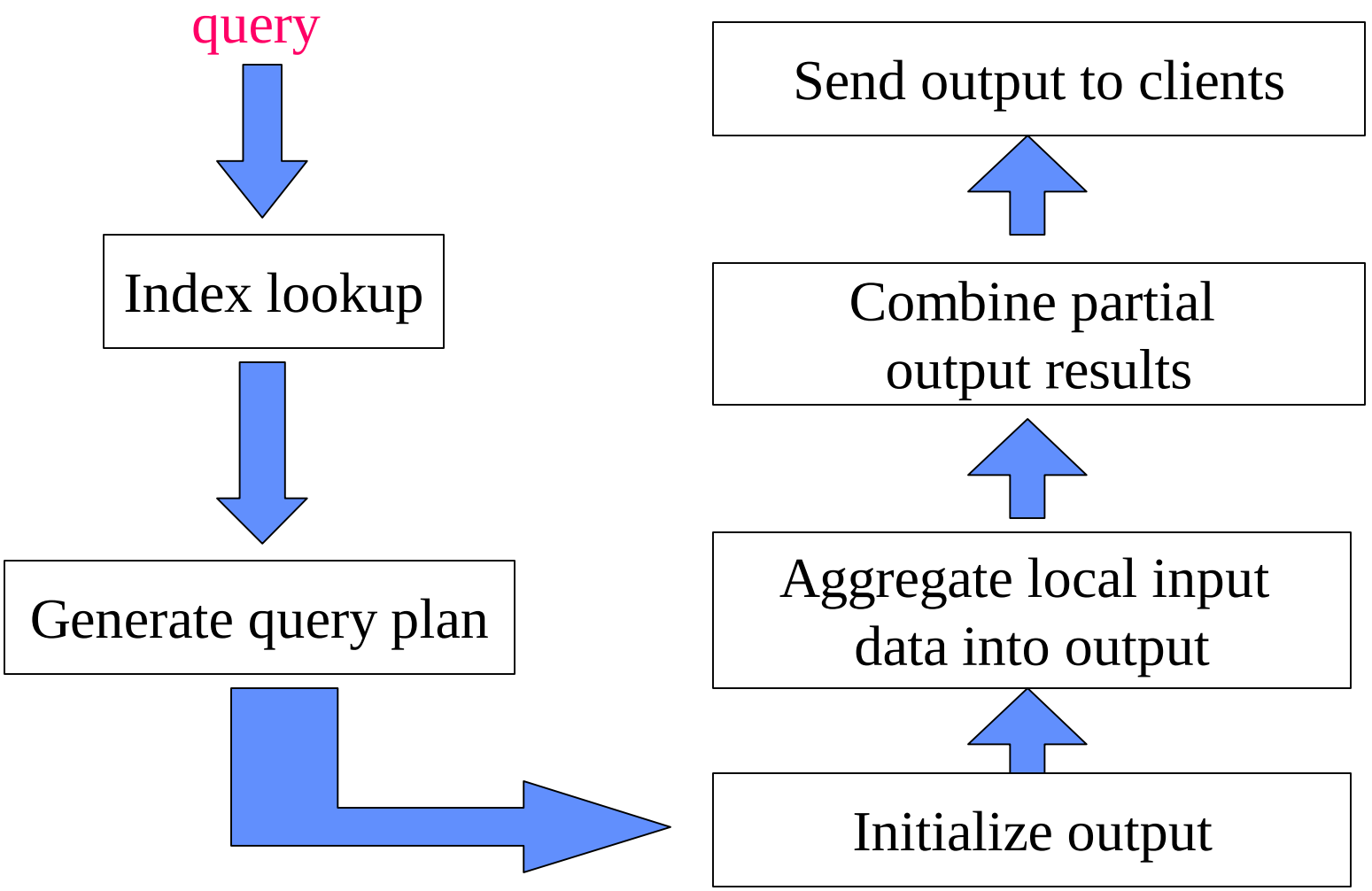
Query Execution in Active Data Repository

- **An ADR Query contains a reference to**
  - the data set of interest,
  - a query window (a multi-dimensional bounding box in input dataset's attribute space),
  - default or user defined index lookup functions,
  - user-defined accumulator,
  - user-defined projection and aggregation functions,
  - how the results are handled (write to disk, or send back to the client).
- **ADR handles multiple simultaneous active queries**

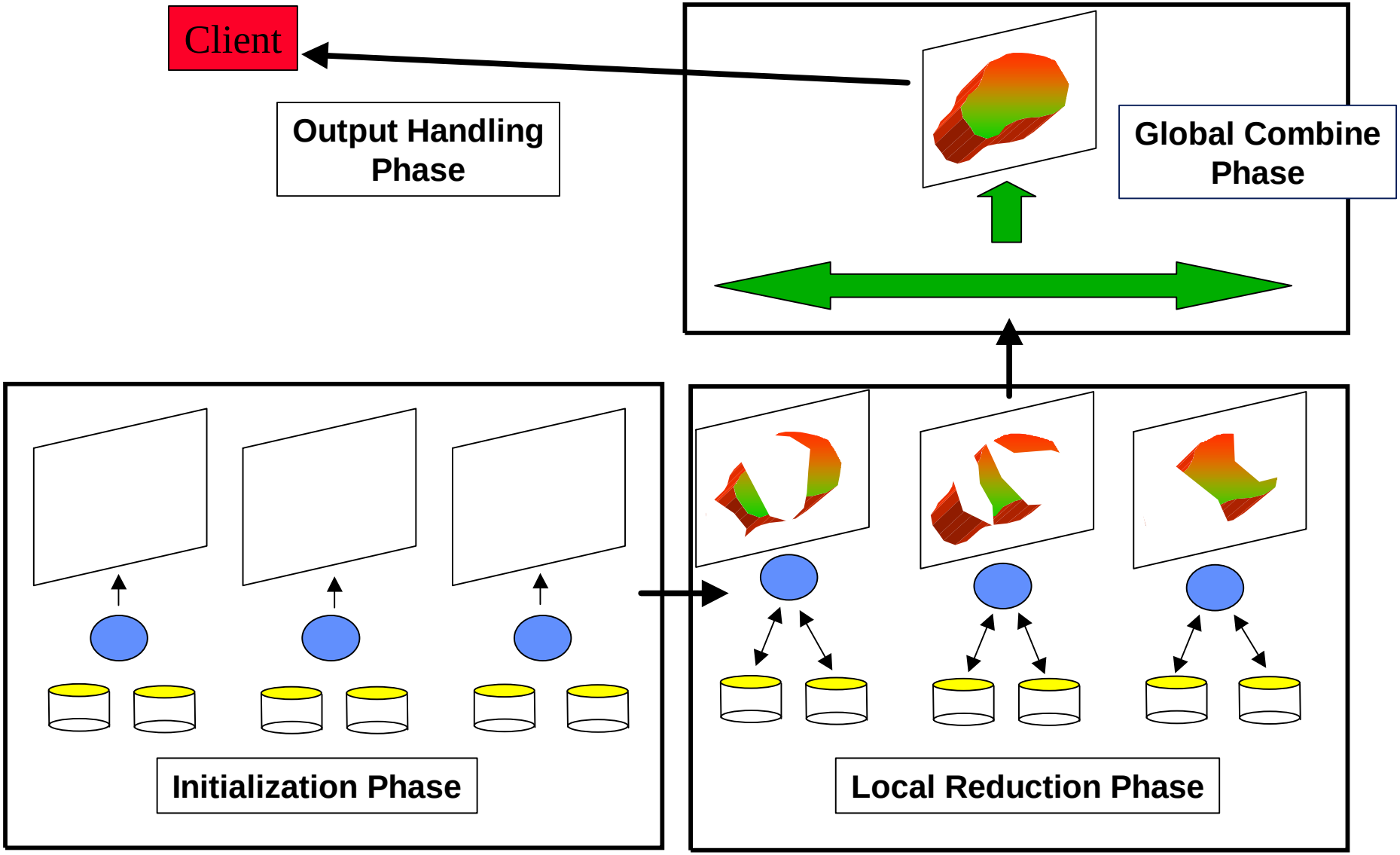
Query Execution in ADR

- **Query execution phases:**
  - *Query Planning*: Find local data blocks that intersect the query. Create in-core data structures for intermediate results (accumulators).
  - *Local Reduction*: Retrieve local data blocks, and perform mapping and aggregation operations.
  - *Global Combine*: Merge intermediate results across processors.
  - *Output Handling*: Create final output. Write results to disk, or send them back to the client.
- **Each query goes though the phases independent of other active queries**

ADR Back-end Processing



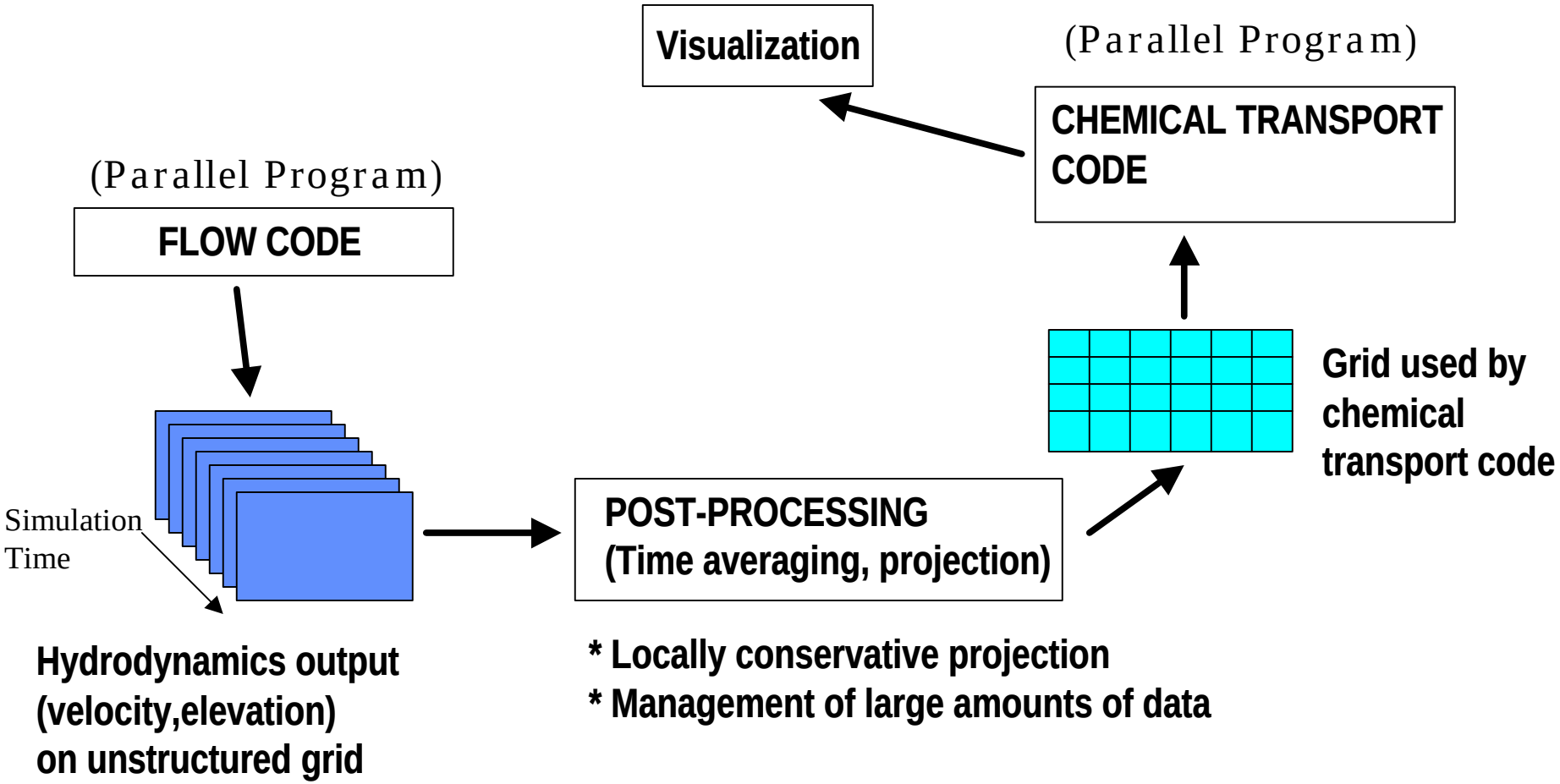
ADR Back-end Processing

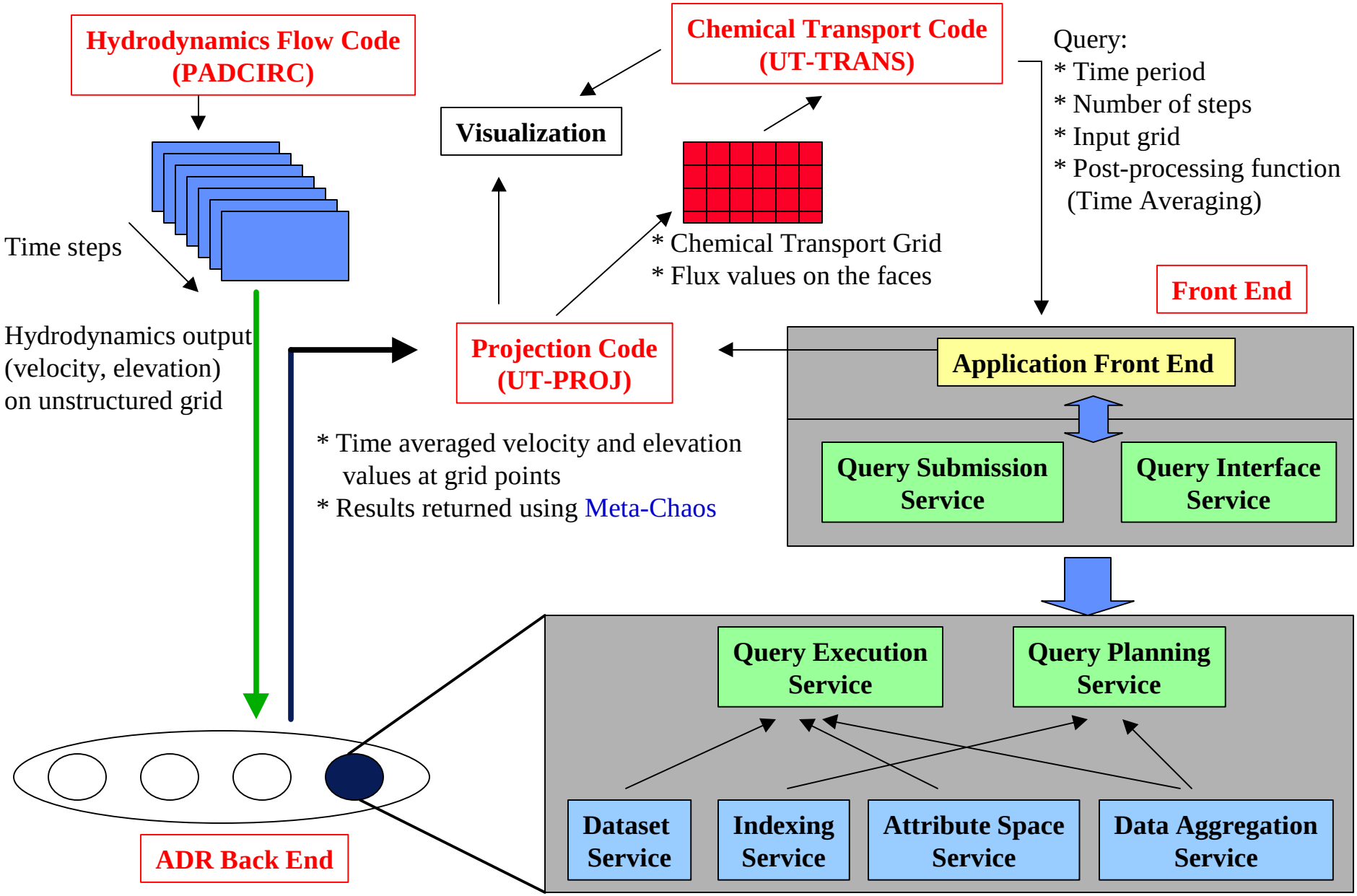


Current Active Data Repository Applications

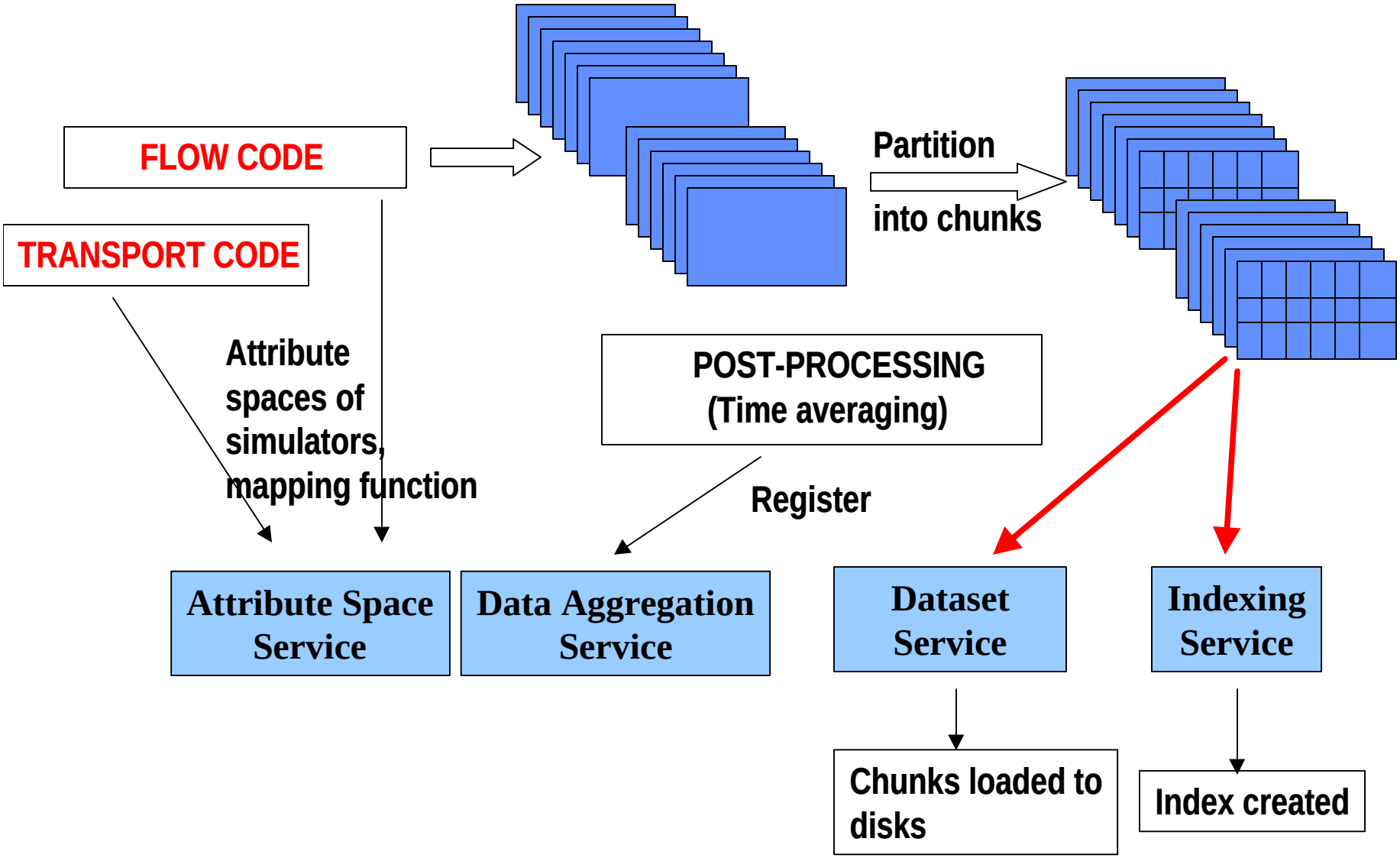
- **Bays and Estuaries Simulation System**
  - Water contamination studies
  - Hydrodynamics simulator is coupled to chemical transport simulator
- **Virtual Microscope**
  - a data server for digitized microscopy images
  - browsing, and visualization of images at different magnifications
- **Titan**
  - a parallel database server for remote sensed satellite data

Bays and Estuaries Simulation System




Hydrodynamics Flow Code (PADCIRC)
Chemical Transport Code (UT-TRANS)
Visualization
Query:
* Time period
* Number of steps
* Input grid
* Post-processing function (Time Averaging)
Time steps
* Chemical Transport Grid
* Flux values on the faces
Front End
Hydrodynamics output (velocity, elevation) on unstructured grid
Projection Code (UT-PROJ)
Application Front End
* Time averaged velocity and elevation values at grid points
* Results returned using Meta-Chaos
Query Submission Service
Query Interface Service
Query Execution Service
Query Planning Service
Dataset Service
Indexing Service
Attribute Space Service
Data Aggregation Service
ADR Back End

Bays and Estuaries Simulation System (Data Loading/Customization)



Bays and Estuaries Simulation System

Attribute Space Service

* Input and output attribute spaces are registered
  - Input Attribute Space: grid points, time steps
  - Output Attribute Space: grid points
* Mapping functions to map input points to output points are registered.

Dataset Service

* An Iterator function is registered
  - understands data structure of a chunk
  - returns input points and data values stored in the chunk

Indexing Service

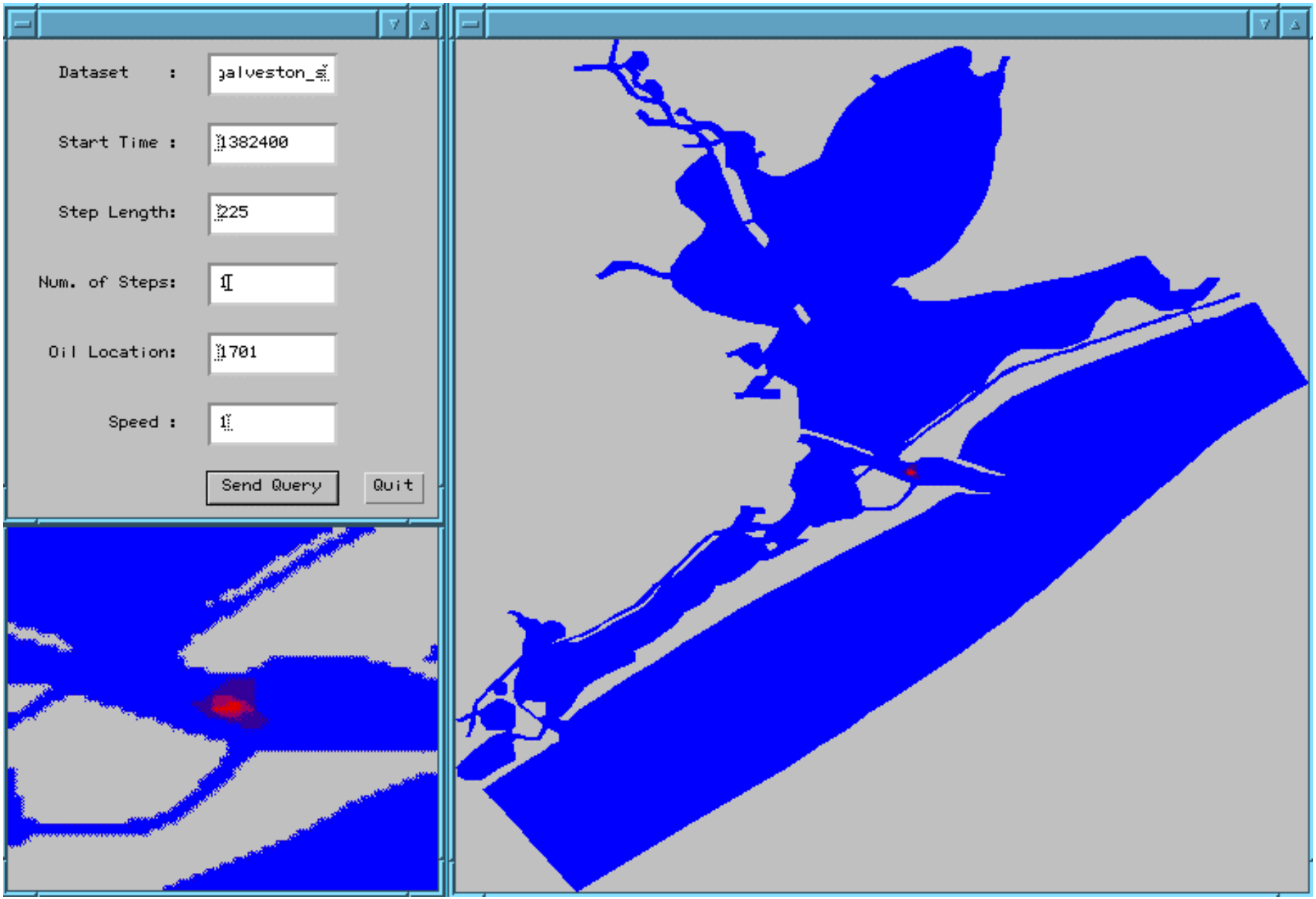
* Functions to read index metadata, to search/return the chunks that intersect the query window are registered.

Data Aggregation Service

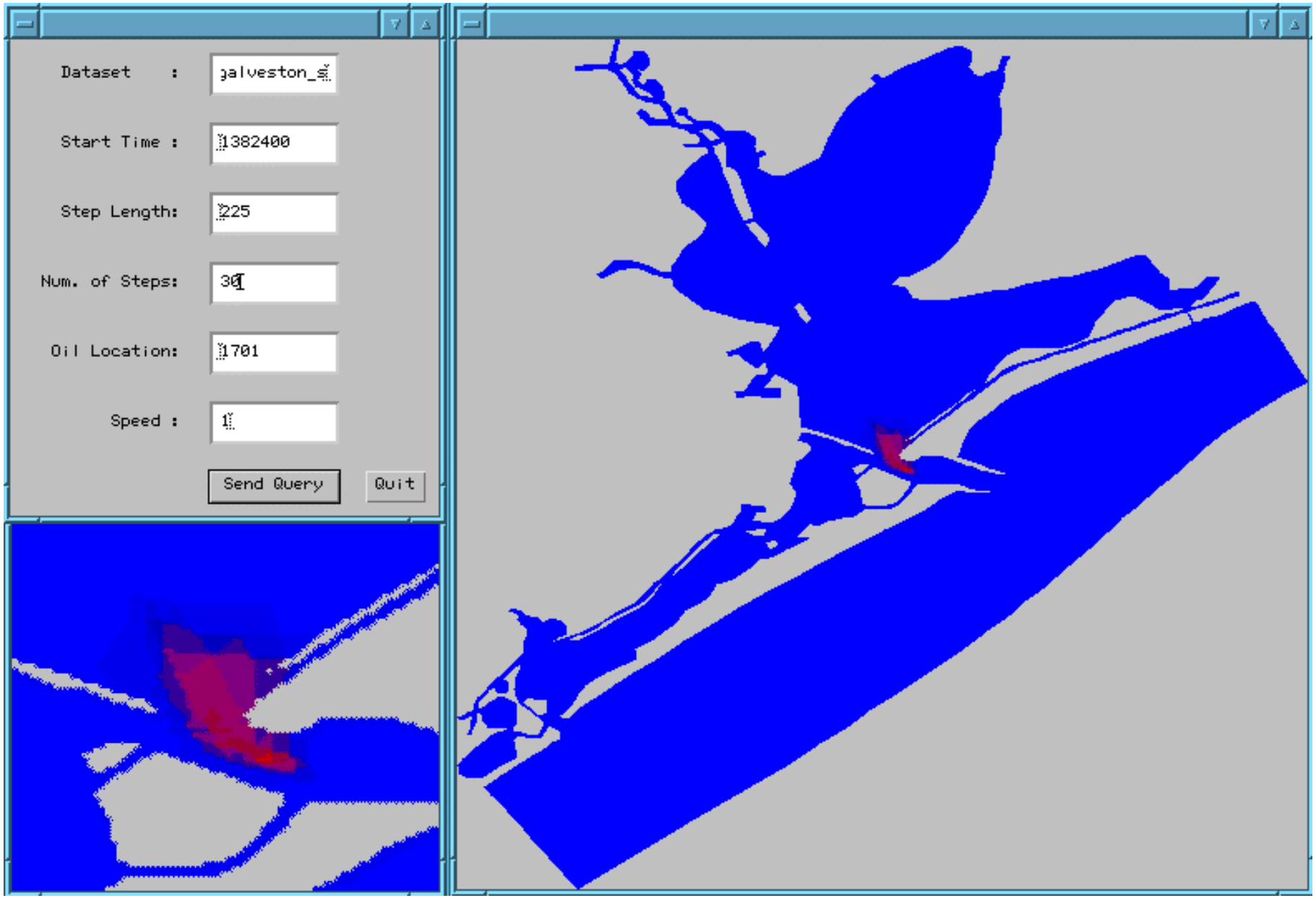
* Functions
  - to define output or intermediate data structure (accumulator)
  - to iterate over output elements
  - to aggregate input data values with output data values (for time averaging)
  - to create final output from intermediate data structure

  are registered.

Initial Oil Spill



Oil spill after 30 time steps (1.9 hours)



Experimental Results

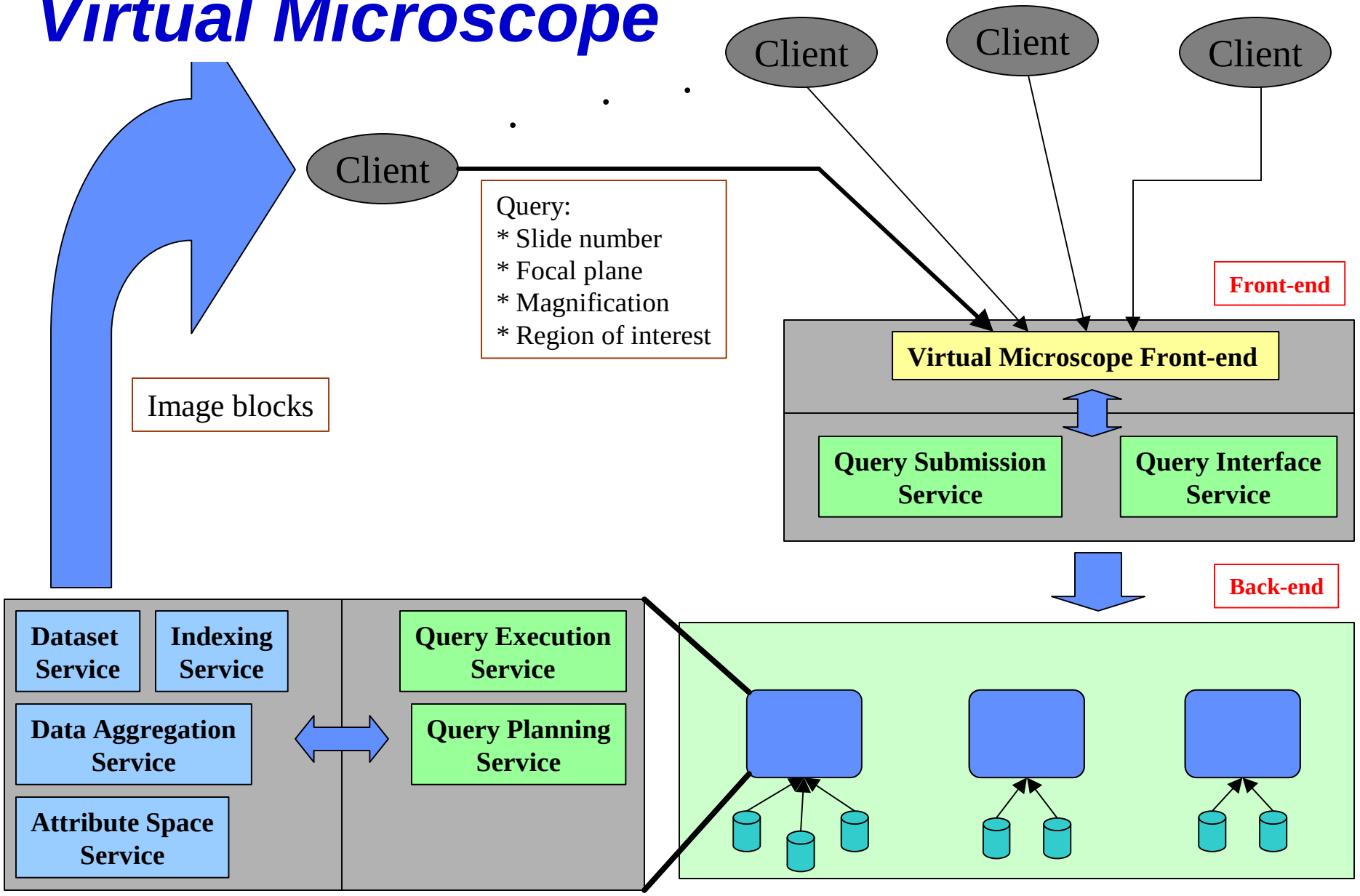
- ADR back-end was run on 8 nodes (with 2 local disks per node) of an IBM SP2.
- A 2D grid that models Galveston bay with 2113 grid points.
- A dataset of 8 days of hydrodynamics simulator output (using simulation time steps of 15 seconds).
  - Data set was partitioned into data blocks, each of which is 128 Kbytes, and contains 33 grid points and 323 time steps. A total of 9152 blocks.
  - Data blocks declustered across all the disks.
- Meta-Chaos was used for sending results from ADR to simulation code.

Experimental Results

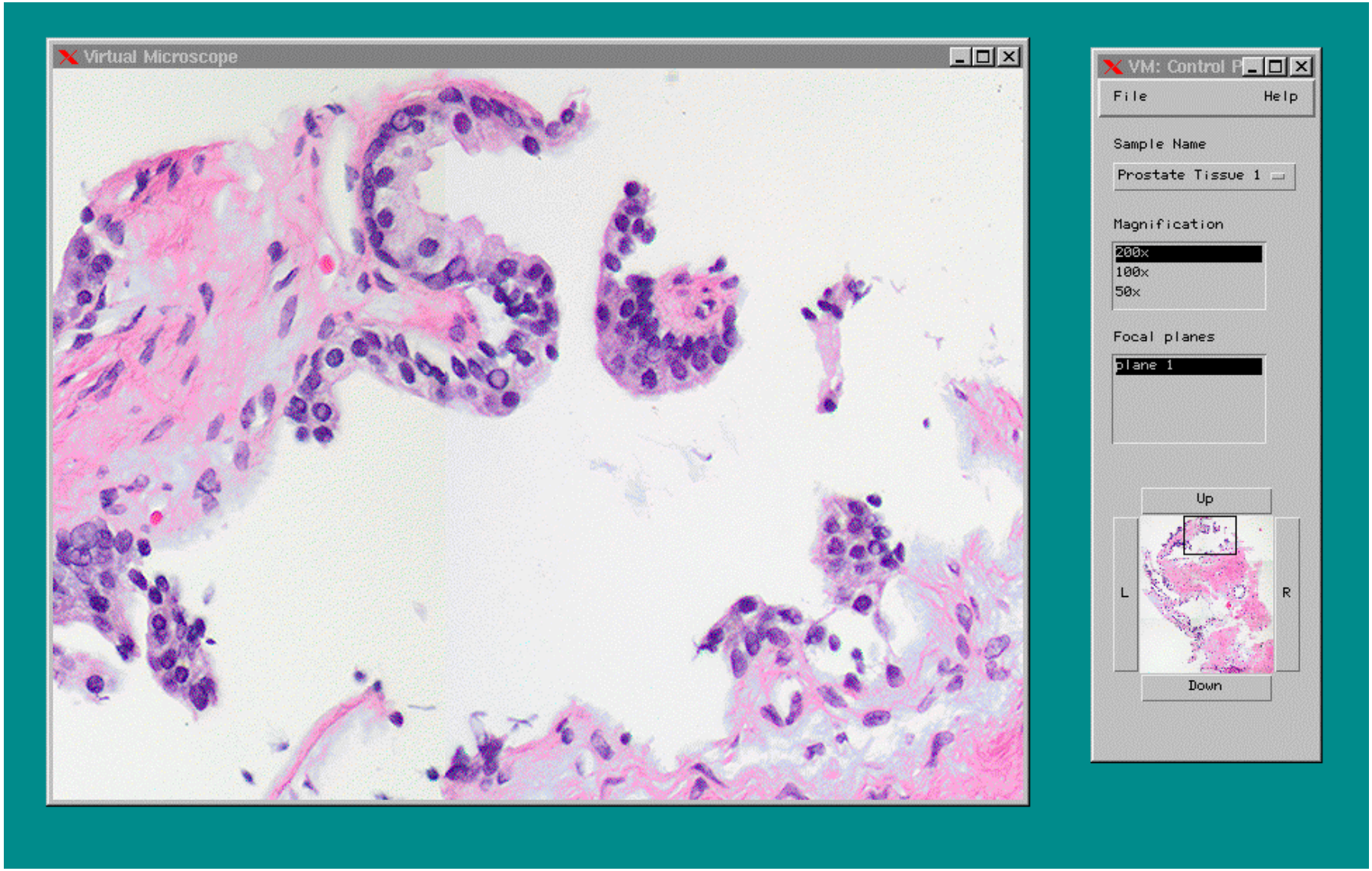
Query	Total	Query Planning	Local Reduction	Global Combine	Output Handling
9000	1.35	0.56	0.43	0.03	0.33
4500	1.00	0.42	0.23	0.02	0.33
2250	0.90	0.35	0.15	0.03	0.37
1125	0.85	0.38	0.10	0.03	0.34
225	0.84	0.38	0.08	0.03	0.35

An end-to-end 2-hour oil spill simulation takes 300 secs. using chemical transport step of 225 secs. (i.e., averaging over 15 hydrodynamics steps)

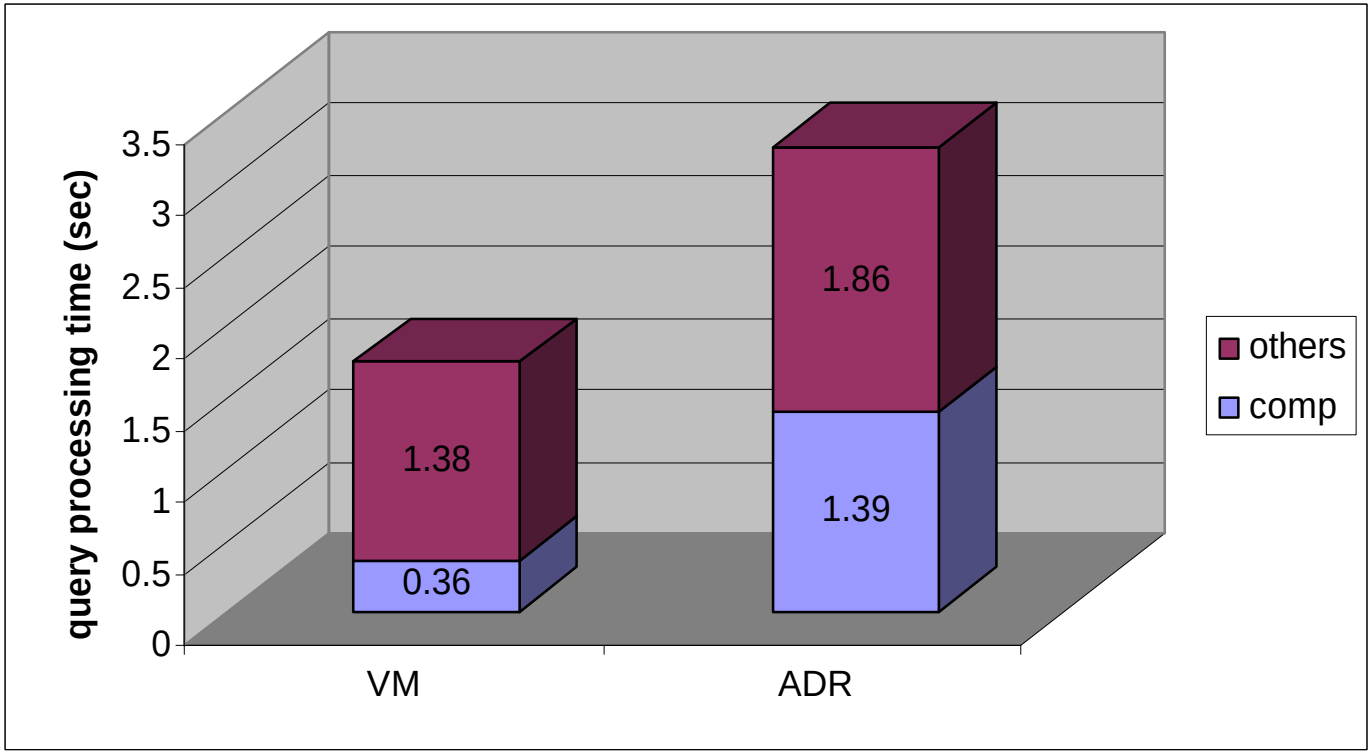
Virtual Microscope



Virtual Microscope Client

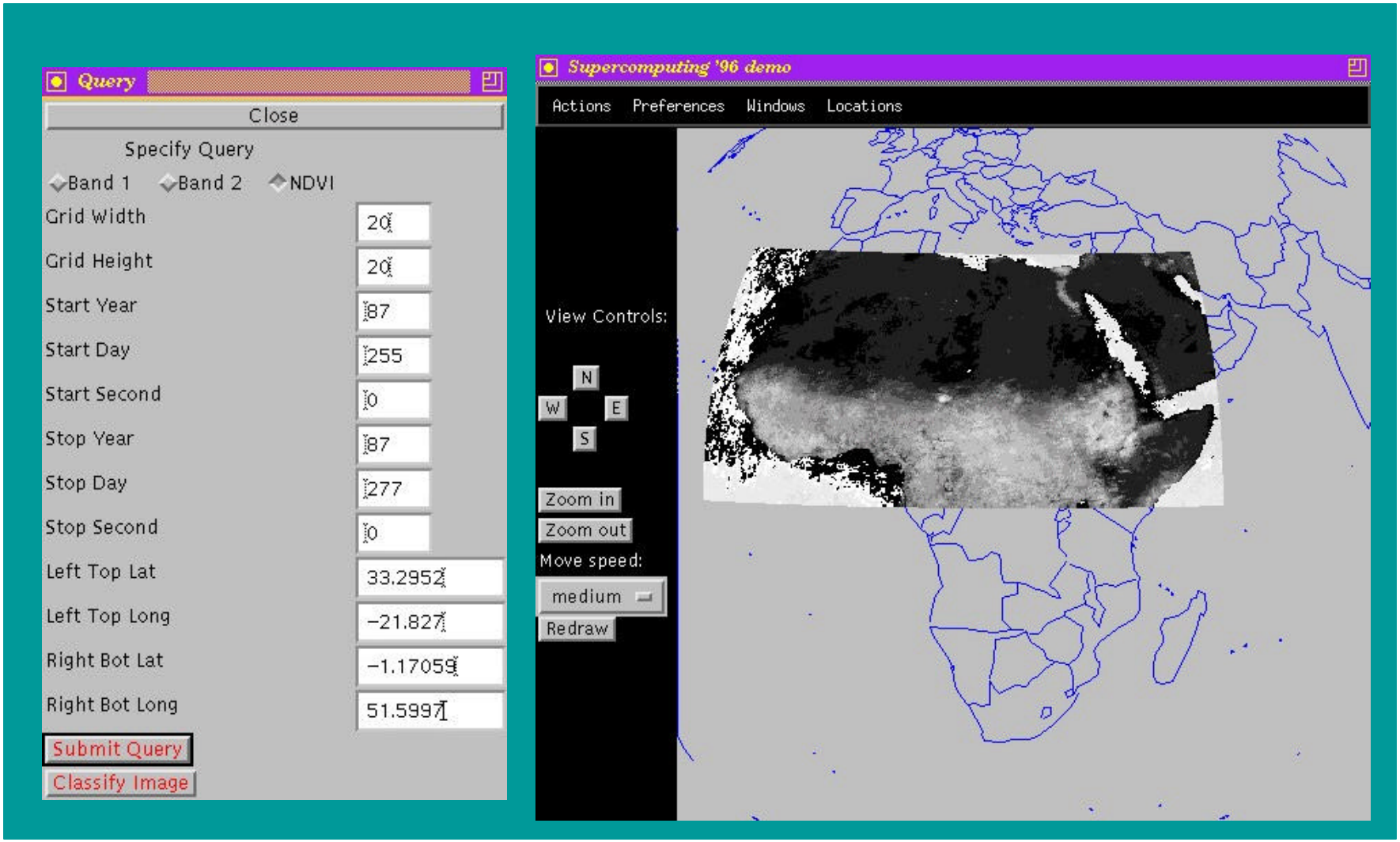


VM Performance



- **Running on 8 back-end nodes, 1 disk/node**
- **Input data size = 72 MB (357 data blocks)**

Titan - Satellite Data Processing



Example: Satellite Data Processing

Attribute Space Service

* Register attribute spaces
  - lat/lon/time, Goodes projection, etc.
* Register mapping functions between attribute spaces

Dataset Service

* Partition IFOVs into data chunks
* Register iterator functions to return IFOVs from chunks

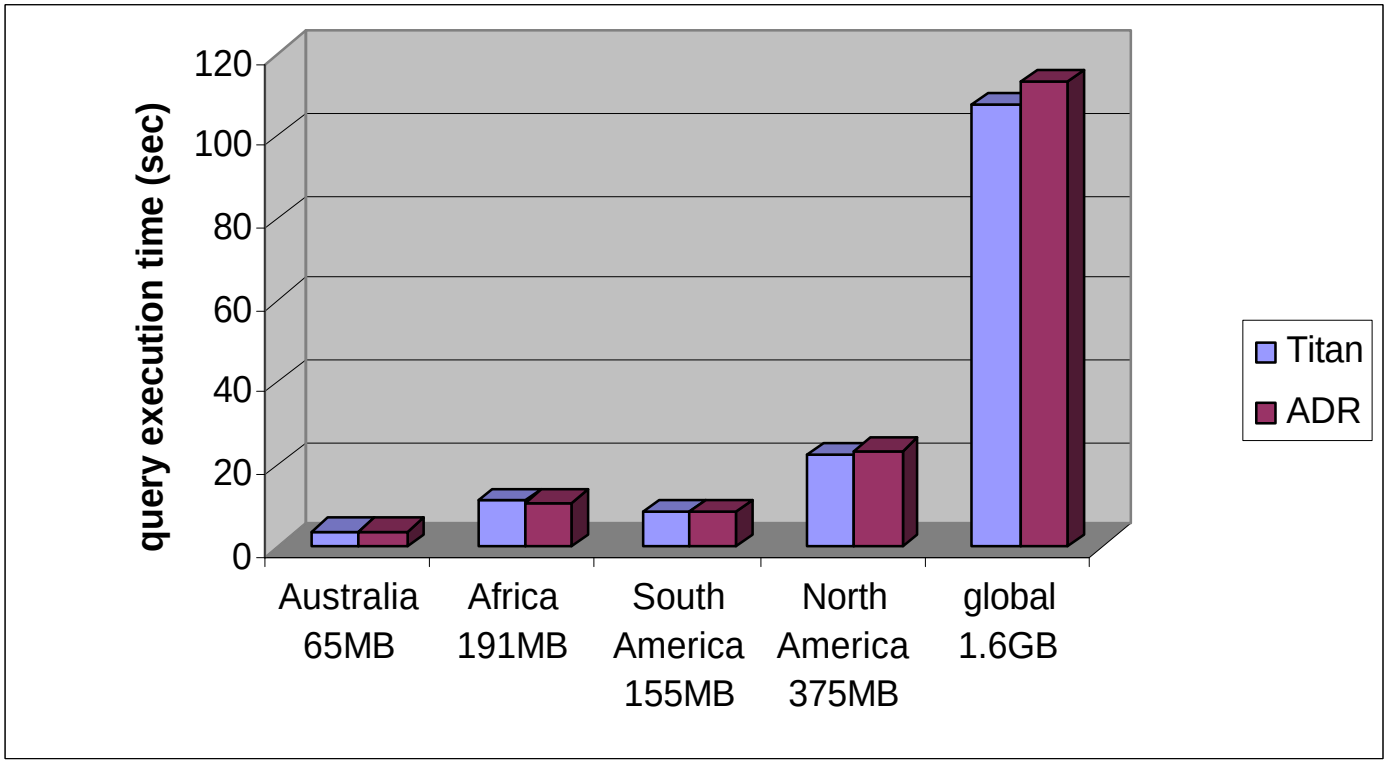
Indexing Service

* Build an R-tree to index the IFOV chunks
* Use lat/lon/time of IFOV chunks as bounding rectangles

Data Aggregation Service

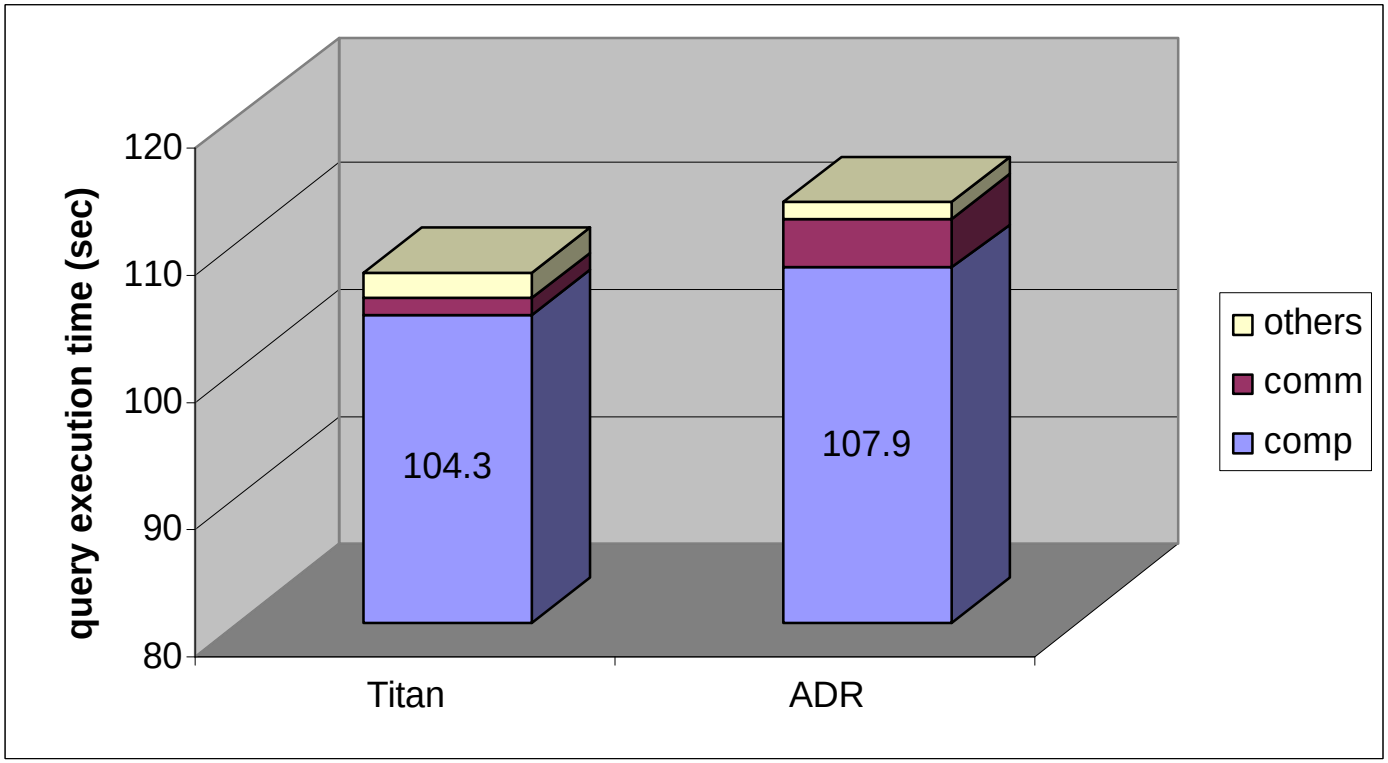
* Register functions to:
  - Initialize the output image
  - Compute vegetation index of an output pixel with a given IFOV
  - Select *clearest* vegetation index out of a set of IFOVs

Satellite Data Processing Performance



- **Running on 14 SP-2 RS6000/390 nodes, 4 disks/node**

Global Query Breakdown



Average input data size read per back-end node = 114 MB