CSCS

Swiss National Supercomputing Centre

CrayViz
A Tool for Visualizing Job Status and Routing in 3D on the Cray XT3

John Biddiscombe and Neil Stringfellow

CSCS

Overview

- A set of tools were used or developed at CSCS to analyse a particular failure pattern on the Cray XT3
  - The failure mechanism has now been corrected !!
- The initial tools were text based and helped in the analysis
- It was soon evident that a more powerful tool was required to visualize the state of the system
- The tool which was developed showed potential, given further development, to analyse the working system

Background : The failures

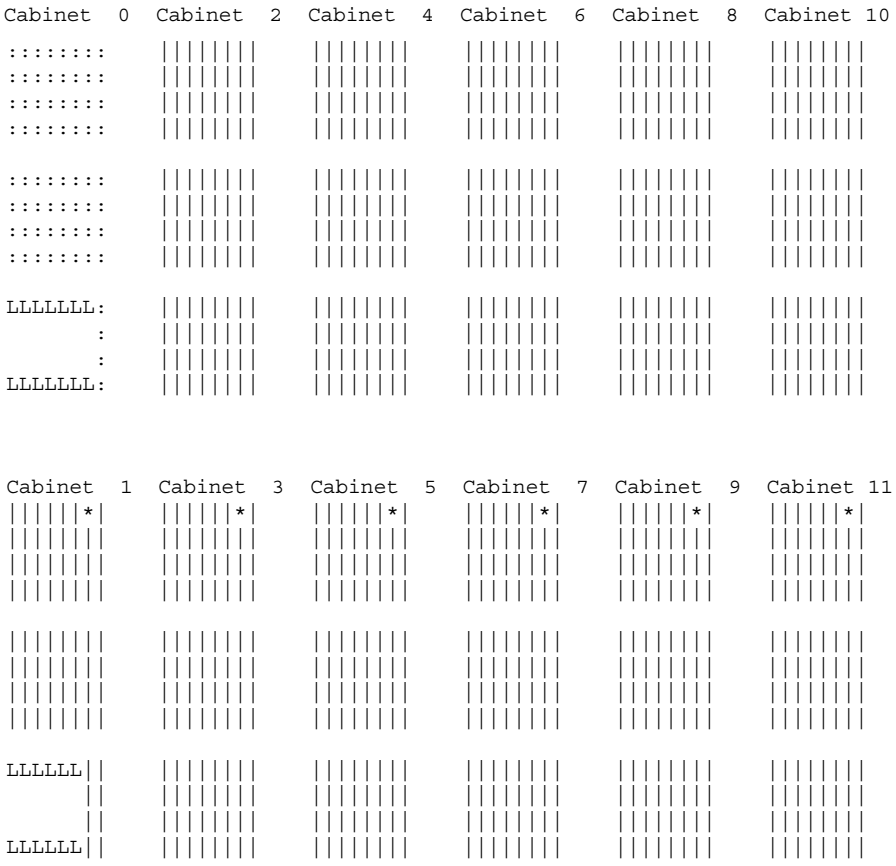
- A whole set of nodes would fail in a pattern
  - On our one cabinet machine this was in chassis 2, for the SeaStar chips in position 0 on every blade
  - On the larger machine things were sometimes more complex, but normally on positions in blade 6 in cabinets
- The failures showed up in the database of downed nodes when a job failed to start
- The `ping_node`/`ping_list` utilities often showed these nodes as bad without an attempted job launch
  - From different login nodes, different patterns could be seen
  - **This allowed the possibility of identifying the most important individual node in the failure pattern**

The tools available

- The current state of the machine, (as maintained by the service database), can be seen using the `xtshowmesh` utility
  - Output from `xtshowmesh` was designed for Red Storm (i.e. the layout of nodes was using the topology of Red Storm not our system)
  - The XT3 machines at CSCS are connected as a Torus in all 3 dimensions
- A utility called `ping_list` which we obtained from PSC is able to see if a node is reachable from the login node
  - `ping_list` is a standalone multi-node version of `ping_node` from Sandia/Cray
- These two tools combined allowed us to get a better picture of the state of the machine
  - (Now, `xtshowcabs` replaces `xtshowmesh` displaying nodes in a more universal XT3 form).

Simple scripting for a new tool

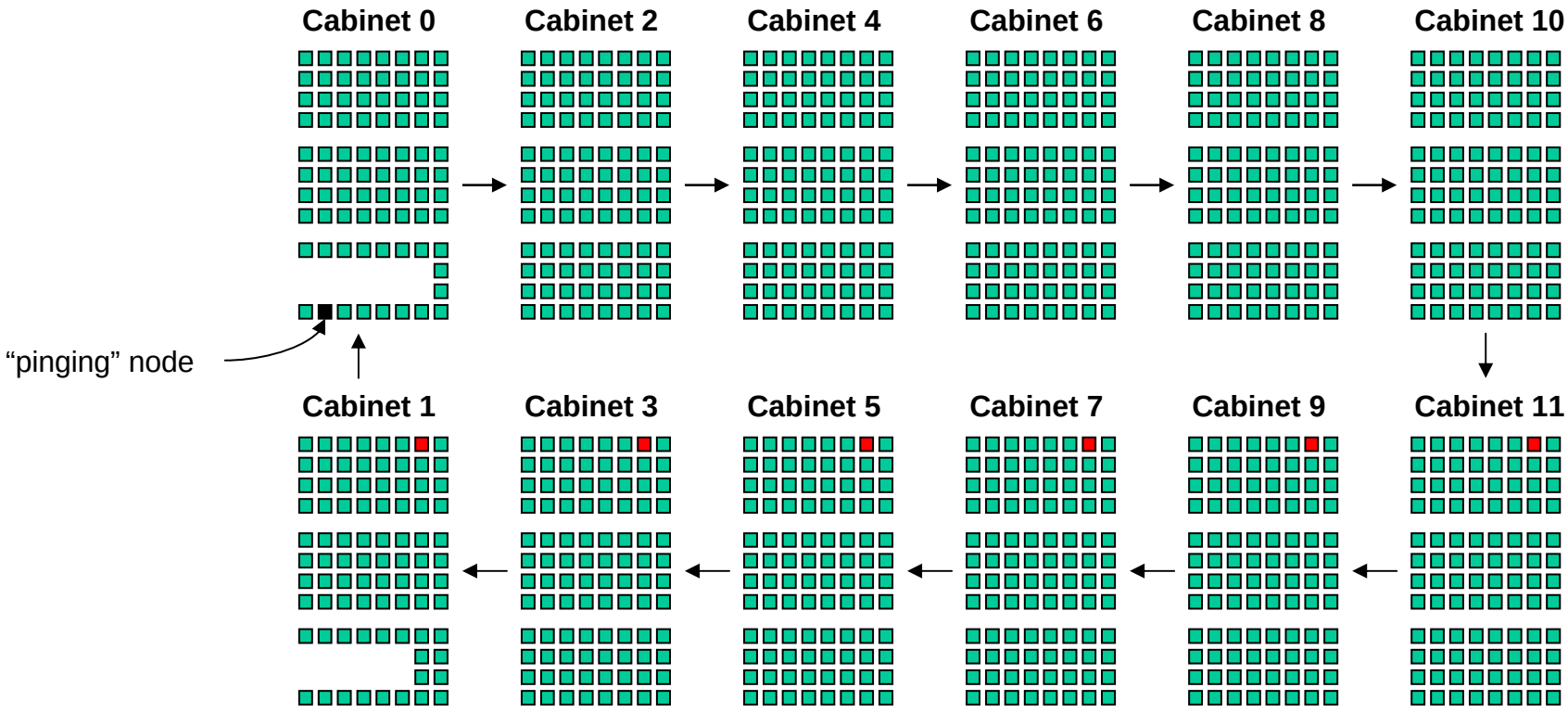
- The combined the output of the two utilities to see the failures in cabinets as shown on the next slides




```
Legend:
     nonexistent node
L    unallocated Linux node
:    free interactive compute node
|    free PBS compute node
X    failed compute node
*    failed node in SDB and ping_list
-    failed node in ping_list
&    failed node in ping_list with running job
```

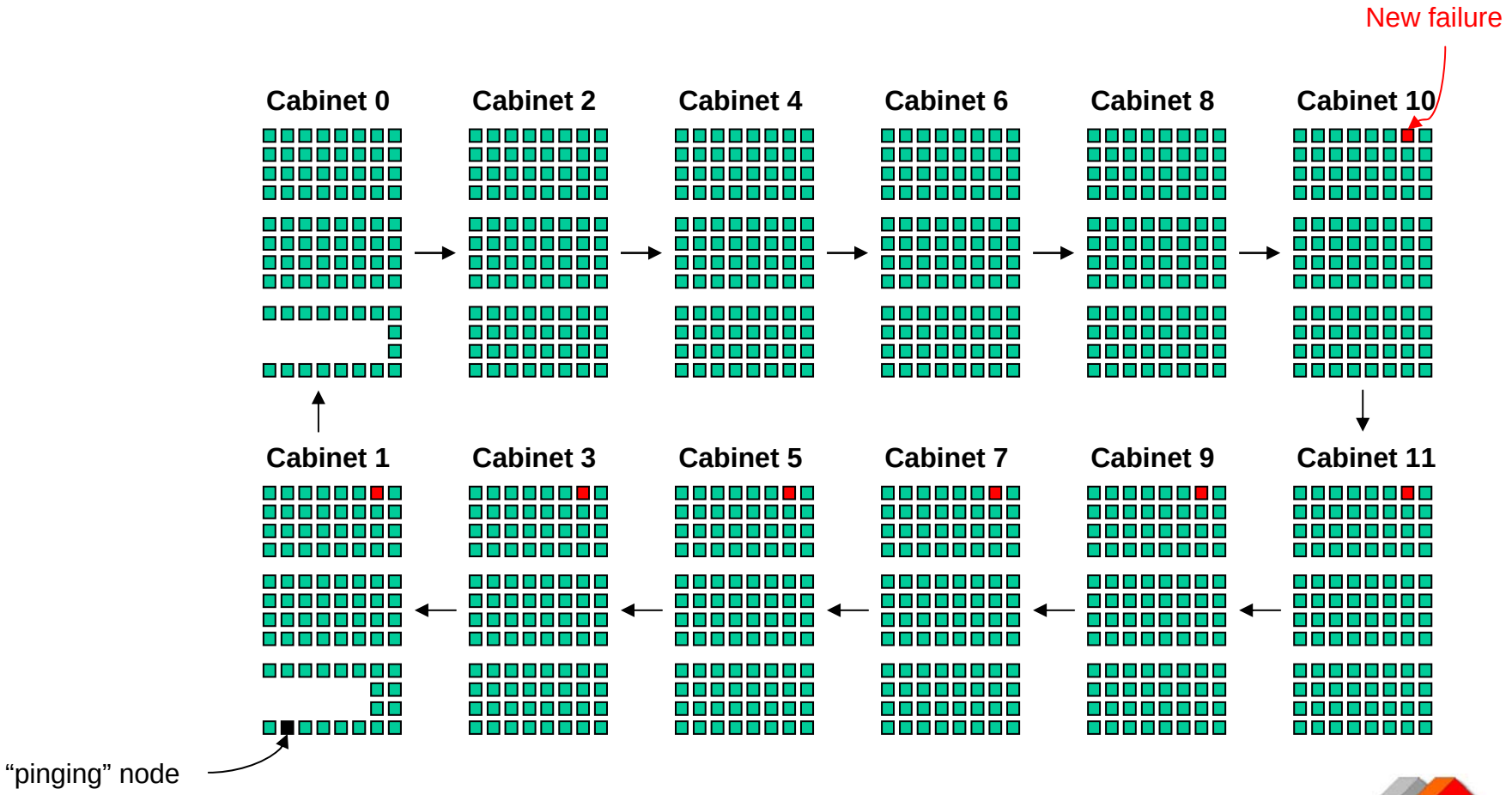
Example : View from cabinet 0

- There is one failed node on each of the odd numbered cabinets



Example : View from cabinet 1

- There is an additional failure on cabinet 10



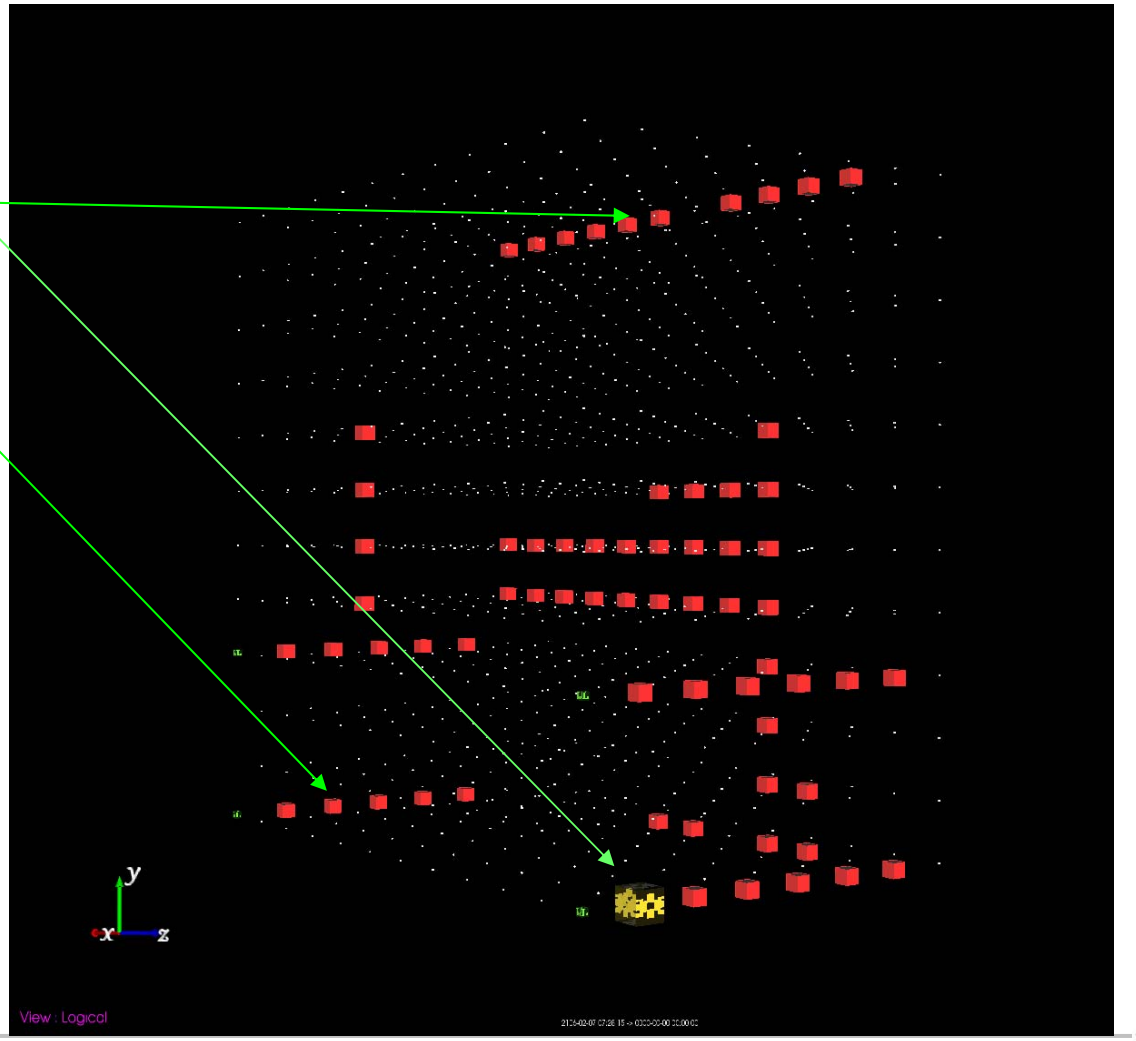
View in 3D of real node fail output

- The pattern of failed nodes is slightly clearer when viewed in 3D

Login Node

Failed Nodes

NB. The service/linux nodes also appear as failed

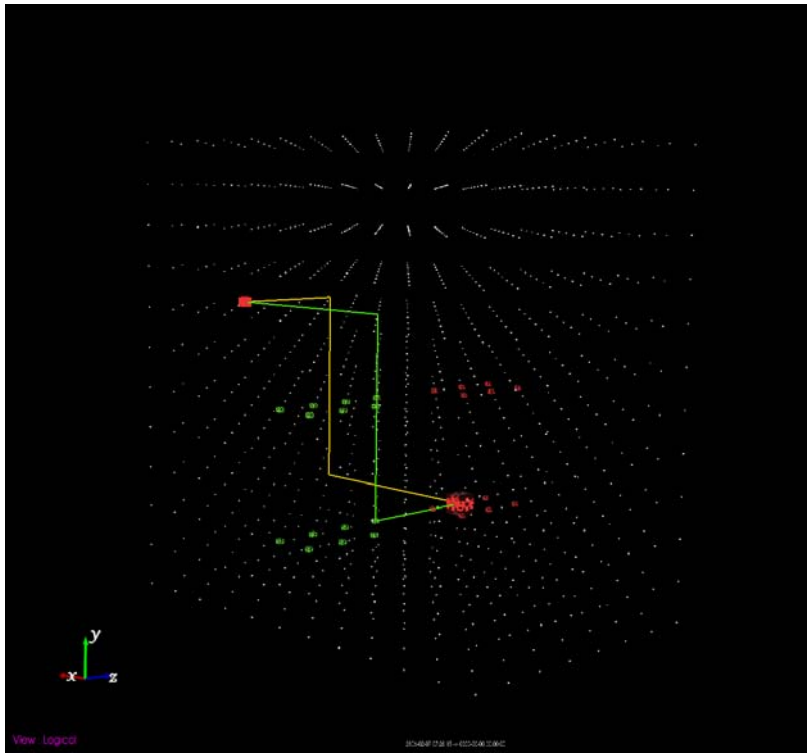


Routing tables - paths in the mesh

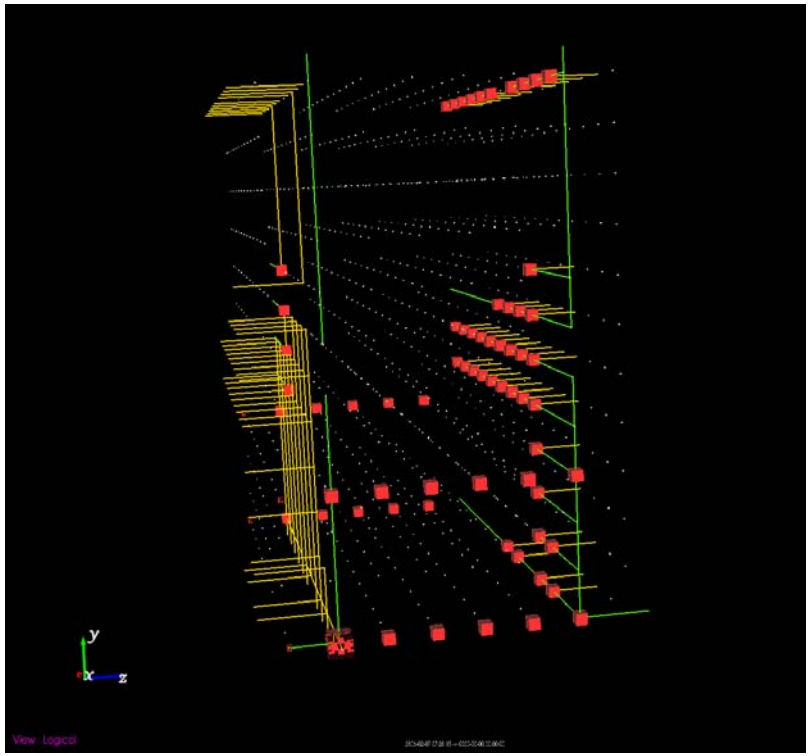
- The routing tables for the XT3 are generated at boot time, and are static
  - bad for avoiding bottlenecks and failures
  - good for predicting paths between processors
- These paths enable us to interpret patterns of failures
  - Different patterns of failures when viewing the system from different processors
- One bad SeaStar connection can make several processors become “invisible”
- The route from B to A is normally *not* the reverse of the route from A to B

How are the nodes connected

We want to know how the physical layout of processors and the connections between them relate to the topology of the torus.



A single route (forward and return) with the start and end points rotated into the centre of the cube



Routes to all failed nodes, links passing off one edge and onto the other appear as dangling sticks

What do (did) we want to know ?

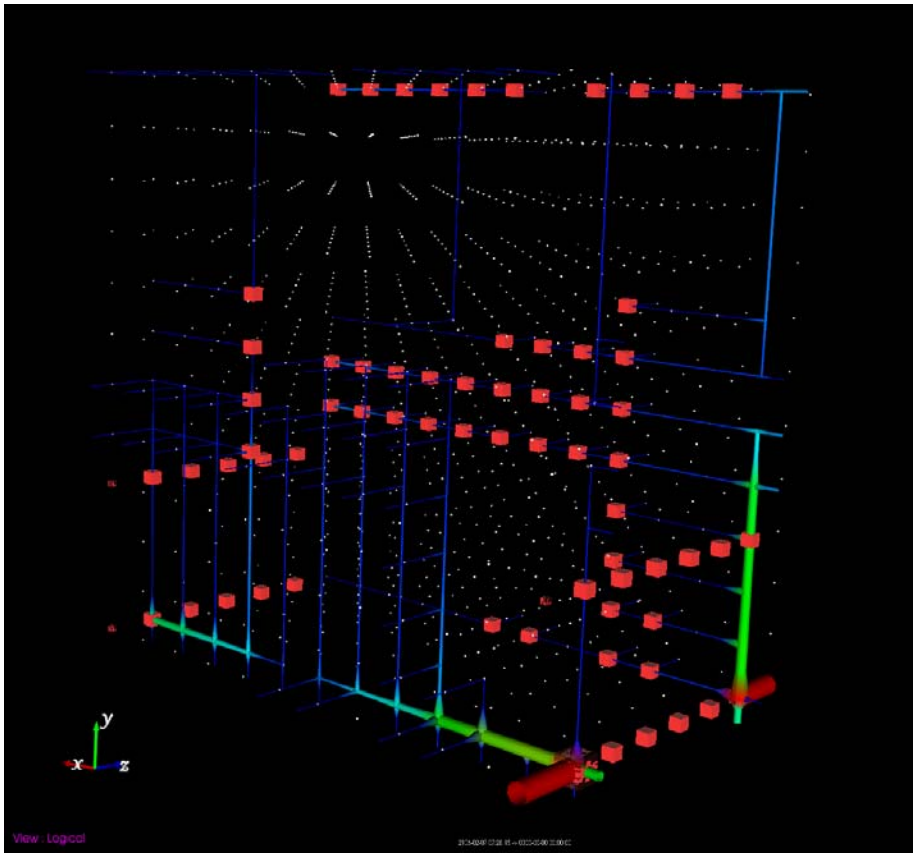
- Are node failures on the physical cabinets in a pattern on the torus ?
- Are there common routes which need to be taken from the pinging nodes to the pinged nodes ?
- How are the jobs distributed on the machine ?
  - Failures on the nodes might only become visible after a job has tried to launch on a set of nodes
- The XT3 “problem” was resolved by increasing the voltage on the SeaStar chips just as we were about to “solve the mystery” of the failing nodes and how they fit the patterns we could see.

Route analysis in 3D

We wanted to see which links between nodes were the common denominators in the routes to the failed nodes

In this image, each link from the ping (login) node (and back) to a failed node is traversed and the count is incremented each time the link is used

we colour and thicken tubes around the links to show the amount of usage and thereby identify the most probable points where a failed SeaStar might have caused our pattern of failures.



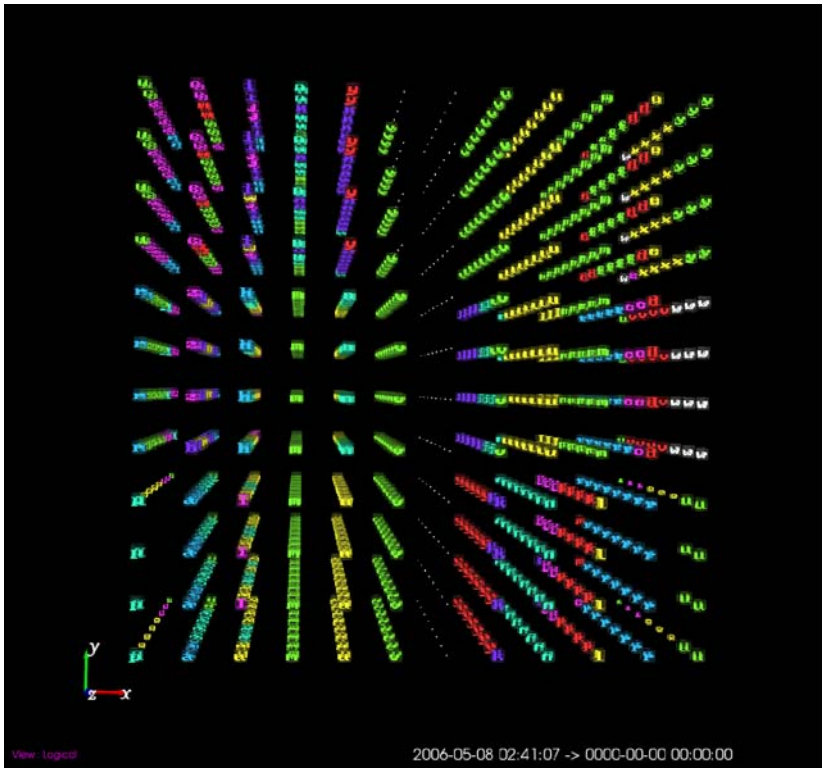
Adding MySQL abilities to CrayViz

- Data about all jobs started and stopped is logged in an SQL database on the SDB node.
- Instead of parsing the output of `xtshowmesh` or `xtshowcabs` (after SSH'ing into the machine) we can instead query job information directly from the database (even over the network, once the SDB machine was fitted with a network card)
- Not just data for 'now', but yesterday and the day before too (as far back as the machine was installed and data collection began)
- System information is also available – Mapping of Processor to jobs, IO, Service, Login nodes, and a wealth of additional information – even topological information (whether links between processors are present).
- Client display can be on any machine, even a laptop at a conference.
- Any XT3-like machine can be examined
- The Database can be dumped to a desktop or laptop and taken away for experimentation.

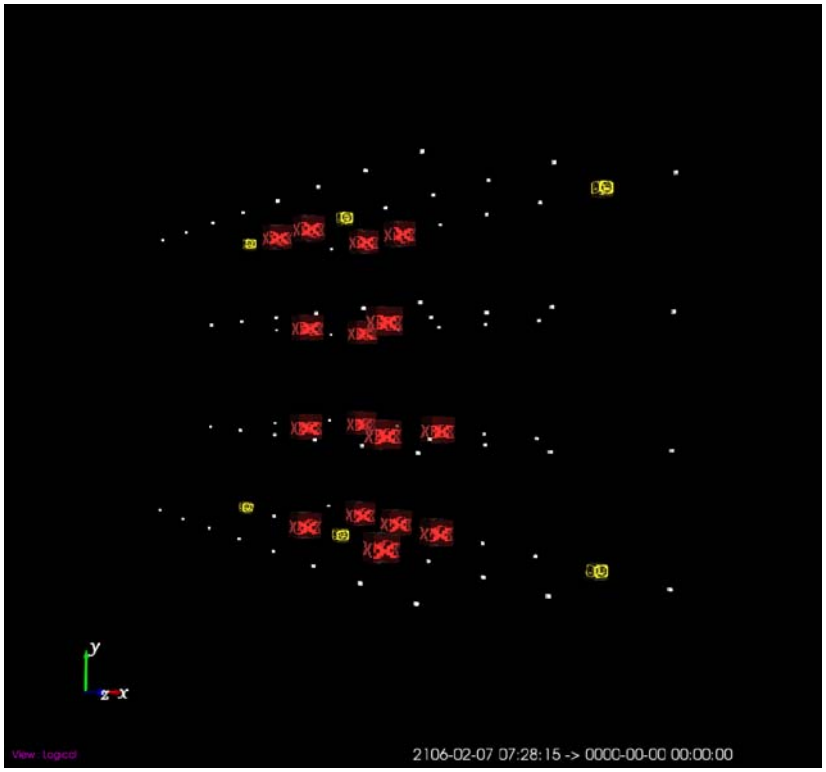
System configuration

Here we see two of the XT3 machines at CSCS, the configuration of the machines; login, IO, service and even missing nodes, is detected automatically from the SQL database

The only hardcoded information is the actual topology (how routes are generated)



palu – a 12 cabinet configuration

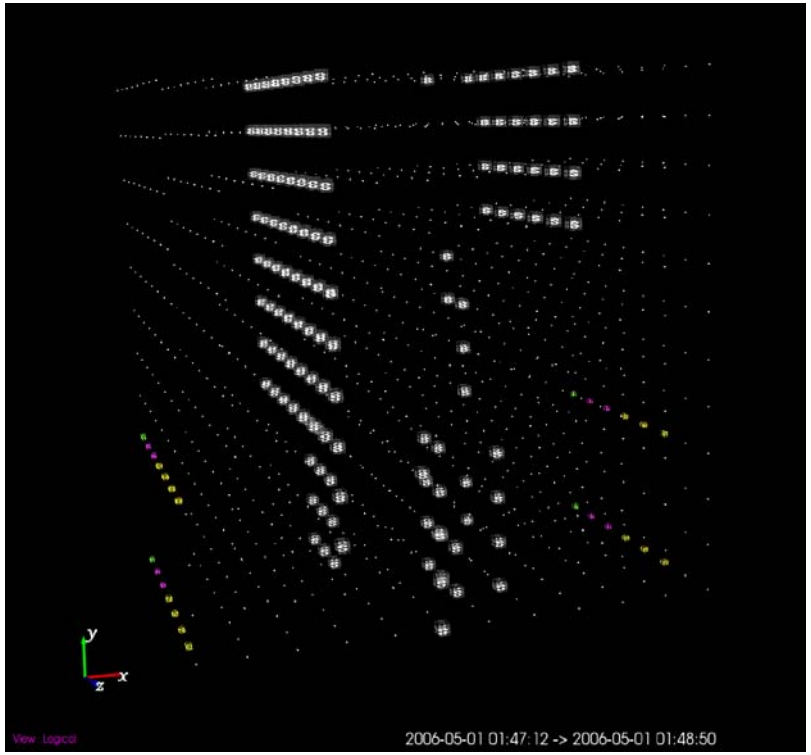


gele : a single cabinet

Viewing job distribution

The tool opens up interesting possibilities for viewing job distribution on the machine

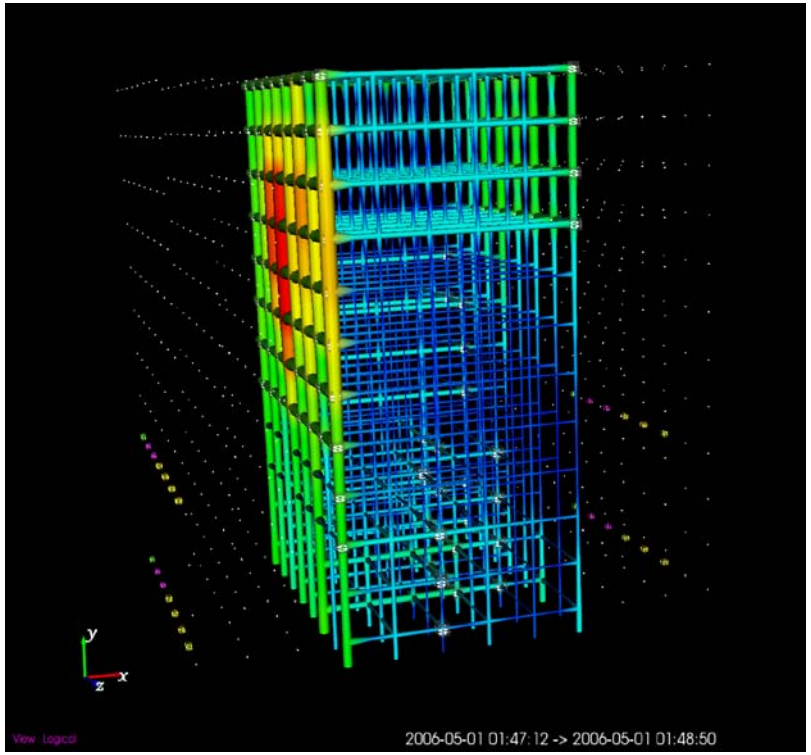
- are nodes placed close together on the machine?
- are nodes placed in a good topology for the job?
- Allocating processors from 1-N is not often a good way of doing it.
- This job is quite fragmented
- How does this affect performance



Viewing “theoretical” job traffic

The “theoretical” traffic patterns between nodes for a particular job can be examined

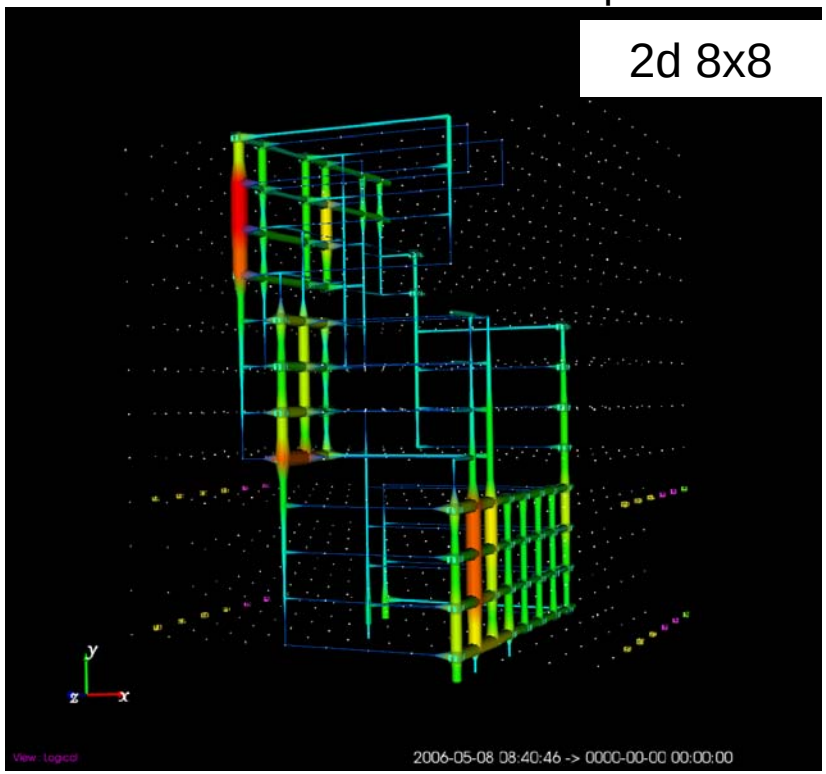
- by theoretical we mean we are not collecting real data – instead predicting routes and route usage based on possible communication patterns
- Fat red tubes represent links with higher traffic, thin blue tubes are lower traffic.

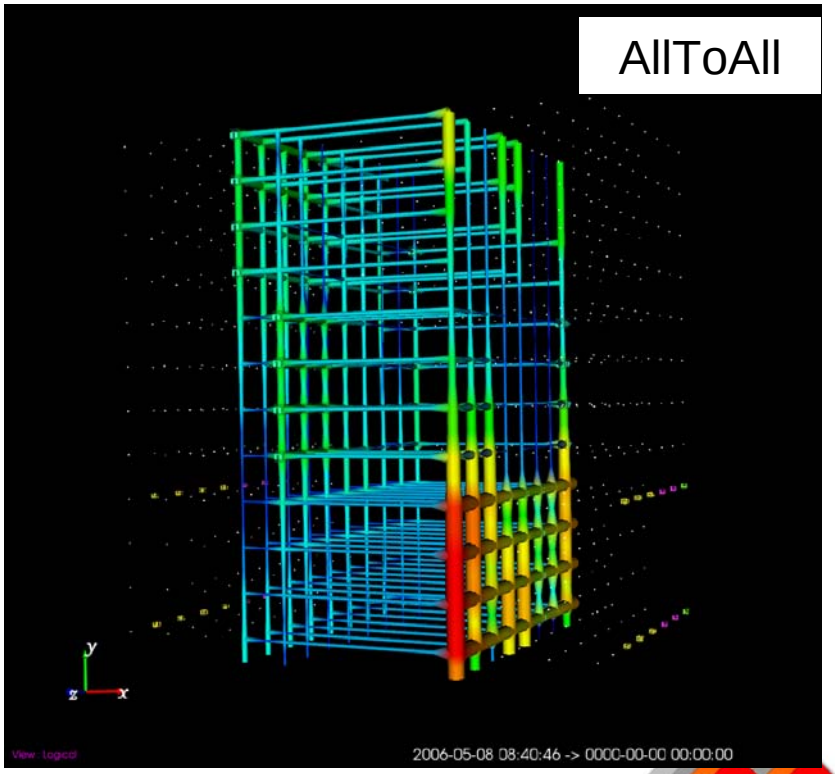


Different traffic patterns

Two different traffic patterns have been considered

- All To All : Each compute node sends and receives packets to all other compute nodes. (e.g. some molecular or N body simulations)
- Nearest neighbour - Grid 2D or 3D : Computes nodes send packets to their nearest neighbour nodes in 2d or 3D grids
- The routing patterns are very different when considering a single job using a different communication pattern

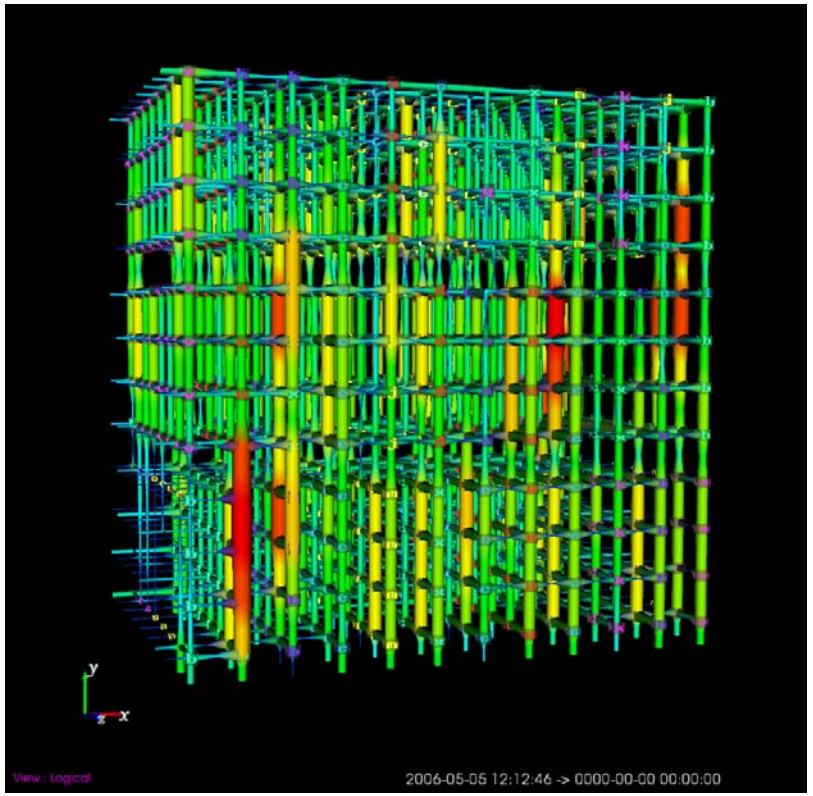




Looking at other problems

Removing a blade from the system

- If a blade is removed, then the routing araound that region will cause a huge change in the traffic pattern.
- The image here shows all jobs running at an instant of time with the traffic pattern (hypothetical) shown.
- Consider where the red hotpots will be when a region of mesh disappears from the routing space.
- We have not implemented the complete routing algorithm to handle the case of missing/failed SeaStars



Using CrayViz

What sort of question can CrayViz help to answer?

- If a job J ran for N hours and generated X amount of data last month, but when run this week only generated 30% of the data, what caused the difference?
  - We can look at the distribution of processors (both for job J and all other concurrent jobs) at both times to see if network bottlenecks might be the problem
  - We can *animate* the other jobs running during the lifetime of job J to see if some unusual activity might be to blame. Animation provides an important visual means of quickly interpreting job data.
- If job J crashed unexpectedly, can we see if some system related issue caused by other jobs or nodes was to blame

Further Development

- Possible communication performance data could be taken from real jobs and mapped to the topology
- The tool uses the SDB SQL database to do nearly all the work, only the routing tables must be “known”, obtaining the required information about routing would make the tool completely portable.
- The SQL database is an excellent means of storing retrieving the data, if all HPC machines used the same format - one tool would work for all.
- Use of the tool to complement a future compute processor allocator

Acknowledgement

- Thanks to Pittsburgh Supercomputing Centre for providing us with the `ping_list` utility