DataCutter

Joel Saltz

Alan Sussman

Tahsin Kurc

University of Maryland, College Park

and

Johns Hopkins Medical Institutions

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

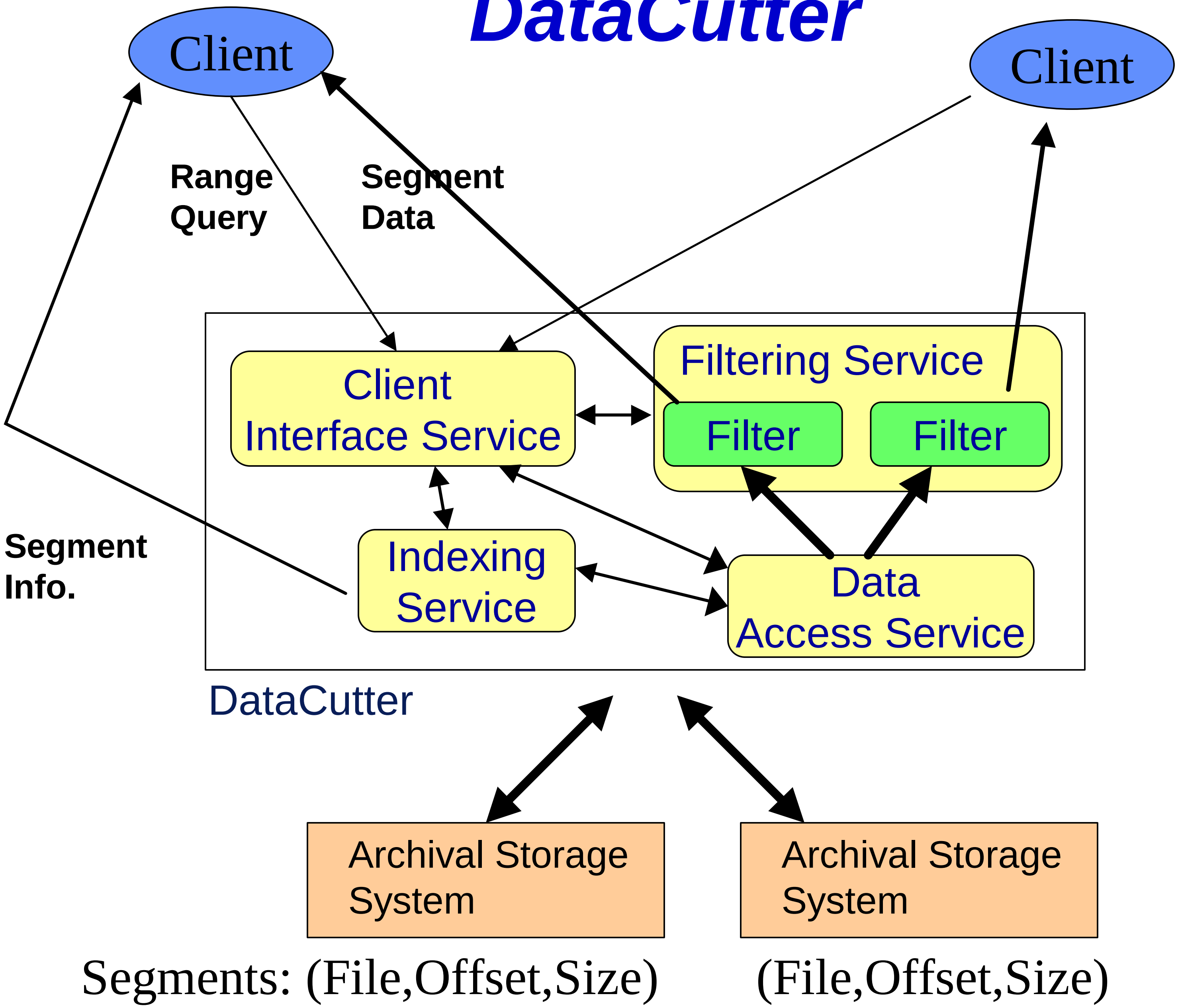
DataCutter

- **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

- **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



Client
Client
Range Query
Segment Data
Segment Info.
Client Interface Service
Filtering Service
Filter
Filter
Indexing Service
Data Access Service
DataCutter
Archival Storage System
Archival Storage System
Segments: (File,Offset,Size)
(File,Offset,Size)

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
- **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**

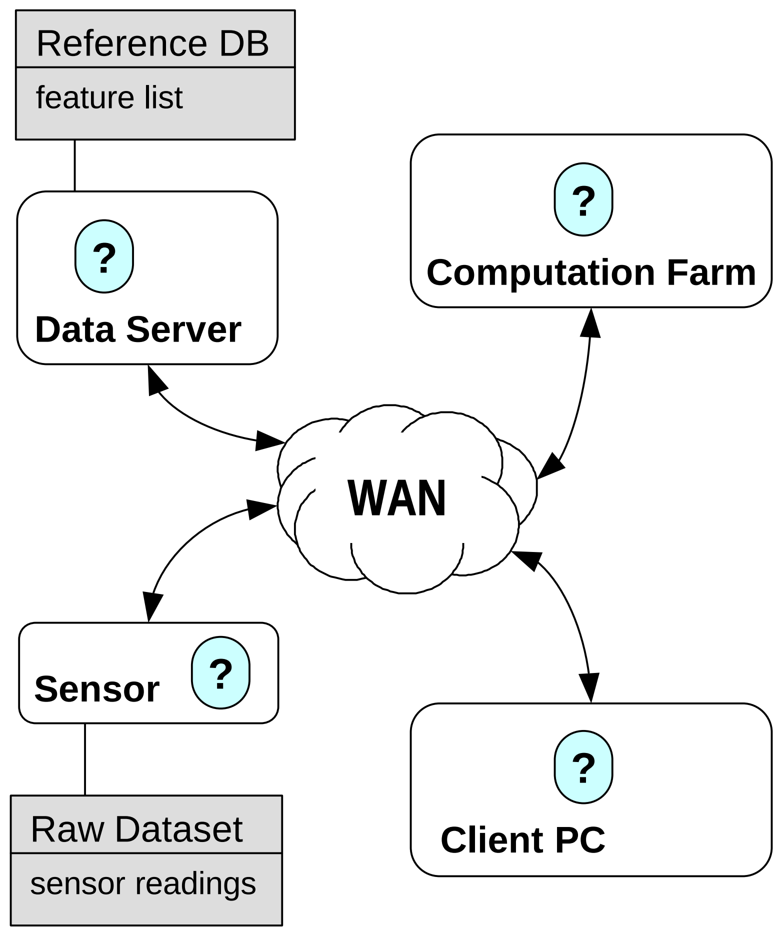
A Motivating Scenario

Sample Application:

- generate 3D reconstructed view
  from new set of sensor readings
- compare features with reference db

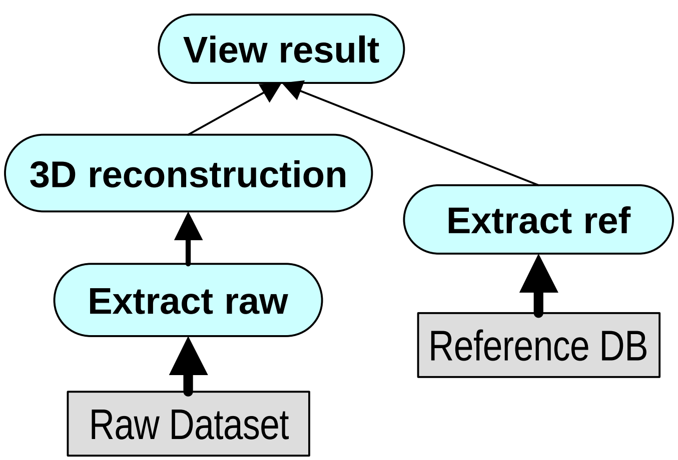
Grid Configuration:

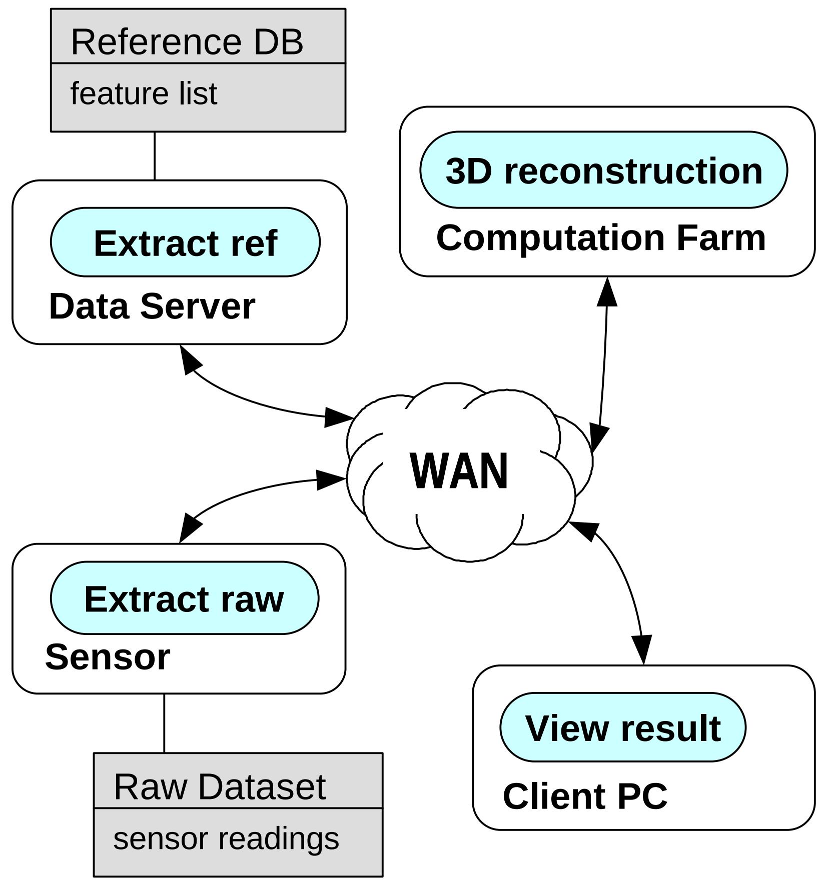
- remote data server - reference db
- sensor host - large raw readings
- parallel computation farm available
- 3D reconstruction computationally intensive



A Motivating Scenario (2)

Application :
// process relevant raw readings
// generate 3D view
// compute features of 3D view
// find similar features in reference db
// display new view and similar cases





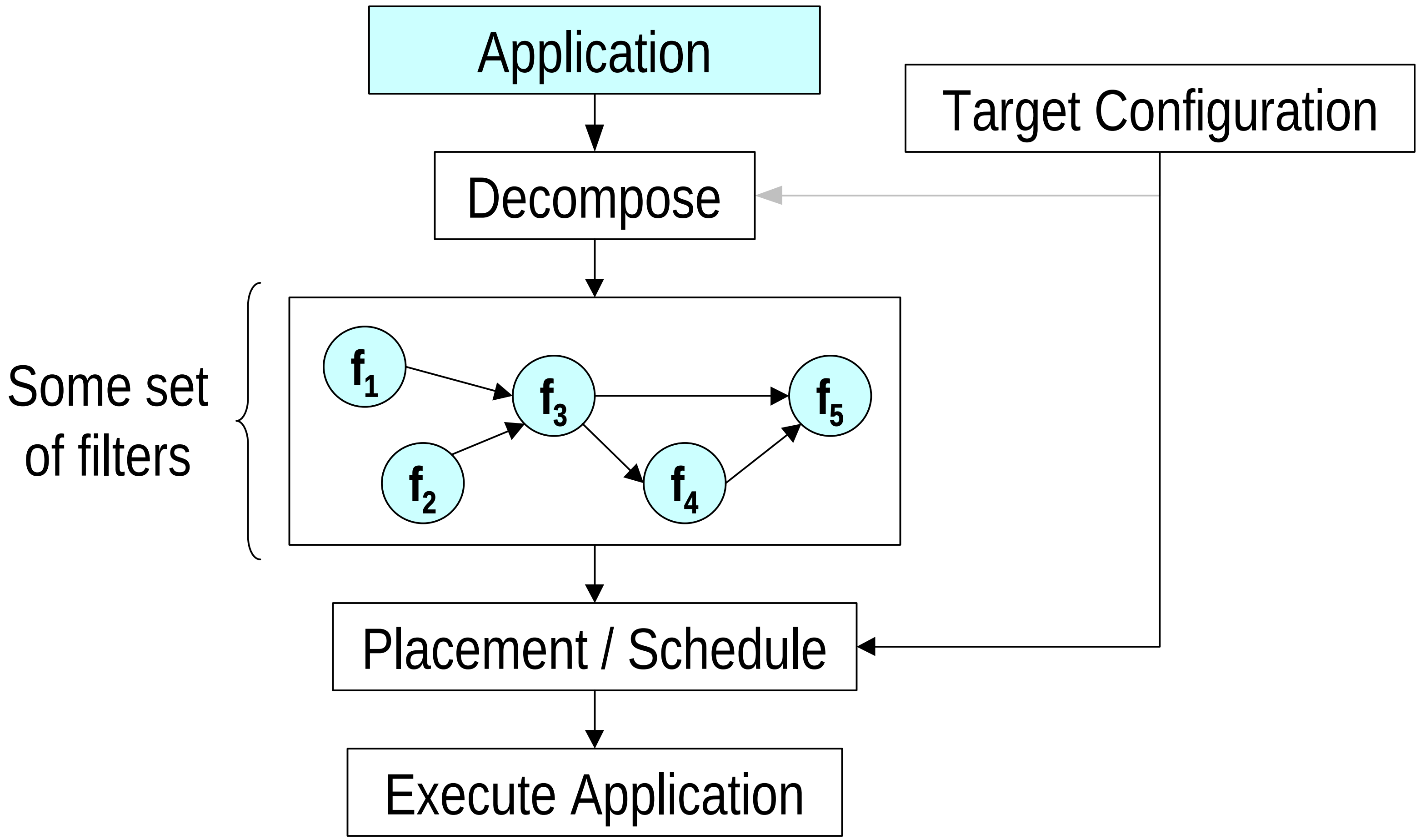
Filters

- **Filters**
  - communicate with other filters *only* using streams
  - cannot change stream endpoints
  - are allowed to pre-disclose dynamic allocation of memory/scratch space in *init* phase, before *processing* phase
- **Advantages**
  - location independence
  - easier scheduling of resources
  - filter stop and restart is defined explicitly in model

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

Filter Framework

```
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

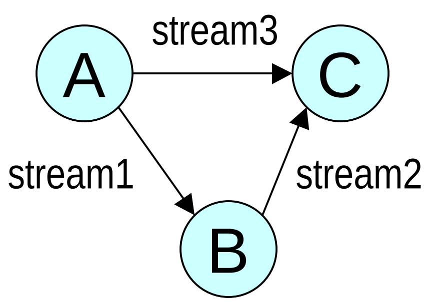
[filter.B]

ins = stream1

outs = stream2

[filter.C]

ins = stream2 stream3



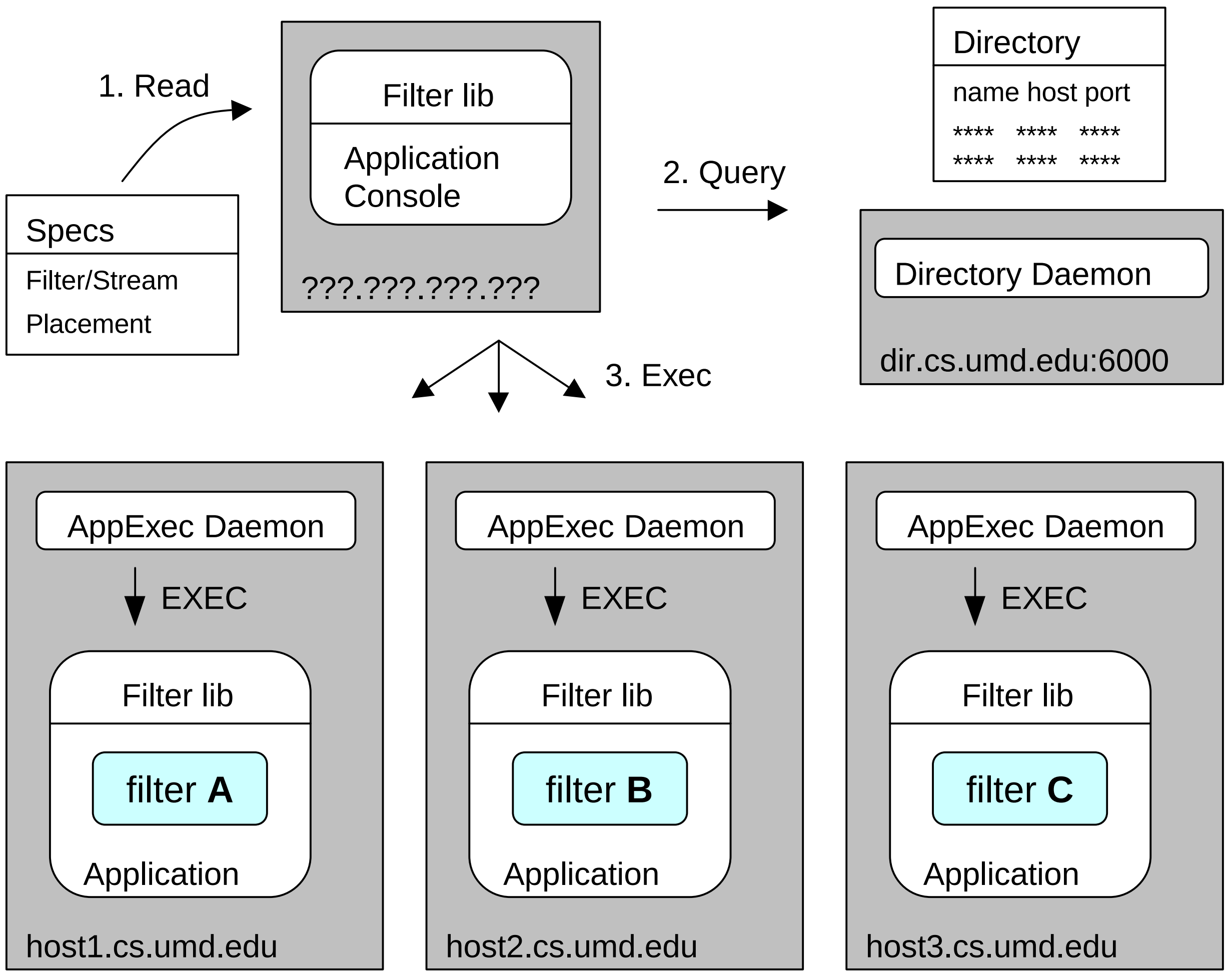
[placement]

A = host1.cs.umd.edu

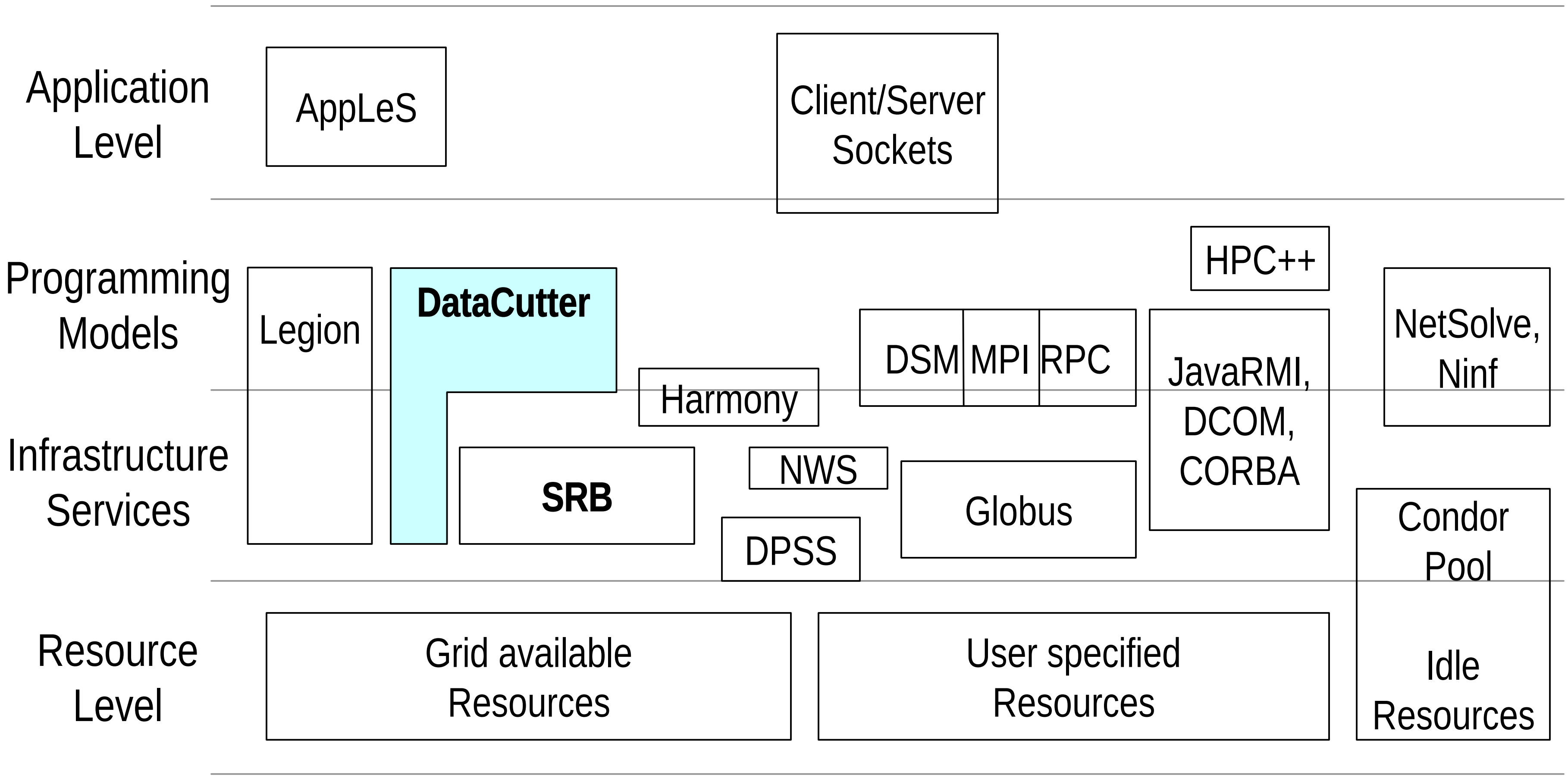
B = host2.cs.umd.edu

C = host3.cs.umd.edu

Execution Service



Related Work



Integrating DataCutter with the Storage Resouce Broker

Storage Resource Broker (SRB)

- **Middleware between clients and storage resources**
- **Remote Access to storage resources.**
  - Various types :
    - File Systems - UNIX, HPSS, UniTree, DPSS (LBL).
    - DB large objects - Oracle, DB2, Illustra.
  - Uniform client interface (API).

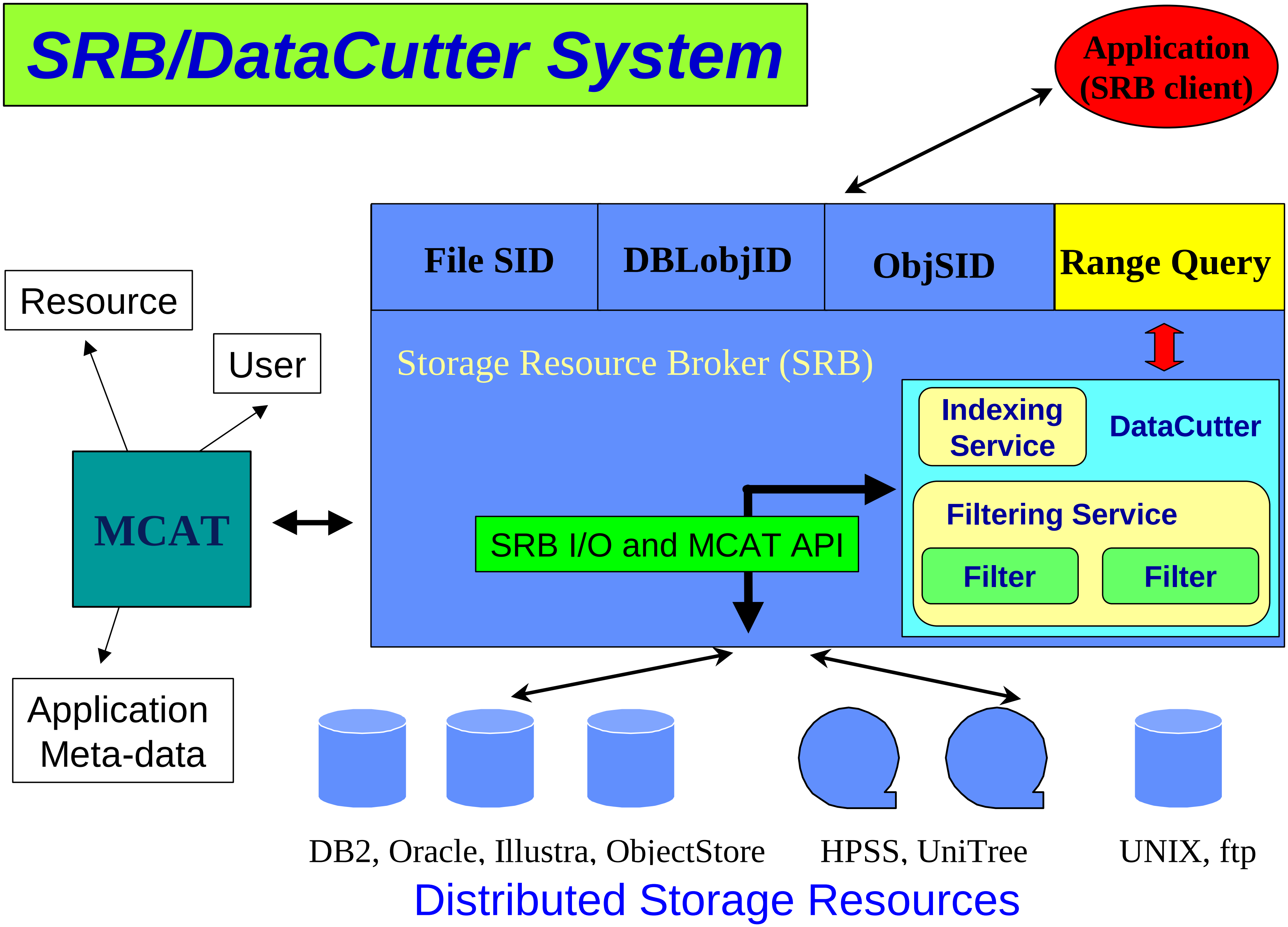
Storage Resource Broker (SRB)

- **MCAT - MetaData Catalog**
  - Datasets (files) and Collections (directories) - inodes and more.
  - Storage resources
  - User information - authentication, access privileges, etc.
- **Software package**
  - Server, client library, UNIX-like utilities, Java GUI
  - Platforms - Solaris, Sun OS, Digital Unix, SGI Irix, Cray T90.

SRB/DataCutter - Prototype Implementation

- **Support for Range Queries**
  - Creation of indices over data sets (composed set of data files)
  - Subsetting of data sets
    - Search for files or portions of files that intersect a given range query
  - Restricted filter operations on portions of files (data segments) before returning them to the client (to perform filtering or aggregation to reduce data volume)

SRB/DataCutter System



SRB/DataCutter Client Interface

- **Creating and Deleting Index**

```
int sfoCreateIndex(srbConn *conn, sfoClass class, int catType,
                   char *inIndexName, char *outIndexName,
                   char *resourceName)
```

```
int sfoDeleteIndex(srbConn *conn, sfoClass class, int catType,
                   char *indexName)
```

SRB/DataCutter Client Interface

- **Searching Index -- R-tree index**

```
typedef struct {
  int    dim;   /* bounding box dimensions   */
  double *min;  /* minimum in each dimension */
  double *max;  /* maximum in each dimension */
} sfoMBR; /* Bounding box structure */

typedef struct {
  sfoMBR   segmentMBR;      /* bounding box of the segment */
  char     *objID;          /* object in SRB that contains the segment */
  char     *collectionName; /* collection where object is stored */
  unsigned int offset;      /* offset of the segment in the object */
  unsigned int size;        /* size of segment */
} segmentInfo; /* segment meta-data information */

typedef struct {
  int        segmentCount;  /* number of segments returned   */
  segmentInfo *segments;    /* segment meta-data information */
  int        continueIndex; /* continuation flag             */
} indexSearchResult; /* search result structure */
```

SRB/DataCutter Client Interface

- **Searching Index -- R-tree index**

```
int sfoSearchIndex(srbConn *conn, sfoClass class,
                   char *indexName, void *query,
                   indexSearchResult *myresult,
                   int maxSegCount)
```

```
typedef struct {
    int    dim;
    double *min, *max;
} rangeQuery;
```

```
int sfoGetMoreSearchResult(srbConn *conn, int continueIndex,
                           indexSearchResult *myresult,
                           int maxSegCount)
```

Applying Filters

```
typedef struct {
   segmentInfo segInfo;  /* info on segment data buffer after filter oper. */
   char        *segment;    /* segment data buffer after filter is applied      */
} segmentData;

typedef struct {
    int         segmentDataCount;   /* #segments in segmentData array */
    segmentData *segments;       /* segmentData array */
    int         continueIndex;          /* continuation flag */
} filterDataResult;
```

Applying Filters

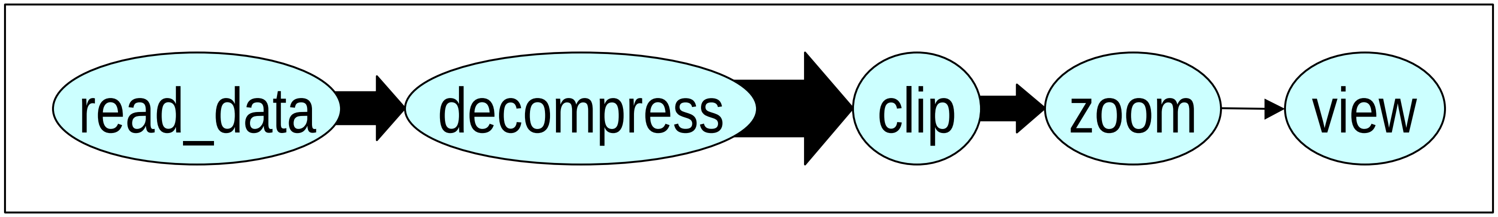
```
int sfoApplyFilter(srbConn *conn, sfoClass class, char *hostName,
                    int filterID, char *filterArg,
                    int numOfInputSegments,
                    segmentInfo *inputSegments,
                   filterDataResult *myresult,
                   int maxSegCount)
```

```
int sfoGetMoreFilterResult(srbConn *conn, int continueIndex,
                            filterDataResult *myresult,
                            int maxSegCount)
```

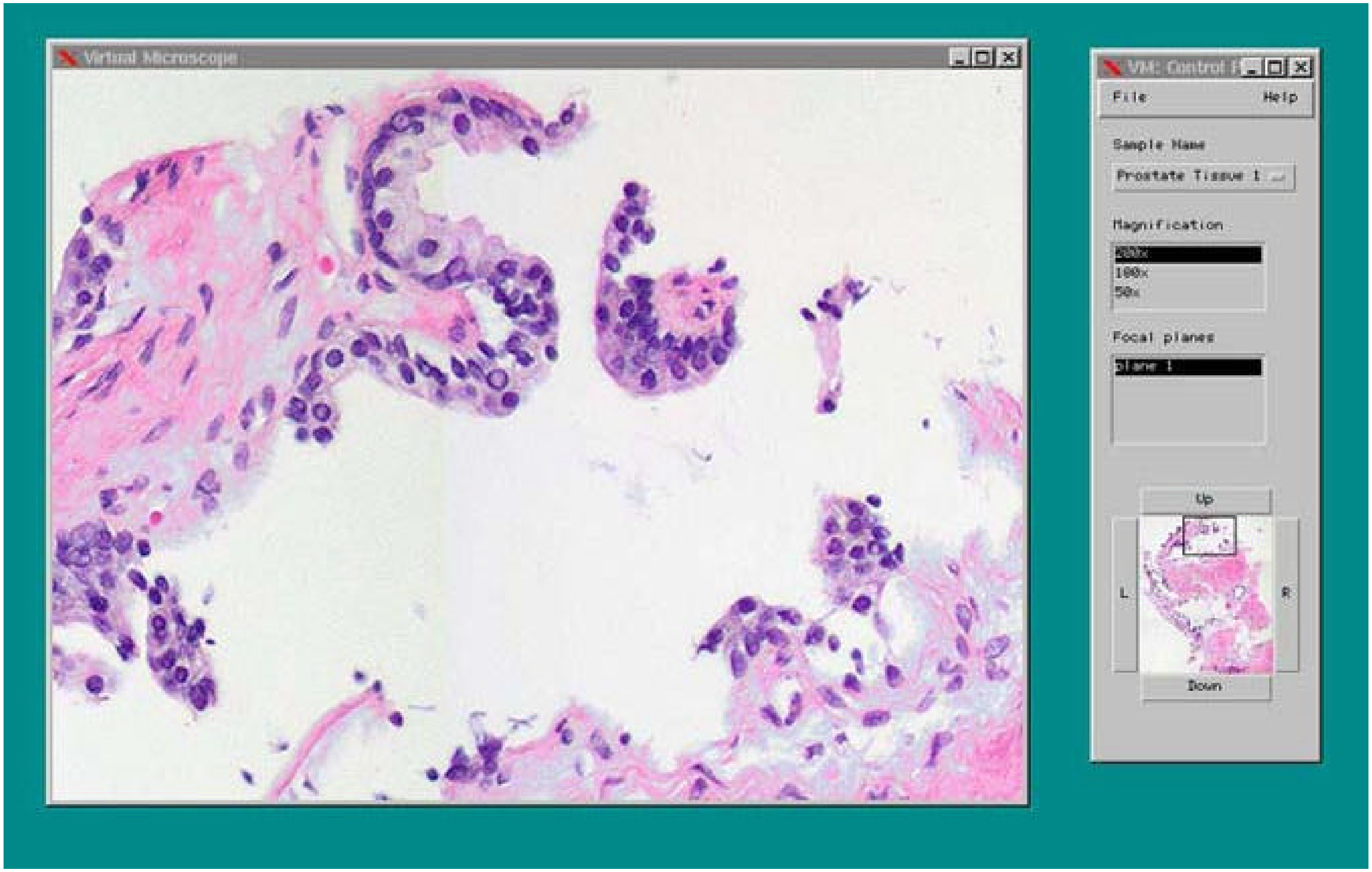
Application: Virtual Microscope

- **Interactive software emulation of high power light microscope for processing/visualizing image datasets**
- **3-D Image Dataset (100MB to 5GB per focal plane)**
- **Client-server system organization**
- **Rectangular region queries, multiple data chunk reply**

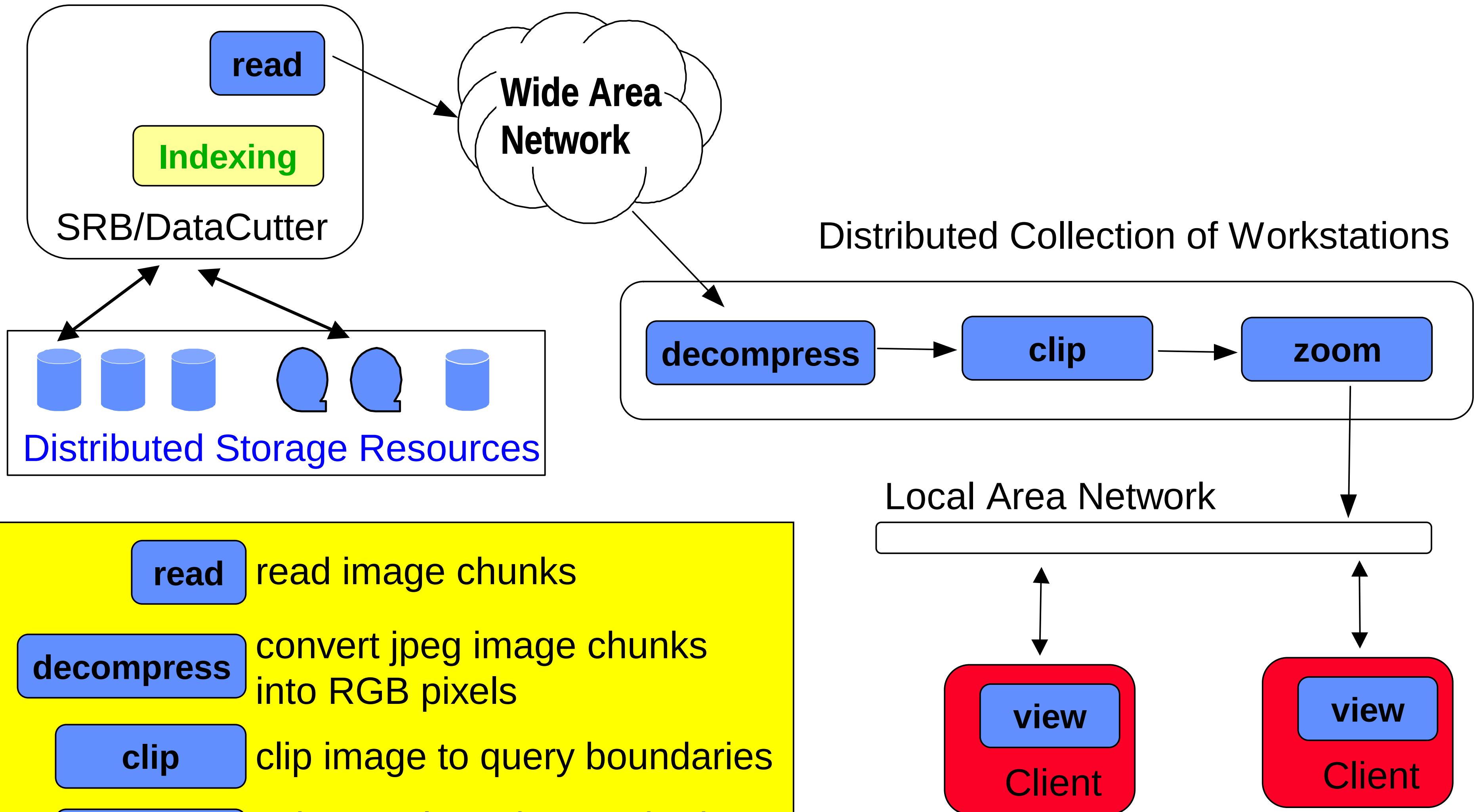
- **pipeline style processing**



Virtual Microscope Client



VM Application using SRB/DataCutter



read	read image chunks
decompress	convert jpeg image chunks into RGB pixels
clip	clip image to query boundaries
zoom	sub-sample to the required magnification
view	stitch image pieces together and display image