Illuminating the Shadow Mesh

Jon Goldman (jgoldma@sandia.gov) *and*
Warren Hunt (wlhunt@sandia.gov),
Sandia National Laboratories

ABSTRACT: *Sandia's Red Storm is currently the largest XT3 installation in the world. The complexity of Red Storm, with its arrangement of 13,280 compute, service, and I/O nodes requires new analysis paradigms be developed to help better understand the state of the machine, especially when problems arise. Fully understanding issues that affect routing would require being able to picture on the order magnitude of 176,358,400[*] potential routes in the system. To aid in this endeavor we have created a suite of three-dimensional visualization and database tools.*

KEYWORDS: Sandia, Cray, XT3, Red Storm, visualization, illumination, graphics, routing, shadow, mesh

1. Introduction

1.1 Sandia's Red Storm

Sandia National Laboratories architected the Red Storm design[1]. In 2002 Cray Inc. was named vendor partner for carrying out the design details, engineering, and building Red Storm. In 2004[2] Cray Inc took the XT3 to market, and Sandia's installation commenced in 2005.

Red Storm's sheer size in the number of nodes and requisite computing cabinets, wires, etc. makes it a daunting task to fully comprehend and understand. The authors, as part of the visualization team in the Scientific Applications and User Support department at Sandia, are working closely with the System Manager[3] of Red Storm to create a visually-based analytical tool. The project was begun in 2005. Initially the software was engineered to analyze Red Storm's mesh from a logical and physical[4] perspective. Later it was adapted to support routing and to help visualize the Shadow Mesh, which is the focus of this paper.

The following image gives an idea of the size of Red Storm. It shows the cabinet layout and distribution of dedicated red/black cabinets (the compute sections are switchable between classified and unclassified operating modes). Note that each compute cabinet houses 96 compute nodes:

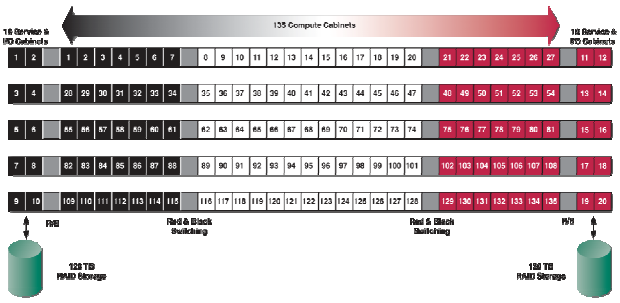

Figure 1: Cabinet Layout

1.2 Visual Analysis Overview

1.2.1 LuzRed

Our software analysis tool, LuzRed, is actually a suite of programs written in C++, python, and Perl. The visualization part of LuzRed is written in C++ and uses the Visualization ToolKit (VTK)[5] for scientific visualization, Qt[6] for the graphical user interface, and CMake[7] for cross platform builds on Windows and Linux.

[*] $13{,}280^2 = 176{,}358{,}400$

Some of the database processing and communications parts of LuzRed were written in Python and Perl. We use MySQL client/server API to build and access various databases. We mirror portions of the XTAdmin database that runs on Red Storm, and for routing analysis we create a new MySQL database.

Ultimately we would like as many data sources to LuzRed be automatically or semi-automatically updated. For example determining when the XTAdmin mirror needs refreshing is an area of ongoing work.

1.3 End User

Although LuzRed is still in development, it is designed to serve the analysis needs of several categories of users. As a research tool, LuzRed allows better understanding of job allocation, distribution, routing, and other resource issues. As a management tool, LuzRed provides real-time status and assessments of the health of the system.

A system manager should be able to leverage his or her knowledge of the machine and associated software systems and databases to perform sophisticated analysis and make correspondences between the various visual representations that our application presents.

A system engineer can use LuzRed to locate failures on the machine. To be truly successful LuzRed would allow an administrator to distinguish between failure cases where Red Storm requires a reboot vs. deferring taking action[8]. There is no dynamic rerouting in the XT3, due to static route tables and hardware limitations, therefore a reboot will eventually be required. Rebooting is the only way to rebuild the route tables and resolve the Seastar failure. This issue relates to the concept of the shadow mesh, which is discussed in detail below in section 2. The Shadow Mesh.

The following image shows the latest incarnation of the visualization tool:

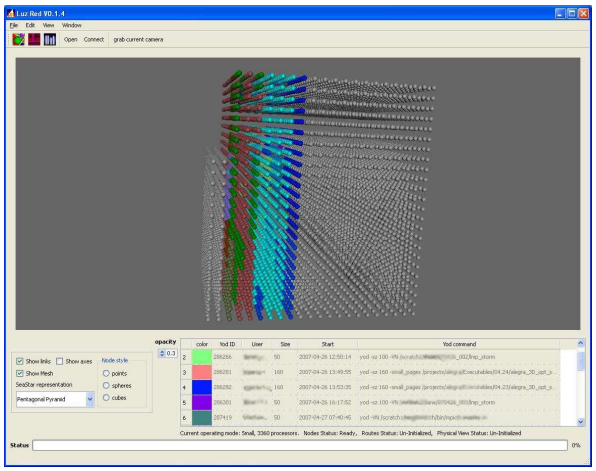

Figure 2: LuzRed Main Panel

We envision that the system manager might be interested in running the visualization program to examine and understand the job allocation mechanism; an administrator might sit with the manager and perform database queries to answer questions such as, “Did job X die as a result of a network routing problem?” A hardware technician might want to look at snapshots of routes, either real-time or for historical queries.

1.4 2D or not 2D that is the 3D question

Early in the development of LuzRed we cogitated over whether to provide 2D as well as 3D representations of the analysis. Colleagues suggested that a 2D view could provide as much information as 3D with less ‘clutter’.

While the authors agree that 2D renderings can prove useful, and less disorienting, the inherent 3D coordinate system and layout of the XT3 mesh, and the complexity of Sandia’s machine, led us to focus on 3D visualization techniques. To date the only 2D representation we support is the job layout similar to what the `xtshowmesh` command displays:

Figure 3: Job Display

However, we have prototyped/explored 2D ideas of displaying job information in ways such as this:

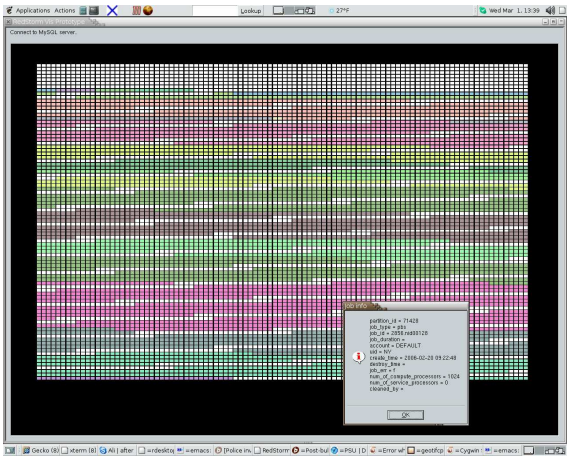

Figure 4: Prototype 2D

2. The Shadow Mesh

The term shadow mesh is inspired by the idea of areas of the system that have become darkened, or not “visible” from one or more nodes. In the same sense that a camera can not see objects that are

obscured by other opaque objects, nodes in the system that are unreachable, either because a Seastar or Seastar link failed or was disabled, are considered 'in shadow.'

A precise definition of the shadow mesh is the aggregate set of nodes that are unreachable on routes from/to other nodes via a particular Seastar node or link.

We can think of the shadow mesh as the sum of all nodes whose communication through the system has been reduced. However, the role of LuzRed is to utilize visualization to enable increased machine performance and better decision making by system administrators. By quantifying the uncertainty introduced by a Seastar failure, LuzRed allows the system administrator to respond to the failure in a more efficient manner. Perhaps the failure does have a catastrophic impact on the system, or maybe an important job can be allowed to complete before restoring the system. By computing shadows from different viewpoints, we can better understand a complex environment and make better decisions.

2.1 Camera Nodes

A camera node, or viewpoint, is defined as a node for which routes originate and terminate. In the context of this work we considered only a subset of camera viewpoints, only nodes in the service partition.

The following image shows a shadow mesh visualization:

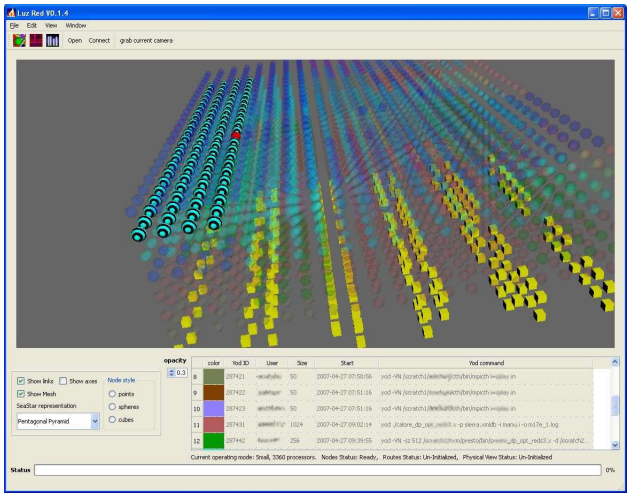

Figure 5: Shadow Mesh

The shadow mesh image depicts the camera (service) nodes as yellow cubes. The Seastar that is down is represented by the red sphere in the middle of the row of cyan and black striped spheres. The rest of the spheres in the image represent compute nodes, and are colored by job (Yod ID). The cyan spheres with stripes are the nodes that are in shadow of the down Seastar from the perspective of the camera nodes. We make transparent nodes that we want to see, but are of less interest in this particular analysis.

The next image shows another view of the shadow, with the service nodes on the left, and the downed Seastar and resulting shadow on the right plane:

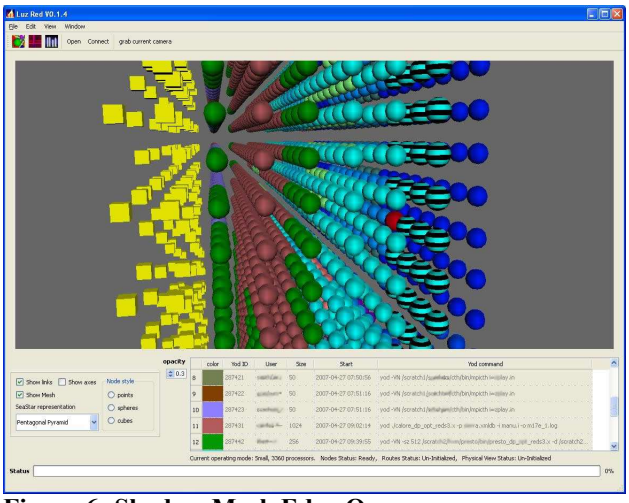

Figure 6: Shadow Mesh Edge On

2.2 Shadow Flavours

The *service* shadow mesh is defined as the aggregate set of nodes that are unreachable from/to the PBS, Lustre service nodes, and sdb.

The *data path* shadow mesh is defined as the aggregate set of nodes that are unreachable from/to the I/O nodes.

To date we have focused on looking at the service shadow. In fact there are other classes of shadows that should be explored, for example the compute mesh has its own distinct shadow with respect to itself. The following image shows various service shadow viewpoints:

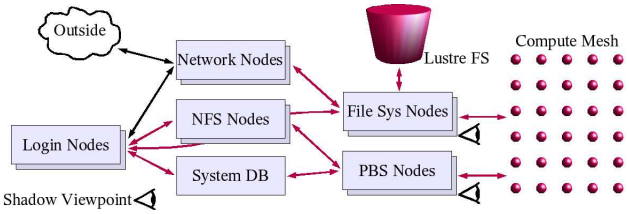

Figure 7: Red Storm Service Shadow Viewpoints

2.3 Islands of the Mesh

One interesting concept that arises when exploring the shadow mesh is the question of which parts of the machine are still usable after one or more Seastars or links go down. This is akin to the question of which future job submissions will run to completion. Asked another way: Is it possible, following a routing related failure, to identify a portion

of the mesh for which a job can be submitted and executed and run to completion (i.e. capture the results of the job)? The answer to this question is non-trivial, to say the least.

Most likely the answer is 'no' for any job that tries to write results data to a file and for which a Seastar goes down on a Lustre node in the service partition. For other areas of the mesh, it is not so clear. Timing is also an issue. Once a job is submitted via PBS, if the submitting PBS node's Seastar or login node goes down, the job may very well still run to completion. We expect as this research continues, we will be able to answer this question for a subclass of queries for certain node types (PBS, login, compute, etc.) of the mesh.

3. LuzRed Data Sources

3.1 Data Sources in Red Storm

LuzRed employs several databases. MySQL is used extensively for analyzing jobs in historical and real-time contexts. The routes are also stored in MySQL (see Appendix A) — we have experimented with two variations to store the routes.

Other information is created and stored in flat files— a lookup table file is generated using the output from the "`rtr -Ii`" and the output from the "`xtnidname`" to easily convert between logical, physical, and NID node addresses. The address table lookup has this format:

```
…
c9-1c1s0s2,[9,5,23],4738
c9-1c1s0s3,[9,4,23],4739
c9-1c1s1s0,[9,7,22],4740
…
```

For the shadow mesh we store the list of service nodes — PBS, lustre, login, sdb, 10GB network, in a file that has this format:

```
…
nid00076,c0-0c2s3n0,login
nid00080,c0-0c2s4n0,lustre
nid00087,c0-0c2s5n3,10GB
nid00099,c0-1c0s0n3,unused
nid00112,c0-1c0s4n0,pbs
nid00192,c0-2c0s0n0,sdb
…
```

3.2 MySQL Issues

The Red Storm XTAdmin database is extensively used for our visual analysis. However, as we progressed with our work we found it necessary to mirror the MySQL database(s) on a separate machine. Although this requires extra software to be written, and work to monitor and maintain the mirror, it has the following advantages:

1. We are not completely dependent upon the stability of Red Storm, i.e. booting due to maintenance and servicing;
2. If the login or other service nodes are not accessible we can still perform our analysis;
3. We are not dependent upon the machine configuration (classified vs. unclassified)
4. We mirror the database(s) on a non- Red Storm node, away from the XT3 installation and building. This increases significantly the uptime and availability to the XTAdmin database.

3.2.1 XTAdmin mirror

As described above, we mirror portions of the XTAdmin on a system that is not co-located with Red Storm. We do not copy the entire database, mainly for speed of loading, and because our analysis to date did not require data from all of the tables.

The XTAdmin mirror is updated either manually or in a semi-automated fashion. We implemented a python client server program, dbMonClient and dbMonServer, respectively using the python socket library. The role of dbMon {Client|Server} is to periodically monitor the XTAdmin database on Red Storm and look for changes to the database as a result of new jobs starting or old jobs ending. If the monitor detects a change the mirror XTAdmin database on our visualization machine is updated with a snapshot from the actual XTAdmin running on Red Storm.

3.3 Building the Routes

The routes flat-file database is generated whenever Red Storm is rebooted. A script is run that dumps a flat text file using the output from "`rtr -IR`." These files can be quite large. In the small operating mode[9] the file is 3.1Gigabytes. We can expect this file size to increase in proportion to the square of the mesh. Thus for jumbo mode[10], which is approximately four times small mode, we should see a sixteen fold increase in the routes file, or about 48 Gigabytes.

A MySQL routes database is generated by processing this text file using a C++ program. We implemented two slightly different forms of the database, which provide trade-offs in speed of queries vs. time-to-build. One format, which we call, routes_hop_per_row Database schema:, has a simple organization to store each hop of the routes in its own row.[11] The other database routes_hops_combined Database schema:, stores an entire route in a single row of the database by encoding the hops of the route in a text field. See Appendix A for database schemas.

3.3.1 Performance Concerns

As of this writing it takes approximately 1 hour to build the routes_hops_combined database, and 17

hours[12] to build the routes_hop_per_row database for Red Storm's small size Operating Mode.

While building the database is required only at boot-up, we would like to improve this turn-around time. Several ideas are being explored:

- Parallel or cluster implementations of MySQL or other database technologies;
- Reduce the number of MySQL INDEXes;
- Move the MySQL server to a 64-bit platform[13];
- Store the entire routing table in virtual memory.[14]

4. Core LuzRed Visualization Functionality

This paper has focused to this point on the shadow mesh and database issues. At this point we would like to delve into some detail of the core features of the LuzRed visualization. The graphics application currently supports three main types of view: logical, physical, and routes[15].

4.1 Logical View

The Logical View is essentially a graphical representation of the Red Storm logical coordinate system, with nodes represented by spheres, and colored by its Yod ID through a simple lookup table. The color in 3D space matches the color square next to each job, on the main GUI panel (see Figure 2: LuzRed Main Panel).

The Logical View snapshot below shows how we can hone in on a specific job. It is selected and highlighted in the Yod table on the GUI, with the corresponding nodes for that job drawn opaque on the main 3D view window, and all other nodes draw transparently:

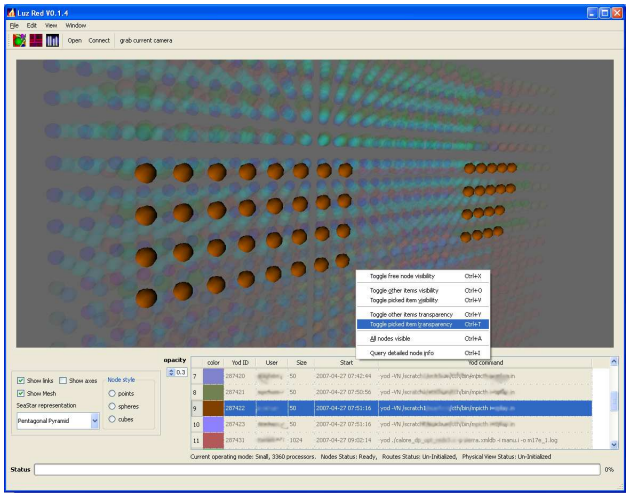

Figure 8: Job highlighting

4.2 Physical View

The physical view attempts to depict a layout of the Red Storm nodes in a manner that approximates the real physical layout of the nodes, cabinets, and cages.

Below we see a snapshot of jobs mapped to the physical view. Each "pizza box" represents a node in the mesh.

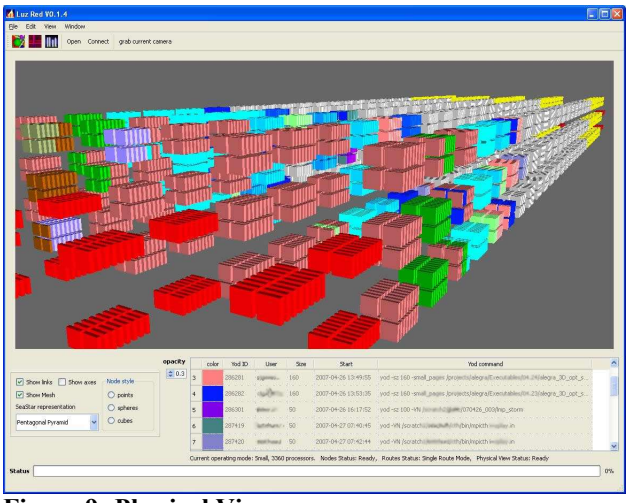

Figure 9: Physical View

4.3 Routes View

The routes view in LuzRed is mapped to Red Storm's logical coordinate system:

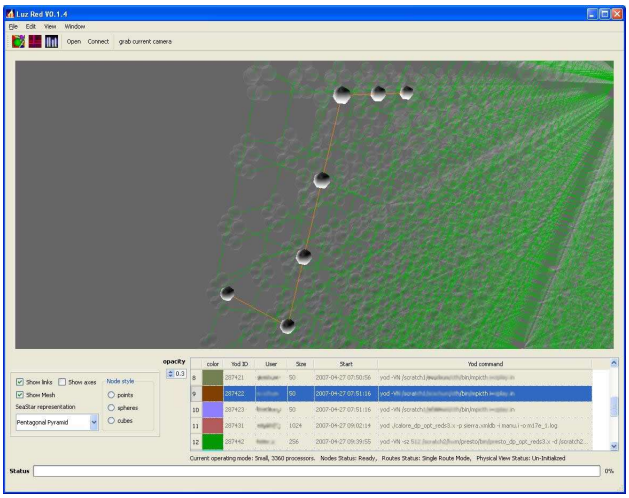

Figure 10: Routes View

In this view we depict the Seastars as a cluster of six spheres (one for each Seastar port). The links are represented by green lines. In the image above a route has been highlighted by coloring its links orange.

4.4 Security

The restrictive environment around Red Storm poses challenges to accessing system information and developing supporting applications. Because of the two classification levels available on the machine, the machine lives inside of a facility similar to a vault. All communications with the machine are well secured, which also implies that the firewall protecting the system makes it difficult for LuzRed to access the system management workstation for current information. As discussed earlier, and for ease of development, snapshots of the system data bases are mirrored on servers that provide more convenient access. LuzRed must also detect and adjust its visualization according to how Red Storm is switched or partitioned between the classified and unclassified ends. Developing LuzRed has been challenging because of the wide dispersal of hardware and software expertise, and the one-of-a-kind nature that is Red Storm.

5. Implementation

5.1 VTK, Qt, and ParaView3

A principle goal of LuzRed is to provide a robust and effective tool for the variety of prospective users. Our goals were to create a tool that could provide advanced visualization capabilities, user friendliness, and provide real-time performance. For performance consideration we chose to implement the application in C++. The user interface is built using Qt, a cross-platform GUI toolkit.

At the onset of development for LuzRed, development work had recently been underway on ParaView3[16]. If we were starting the project today, we would probably more seriously consider using ParaView3 as the basis for the visualization tool to LuzRed.

6. Conclusion

We feel that while significant progress has been made in using visualization to analyze aspects of Red Storm job and routing, much work remains to be done. The addition of certain graphics features to the logical and physical views of LuzRed, such as text annotation, hypercylinder[17] representation, and multiple views, will make LuzRed a better tool and aid faster analysis.

Other areas of the LuzRed system that need improvement are MySQL performance issues, as mentioned earlier, and automation of certain data streams.

There is much interesting work to continue in the shadows realm: exploring other shadow types, and refining our algorithms for quicker reporting.

Dynamic rerouting is another interesting research topic. XT3 hardware limitations preclude the ability at this point to consider dynamic re-routing of the mesh when nodes are down. However it is useful to contemplate this ability/feature, for some future design architecture.

Acknowledgments

The authors would like to thank Sandia folks—Bob Balance for stimulating this work and providing technical guidance at many levels, Dr. John Greenfield for encouraging us to submit to CUG2007 in the first place, Jon Stearley and Vitus Leung for discussions about routing and the shadow mesh. We'd also like to thank Cray people that helped us understand Red Storm better: Victor Kuhns, Bob Purdy, and Barry Oliphant.

About the Authors

Jon Goldman works on Scientific Visualization at Sandia, and enjoys good coffee. Jon has an undergraduate degree in Electrical Engineering and a Masters degree in Computer Science.

Warren Hunt received a BS degree in computer science from the University of New Mexico, and will complete the MS degree in computer science in July 2007, also from the University of New Mexico. His research interests are in scientific visualization and scientific computing. He is a member of IEEE and ACM.

Appendix A: MySQL databases

routes_hop_per_row Database schema:

```
+---------------------+-------------+------+-----+---------+-------+
| Field               | Type        | Null | Key | Default | Extra |
+---------------------+-------------+------+-----+---------+-------+
| beg                 | smallint(6) | YES  | MUL | NULL    |       |
| end                 | smallint(6) | YES  |     | NULL    |       |
| hop_src             | smallint(6) | YES  | MUL | NULL    |       |
| hop_dst             | smallint(6) | YES  | MUL | NULL    |       |
| hop_src_output_port | tinyint(4)  | YES  |     | NULL    |       |
| hop_dst_input_port  | tinyint(4)  | YES  |     | NULL    |       |
+---------------------+-------------+------+-----+---------+-------+
```

Sample route:

```
mysql> select * from route where beg=1200 and end=1103;
+------+------+---------+---------+---------------------+---------------------+
| beg  | end  | hop_src | hop_dst | hop_src_output_port | hop_dst_input_port |
+------+------+---------+---------+---------------------+---------------------+
| 1200 | 1103 |    1200 |    1201 |                   0 |                   5 |
| 1200 | 1103 |    1201 |    1202 |                   0 |                   5 |
| 1200 | 1103 |    1202 |    1203 |                   0 |                   5 |
| 1200 | 1103 |    1203 |    1040 |                   0 |                   5 |
| 1200 | 1103 |    1040 |    1041 |                   0 |                   5 |
| 1200 | 1103 |    1041 |    1042 |                   0 |                   5 |
| 1200 | 1103 |    1042 |    1043 |                   0 |                   5 |
| 1200 | 1103 |    1043 |    1039 |                   3 |                   2 |
| 1200 | 1103 |    1039 |    1035 |                   3 |                   2 |
| 1200 | 1103 |    1035 |    1031 |                   3 |                   2 |
| 1200 | 1103 |    1031 |    1027 |                   3 |                   2 |
| 1200 | 1103 |    1027 |    1119 |                   3 |                   2 |
| 1200 | 1103 |    1119 |    1115 |                   3 |                   2 |
| 1200 | 1103 |    1115 |    1111 |                   3 |                   2 |
| 1200 | 1103 |    1111 |    1107 |                   3 |                   2 |
| 1200 | 1103 |    1107 |    1103 |                   3 |                   2 |
+------+------+---------+---------+---------------------+---------------------+
```

routes_hops_combined Database schema:

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| beg   | smallint(6) | YES  | MUL | NULL    |       |
| end   | smallint(6) | YES  |     | NULL    |       |
| hops  | text        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Sample route:

```
mysql> select * from route where beg=1200 and end=1103;
-----------------------------------------------------------+
|beg    |       end     |       hops
-----------------------------------------------------------+
|   1200    |   1103    |   c2-1c2s4s0l0v1;c2-1c2s4s1l0v1;c2-1c2s4s2l0v1;c2-1c2s4s3l0v1;c2-0c0s4s0l0v1;c2-
0c0s4s1l0v1;c2-0c0s4s2l0v1;c2-0c0s4s3l3v1;c2-0c0s3s3l3v1;c2-0c0s2s3l3v1;c2-0c0s1s3l3v1;c2-0c0s0s3l3v1;c2-
0c2s7s3l3v1;c2-0c2s6s3l3v1;c2-0c2s5s3l3v1;c2-0c2s4s3l3v1;c2-0c2s3s3; |
+------+------+-----------------------------------------------------------------------------------------------------------------------
```

Mirror of XTAdmin used in LuzRed:

```
+----------------------+
| Tables_in_XTAdmin    |
+----------------------+
| partition            |
| partition_allocation |
| processor            |
| service_processor    |
| yod                  |
| yod_allocation       |
| ---------------------|
```

References and Endnotes

[1] http://www.sandia.gov/news-center/news-releases/2002/comp-soft-math/redstorm.html
[2] http://en.wikipedia.org/wiki/Cray_XT3
[3] Bob Ballance, Scientific Computing Systems department, Sandia National Laboratories
[4] Red Storm's nodes are logically laid out in a regular 3D volumetric grid. The "physical" layout corresponds more closely to actual cabinets, rows of cabinets, module boards, etc. of the actual system as it resides on site at Sandia.
[5] Visualization ToolKit— See: http://www.vtk.org
[6] See www.trolltech.com
[7] http://www.cmake.org/HTML/Index.html
[8] Error or malfunctioning cases and determining situations when Red Storm does not need to be rebooted is a fairly large and open topic. If the work described in this paper influences the uptime of the machine, we will have considered LuzRed largely successful.
[9] Small mode = 3360 compute nodes, 320 service and I/O nodes
[10] Jumbo mode = 12960 compute nodes, 320 service and I/O nodes
[11] See Appendix.
[12] Queries are reasonably fast, in the range of seconds or less. To query and generate an entire shadow mesh takes approximately two minutes, quite acceptable for our needs.
[13] The shadow mesh was generated from a MySQL database installed on a dual-processor Intel Xeon 3.2GHz machine running Red Hat Enterprise Release 3
[14] For small mode system, using STL map container as the storage semantics, we estimate the entire route table can be stored in about 5GB of RAM.
15 The alert reader may notice some similarities to the work found in *CrayViz, a Tool for Visualizing Job Status and Routing in 3D on the Cray XT3*, John Biddiscombe, and Neil Stringfellow, Swiss National Supercomputing Centre, and *XT3 Operational Enhancements* Chad Vizino, Nathan Stone, J. Ray Scott {vizino,nstone,scott}@psc.edu Pittsburgh Supercomputing Center, both papers presented at Cray User Group Conference 2006, Lugano, Switzerland, May 8-11, 2006. We used some of the ideas from the Graphical Monitor listed in the Vizino et al. al. paper. We did not use ideas directly from Biddiscombe et al but other Sandians with whom we work did see the Biddiscombe presentation at CUG 2006, and discussions over 2006 did influence our work.
[16] http://www.paraview.org/Wiki/ParaView_III
[17] Sandia's Red Storm topology is not the general torus of other installations, but rather a hypercylinder, where only the Z-axis wraps.