Distributed Processing of very large datasets with DataCutter

By: Michael Beynon, Tahsin Kurc, et. Al.

Presented by: Cassie Thomas

What is DataCutter?

- A suite of Middleware for subsetting and filtering multi-dimensional datasets stored on archival storage systems
- Subsetting through Range Queries
  - a hyperbox defined in the multi-dimensional space underlying the dataset
  - items whose multi-dimensional coordinates fall into the box are retrieved.

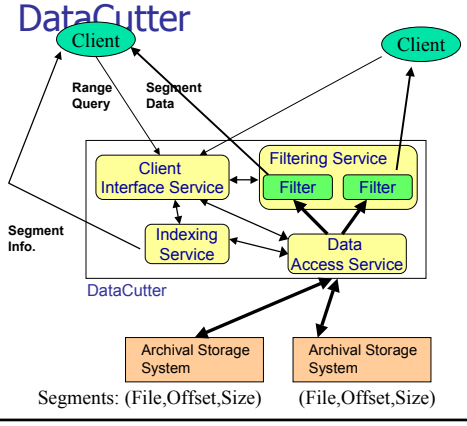
DataCutter (cont)

- Restricted processing (filtering/aggregations) through Filters
  - to reduce the amount of data transferred to the client
  - filters can run anywhere, but intended to run near (i.e., over local area network) storage system
  - based on filter-stream programming model -- to optimize use of limited resources, such as memory and disk space

DataCutter (cont)

- Restricted processing (filtering/aggregations) through Filters
  - to reduce the amount of data transferred to the client
  - filters can run anywhere, but intended to run near (i.e., over local area network) storage system
  - based on filter-stream programming model -- to optimize use of limited resources, such as memory and disk space

DataCutter



DataCutter Architecture

- Client Interface Service
  - Manages client connections and client requests
  - Manages data and information flow between different services
- Indexing Service
  - Two-level hierarchical indexing -- summary and detailed index files
  - Customizable --
    - Default R-tree index
    - User can add new indexing methods

DataCutter Architecture

- Filtering Service
  - Manages filters (registered in the system)
  - Users can add/run new filters
  - Supports multiple concurrent filters and filter instances
    - A filter instance is a set of filters used collectively to perform computation
- Data Access Service
  - Manages storage/retrieval of data from the tertiary storage
  - Low level system dependent I/O operations

DataCutter -- Subsetting

- Datasets are partitioned into segments
  - used to index the dataset, unit of retrieval
- Indexing very large datasets
  - Multi-level hierarchical indexing scheme
  - Summary index files -- to index a group of segments or detailed index files
  - Detailed index files -- to index the segments

DataCutter -- Filters

- Filters
  - Specialized user program to process data (segments) before returning them to the client
- Filter-stream programming model
  - Originally developed for Active Disks environment (Acharya, Uysal, and Saltz)
  - Based on stream abstraction
    - A stream denotes a supply of data
    - Streams deliver data in fixed size buffers
    - Communication of a filter with its environment is restricted to its input and output streams
  - init, process, finalize interface

Application Model

- Application starts with a single process, called console
  - console uses the filtering service to instantiate user-defined filters on other hosts
- Detached vs. Pass-Thru
  - Filter output is not consumed by console -- Detached
  - Filter output is consumed by console -- Pass-Thru
- Stand-Alone vs. Client-Server
  - Console process initiates the work -- Stand-Alone
  - Client process sends work (e.g., query) to be dispatched to filters by the console -- Client-Server

Filters

- A Filter
  - communicates with other filters only using streams
  - cannot change stream endpoints
  - is allowed to pre-disclose dynamic allocation of memory/scratch space in init phase, before processing phase
  - has to pre-disclose minimum required buffer size for each of its streams
- Advantages
  - location independence
  - easier scheduling of resources
  - filter stop and restart is defined explicitly in model

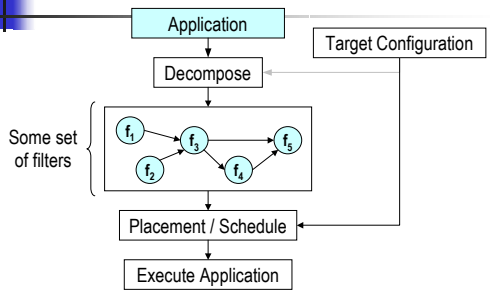
Streams

- Means of data movement
  - A stream is associated with a name
  - Delivers data in fixed size buffers: A filter has to pre-disclose minimum required buffer size for each of its streams.
- Two types of streams
  - File Streams: used to access files.
    - A stream can only access user-defined files.
    - Access to files is done through file segments.
    - A segment is accessed in fixed size buffers, and may cover the entire file.
  - Pipe Streams: uni-directional data flow between two filters

Placement

- The dynamic assignment of filters to particular hosts for execution is placement (mapping)
- Optimization criteria:
  - Communication
    - leverage filter affinity to dataset
    - minimize communication volume on slower connections
    - co-locate filters with large communication volume
  - Computation
    - expensive computation on faster, less loaded hosts

Restructuring Process



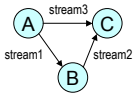
Software Infrastructure

- Prototype implementation of filter framework
  - C++ language binding
  - manual placement
  - wide-area execution service
  - one thread for each instantiated filter

```
class MyFilter : public AS_Filter_Base {
public:
    int init(int argc, char *argv[ ]) { … };
    int process(stream_t st) { … };
    int finalize(void) { … };
};
```

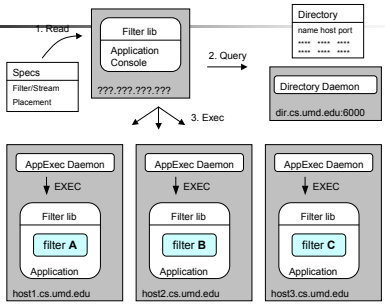
Filter Connectivity / Placement

```
[filter.A]
outs = stream1 stream3
[filter.B]
ins = stream1
outs = stream2
[filter.C]
ins = stream2 stream3
```




```
[placement]
A = host1.cs.umd.edu
B = host2.cs.umd.edu
C = host3.cs.umd.edu
```

Execution Service



Related Work