A slice of the NERSC data collect system
• **NERSC is moving between data centers.**
  
  – Old data center is a mis-mash of systems.
  
  • 3 separate building control systems
    
    – One is Windows 98 based
    
    – Two are embedded Windows
      
      • This one is browser based, except..
      
      • You load an active-x object to check java version
        
        • And if you do not have the correct version, downloads it from a 3rd party website..
      
      • Then you load java to run a java-ws program..
      
      • Which then downloads and runs a Windows DLL to provide window decoration.
    
  – New data center has no chillers.
A Slice of what NERSC is doing at CRT

- **Introduce several new ideas.**
  - Clean slate – no legacy to clean up or rebuild.
  - Security is paramount.
    - Most BAS/metering protocols are insecure.
  - Ability to Fail fast
  - Modularity
    - We demand the ability to replace anything, at any time, for any reason.
  - Use services that are designed for HA
    - We want failover, maintainability, and scalability
  - Zero downtime.
    - We are the first up, and the last down
A new way of doing things.

- System must be first one up, and the last one down.
- System has many sensors, collectors and networking.
  - 1600+ temperature sensors
  - 800+ power meters
    - Substations master breakers, Substation breakers, panel breakers
  - 600+ power strips (Raritan, 50/60 amp is the preferred vendor/model)
    - With breaker level monitoring – 6 breakers per strip.
  - 200+ bacnet devices
    - Many with logical sub-devices.
  - 2k+ ethernet ports
  - 24+ network switches
  - 4 Seismometers
- The data roach hotel.
  - Data checks in, but never checks out..
    - HPSS is your friend.
More Requirements

- No plugins.
  - No java, no active-x, no flash

- HTML5 support.

- Desired to work on Cell phones, tablets, Linux, OSX, Windows.

- Stream processing
  - Ability to calculate a PUE in realtime
  - Fault detection
  - Security incidents
  - Building Control
  - Filtering

- Instrument the data collectors themselves.
I am not a number!

- **Metrics 2.0 implementation**
  - What the heck is metrics 2.0?
  - A different way of representing data
  - Attaching meta information to the metric.
  - See [http://metrics20.org](http://metrics20.org)

**Traditional systems**

```
collectd.dfs1.df.srv-node-dfs10.df-complex.used

diskspace...srv_node_dfs10.byte_used
{
    server: dfs1
}
```

**Metrics 2.0**

```
{
    server: dfs1
    what: disk space
    mountpoint: srv/node/dfs10
    unit: B
    type: used
    metric_type: gauge
}
```

```
meta: {
    agent: diamond,
    processed_by: statsd2
}
```
New technologies

- **Ovirt (RHEL) VM management**
  - VLAN tagging/trunks
    - PXEboot needs untagged packets.
  - Dense servers – 8 nodes in 4U slot.
    - Supermicro Fat-twin based

- **Rancher with Docker containers**

- **Split networks**
  - Have to connect to 10 different networks.
  - 3 public networks
  - 7 private networks
CRT Data Collect Network
CRT Switch Network

- 16 SG300/52 switches
- 4 SG300/52MP switches
- 2 Fiber aggregation switches

Green = To Be Installed.

All switches have one fiber based uplink to each aggregation switch. This is not drawn in this diagram due to space.

The SG300/52MP are POE power for the onewire and modbus gateways, split across 2 or 3 switches in each area.

There are 38 onewire and 21 modbus gateways in the network.
Logical Data Flow
What the logical translates into.

- 15 Elasticsearch (ingest) nodes w/SSD and 64GB ram
- 5 Elasticsearch (archive) nodes w/8TB disk and 64GB ram
- 3 dedicated Master nodes (both Consul & Elasticsearch) (vm)
- 2 internal logstash2amqp nodes (ha) (vm)
- 1 Kibana/Elasticsearch worker node (no data/not a master) (vm)
- 4 internal amqp2logstash nodes (vm)
- 3 rabbitmq-servers in ha/mirror cluster mode (vm)
- 6 node redis cluster (vm)
- 8 Rancher/docker nodes
  - 1 VM for Substation network
  - 2 VM for UPS network
  - 2 real nodes (dual socket, 32 cores, 64GB of ram)
  - 3 floating VM's
Freeboard and Metrics 2.0 using rabbitmq/Stomp/Websocket
- **http://ovirt.org**
- **Redhat's RHEV product, opensource version.**
  - We are currently on the 3.6.x version.
oVirt

- KVM/qemu based
- GlusterFS, nfs, iscsi volume management
- Web interface
- Understands VLANs, bridges
- Can talk to Openstack components
- Manages images, templates for you.
- **http://rancher.com**
  - Recently released 1.0.1
    - Oh no.. a x.0 release!
    - Does have some irritating bugs.
    - But it does solve several major problems!
- Supports docker, kubernetes, and Docker Swarm.
- Internal, encrypted network between containers.
- HTML and CLI interfaces.
  - Docker compose file support
- Catalog for applications.
- Running on both VM's and bare metal systems.
- They even have a Docker based OS – RancherOS
  - Like CoreOS/AtomicOS, but geared towards virtual machines.
- Convoy for docker volume management using GlusterFS/NFS.
• One click deployment of application stack.
  – Scales containers, creates internal network, manages volumes
  – Rollback, upgrades are simplified.
  – Schedules containers on the proper nodes.
    • Based on tags.
    • We use this for network routing for example.
Add Onewire Collectors Stack

(Experimental) Onewire server, workers, and jobber

Template Version

0.1

New Stack

Name

donewire

Description

Configuration Options

Worker Nodes:

Number of workers for collection.

Start services after creating

Launch Cancel
### Stacks

<table>
<thead>
<tr>
<th>Stack Name</th>
<th>Status</th>
<th>Image</th>
<th>Services</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>logspout</td>
<td>Active</td>
<td>rancher/logspout:0.2.0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>modbus</td>
<td>Active</td>
<td>docker-registry.cfn.nerc.gov:5000/pbimaster_worker_1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>onewire</td>
<td>Active</td>
<td>docker-registry.cfn.nerc.gov:5000/minik8s/jobber:v0.9</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>rq-dashboard</td>
<td>Active</td>
<td>docker-registry.cfn.nerc.gov:5000/minik8s/rq-dashboard:v0.1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Footer Information**

Production..
Consul for service registration/discovery.

- Service discovery/registration with Consul
  - Uses Nagios checks for health and integration in nagios.
  - DNS routing and load balancing
- Used by onewire stack to route onewire api requests to the proper gateway.
- Nagios also checks for service health.
- Also used to provide round-robin DNS name service for internal services.
- Future uses include configuration templating and key value pairs.