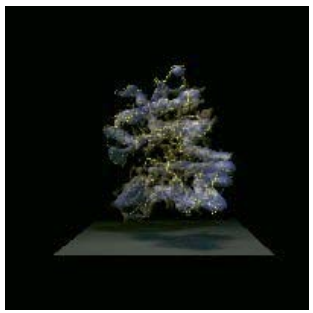
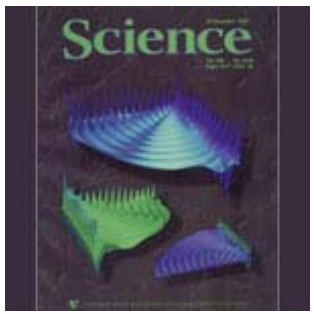
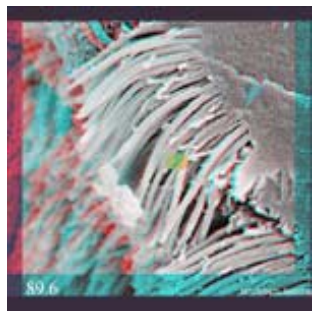
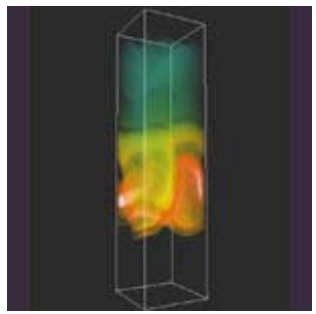
Considerations for a Distributed Visualization Architecture (DiVA)

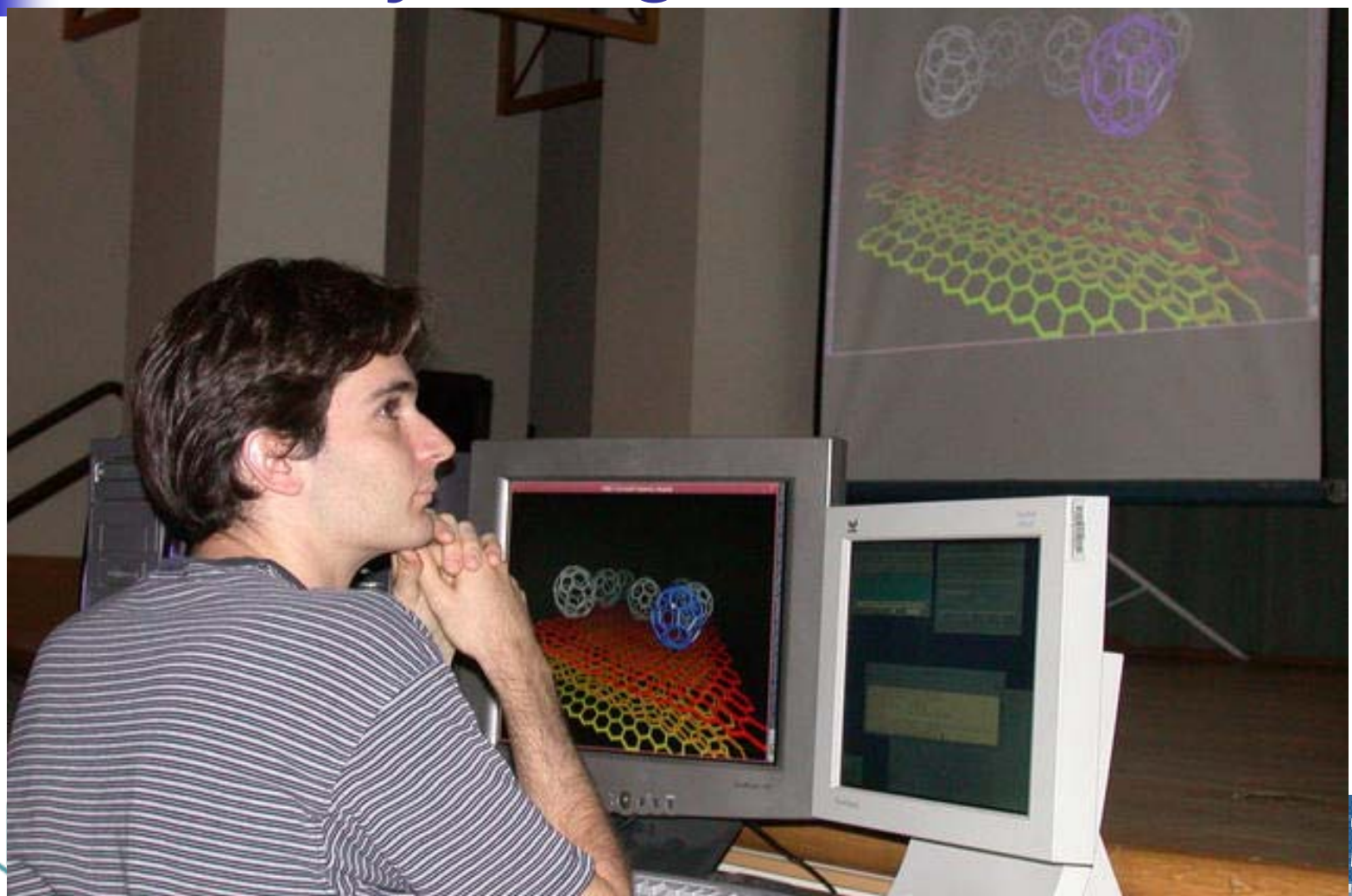



John Shalf
NERSC/CRD Visualization Group

BERKELEY LAB

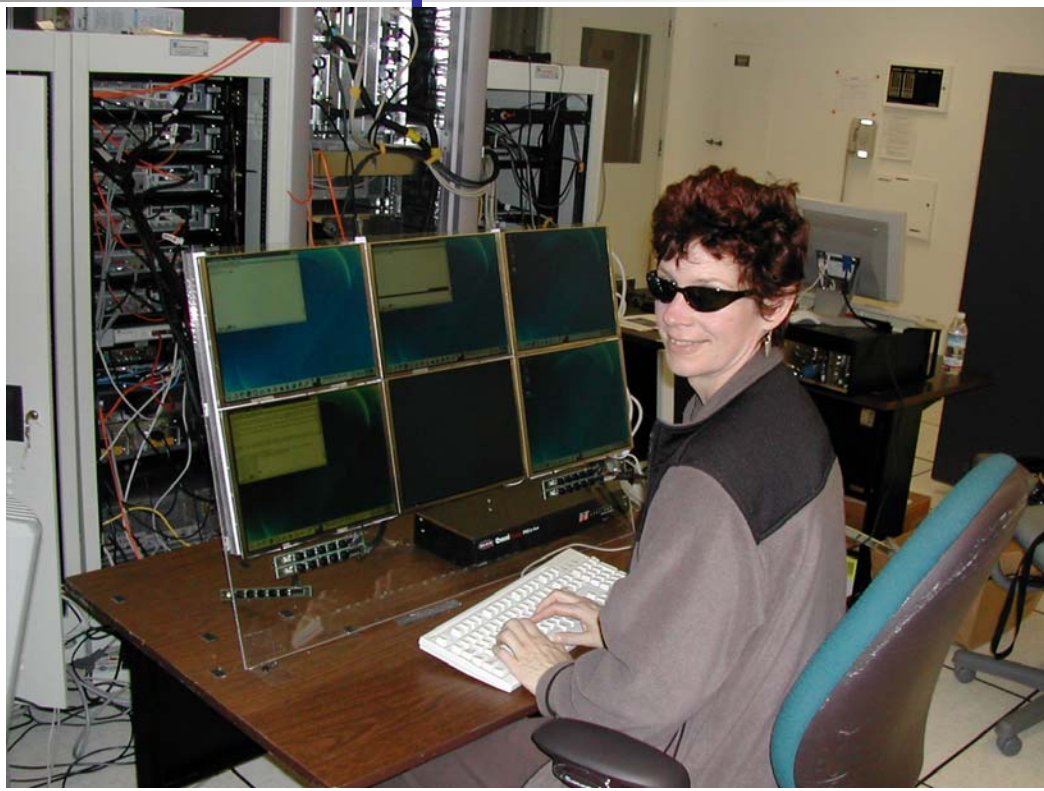
Office of Science
U.S. DEPARTMENT OF ENERGY

The way things were…



But it got more complicated…

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.



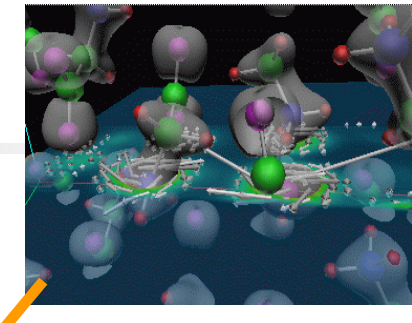
And even more complicated…

Shaky City

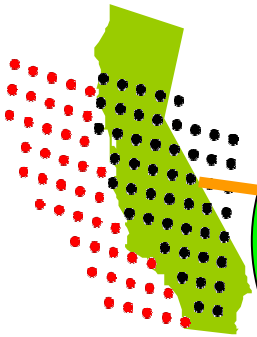
Data Caches

HPC Resources

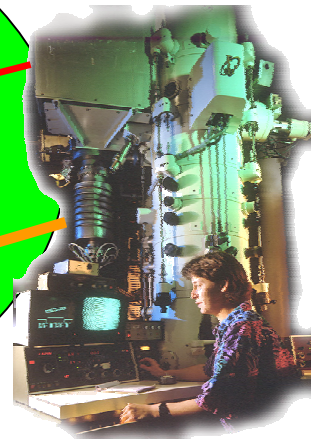
Simulations



Sensor Nets



STM



Collaborators

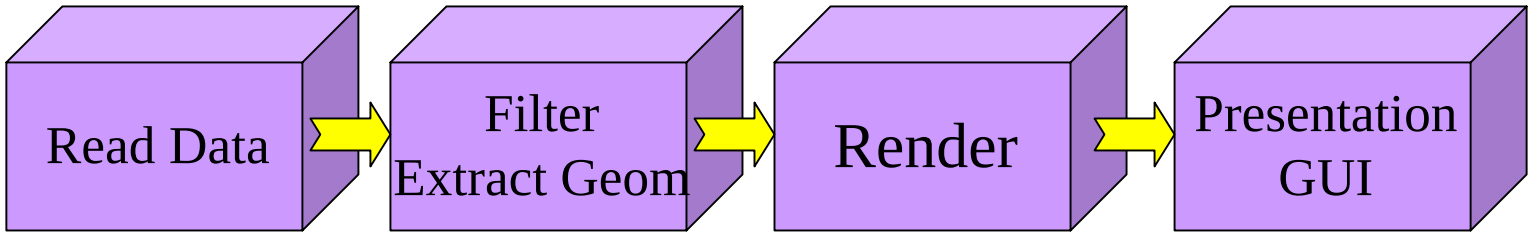
Dr Jane

Handheld Devices

Berkeley Lab

Office of Science
U.S. Department of Energy

Canonical Data Analysis Pipeline



Remote data analysis applications attempt to optimize pipeline

- Repartition the pipeline
- Collapse stages of the pipeline
- Parallelization: SIMD and pipelined
- Improve throughput between stages
  - Data reduction / Progressive Transmission (info proc & encoding)
  - Protocol/transfer acceleration (hardware, drivers & protocols)
- Each method optimal for a narrow set of conditions

Where are we now?

- Despite years of effort and demonstrations of remote vis technology, users predominantly use serial desktop tools
  - Download data to workstation and use locally
  - Use serial tools over remote X11 connections (just to avoid moving the data to a local workstation… that's desperation!)
- Fractured component technology and remote vis efforts
  - Open Source Frameworks (Parallel VTK, OpenDX)
  - Commercial tools/frameworks (CEI Ensight, AVS Express, …
  - Standalone tools (VisIT, Visapult, Terascale Browser)
  - Lack of generality
- Do any of these tools offer a comprehensive solution that works on the emerging Supercomputer Architectures?
  - No?
- Will they ever interoperate?
  - Not likely without common architecture to write to…

We Need a DiVA!

A "Distributed Visualization Architecture"

- We will not be able to tackle emerging data analysis problems without distributed/parallel remote visualization systems!
  - Remote visualization has repeatedly demonstrated advantages
- We won't be able to do remote/distributed visualization effectively without a common framework that enables us to share/combine our work!
  - There has been no common delivery platform to enable pervasive adoption by users
- Frameworks/Architectures are
  - Rigid formalisms encoding (*enforcing*) best practices
  - A way to encode for well-understood (*menial*) tasks so developers can focus on high level concepts
  - A way to encode things we understand and have already thought out (*familiar/commonly used techniques are what we consider "menial"*)
  - A method that does not readily accommodate new concepts *(but what does?)* So we should expect to primarily encode current practices.

What to Expect of a "Distributed Visualization Architecture" (DiVA)

- Modular component framework supporting community contributions
  - Supports discoververy of distributed/parallel components
  - Supports remote analysis (*eg. Latency tolerance, desktop interactivity*)
  - Supports streaming/out-of-core/progressive execution model
- Decouple BackEnd distributed components from presentation/GUI
  - Permits reuse of same compute-intensive components for different presentation methods and interfaces contexts
  - Means we need a standard way to talk to back end components
  - OGSA for visualization tools? (*grid speak for service abstraction…)*
- Requires Robust internal data model(s)
  - Essential feature of other community frameworks like OpenDX, AVS, and VTK
  - Encode basic vis & science data structures *(FEM, Geometry, Block-structured)*
  - Domain Decomposition, hierarchical representations, progressive encoding, information indices (*commonly neglected in current frameworks!)*
  - *Must end current balkanization of data formats / data models.*

What to Expect of a DiVA (cont…)

- Effortless selection and placement of components on distributed computers and load-balancing
  - Requires a mature Grid (eg. Grid Application Toolkits)
  - Requires common data model (or collection thereof)
  - Requires robust performance model and runtime instrumentation for “Mapping”
- Basic Data Transport between network-connected components
  - Stream/discretized : reliable/unreliable)
  - Negotiate QoS with new switched circuit networks.
  - Can leverage heavily on data model for higher level info representation
- Integration with Storage Resource Management
  - Replica Catalogs and shared virtual file spaces
  - Includes data staging, cataloging, scheduling of preprocessing tasks
  - Essential for efficient use of scarce network resources
- Needs are applicable beyond interactive visualization!
  - Data Mining, feature extraction, data summarization (batch)
  - Interactive Visualization and Analysis (interactive)
  - Data Preprocessing, reorg. and indexing, for interactive vis. (batch)

DiVA needs to do….

- All of the stuff that vis people do *not* want to do!
- All of the stuff that vis people are no good at doing!

Simple Example (security)

- Launching our distributed components
  - Secure launching
  - Authenticated sockets
  - Encrypted sockets

Vis Security (in practice)

- Commonly Used Security Options for Distributed Vis Applications
  - .rhosts
  - ssh
  - GSI/PKI
- Examples in “the wild”
  - SGI Vizserver: (who needs security? You’re on a VPN -- right??)
  - Ensight & Visapult (login to rmt. host)
  - VisIt & AVS3-5 (ssh to launch, but no authentication for TCP)
  - Triana (everything is fine as long as you use a JVM)

Vis Security (in practice)

- Commonly Used Security Options for Distributed Vis applications
  - .rhosts
  - ssh
  - GSI/PKI
- Examples in “the wild”
  - SGI Vizserver: (who needs security?)
  - Ensight & Visapult (login to rmt. host)
  - VisIt & AVS3-5 (ssh to launch, but no authentication for TCP)
  - Triana (everything is fine as long as you use a JVM)
- Overall Conclusion
  - Vis people suck at security
  - Security is not a core competency of vis application developers
  - We need domain-specific APIs *(simpler, easier, encode best practices)*

Copy Data -- Globus/GASS

```
int CopyFile (const char* source,
              const char* target)
{
  globus_result_t                       result;
  globus_url_t                          source_url;
  globus_io_handle_t                    dest_io_handle;
  globus_ftp_client_operationattr_t     source_ftp_attr;
  globus_gass_transfer_requestattr_t    source_gass_attr;
  globus_gass_copy_attr_t               source_gass_copy_attr;
  globus_gass_copy_handle_t             gass_copy_handle;
  globus_gass_copy_handleattr_t         gass_copy_handleattr;
  globus_ftp_client_handleattr_t        ftp_handleattr;
  globus_io_attr_t                      io_attr;
  int                                   output_file = -1;

  if ( globus_url_parse (source_URL, &source_url) != GLOBUS_SUCCESS )
  {
    printf ("can not parse source_URL \"%s\"\n", source_URL);
    return (-1);
  }


  if ( source_url.scheme_type != GLOBUS_URL_SCHEME_GSIFTP &&
       source_url.scheme_type != GLOBUS_URL_SCHEME_FTP    &&
       source_url.scheme_type != GLOBUS_URL_SCHEME_HTTP   &&
       source_url.scheme_type != GLOBUS_URL_SCHEME_HTTPS  )
  {
    printf ("can not copy from %s - unsupported protocol\n", source_URL);
    return (-1);
  }
```

```
  globus_gass_copy_handleattr_init  (&gass_copy_handleattr);
  globus_gass_copy_attr_init        (&source_gass_copy_attr);
  globus_ftp_client_handleattr_init (&ftp_handleattr);
  globus_io_fileattr_init           (&io_attr);
  globus_gass_copy_attr_set_io      (&source_gass_copy_attr, &io_attr);
  globus_gass_copy_handleattr_set_ftp_attr
                          (&gass_copy_handleattr, &ftp_handleattr);
  globus_gass_copy_handle_init      (&gass_copy_handle,
                                     &gass_copy_handleattr);

  if (source_url.scheme_type == GLOBUS_URL_SCHEME_GSIFTP ||
      source_url.scheme_type == GLOBUS_URL_SCHEME_FTP    )
  {
    globus_ftp_client_operationattr_init  (&source_ftp_attr);
    globus_gass_copy_attr_set_ftp        (&source_gass_copy_attr,
                                          &source_ftp_attr);
  }
  else {
    globus_gass_transfer_requestattr_init (&source_gass_attr,
                                           source_url.scheme);
    globus_gass_copy_attr_set_gass       (&source_gass_copy_attr,
                                &source_gass_attr);
  }
 output_file = globus_libc_open ((char*) target, O_WRONLY | O_TRUNC
| O_CREAT, S_IRUSR  | S_IWUSR | S_IRGRP | S_IWGRP);

  if ( output_file == -1 )
  {
    printf ("could not open the destination file \"%s\"\n", target);
    return (-1);
  }
```

Copy Data -- Globus/GASS

```
 if ( globus_io_file_posix_convert (output_file, GLOBUS_NULL,
&dest_io_handle)
       != GLOBUS_SUCCESS)
  {
    printf ("Error converting the file handle\n");
    return (-1);
  }

  result = globus_gass_copy_register_url_to_handle (
              &gass_copy_handle,
              (char*)source_URL,
              &source_gass_copy_attr,
              &dest_io_handle,
              my_callback,
              NULL);

  if ( result != GLOBUS_SUCCESS )
  {
    printf ("error: %s\n", globus_object_printable_to_string
(globus_error_get (result)));
    return (-1);
  }
  globus_url_destroy  (&source_url);

  return (0);
}
```

Copy Data -- GT3/OGSA

```
Package org.globus.ogsa.gui;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.net.URL;
import java.util.Date;
import java.util.Vector;
import javax.xml.rpc.Stub;
import org.apache.axis.message.MessageElement;
import org.apache.axis.utils.XMLUtils;
import org.globus.axis.gsi.GSIConstants;
import org.globus.ogsa.ServiceProperties;
Import org.globus.ogsa.base.multirft.MultiFileRFTDefinitionServiceGridLocator;
import org.globus.ogsa.base.multirft.RFTOptionsType;
import org.globus.ogsa.base.multirft.RFTPortType;
import org.globus.ogsa.base.multirft.TransferRequestElement;
import org.globus.ogsa.base.multirft.TransferRequestType;
import org.globus.ogsa.base.multirft.TransferType;
import org.globus.ogsa.impl.security.authentication.Constants;
import org.globus.ogsa.impl.security.authorization.NoAuthorization;
import org.globus.ogsa.utils.AnyHelper;
import org.globus.ogsa.utils.GetOpts;
import org.globus.ogsa.utils.GridServiceFactory;
import org.globus.ogsa.utils.MessageUtils;
import org.gridforum.ogsi.ExtendedDateTimeType;
import org.gridforum.ogsi.ExtensibilityNotSupportedFaultType;
import org.gridforum.ogsi.ExtensibilityType;
import org.gridforum.ogsi.ExtensibilityTypeFaultType;
import org.gridforum.ogsi.Factory;
import org.gridforum.ogsi.HandleType;
import org.gridforum.ogsi.InfinityType;
import org.gridforum.ogsi.LocatorType;
import org.gridforum.ogsi.OGSIServiceGridLocator;
import org.gridforum.ogsi.ServiceAlreadyExistsFaultType;
import org.gridforum.ogsi.TerminationTimeType;
import org.gridforum.ogsi.WSDLReferenceType;
import org.gridforum.ogsi.holders.ExtensibilityTypeHolder;
import org.gridforum.ogsi.holders.LocatorTypeHolder;
import org.gridforum.ogsi.holders.TerminationTimeTypeHolder;
import org.globus.gsi.proxy.IgnoreProxyPolicyHandler;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class RFTClient {
 public static void copy (String source_url, String target_url)
 {
  try {
   File requestFile      = new File (source_url);
   BufferedReader reader = null;

   try  {
    reader = new BufferedReader (new FileReader (requestFile));
   }  catch (java.io.FileNotFoundException fnfe) { }

   Vector requestData = new Vector ();
   requestData.add (target_url);

   TransferType[] transfers1          = new TransferType[transferCount];
   RFTOptionsType multirftOptions = new RFTOptionsType ();
```

Copy Data -- GT3/OGSA

```
multirftOptions.setBinary          (Boolean.valueOf  (
     (String)requestData.elementAt (0)).booleanValue ());
multirftOptions.setBlockSize       (Integer.valueOf  (
     (String)requestData.elementAt (1)).intValue     ());
multirftOptions.setTcpBufferSize   (Integer.valueOf  (

     (String)requestData.elementAt (2)).intValue     ());
multirftOptions.setNotpt           (Boolean.valueOf  (
     (String)requestData.elementAt (3)).booleanValue ());
multirftOptions.setParallelStreams (Integer.valueOf  (
     (String)requestData.elementAt (4)).intValue     ());
multirftOptions.setDcau(Boolean.valueOf(
     (String)requestData.elementAt (5)).booleanValue ());
int i = 7;

for (int j = 0; j < transfers1.length; j++)
{
  transfers1[j] = new TransferType ();

  transfers1[j].setTransferId       (j);
  transfers1[j].setSourceUrl        ((String)requestData.elementAt (i++));
  transfers1[j].setDestinationUrl ((String)requestData.elementAt (i++));
  transfers1[j].setRftOptions       (multirftOptions);
}

TransferRequestType transferRequest = new TransferRequestType ();
transferRequest.setTransferArray (transfers1);
int concurrency = Integer.valueOf
            ((String)requestData.elementAt(6)).intValue();
if (concurrency > transfers1.length)  {
```

```
    System.out.println ("Concurrency should be less than the number"
                        "of transfers in the request");
    System.exit (0);
  }
  transferRequest.setConcurrency (concurrency);
  TransferRequestElement requestElement =
                  new TransferRequestElement ();
  requestElement.setTransferRequest (transferRequest);
  ExtensibilityType extension =
                  new ExtensibilityType ();
  extension = AnyHelper.getExtensibility (requestElement);
  OGSIServiceGridLocator factoryService =
                  new OGSIServiceGridLocator ();
  Factory factory = factoryService.getFactoryPort (
                  new URL (source_url));
  GridServiceFactory gridFactory =
                  new GridServiceFactory (factory);
  LocatorType locator = gridFactory.createService (extension);
  System.out.println ("Created an instance of Multi-RFT");
  MultiFileRFTDefinitionServiceGridLocator loc =
                  new MultiFileRFTDefinitionServiceGridLocator ();
  RFTPortType rftPort = loc.getMultiFileRFTDefinitionPort  (locator);
  ((Stub)rftPort)._setProperty (Constants.AUTHORIZATION,
                  NoAuthorization.getInstance());
  ((Stub)rftPort)._setProperty (GSIConstants.GSI_MODE,
                  GSIConstants.GSI_MODE_FULL_DELEG);
  ((Stub)rftPort)._setProperty (Constants.GSI_SEC_CONV,
                  Constants.SIGNATURE);
  ((Stub)rftPort)._setProperty (Constants.GRIM_POLICY_HANDLER,
                  new IgnoreProxyPolicyHandler ());
} catch (Exception e)  { System.err.println (MessageUtils.toString (e)); } } }
```

Copy Data -- GAPI (SAGA)

```
#include <GAPI.h>

int CopyFile (const char* source_url,
              const char* target_url)
{

  try
  {
    GAPI_File *file = new GAPI_File (source_url);
    file->copy (target_url);
  }
  catch (GATException e)
  {
    printf  (e.ErrorString ());
    return  (e.ErrorCode   ());
  }
  return (0);
}
```

Approaches

- Application developers gravitate towards APIs
  - They don't give a damn about protocols!
  - *(Chromium example)*
- Get a bunch of apps people together to hammer out "abstract APIs"
  - GridLab GAT
  - RealityGrid
  - DiVA
  - SAGA-RG
- Some APIs cannot be simplified (*but many can*)
  - Experts in these areas (eg. Security) don't seem to understand just how little we need!

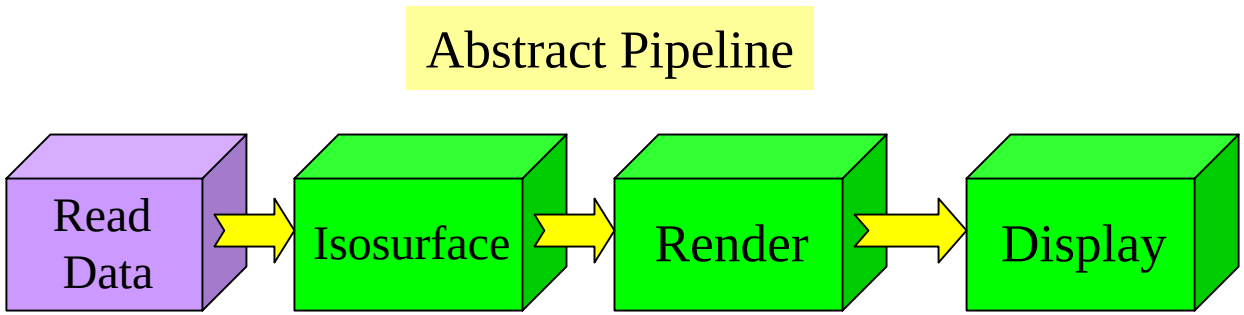
But there's more to it than that

- Not all of the problems we face are related to APIs
- There are some "systems" level issues
  - Resource discovery
  - Component discovery
  - Brokers that understand workflow dependencies
  - Vis-oriented transport protocols
    - *GridFTP is terrible for vis*
    - *New network services like lambda switching & application controlled PVCs*

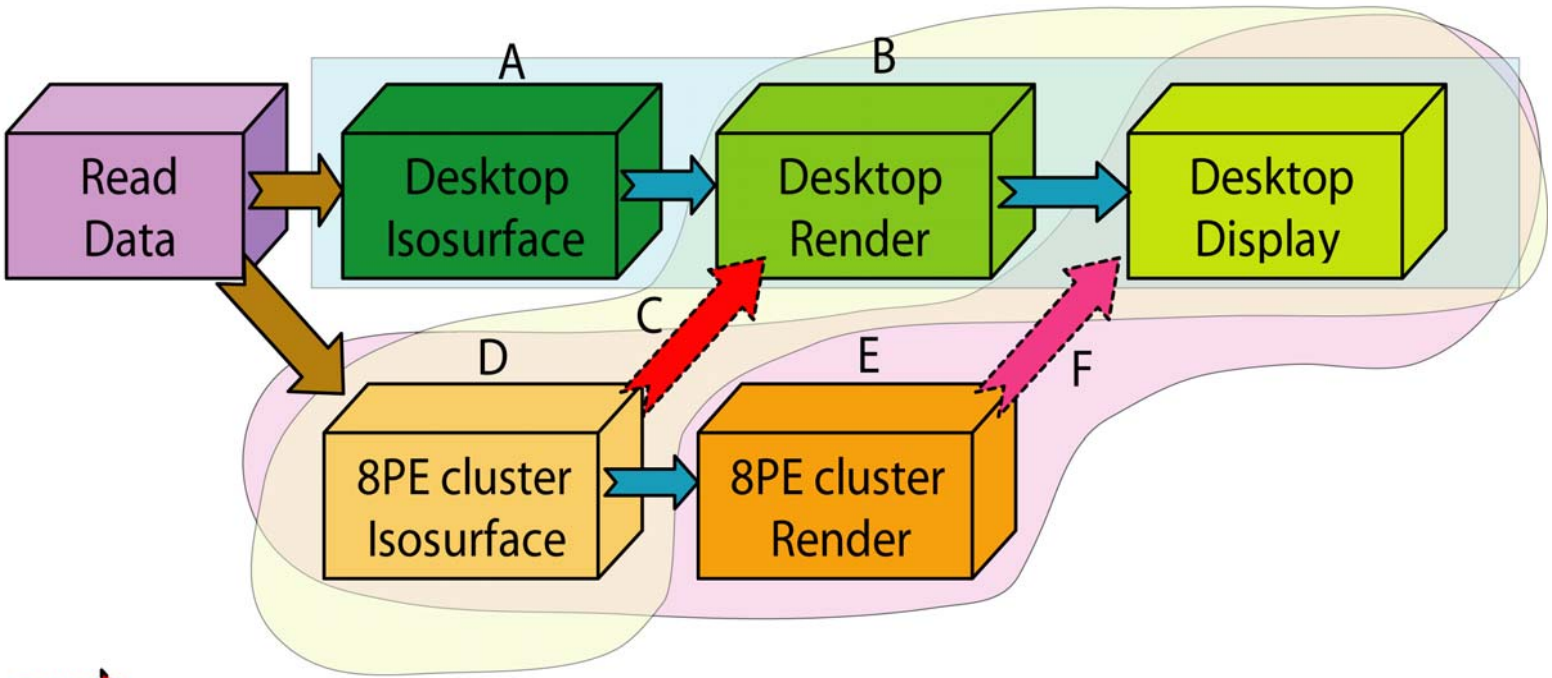
Example: Resource Discovery

- Current Approach
  - Use MDS or else!!!
  - MDS + info providers make data easy to read, but hard for users edit! (not symmetric)
  - Authentication, authorization, access
- What we want (for component discovery)
  - Local
  - Machine
  - Organizational

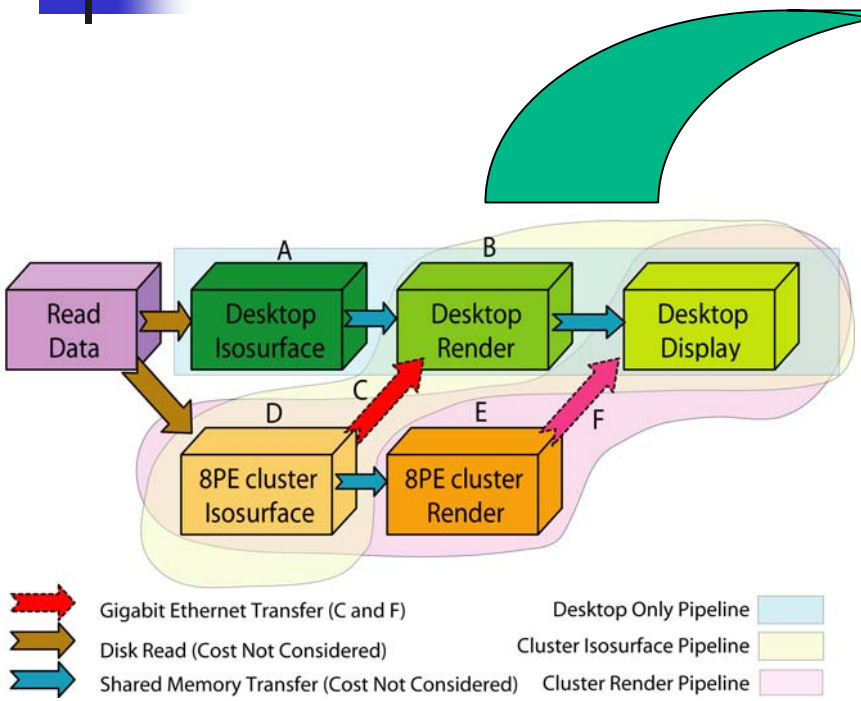
A Simplified Example of Vis Pipeline Responsiveness

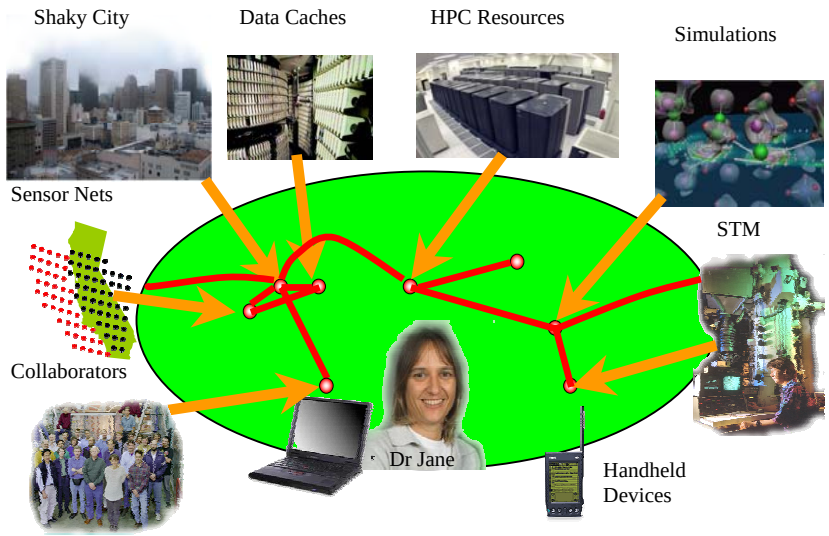


Mapping Problem



Mapping Problem





Workflow Performance Parameters

- Dynamic Response Constraints and Parameters
  - Responds dynamically to runtime/user-defined constraints
    - Display Framerate
    - Datasets/sec Throughput (eg. Shuttling through datasets)
    - Recompute on param change (eg. Change isosurface level)
  - Respond to runtime resource constraints
    - Contract violation
    - hardware/network failure (fault tolerance)
  - Respond to runtime dynamic data requirements
    - Different data payloads or algorithm performance based on algorithm parameter choices
    - Different data payloads or algorithm performance due to changing data characteristics

Distributed Workflow Mapping

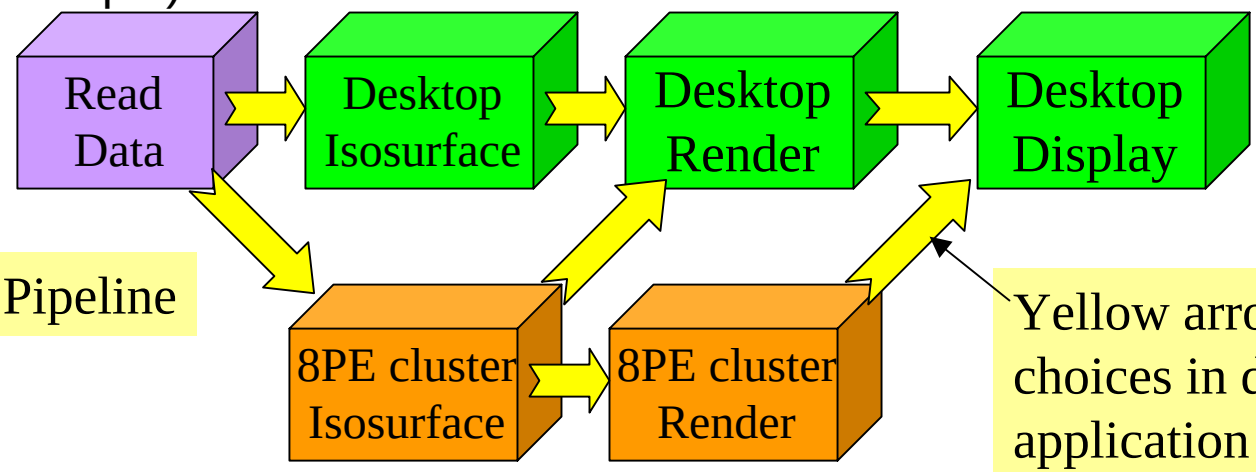
- **Level 1**: Baseline (*map of the pipeline onto the virtual machine is explicit*)
  - Uniform Security, I/O, data model compatability (basic Grid services)
  - Ability to explicitly launch apps on a static map of machines.
- **Level 2**: Static Maps (*optimal **initial** mapping of application to virtual machine*)
  - Get a static mapping of resources that provides best overall performance
  - Requires predictive performance models (heuristic, parameterized/algorithmic, statistical/history-based)
- **Level 3**: Dynamic Maps (*runtime optimization*)
  - Requires continuous instrumentation feedback to the parameterized models of performance.
  - Must support multiple parallel pipelines dynamically refactored depending on response profile (which map can respond most rapidly)
  - Requires commensurability between different methods that produce the same image

Distributed Workflow Mapping

- **Level 1**: Baseline (*map of the pipeline onto the virtual machine is explicit*)
  - Uniform Security, I/O, data model compatability (basic Grid services)
  - Ability to explicitly launch apps on a static map of machines
- **Level 2**: Static Maps (*optimal **initial** mapping of application to virtual machine*)
  - Get a static mapping of resources that provides best overall performance
  - Requires predictive performance models (heuristic, parameterized/algorithmic, statistical/history-based)
- **Level 3**: Dynamic Maps (*runtime optimization*)
  - Requires continuous instrumentation feedback to the parameterized models of performance.
  - Must support multiple parallel pipelines dynamically refactored depending on response profile (which map can respond most rapidly)
  - Requires commensurability between different methods that produce the same image

A Simplified Example of Vis Pipeline Responsiveness

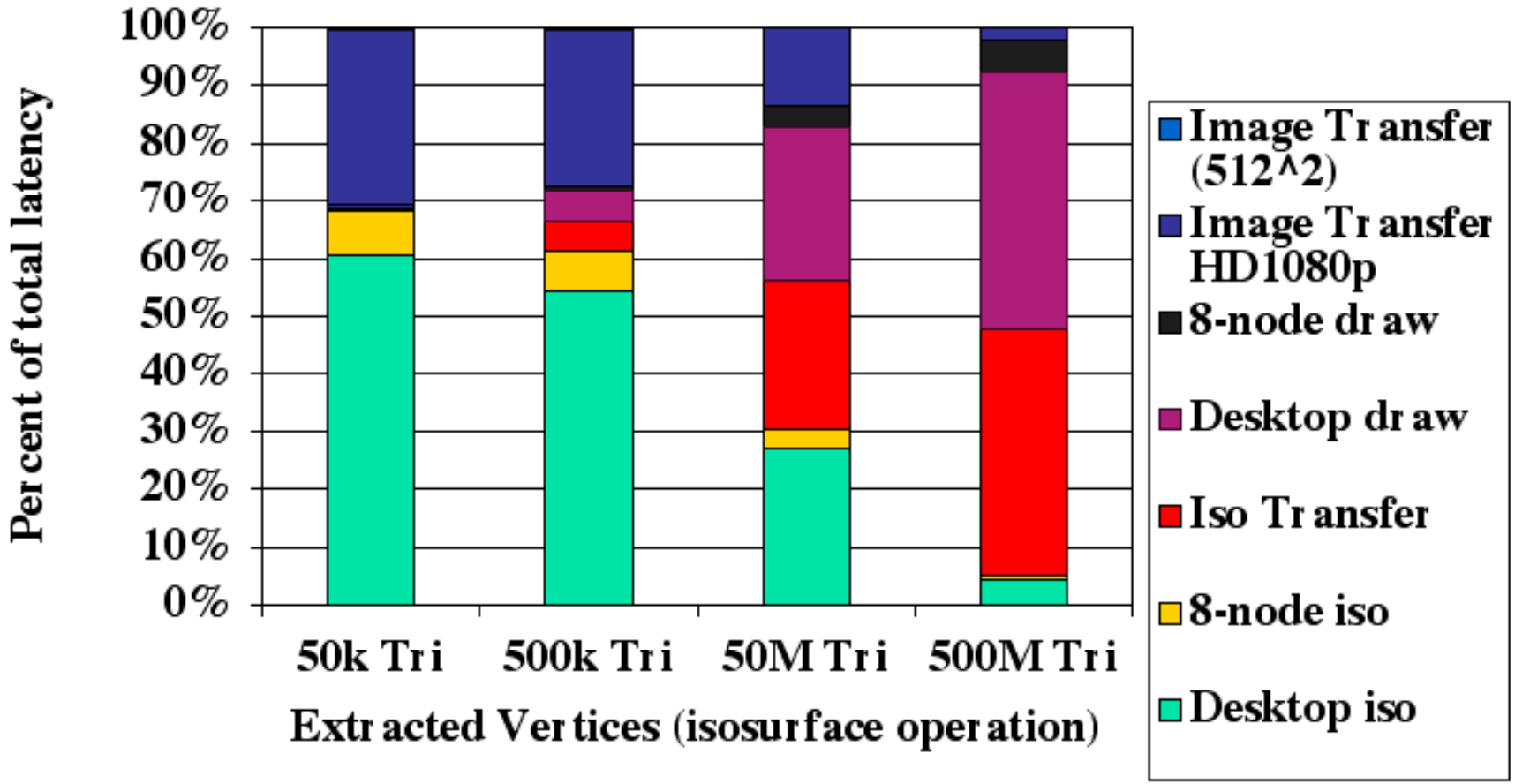
- A simple (*cooked*) performance model
  - 50M triangles/sec (24-byte tri-strips) Graphics HW (1/8 for 8 PEs)
  - 1 Second to compute isosurface with one processor (1/8 for 8PEs)
  - 1 Gigabit Network with perfect performance
  - Perfect Speedup for parallel algorithms
  - The real world will offer a more complex performance model (just an example)



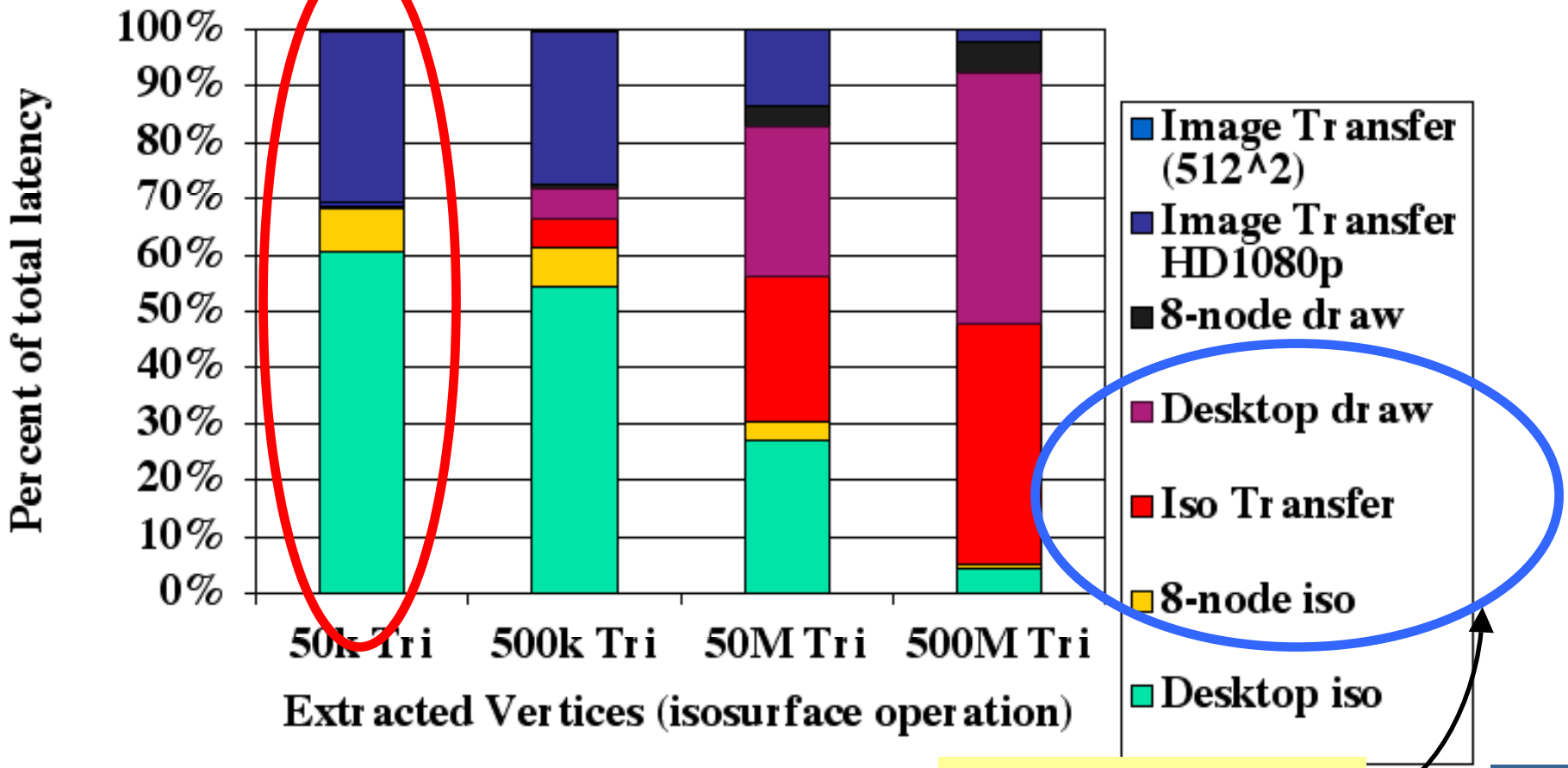
Concrete Pipeline

Yellow arrows indicate choices in distributed application data flow

Vis Pipeline Responsiveness

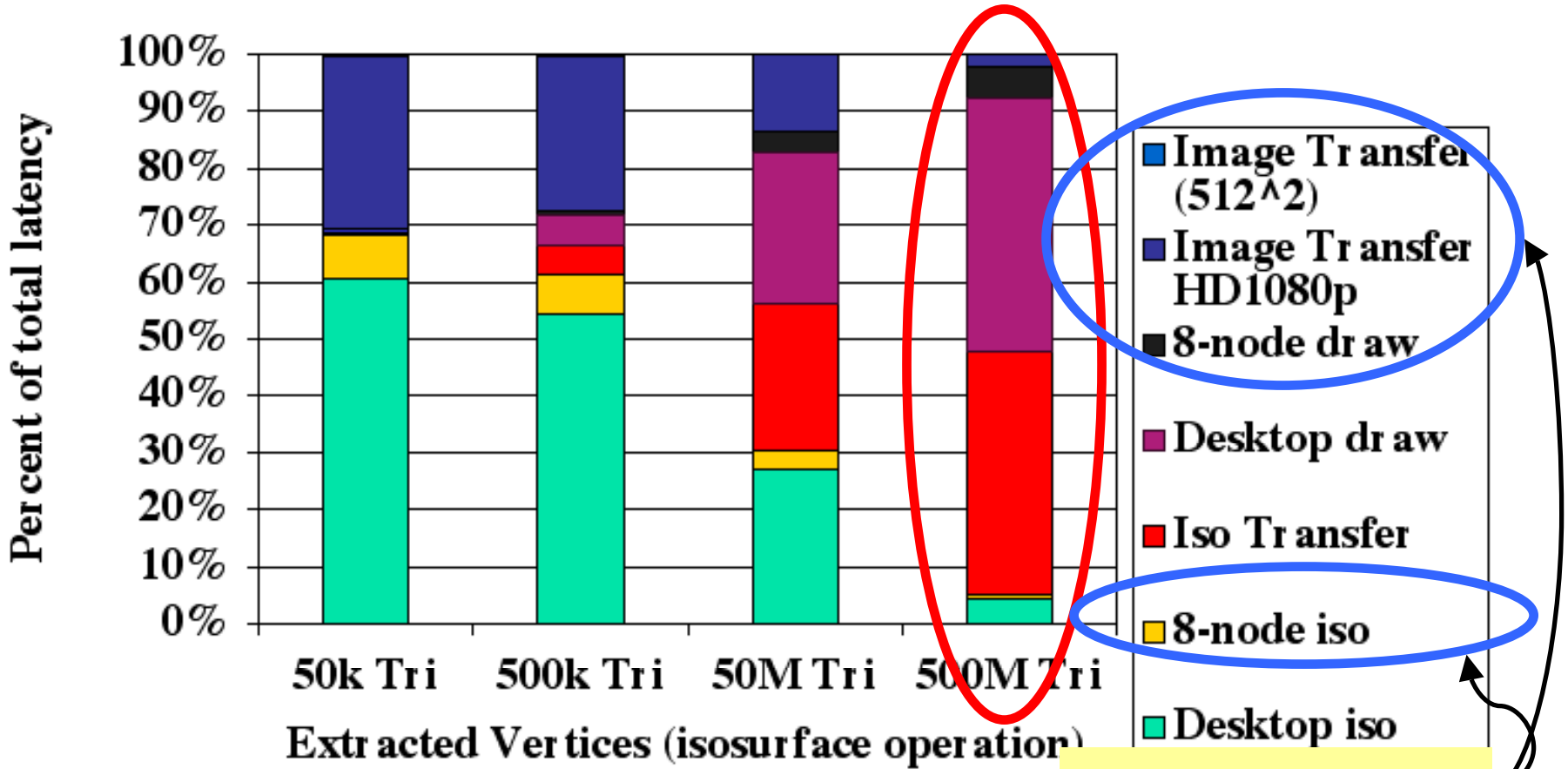


Vis Pipeline Responsiveness



Best Throughput

Vis Pipeline Responsiveness



Conclusion on Pipeline Example

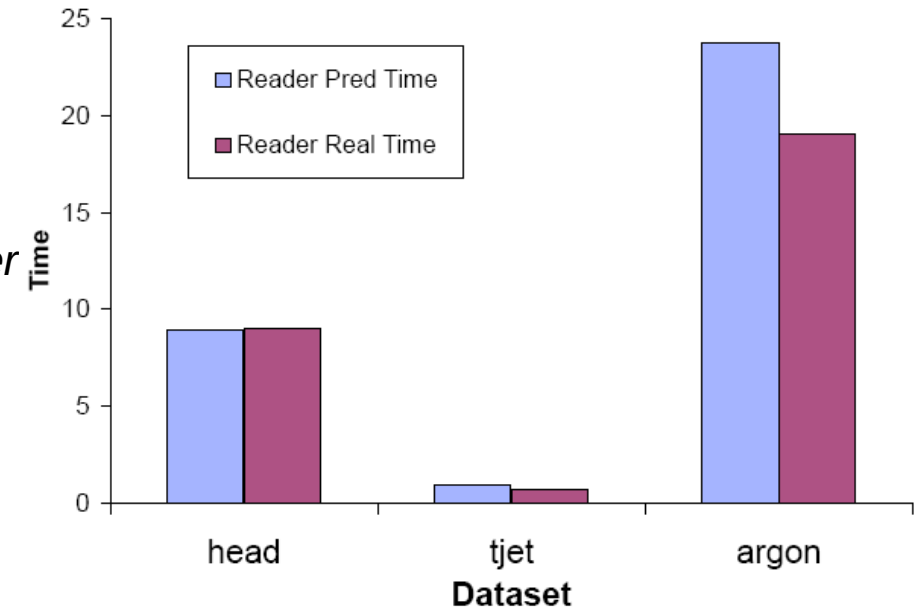
- Just simple change in isolevel completely changes optimal pipeline selection!
- No single remote vis methodology is best in all circumstances (even at runtime)!
- Must have commensurable visual output from many different methods
- Simply scheduling resources for these overlapping pipelines will be hard, muchless auto-selecting between them!
- Must have a common framework to deliver a dynamic multi-pipeline visualization capability.
  - so we can focus our effort on the “hard stuff”!

Performance Modeling and Pipeline Optimization

- Goal: automate the process of placing components on distribute resources.
- Approach: model performance of individual components, optimize placement as a function of performance target.
  - Optimize for interactive transformation.
  - Optimize for changing isocontour level.
  - Optimize for data throughput.
- Find correct performance model
  - Analytic
  - Historical
  - Statistical/Heuristic
- Ensure performance model is *composable*
- Results: Quadratic order algorithm, high degree of accuracy

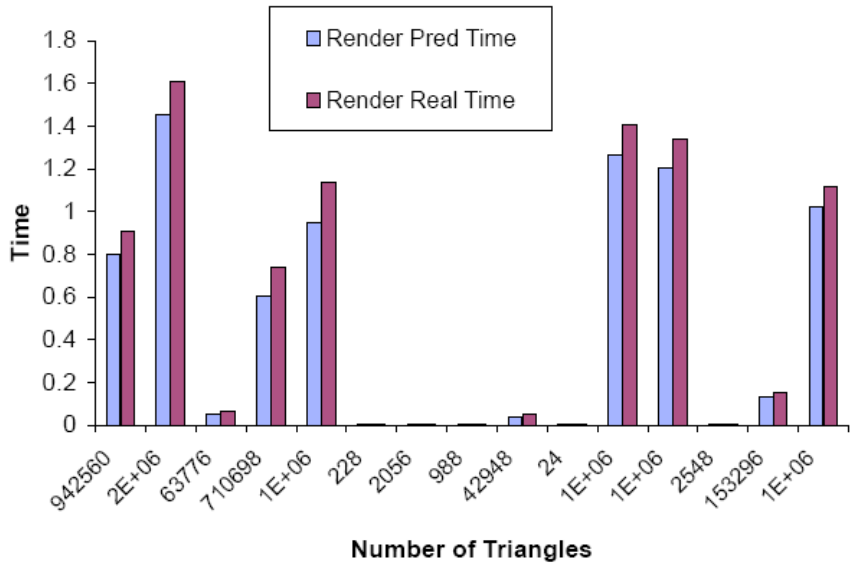
Performance Modeling and Pipeline Optimization

- Single workflow:
  - Reader -> Isosurface -> Render -> Display
- Reader performance:
  - Function of:
    - Data Size
    - Machine constant
  - $T_{reader}(n_v) = n_v * C_{reader}$



Performance Modeling and Pipeline Optimization

- Render Performance:
  - Function of:
    - Number of triangles,
    - Machine constant.



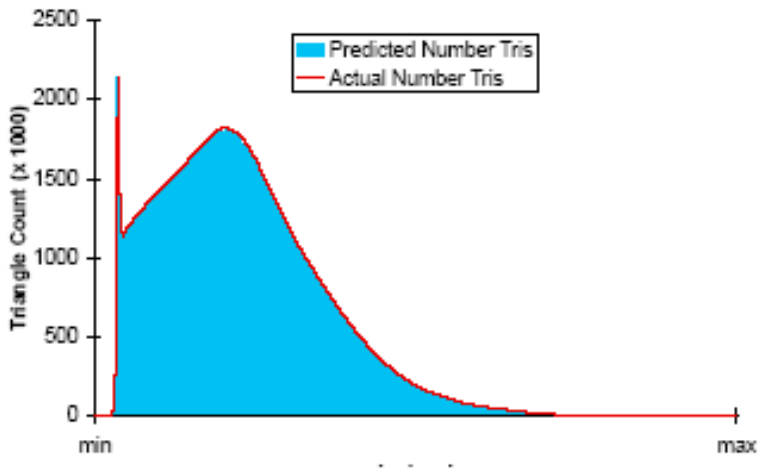
  - $T_{render} = n_t * C_{render} + T_{readback}$

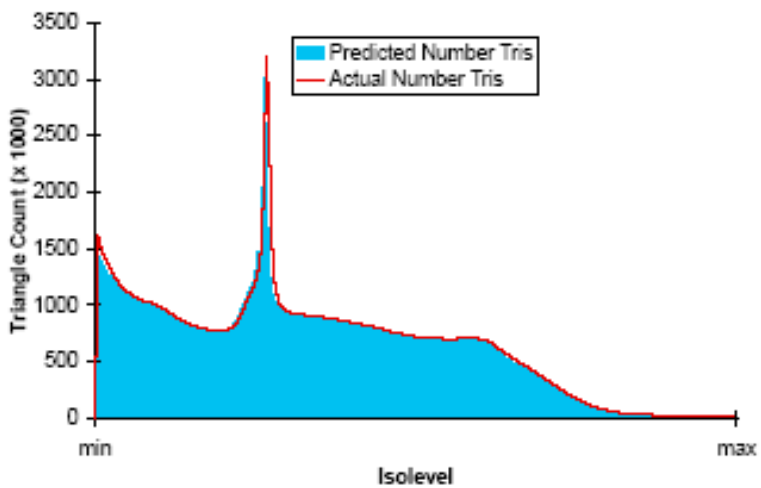
Performance Modeling and Pipeline Optimization

- Isosurface Performance:
  - Function of:
    - Data set size,
    - Number of triangles generated (determined by combination of dataset and isocontour level).
  - Dominated number of triangles generated!
  - $T_{iso}(n_t,n_v) = n_v * C_{base} + n_t * C_{iso}$

Performance Modeling and Pipeline Optimization

- Precompute histogram of data values.
- Use histogram to estimate number of triangles as a function of iso level.

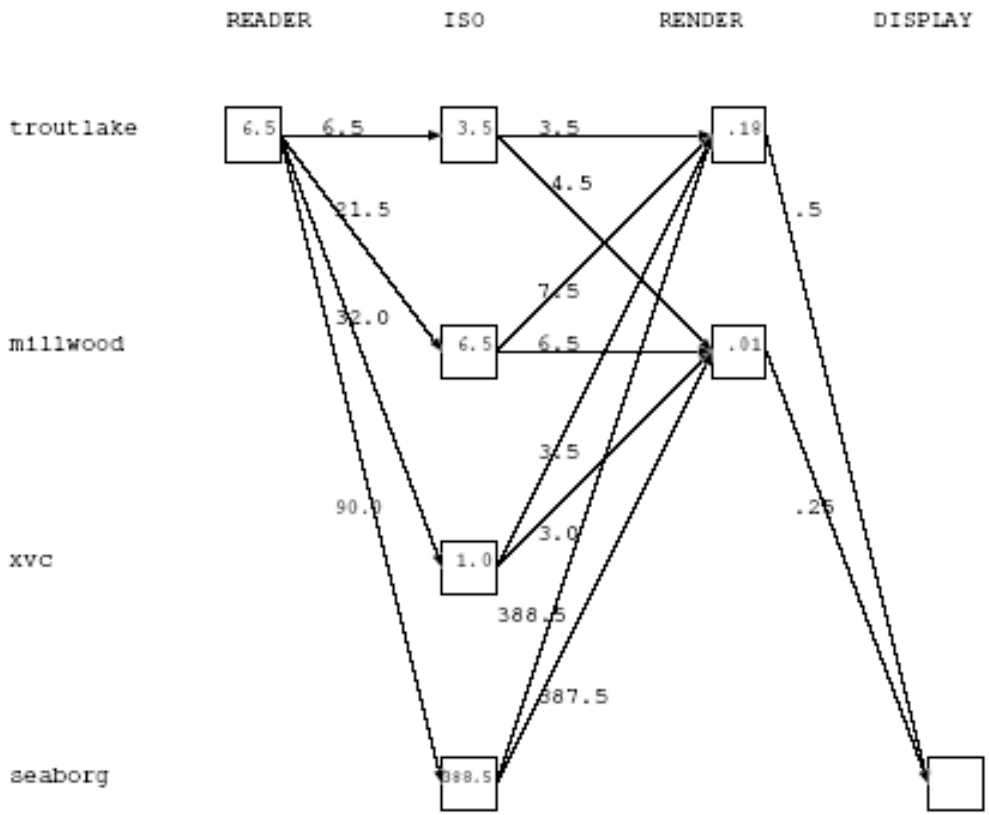




Performance Modeling and Pipeline Optimization

- Optimize placement using Djikstra's shortest path algorithm.
- Edge weights assigned based upon performance target.
- Low-cost algorithm:

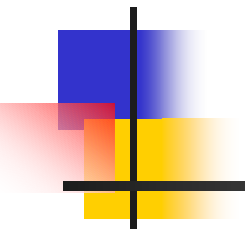
  O(Edges + NlogN)

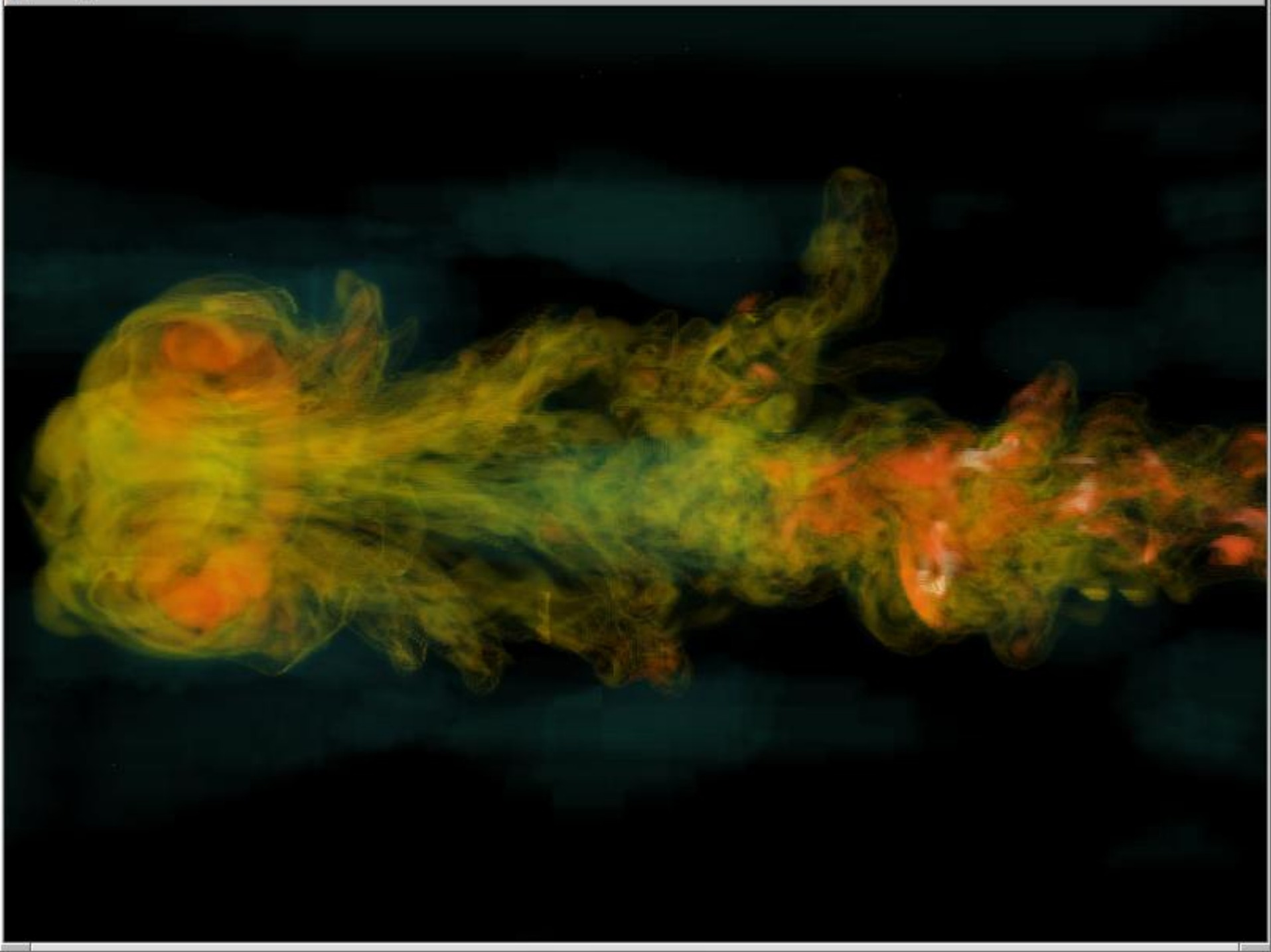


Conclusions

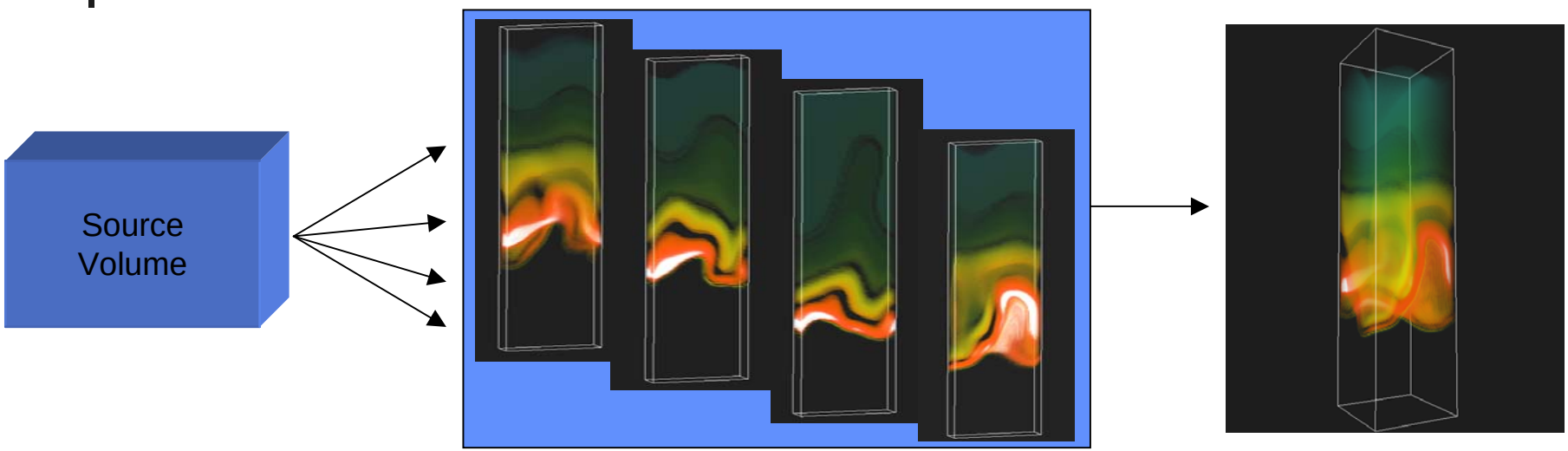
- “Microbenchmarks” to estimate individual component performance.
  - Per-dataset statistics can be precomputed and saved with the dataset.
- Quadratic-order workflow-to-resource placement algorithm.
- Optimizes pipeline performance for an specific interaction target – relieves users from task of manual resource selection.

Networks





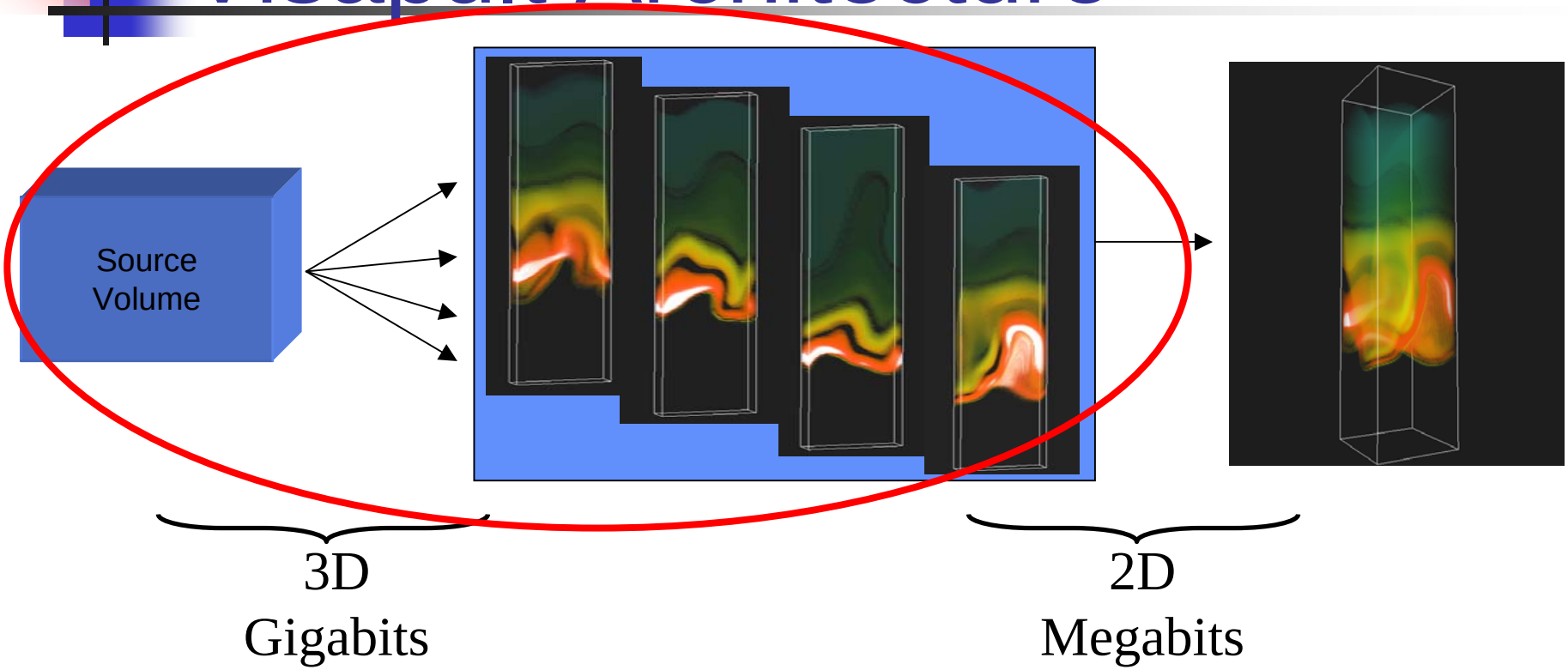
Visapult Architecture



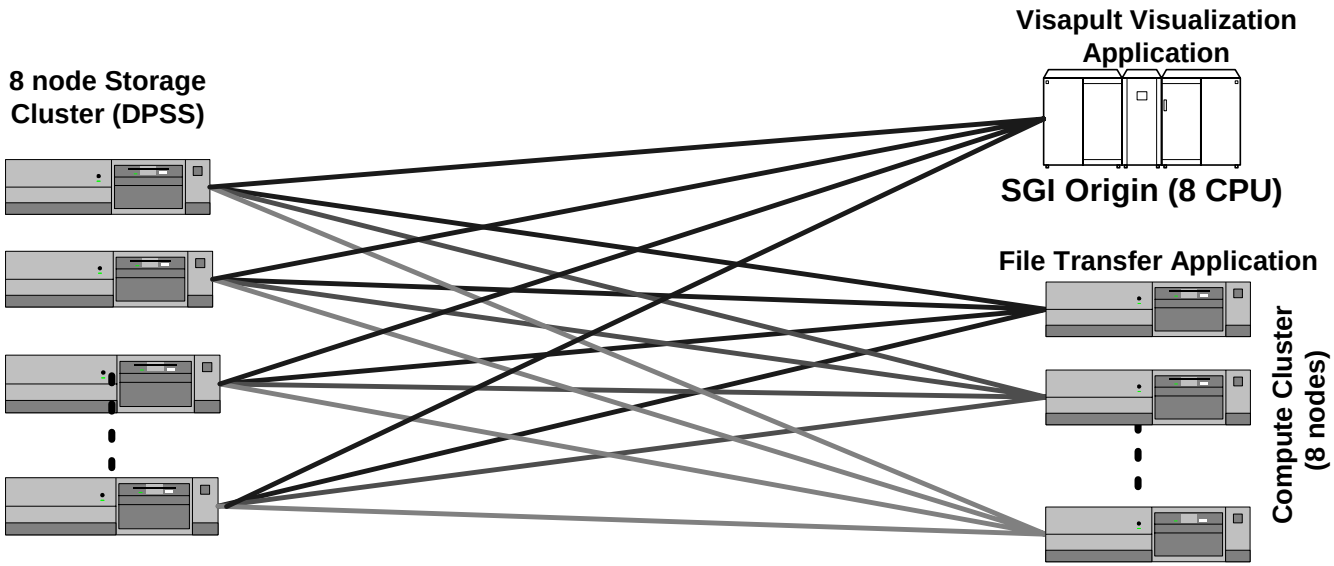
3D
Gigabits

2D
Megabits

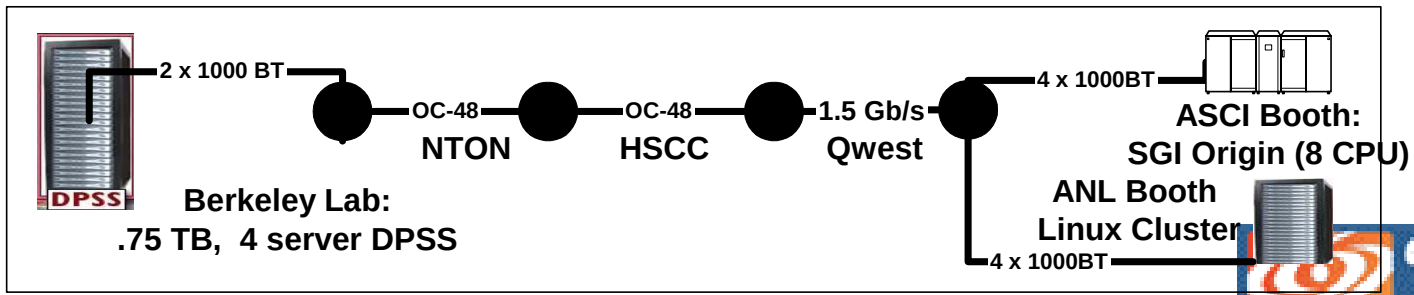
Visapult Architecture



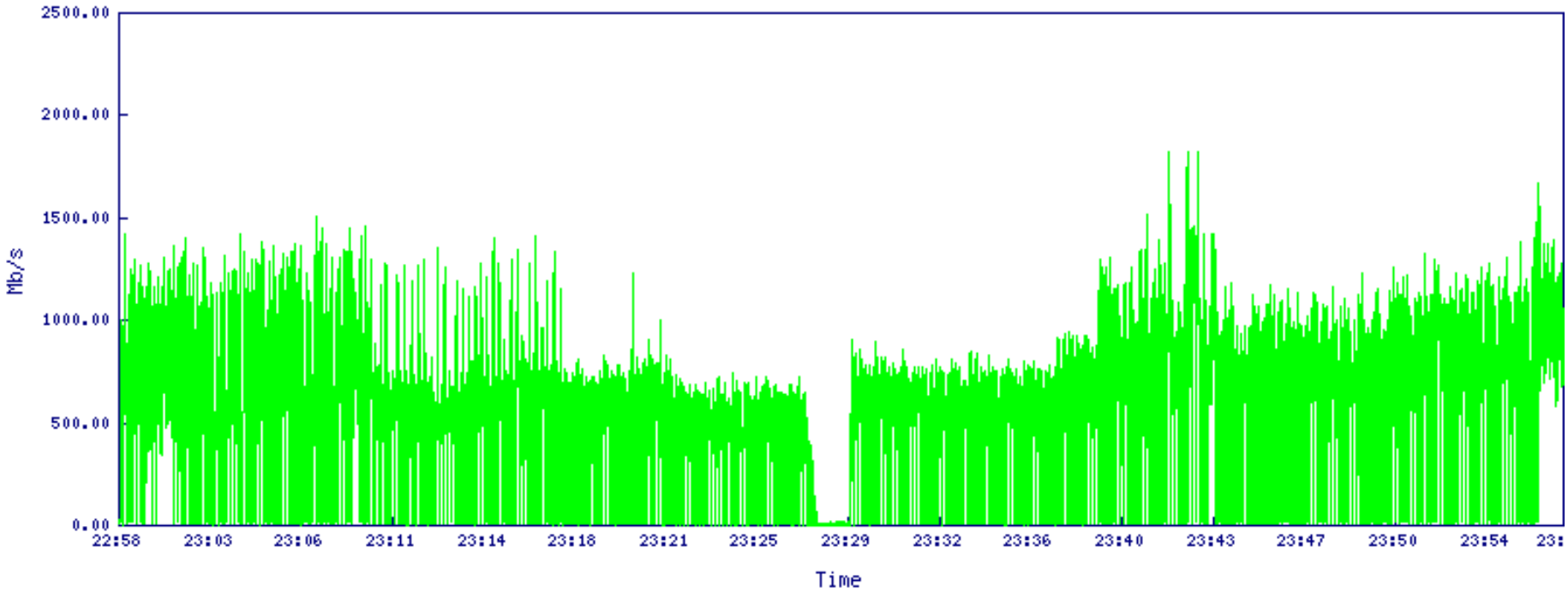
SC2000 Demo Configuration



Network Throughput: 5 sec peak 1.48 Gbits/sec (72 streams: 20.5 Mbits/stream); 60 minute sustained average: 582 Mbits/sec



SC2000 Network Throughput



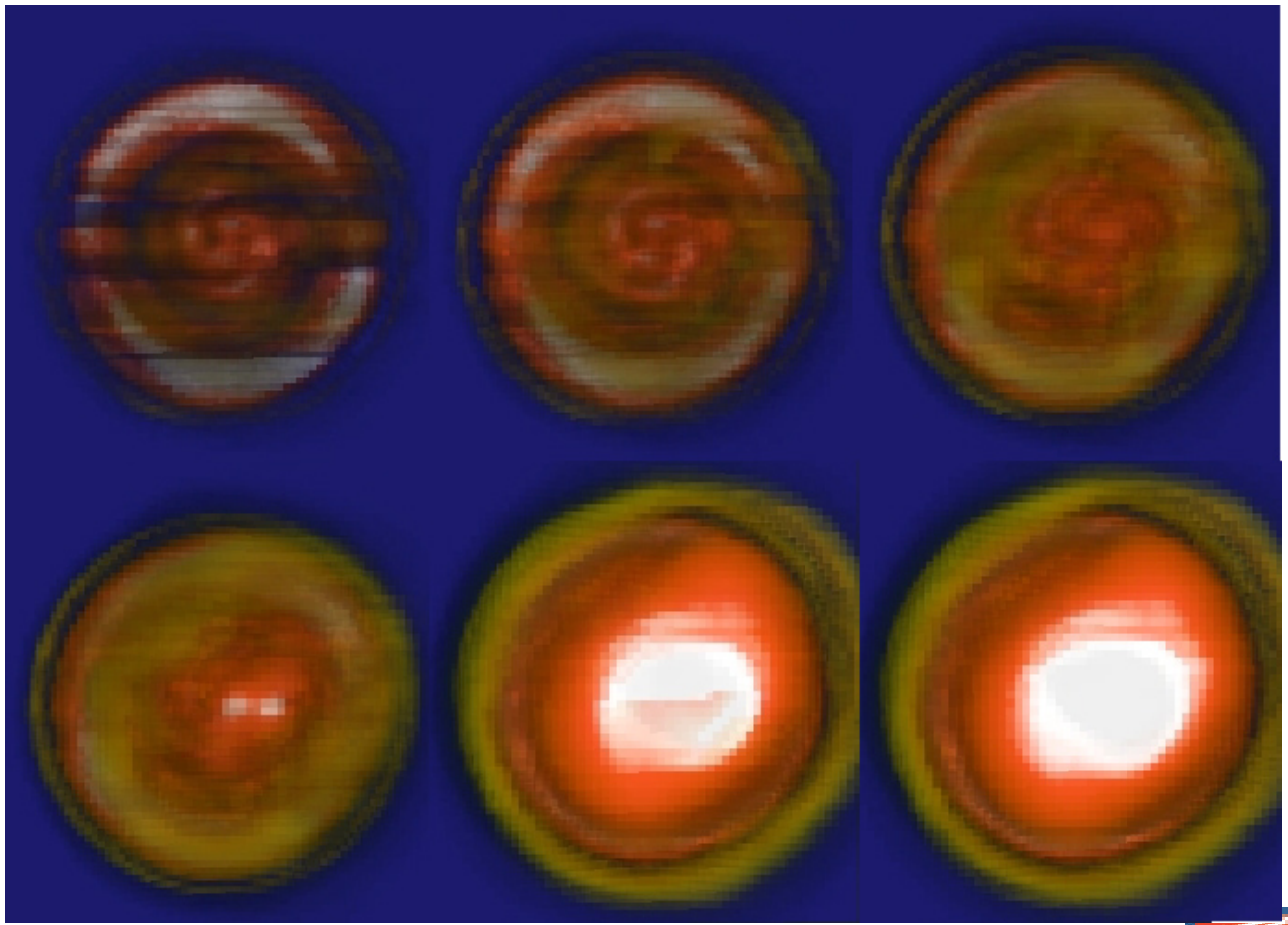
Refactoring the Design

- Congestion avoidance
  - Good for internet
  - Bad bad baaaad for PVCs and other dedicated networks. (switched lambdas?)
- Multistream TCP
  - Erratic performance
  - Requires a lot of tuning
  - Unfriendly to other users
  - Unfriendly to visualization applications
- We want full control of the “throttle”
  - Very much like network video

Refactoring the Design

- TCP is the wrong thing for interactive vis!
  - Layer 3 latency/jitter (all buffering effects)
  - Poor response to bursty traffic
  - Vis needs interactivity and minimal latency!
- Network Video / UDP streams
  - Present packets to app. immediately (low latency)
  - Full control of data rate
  - Lossy, but effects of loss can be managed
- SOCK_RDM

Effect of Loss on Visapult



Evolving Binary Black Hole Merger Simulation (100 timesteps)

Steady @ 16+ Gigabits!

Whats Next?

- Manual throttle (UDP-based protocols) are here to stay.
  - Hopefully SOCK_RDM will cover most needs
  - Whaaa? Those idiots are going to burn down the network! Next big thing: resource management
- RSVP & DiffServ were developed to manage this very situation with regard to network video
- RSVP & DiffServ are never going to happen
  - Gregory Bell, “Failure to Thrive: QoS and the Culture of Operational Networking,” *Proceedings of the ACM SIGCOMM 2003 Workshops, RIPQoS Workshop*.
- Next Big Thing? : Pluggable/Adaptive Congestion Management
  - AIMD for internet (can even mimic multistream TCP behavior)
  - Fixed rate for PVCs and switched lambdas

What is Needed?

- Vis Forum
  - Agree on interfaces
  - Hide the innards
  - Multiple implementations of same interface
  - Reference implementations / OpenSource
- DiVA
- GGF-ACE *(vis security requirements document)*
- Vis participation in SAGA-RG