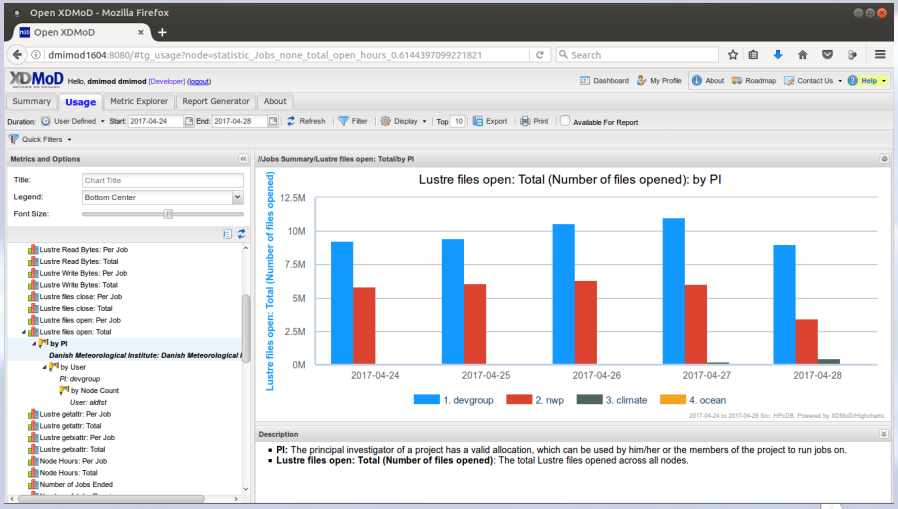
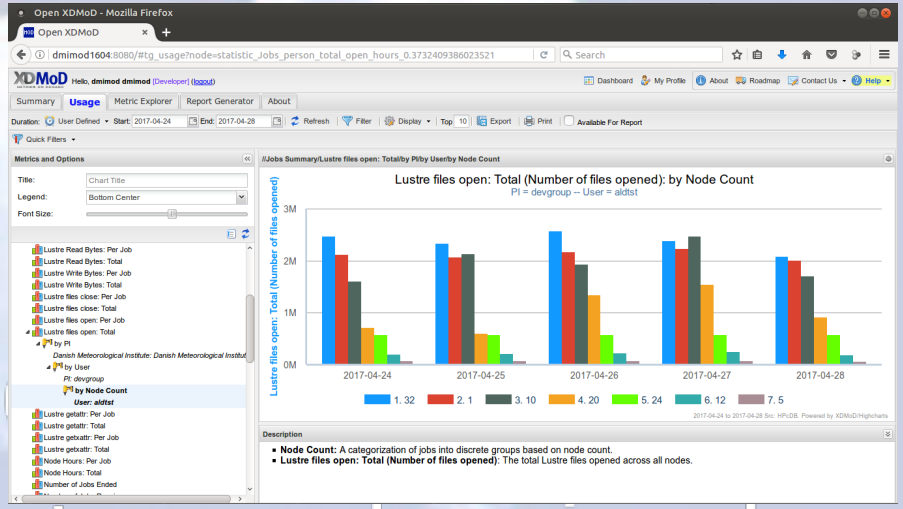
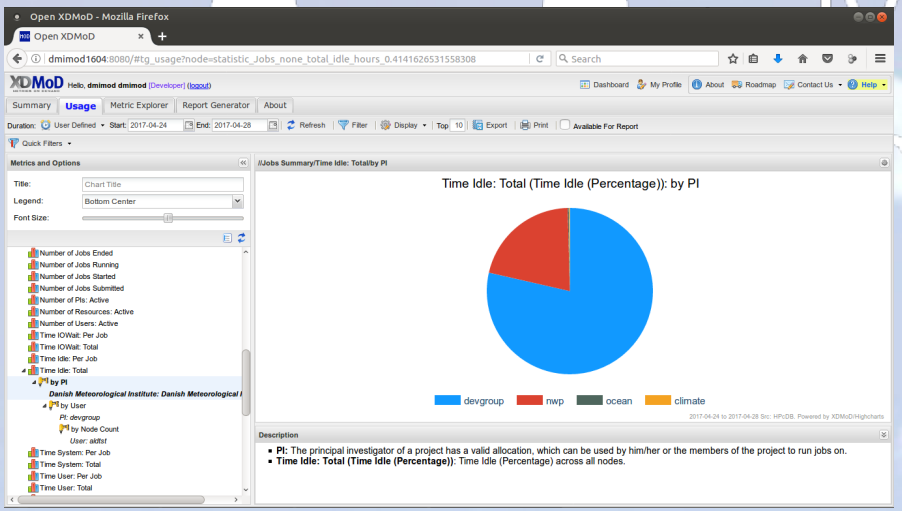
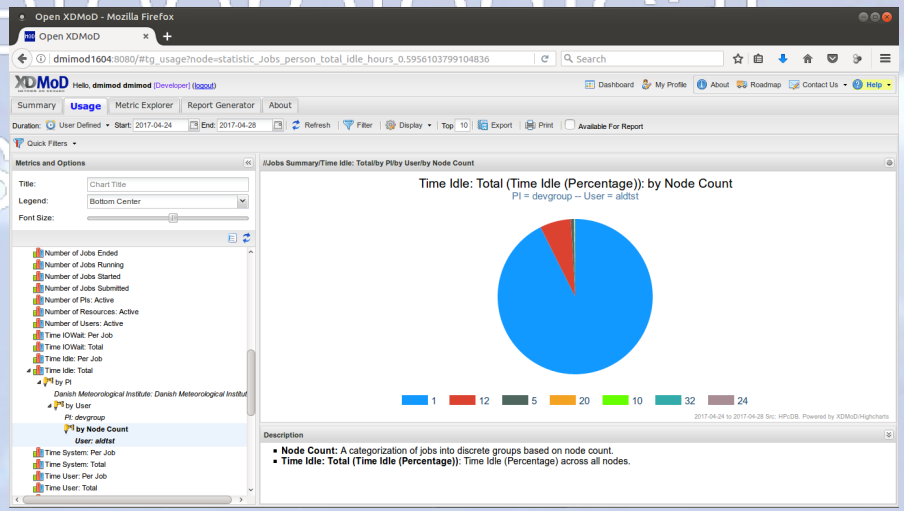
Using Open XDMoD for accounting analytics on the Cray XC supercomputer









Using Open XDMoD for accounting analytics on the Cray XC supercomputer

- Keywords
  - Accounting analytics
  - RUR
  - PBS Pro
  - Open XDMoD
- Presenters
  - Thomas Lorenzen, <tl@dmi.dk>
    Systems Analyst, Danish Meteorological Institute
  - Jason Coverston, <jcovers@cray.com>
    Systems Analyst, Cray

Agenda and intro

- Mission of the site
- The need for an analytics engine
- The need for custom metrics gathering
- Adapting RUR to sample custom metrics
- Getting RUR data into PBS Pro accounting logs
- Adapting Open XDMoD to custom metrics
- First experiences with Open XDMoD
- Future outlook

Mission of the site
The obligations and requirements

- The Danish Meteorological Institute obligates to forecast evolution of weather and ocean for the Kingdom of Denmark.
- The Kingdom of Denmark is not just Denmark but includes Greenland as well.
- Timely forecasts require supercomputing power.
- Resiliency mandate redundancy, either ”2*N” or ”N+1”.
- Two XC systems, one for production and one for research and development, with shared Sonexion and Netapp.
- Hosted at the Icelandic Meteorological Office in a joint partnership.

Mission of the site
The geography

Mission of the site
The site

The need for an analytics engine
The XC needs

- Management requires to know the Return on Investment.
- Systems analyst desire a tool for retrospective usage analysis.
  - Sonexion is a shared ressource, where excessive metadata activity can have adverse effects on all other activities.
  - Production risk being delayed due to metadata bottleneck.
  - The systems comprises both traditional compute nodes and repurposed compute nodes.
  - Resources can easily be wasted if jobs are misplaced.
- Accounting from PBS Pro shall contain all desired metrics.
- The analytics engine shall be extendable to custom metrics.

The need for an analytics engine
The needs external to XC

- The supercomputer is only a part of the full production chain.
- Desire is for the analytics engine to be usable for other batch scheduling systems used outside of the supercomputer, thus being batch system agnostic.

The Danish Meteorological Institute

The need for an analytics engine
The choice

- A tour de internet revealed Open XDMoD (a LAMP web application) as appealing due to the following design objectives.
  - ”… provide management with a tool to monitor utilization, user base, and performance of resources ...”.
  - ”… provide operational staff with the ability to monitor and tune resource performance ...”.
- Further, the following flexibility statements appear very appealing.
  - ”… Open XDMoD … open source tool designed to audit the utilization … by providing a wide range of metrics ...”.
  - ”… Open XDMoD has been created to be adaptable to any HPC environment ...”.

The need for custom metrics gathering

- Out of the box batch accounting only contain limited information.
- Assessing on proper processor utilization and proper formed IO patterns requires more exotic counter samplings.
- These custom metrics shall be gathered on a batch job basis and seamlessly integrated into accounting logs.
- Subsequently shall all accounting data integrate into a graphical analytics engine for visual analysis.
- Here we consider only traditional compute nodes.
- Repurposed compute nodes will be the next step.

Adapting RUR to sample custom metrics

- Counters of interest already exist.
  - /proc/fs/lustre/llite/*/stats – lustre statistics.
  - /proc/stat – processor statistics.
- Define a custom RUR plugin as per RUR documentation.
- Use the energy plugin of RUR as source of inspiration.
- Mostly this is about calculating counter differences.
  - For lustre statistics just do counter differences on selected parameters, here bytes read and written plus number of files opened and closed.
  - For processor statistics calculate the percentage of the time, in which the processor has been in either user, system, iowait or idle state.

Adapting RUR to sample custom metrics

- Sample python code in the rur staging component.

```
with open(''.join(glob.glob("/proc/fs/lustre/llite/*/stats")), 'r') as f:
  for line in f.readlines():
    if re.match('read_bytes|write_bytes',line.strip().split()[0]):
      data[line.strip().split()[0]] = int(line.strip().split()[6])
      continue
    if re.match('open|close|getattr|getxattr$',line.strip().split()[0]):
      data[line.strip().split()[0]] = int(line.strip().split()[1])
      continue
```

Adapting RUR to sample custom metrics

- Sample python code in the rur post component.

```
with open(inputfile, "r") as f:
  for line in f:
    linedata = rur_unwrap_post_data(line)
    input = json.loads(linedata)
    for a in ['read_bytes', 'write_bytes', 'open', 'close']:
      output[a] += input[a]
```

Getting rur data into pbspro accounting logs

- The rur_prologue and rur_epilogue can be called from both alps {pro,epi}logue and pbspro {pro,epi}logue.
- For PBS Pro is the hooks framework the modern way to execute site specific actions.
- The hooks execjob_prologue and execjob_epilogue are the right triggers for rur_prologue and rur_epilogue.
- Furthermore does PBS Pro support the definition of an unlimited number of custom resources.
- These custom resources can be assigned values via hooks.
- These resources and their values will end up in accounting logs.

Getting rur data into pbspro accounting logs

- The way to always start a PBS Pro hook.

```
import pbs
job=pbs.event().job
```

- Sample on how to execute RUR within a hook.

```
If (str(job.Resource_List.arch) == 'XT'):
  rurcmd = '/path/to/rur_{pro,epi}logue.py' + ' -j ' + job.id
  os.system(rurcmd)
  pbs.event().accept()
```

Getting rur data into pbspro accounting logs

- How to define custom resources with qmgr.

```
create resource read_bytes,write_bytes,open,close, type=long
create resource idle,user,system,iowait type=float
```

- Sample on how to assign values to resources within a hook.

```
for dmistats_metric in ["read_bytes", "write_bytes", "open", "close",
"idle", "user", "system", "iowait"]:
  job.resources_used[dmistats_metric] = rur_data[jobid]['dmistats']
[dmistats_metric]
```

Adapting Open XDMoD to custom metrics

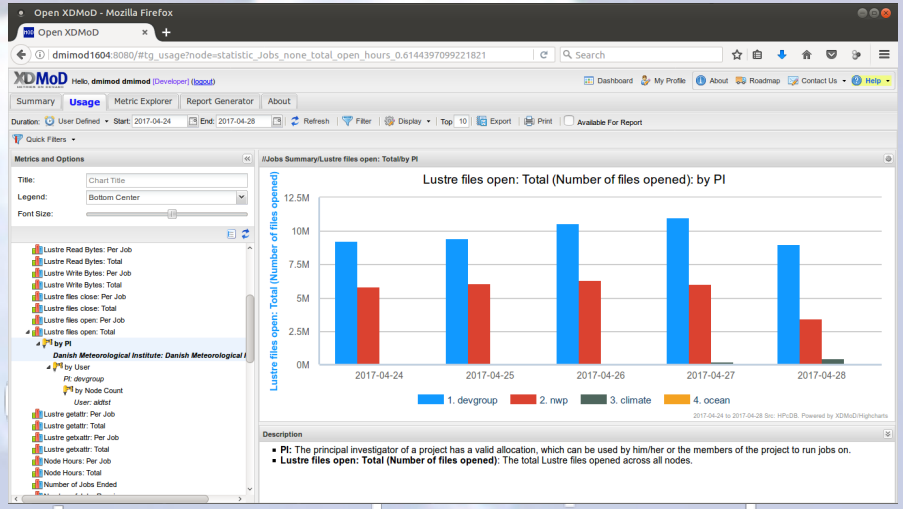
- Open XDMoD is a LAMP web application.
- All newer linux distributions will do.
- The php code needs to be adjusted to cater for custom metrics.
  - Definition of additional mysql table columns.
  - Adjusting how to shred from accounting logs.
  - Adjusting how to ingest the shredded data.
  - Add proper descriptive texts for the custom metrics.
- Requires a little trial and error to get it right.
- Not repeated here but consult the paper or its authors.
- After code adaption is installation most simple.

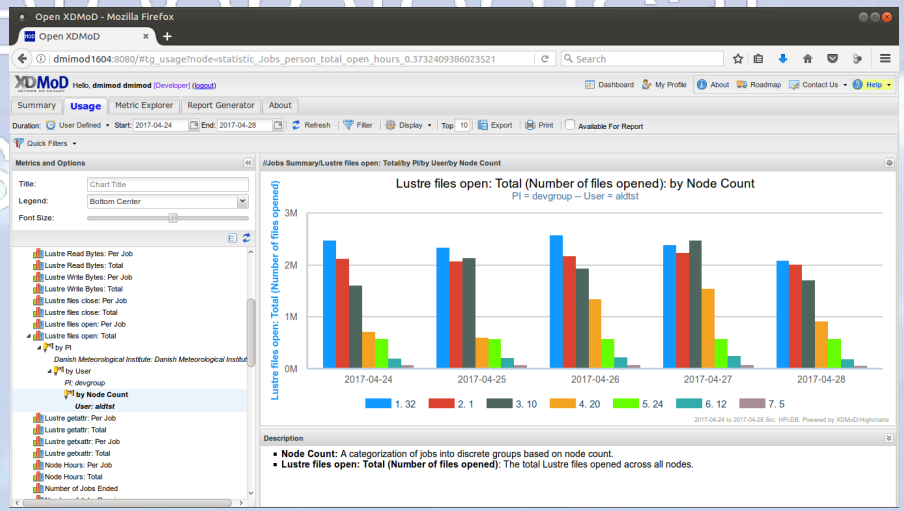
First experiences with Open XDMoD

- Transfer accounting files to Open XDMoD server on a daily basis.
- Shredding.
  `xdmod-shredder -v -r <resource> -f craypbs -d /path/tl/acctfile`
- Ingesting.
  `xdmod-ingestor -v`
- And after a few days will there be enough of data to start getting an overview on.

First experiences with Open XDMoD

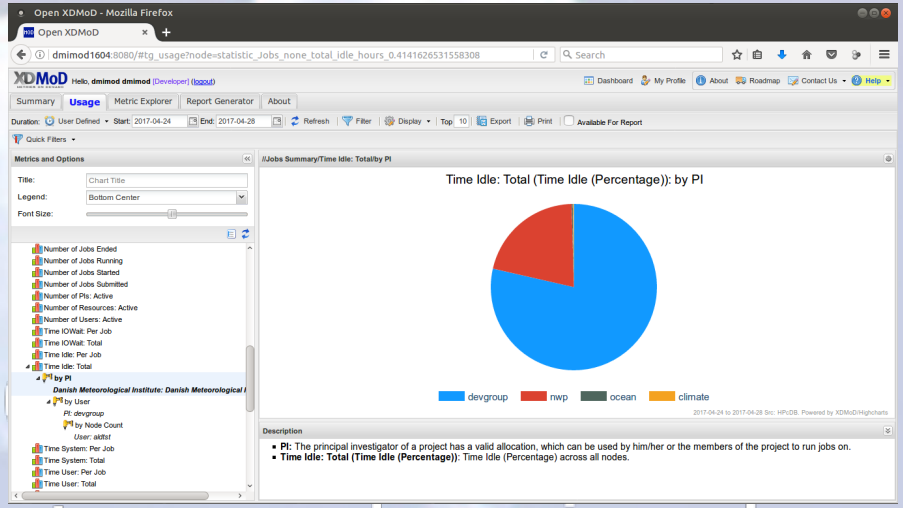
- Start by selecting number of file open operations and then graph per principal investigator.
- Identify which PI does most file open operations and click on a bar for this PI to drill down to user level.
- Not shown here, but a particular user really sticks out, so click on a bar for this user and drille down to node count.
- Top scores for this user are jobs allocating 32 nodes and job only allocating a single core.
- Dialog with user will be easier when he or she can be pinpointed that his or here job allocating 32 nodes are doing suspicious IO.
- The jobs allocating only a single core are suspicious anyway and calls for investigating this further.

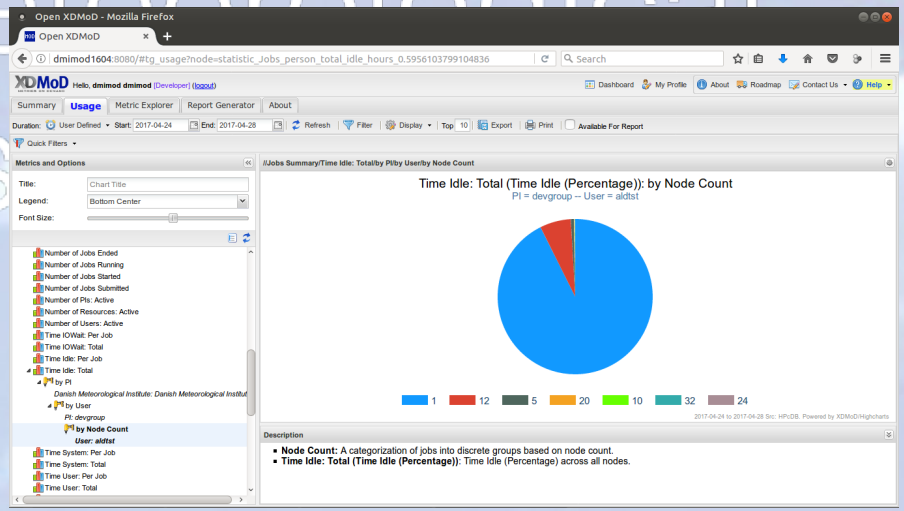




First experiences with Open XDMoD

- Start by selecting percentage of processor being idle and then graph per principal investigator.
- Identify which PI seems to be responsible for most idleness and click on the pie wedge for this PI to drill down to user level.
- Not shown here, but a particular user really sticks out, so click on the pie wedge for this user and drille down to node count.
- Top score for all this idle time comes from single core jobs of this particular user, so this calls for repurposed compute nodes.





Future outlook

- Collecting and analysing custom metrics seem manageable.
- The Open XDMoD interface seems very flexible and intuitive.
- Prototyping drill down to batch job id has been done.
- Having rur metrics on a per alps id basis will give further details.
- Present experiences are only with RUR, which is tightly bound to traditional compute nodes.
- For repurposed compute nodes, which are not job exclusive, is being investigated statistics gathering from cgroups.
- Ingesting data from other workload managers in use should follow similar approach.
- The Open XDMoD tool thus seem very usable and the biggest task seems to be that of finding out, which data to analyse.

Summary and outro

- Mission of the site
- The need for an analytics engine
- The need for custom metrics gathering
- Adapting RUR to sample custom metrics
- Getting RUR data into PBS Pro accounting logs
- Adapting Open XDMoD to custom metrics
- First experiences with Open XDMoD
- Future outlook