

The Virtual Microscope

Tristan Wen
September 26, 2002

Presentation Outline

- Introduction
- System Overview
- VM Server Using ADR
- VM Server Using DataCutter

Introduction

- Virtual Microscope
 - Traditional way
 - Why digitalize slides?
 - Provide simultaneous access to the slides world-widely
 - No need to rely on slides at local institution
 - Suitable for training and conference environments
 - Slides comparing made easy
 - New software modules can be added
 - To perform additional processing
 - Emulation of high power light microscope
 - Usual behavior of a microscope
 - Continuously moving the stage
 - Changing magnification
 - Changing focus

Introduction (cont.)

- Data size is huge
 - For ex. A 200X spot at a single depth of focus
 - Requires 1000X1000 pixels
 - RGB: 1000X1000X3 => 3 MB
 - A complete slide of 3.5X2.5 cm
 - =>50X70 such 200X spot
 - =>50X70X3 MB = 10.5 GB
 - 5 focal planes
 - =>5X10.5 GB = 52.5 GB
 - A hospital can generate thousands slides
 - JHH: 420,000 slides per year

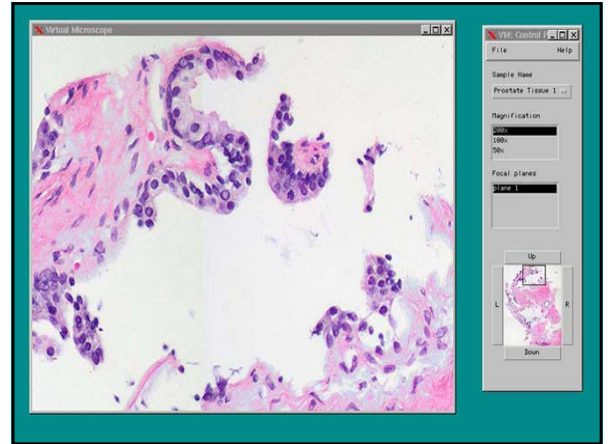
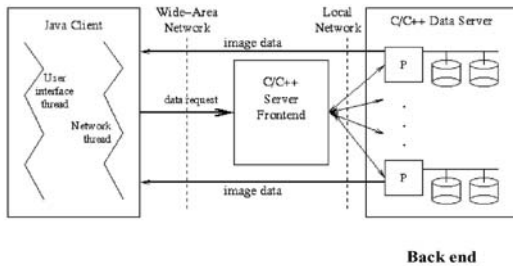
System Overview

- Java based client
 - Basic functionalities
 - Fast browsing
 - Local browsing
 - Changing magnification
 - Changing the focal plane
- Server Front-end
 - Sequential program on one node
 - Interacts with clients
 - Translates clients' requests into queries for data server
 - Relieves data server from being interrupted

System Overview (cont.)

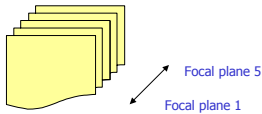
- Data Server
 - Run on a cluster with local disks
 - Retrieves and processes data from multiple disks
 - Two implementations
 - Active Data Repository (ADR)
 - Data Cutter

Virtual Microscope Architecture



Data set in Virtual Microscope

- Data set is in 3 dimensions
 - Each focal plane is a 2-D image



- A reference to the data of interest is described by a range query

Active Data Repository (ADR)

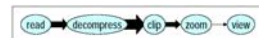
- Used in a homogenous environment
- Consists of a set of modular services
 - Indexing service
 - Default is R-tree indexing
 - Data aggregation service
 - Manages user-provided functions to be used on data
 - Functions to generate final output
- Hilbert curve-based algorithm
 - De-clustering

DataCutter

- Used in a heterogeneous environment
- Component-based model
- Provides a set of core services
 - Indexing service
 - R-tree indexing
 - Filtering service
 - A filter is a user-defined object with methods

DataCutter (cont.)

- Filters in Virtual Microscope



- Read_data (R)
 - Read image chunks that overlap the query
- Decompress (D)
 - Convert jpeg image chunks into RGB pixels
- Clip (C)
 - Only keep portion of chunk inside the query region
- Zoom (Z)
 - Sub-sample to the required magnification
- View (V)
 - Stitch chunks together and display image

Data Caching In Client

- Server interacts with many clients simultaneously
 - Should also achieve small response time
- Cache in client
 - Images are viewed as partitioned tiles
 - Tiles are used as units of caching for portions of the image
 - Least Recently Used (LRU) policy is used
 - Performance boost

Titan: A High-Performance Remote-Sensing Database

Tristan Wen
September 26, 2002

Presentation Outline

- Introduction
- System Overview
- Data Placement
- Query Processing

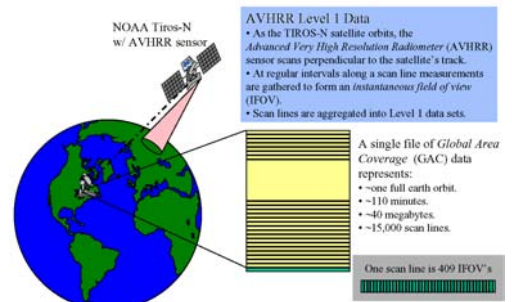
Introduction

- Titan
 - Sensors obtain data about the world
 - Research
 - Land cover changes
 - Temperature and rainfall as well as climate predictions
 - Data volume is huge
 - A coarse-grained satellite data spans 10 days is 4 GB
 - A finer-grained one is 65 GB!
 - Processed output image is comparably small
 - 228 MB for a coarse-grained data

Introduction (cont.)

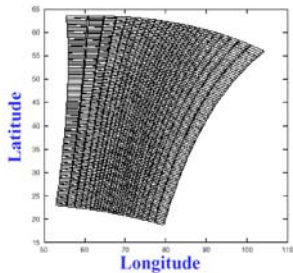
- 2 challenges
 - Low latency retrieval of vary large volumes of spatial-temporal data
 - Requires effective de-clustering and placement of a multi-dimensional dataset onto a disk farm
 - Data size greatly reduced (4 GB-> 228 MB)
 - Requires careful coordination of computation and data retrieval to avoid slowing down either process
- 2 reasons for maintaining the data in a raw form
 - Composition operations can not be reversed
 - No single projection is adequate for all uses

Processing Remotely Sensed Data



Spatial Irregularity

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

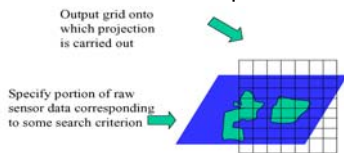


System Overview

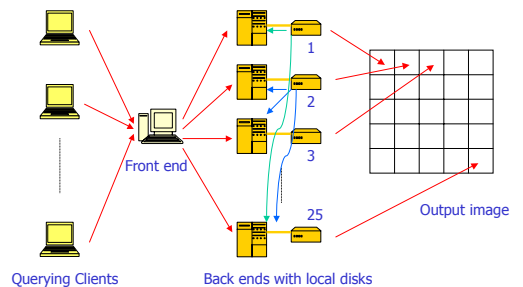
- Querying clients
- Front end
 - One host
 - Interacts with querying clients
 - Partitions data retrieval and computation
 - A simplified R-tree to index data blocks
- Back end
 - A set of processing nodes on a dedicated network
 - Retrieves the data
 - Performs post-processing and composition operations

System Overview (cont.)

- Titan queries
 - Temporal bounds
 - Spatial bounds
 - Sensor type and number
 - Resolution of the output



System Diagram



Data Placement

- Titan addresses the problem of low-latency retrieval of very large volumes of data in 3 ways:
 - Takes advantage of the data format and query patterns
 - Maximize disk parallelism by de-clustering the set of chunks onto disk farm
 - Minimize seek time by clustering the chunks assigned to each disk

Data Partitioning

- Size should be large enough
 - To allow efficient retrieval
- Size should be small enough
 - That most data retrieved is useful
- Data block should be square
 - To provide better indexing
- Therefore, we chose to partition in tiles of 204X204 pixels

Indexing Scheme

- Implemented the index as a simplified R-tree
- Leaf nodes
 - Data blocks, spatial and temporal extent, sensor type, satellite number, position on disk for each data block
 - Arranged in a z-ordering before index is built
 - Sorting the leaves spatially allows access to the index as a range tree

De-clustering

- Goal
 - To maximize disk parallelism
- Mini-max spanning tree algorithm by Moon
- Each disk is assigned at most N/M data blocks

Clustering

- Goal
 - To minimize seek time on individual disks
- Shortest Spanning Path Algorithm (SSP)
 - To linearize the data blocks
 - Uses a heuristic

Query Processing

- 5 phases-executed in a while loop until all the data has been processed
 - Block-read
 - Block-send-check
 - Block-receive
 - Block-read-check
 - Block-consume
- Overlapped I/O
 - To hide larger latency for I/O and inter-processor communication

Query Processing (cont.)

```
while (not all activities are done)
  /* block-read phase */
  issue as many asynchronous disk reads for blocks as
  possible;
  /* block-send-check phase */
  check all pending block sends, freeing send buffers for
  completed ones;
  /* block-receive phase */
  check pending block receives, and for each completed
  one:
    add the receive buffer to the list of buffers that
    must be processed locally;
    if (more non-local blocks must be obtained) issue
    another asynchronous receive;
```

Query Processing (cont.)

```
/* block-read-check phase */
check pending disk reads;
for each completed one, generate asynchronous sends
to the remote consumers;
/* block-consume phase */
if (a block is already available for processing)
  process the block - perform mapping and com-
  positing operations for all readings in the block;
endif
endwhile
```



Conclusion

- 2 applications are similar
 - Deal with multi-dimensional data set (3-D)
- ADR is a generalized parallel data server
 - Titan is a customized ADR
 - Titan \sim Satellite data + ADR



Discussion

- Some interesting topics
 - ADR and Data Cutter
 - Spatial and temporal data indexing
 - Other suitable applications