CMS Grid & Virtual Microscope

Malina Kirn

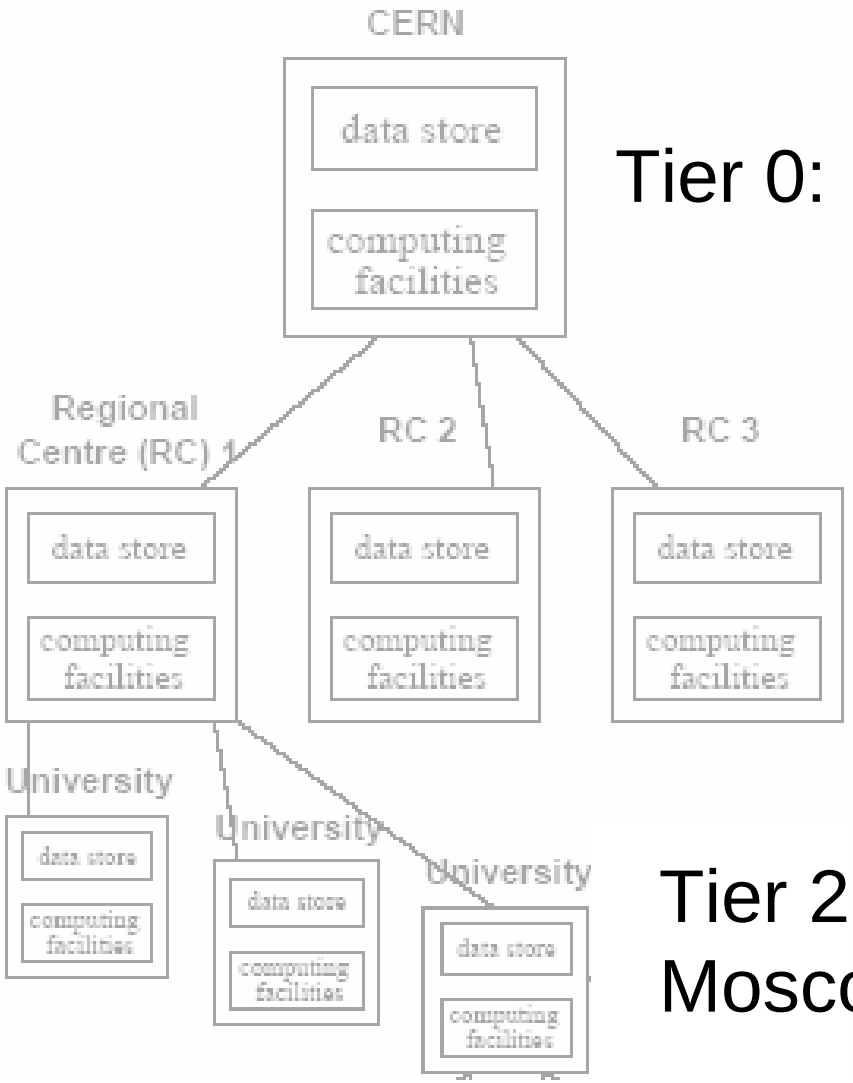
May 1, 2007

CMS Grid

Physics Terminology

- Monte Carlo (MC, aka *generation* or *simulation*): simulated physics; used to test code and to compare to real data
- *Reconstruction* (RECO): turning data in the form of electronic response (such as voltage) into particle physics events (such as particle type, trajectory and energy)
- Analysis: comparison of MC to reconstructed data for extraction of interesting physics

CMS Grid: Structure



Tier 0: CERN

Tier 1: CERN, FNAL, ...

this paper

Tier 2:
Moscow State U., MIT, UCSD, ...

CMS Grid: User Level Tasks

- Analysis of published datasets
  - Automatic resource and data discovery
- Production of specialized samples
  - User provided resources
- Detector studies
  - Dedicated and localized resources

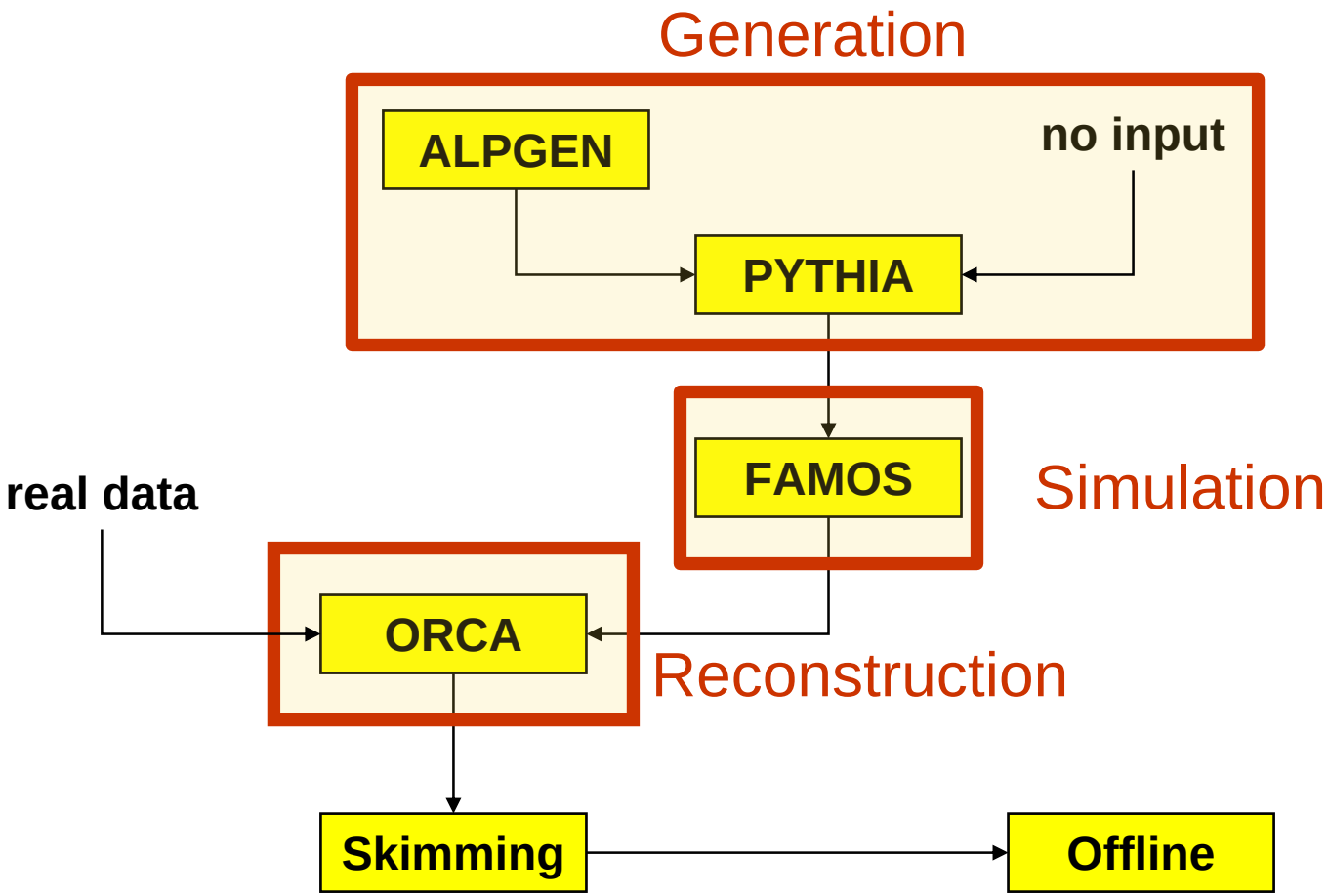
Test Environment

- All sites provide
  - CE = computing element
  - SE = storage element
  - WN = worker nodes
- Use Condor_g & srmcp
- 'Validation' job is run before mass submission to make sure site is responsive.

Software

- ALPGEN: theoretical physics simulator, produces *generation level* output, used for MC only
- PYTHIA: slightly more complicated theoretical physics simulator, produces *generation level* output, used for MC only
- FAMOS: takes generation level input and produces *simulation level* output (turns theoretical physics into what it looks like in an experiment), used for MC only
- ORCA: takes simulation level and *real data* input and produces *reconstruction level* output (turns experiment output into theoretical physics)

Workflow (newer than paper)



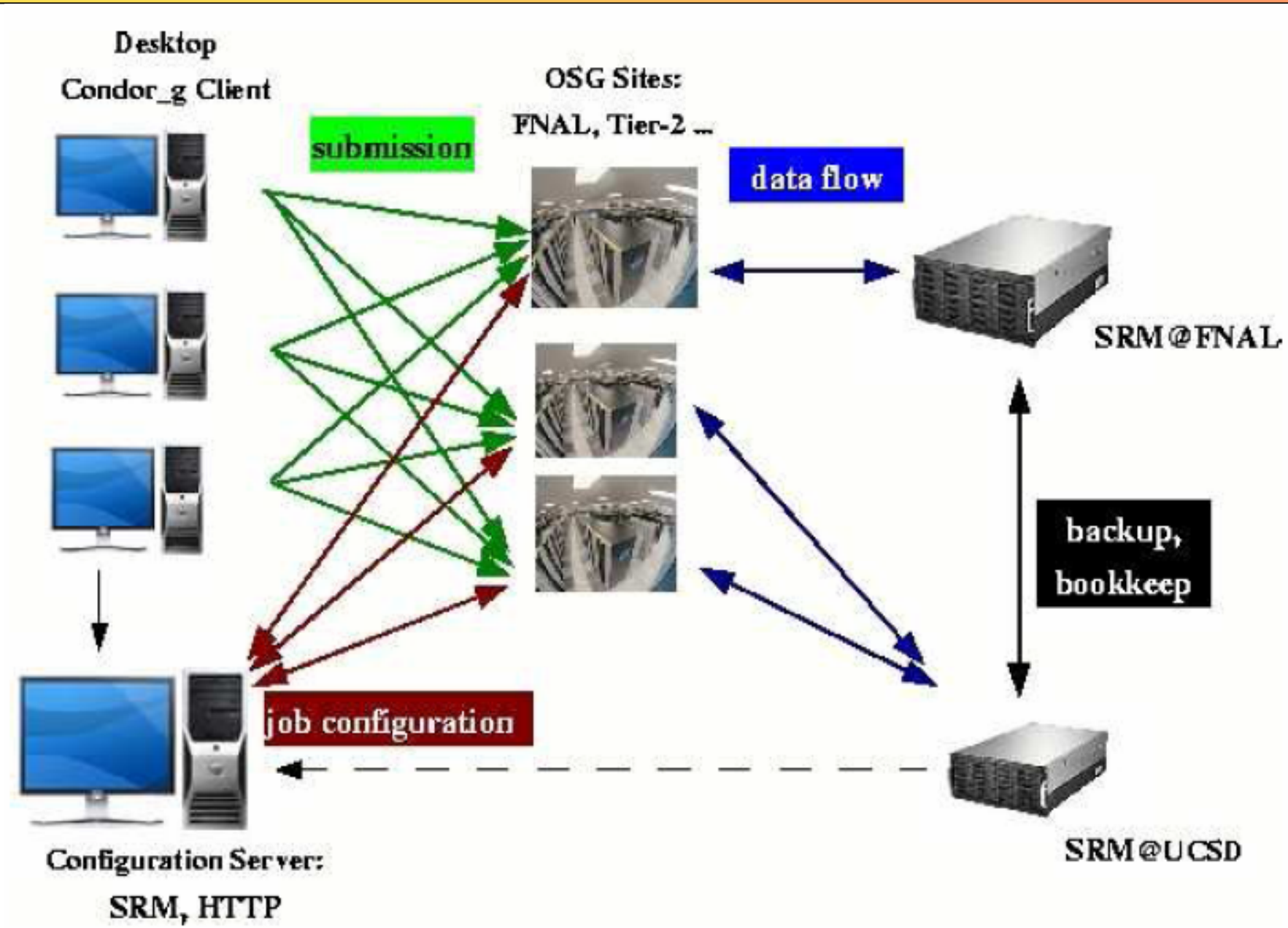
Statistics

Application Type	Events per job	Length per job (hour)	Total CPU time usage (hour)
ALPGEN generation	100-100k	5-6	$\sim 10^5$
ALPGEN-PYTHIA	10k	1	$\sim 10^3$
PYTHIA generation	5k	1	$\sim 10^5$
FAMOS	3k-10k	5-8	$\sim 0.8 \times 10^6$
ORCA	3k-5k	4-5	$\sim 2 \times 10^4$
Skimming	$\sim$ 300k	1-2	$\sim$ 300

Data Type	Events per file	Size per file (MB)	Total storage usage (GB)
Official Dataset			$\sim$ 20000
Text event from ALpgen	10-40k	10-40	$\sim$ 50
CMKIN N-tuples	5-10k	300-500	3000-4000
Root N-tuples	5-10k	30-50	300-400
Offline Root Files	$\sim$ 300k	500-1000	$<$30

Observations

- Data splitting and merging to keep the number of events per file constant
- PYTHIA data not kept (*simulation level*) to reduce storage consumption
- Output directed to only 1 or 2 SEs for simple retrieval
- No logical file naming due to fast evolution
- File names and locations provide metadata
- No automatic resubmission of failed jobs because human intuition is better, most failed jobs are never resubmitted
- Some jobs have access to information allowing them to know if an equivalent job has already been performed
- Grid job failure rate is similar to normal cluster failure rate

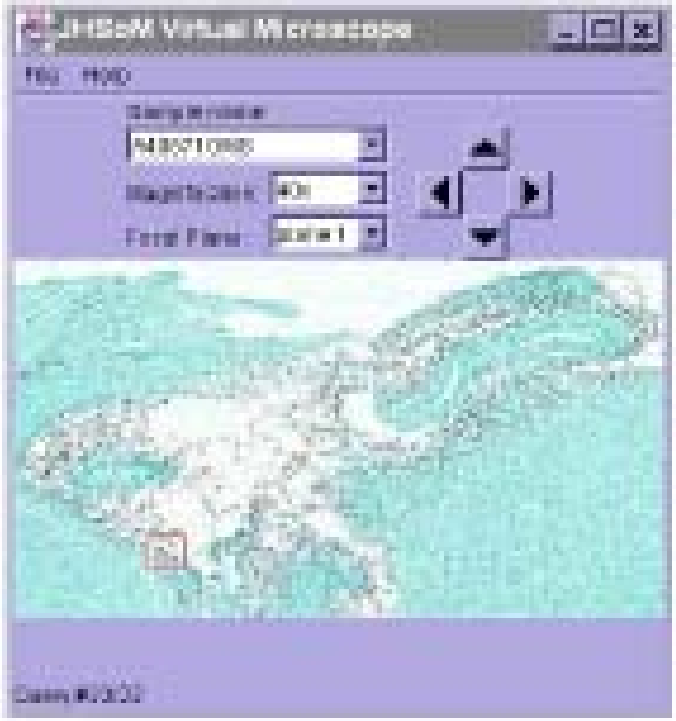

Desktop
Condor_g Client
OSG Sites:
FNAL, Tier-2 ...
submission
data flow
SRM@FNAL
backup,
bookkeep
job configuration
SRM@UCSD
Configuration Server:
SRM, HTTP

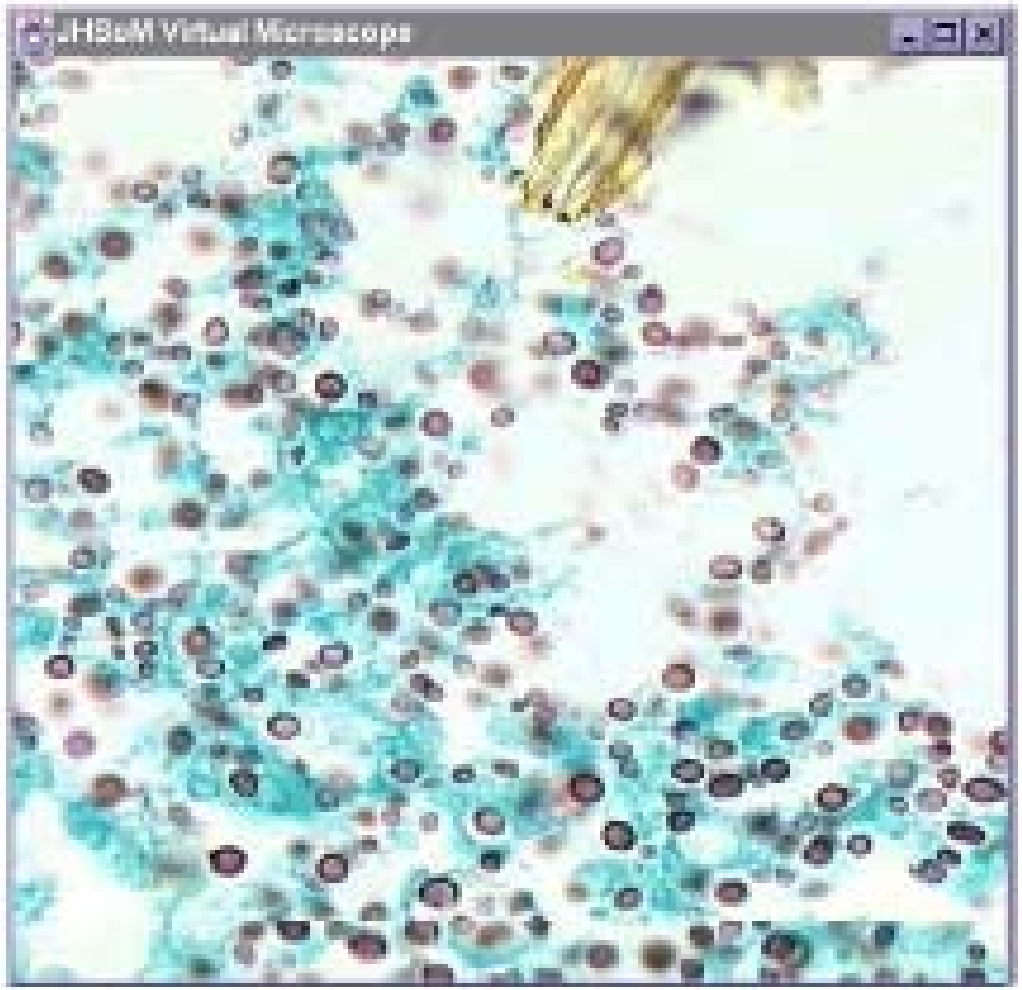
Parameter Tuning

- "The length of execution time for each job and output data are largely tuned to minimize the pressure on the infrastructure and maintain high efficiency"
- Running time < 6 hours for failures
- 10 MB < output < 200 MB for dCache (a hierarchical SRM)
- No directory holds more than a few thousand files for PNFS (?)

Virtual Microscope

Client GUI


JHSoM Virtual Microscope
File Help
Magnification:
Focal Plane:


JHSoM Virtual Microscope

Server Frontend

- Interprets client queries as database queries
- Allows asynchronous client queries to the data server
- Handles clients behind firewalls
- Generally, responses are sent directly to the client from the data server, so frontend generally handles queries, not responses

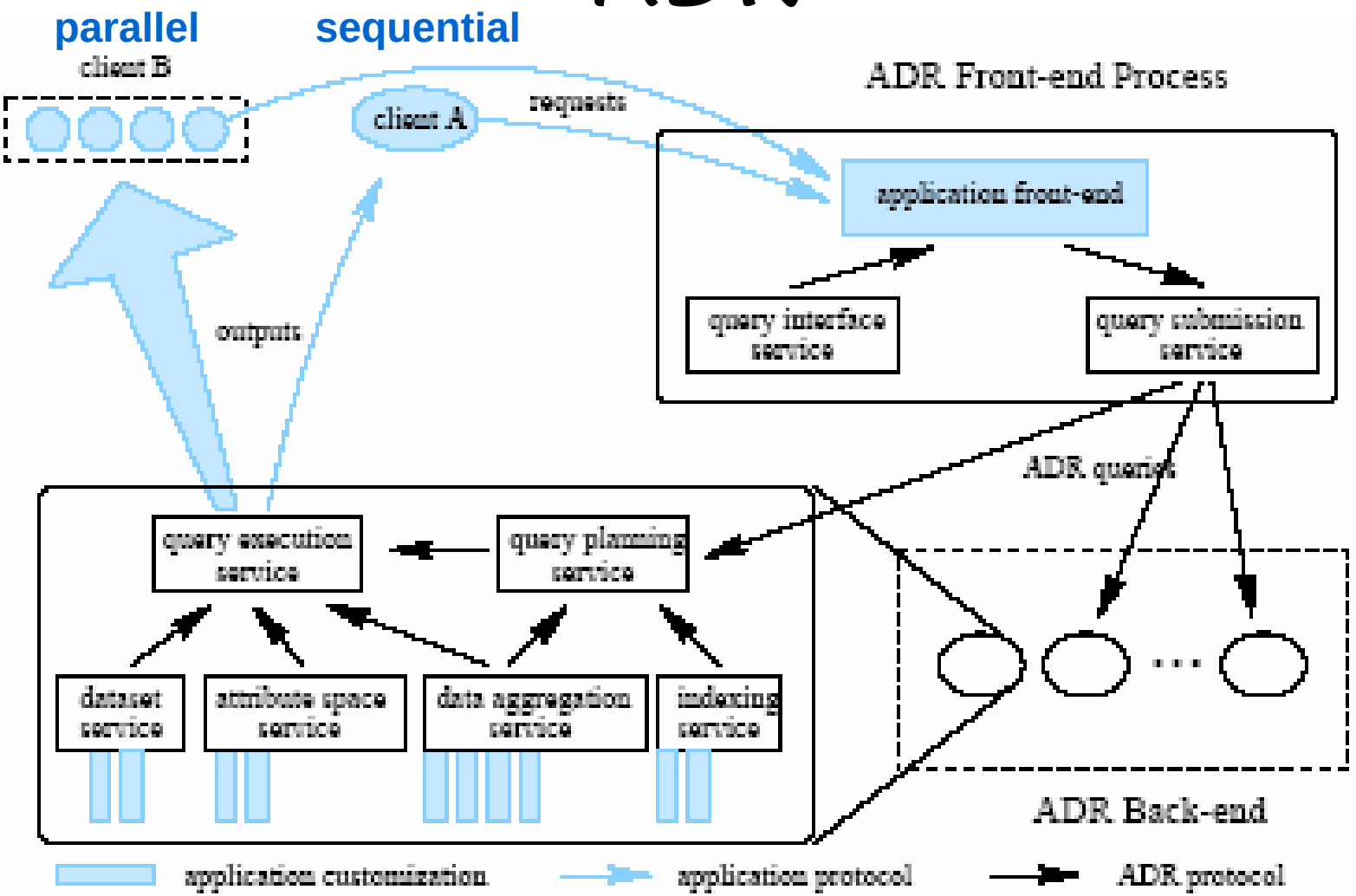
Data Server

- Fast retrieval of data from disk to memory
  - Low latency
  - Efficient directory structure
- Fast processing of queries on data in memory
  - Project high resolution onto resolution corresponding to magnification requested
  - Scalable: parallel or computing cluster
  - Asynchronous operation on multiple blocks

Active Data Repository (ADR)

- Describes every piece of data with coordinates in n-dimensional coordinate space (VM is 3D)
- Retrieves all data inside a range query (range specified in each dimension)
- Aggregates appropriate input data to output data through intermediate accumulator, provided by application

Application Deployment on ADR



ADR Back-end

- *Attribute space service* manages registration and use of application-defined mapping functions
- *Dataset service* manages datasets stored in the ADR back-end and provides utility functions for loading datasets into ADR
- *Indexing service* manages various indices (default and user-provided) for the datasets stored in ADR
- *Data aggregation service* manages application-provided functions to be used in aggregation operations, and functions to generate the final outputs

Datasets in ADR

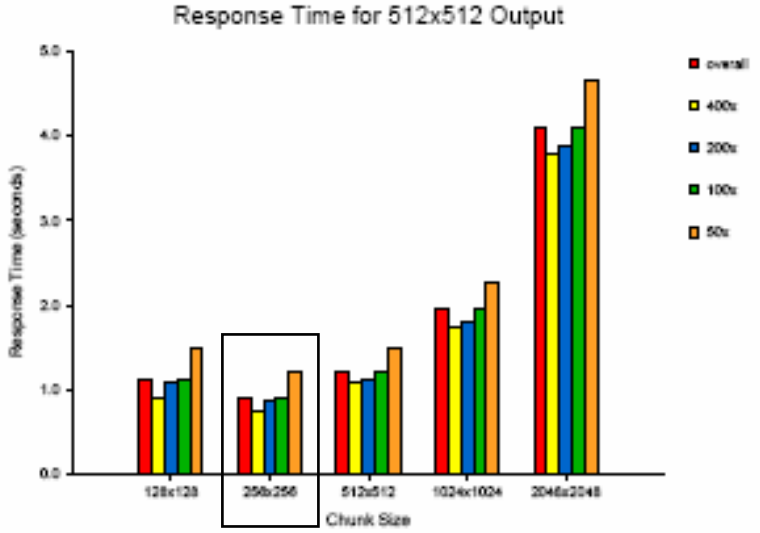
- Data stored and retrieved as chunks for bandwidth where chunks specified by range in n-dimensional space
- Chunks nearby in n-dimensional space distributed evenly across disk space using Hilbert curves by default

Processing in ADR

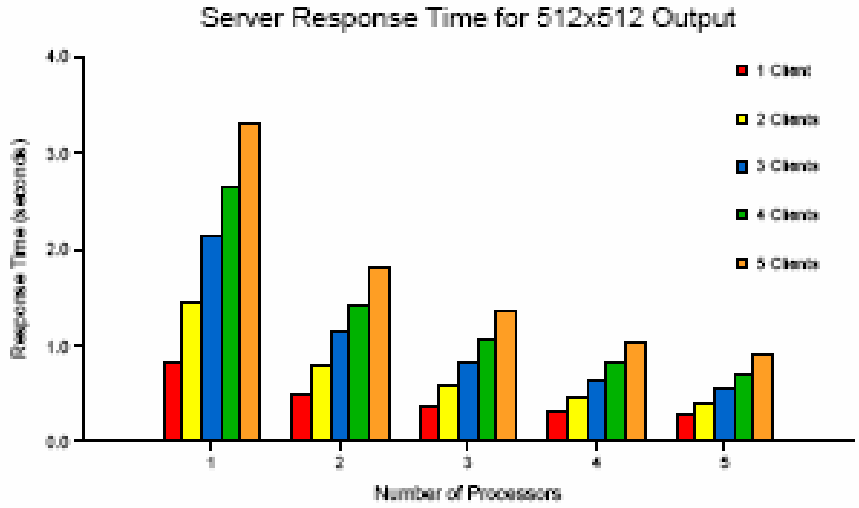
- Query planning
  - Index lookup: find blocks in coordinate range
  - Tiling: if all data doesn't fit in memory
  - Workload partitioning: local accumulator processing
- Query execution
  - Initialization: memory allocated, process started
  - Local reduction: local accumulator
  - Global combine: inter-processor combination of local reductions
  - Output handling: output from combined accumulators

VM on ADR

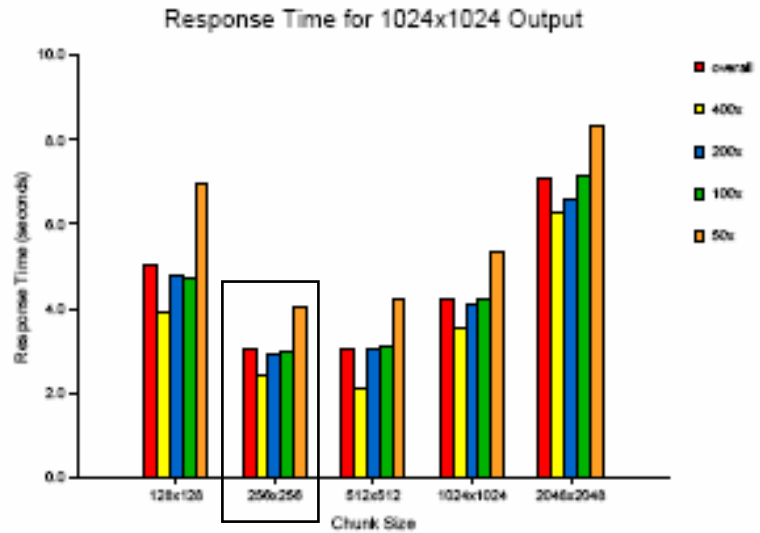
- 2D space used for Hilbert distribution of data across disks
- Third dimension of data kept clustered
- ADR block size tweaked for both disk access efficiency and minimal data waste
- JPEG compression
- Query process: retrieve, decompress, clip (crop), subsample
- Results of local processing on ADR blocks sent directly to client (no global combine)



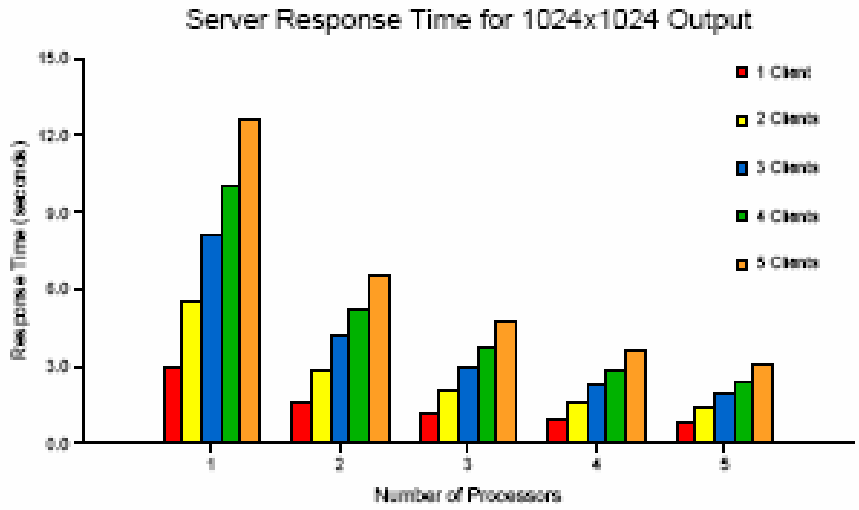
(a) 512x512 output image



(a) 512x512 output image



(b) 1024x1024 output image



(b) 1024x1024 output image

DataCutter

- Indexing service
  - Indexes data by position in n-dimensional space
- Filtering service
  - Supports execution of groups of processes (filters) on different machines concurrently, independently, or in order (emulate parallel, asynchronous, or sequential over different machines)
  - Communication between filters termed a stream, uni-directional pipes that deliver data in fixed size buffers

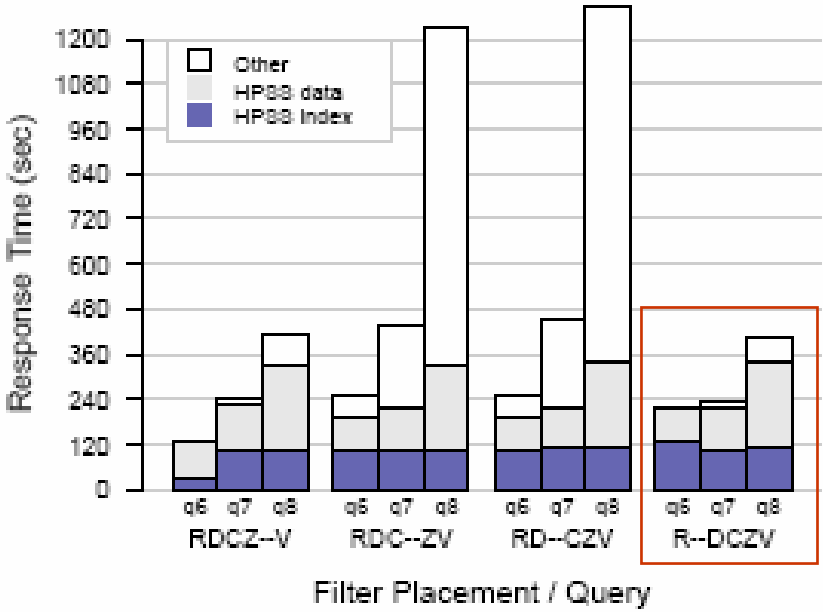
Multi Level Indexing

- Summary index files
  - Associate metadata with one or more data chunks and/or index files
- Detailed index files
  - Provides metadata for all data chunks in some data set

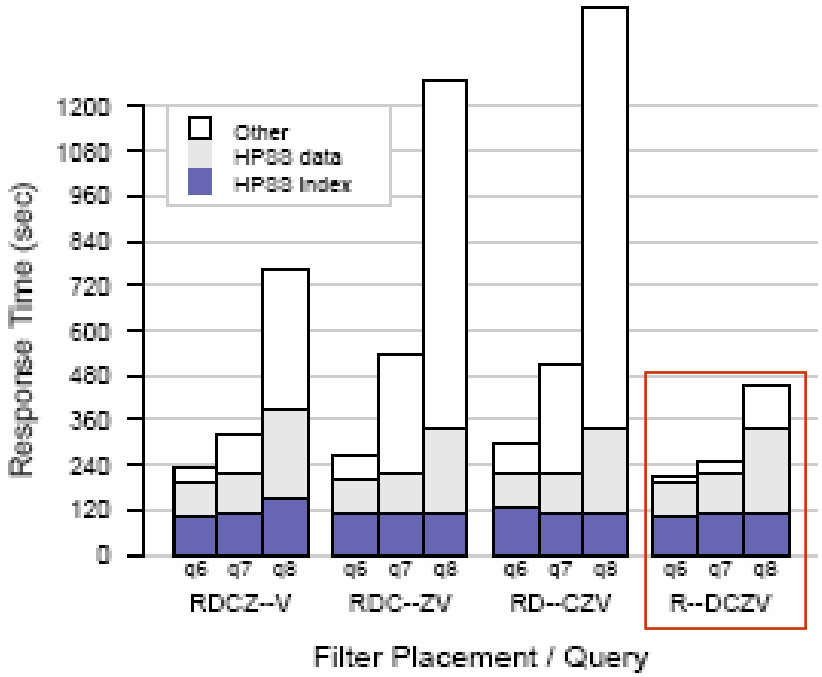
Filter-Stream Optimizations

- Filters placed as locally as possible to the data source
- Transparent copies: identical filters that run concurrently with the goal of no single filter causing a bottleneck
  - How to distribute the buffer?
    - Round robin
    - Round robin based on number of copies on a single host
    - Demand driven (who has been the best performer so far?)

VM on DataCutter



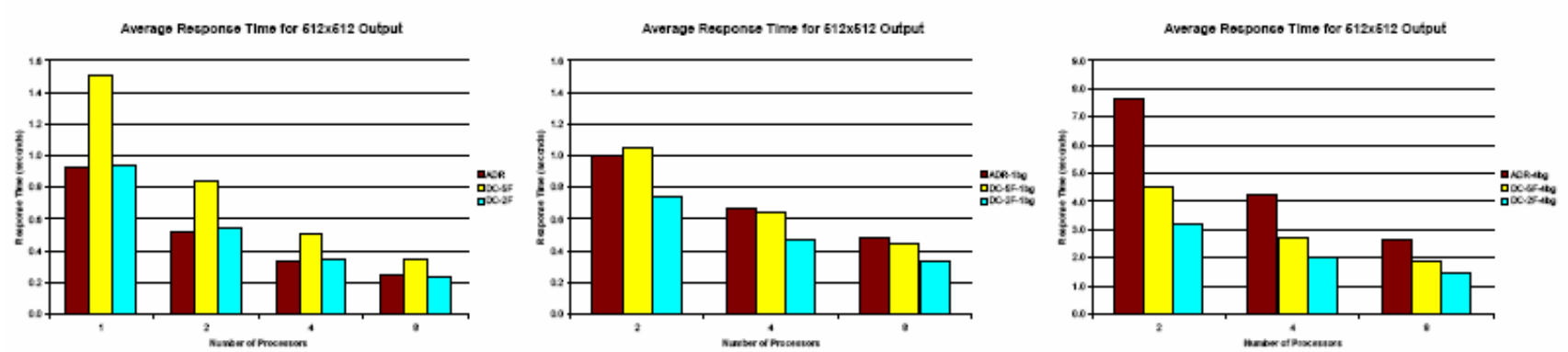
(a) Server has no added load



(b) Server is loaded to double execution time of only the filters on the server

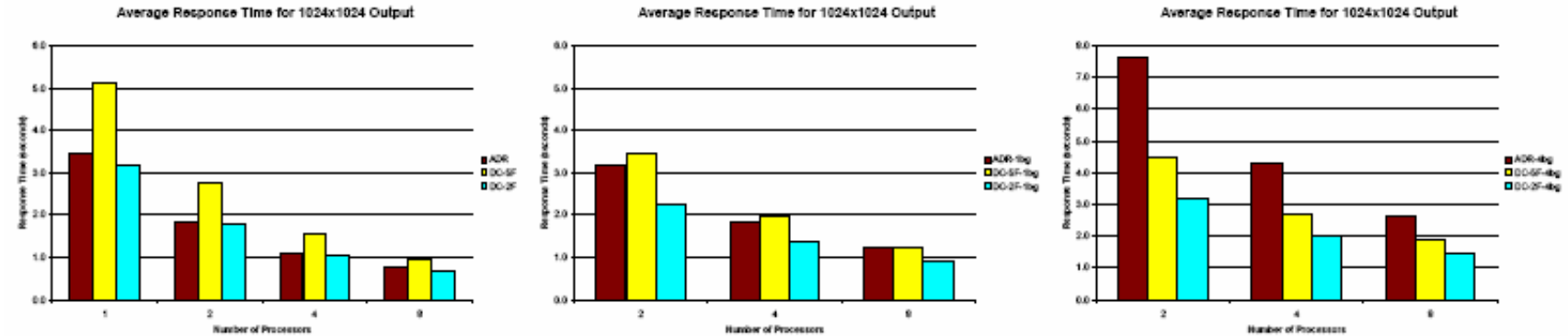
- read_data (R), decompress (D), clip (C), zoom (Z), view (V)

ADR vs DataCutter



(a) No background load (b) 1 background job (c) 4 background jobs

Fig. 11. Average response time of the VM servers for 512x512 output image.



(a) No background load (b) 1 background job (c) 4 background jobs

Fig. 12. Average response time of the VM servers for 1024x1024 output image.

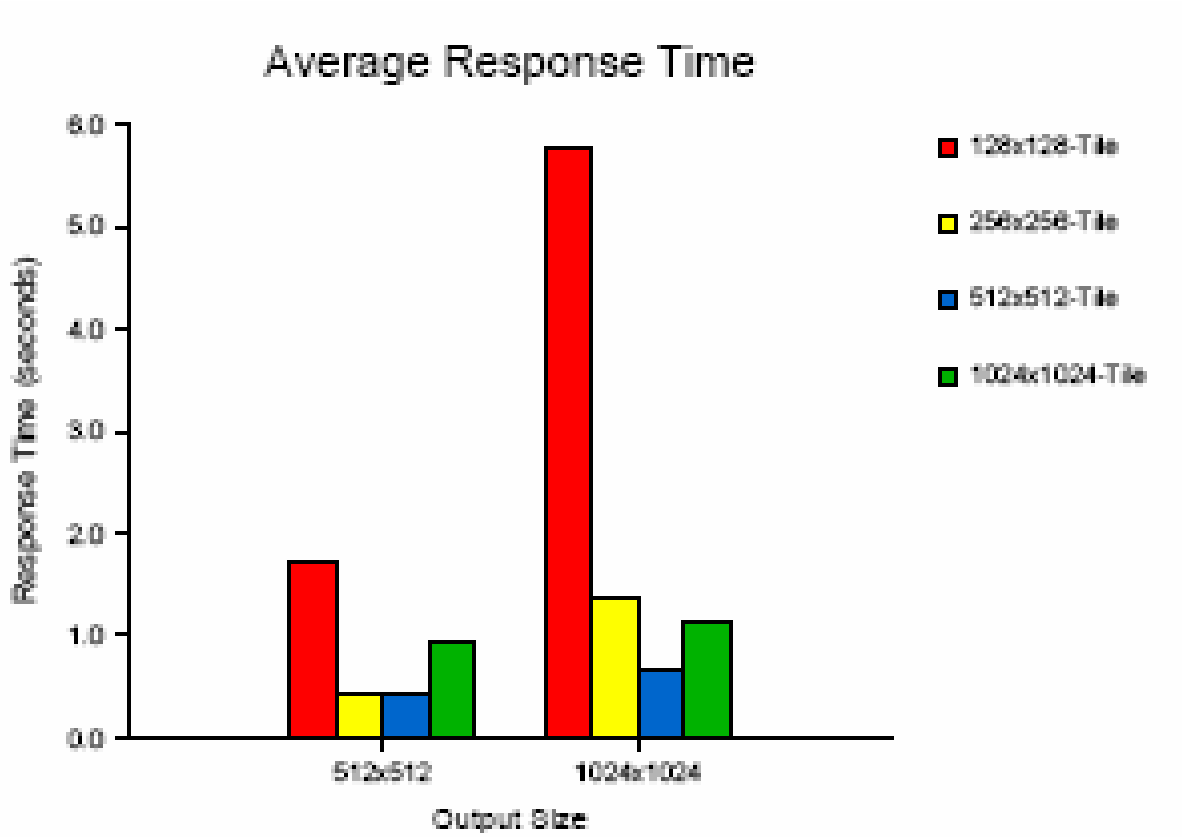
ADR vs DataCutter

- For a small number of clients (e.g., 1 or 2) the ADR server performs better than the DataCutter server versions.
- DataCutter server configurations with fewer instances, but more transparent copies per instance, achieve better performance for small numbers of clients.
- When the number of clients increases, the DataCutter implementations of the VM server with multiple group instances perform better.
- ADR server scales better for bigger queries.
- DataCutter provides sufficient flexibility so that the data server configuration can be modified to accommodate various data access patterns.

Workload Reduction

- Sent image is compressed
- Proxy caching
  - Intermediate proxy between co-located clients and the server caches data
  - Only useful when clients are near proxy and clients share data interest
  - Results in cache miss penalty
- Client caching
  - Two-level: memory and disk, LRU
  - Clients try to perform magnification operations themselves

Client Caching



Questions

- "…appropriately compositing pixels that map to a single grid point, to avoid introducing spurious artifacts into the displayed image." Map to a single grid point?
- R-tree index for locating data points? Is this the same as Multi Level Indexing?
- What's the difference between the attribute space and indexing services in ADR?

Questions

- How do they handle the problem of 52 GB of data/slide?
- Is the first comparison of DataCutter to ADR when having DataCutter processes run entirely on the data server?
- Are streams reconfigurable dynamically?
- dCache & SRM:
  - http://www.dcache.org/manuals/chep2003.pdf

Questions

- Comment: Multiple users should be able to share some common cache or a proxy to reduce server workload.
- Answer:

```
@InProceedings{ BEYNON99,
  author = "Michael Beynon and
                Alan Sussman and Joel Saltz",
  title = "Performance Impact of Proxies in Data Intensive
          Client-Server Applications",
  booktitle = "Proceedings of the 1999 International
                Conference on Supercomputing",
  year = 1999,
  publisher = "ACM Press",
  month = Jun
}
```