GPU Usage Reporting

Cray User Group 2018
Nicholas P. Cardo, CSCS
Miguel Gila, CSCS
Mark Klein, CSCS
May 24, 2018
The Problem

- A batch job is submitted to a compute node containing a GPU
  - Did they utilize the GPU or just the node’s processor?

- Easy to tell if a GPU was requested
  - Can check GRES
  - Can check node name

- Hard to tell if a GPU was used from existing accounting

- How to report GPU usage in a meaningful way
The GPU Hardware @ CSCS

- Piz Daint has 5,319 GPU capable nodes
  - Plus 1,825 nodes without GPUs
- NVIDIA Tesla P100-PCIE-16GB GPUs
- 1 x Intel(R) Xeon(R) E5-2690 v3 @ 2.60 GHz
- 64GB DDR 4

Solution Must be Scalable
nvidia-smi utility

Nid00032: > nvidia-smi -q -d accounting

===============NVSMI LOG===============

Driver Version                      : 384.111
Attached GPUs                       : 1

Accounting Mode                 : Enabled
Accounting Mode Buffer Size     : 1920

Process ID                  : 10757
GPU Utilization         : 0 %
Memory Utilization      : 0 %
Max memory usage        : 291 MiB
Time                    : 272 ms
Is Running              : 0

Process ID                  : 15098
GPU Utilization         : 71 %
Memory Utilization      : 5 %
Max memory usage        : 289 MiB
Time                    : 25194 ms
Is Running              : 0

Process ID                  : 15125
GPU Utilization         : 93 %
Memory Utilization      : 6 %
Max memory usage        : 289 MiB
Time                    : 91777 ms
Is Running              : 0

Process ID                  : 4448
GPU Utilization         : 93 %
Memory Utilization      : 6 %
Max memory usage        : 0 MiB
Time                    : 91899 ms
Is Running              : 0

- Checkout nvidia-smi –h!
- Indications of usage are present
- Different devices provide different info
Objectives

- **Solution Components**
  - Data Capture: ability to capture statistics
    - Extract usage information
  - Data Store: ability to store statistics
    - Quickly store the data for later processing
  - Data Reports: ability to report usage statistics
    - Accounting and Reporting

- **Design Objectives**
  - Determine if a batch job utilized a GPU
    - Start simple and build from there
    - Avoid initial overcomplicating
  - Store statistics with job accounting data
    - Keep data together, less replication
  - Make data available to users
    - They should know
  - Capability for mass reporting
    - Centre level reporting
Challenges

- Limited to counters present in the GPU
  - Device dependent, newer devices may have more reportable counters

- Efficient capture, aggregation, and storage for large jobs
  - One job > 5,000 nodes

- Efficient capture, aggregation, and storage for large quantities of jobs
  - One job on each of 5,000 nodes

- Data Accessability
  - For Users
  - For Centre Reporting
Design – Satisfying Object #1 (Usage)

- Start counting via Slurm prolog
- Stop counting via Slurm epilog

Selected Data
  - GPU seconds accumulated across all nodes
  - Maximum GPU seconds of a node
  - Maximum GPU memory of a node
  - Total GPU memory across all nodes

Difference is usage
Design - Satisfying Object #2 (Data Storage)

- Store data in Slurm job accounting record
  - Keeps all job data together, no separate database or utilities
  - Reuse an existing text field - AdminComment
  - Use JSON format to store multiple pieces of data
Design - Satisfying Object #3 (Data Availability)

- Extractable with `sacct`
  - `sacct -o AdminComment`

- But data is in JSON format
  - LOTS of tools available
  - Selected `./jq` for command-line parsing
    - [https://stedolan.github.io/jq](https://stedolan.github.io/jq)
  - Selected Jansson for compiled library
    - [https://www.digip.org/jansson](https://www.digip.org/jansson)
    - Comes with CLE! `/usr/lib64/libjansson.[a|so]`
Design - Satisfying Object #4 (Reporting Capability)

- Present the data to the users
  - Provide a Summary Report at the end of each batch job’s stdout file
    - Limitation: only works when sbatch is used, acceptable

- Centre Reporting Requirements
  - Since data is in Slurm accounting, it is automatically available for report processing
Implementation

- JavaScript Object Notation (JSON) format

```json
{"gpustats":
  {
    "maxgpusecs": 146,
    "maxmem": 17034117120,
    "gpupids": 1,
    "summem": 17034117120,
    "gpusecs": 146
  }
}
```

- **High Water Marks**
- **GPU Identifier, only 1 installed**
- **Accumulated memory and time**
Adapting RUR

- RUR utilizes a plugin architecture
  - gpustat_stage.py
    - Minor fix to prevent gpusecs from being doubled
  - taskstats_stage.py
    - Minor fix to remove the ALPS requirement
  - file_output.py
    - Build JSON text
    - Write to Slurm accounting record using MySQL
Batch Job Summary Report

Batch Job Summary Report for Job "test1" (6802625) on daint

<table>
<thead>
<tr>
<th>Submit</th>
<th>Eligible</th>
<th>Start</th>
<th>End</th>
<th>Elapsed</th>
<th>Timelimit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-04-12T06:58:40</td>
<td>2018-04-12T06:58:40</td>
<td>2018-04-12T06:58:41</td>
<td>2018-04-12T07:01:19</td>
<td>00:02:38</td>
<td>00:15:00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Username</th>
<th>Account</th>
<th>Partition</th>
<th>NNodes</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>cardo</td>
<td>csstaff</td>
<td>debug</td>
<td>1</td>
<td>18.31K joules</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gpusecs</th>
<th>maxgpusecs</th>
<th>maxmem</th>
<th>summem</th>
</tr>
</thead>
<tbody>
<tr>
<td>146</td>
<td>146</td>
<td>17034117120</td>
<td>17034117120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scratch File System</th>
<th>Files</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>/scratch/snx3000</td>
<td>2</td>
<td>1000000</td>
</tr>
</tbody>
</table>
Early Results

- Customer Feedback

Dear CSCS

Since today I get Batch Job Summary Reports by default at the end of my jobs.

They are very useful. Thanks for enabling the feature. Well done.

Cheers
Next Steps

- Sufficient data now collected, need to define and produce "meaningful" reports

- Analyze the data
  - Understand how to interpret it
  - Identify trends and indicators

- Evaluate if other data elements could be included
  - Solution is very flexible

- Evaluate independence from RUR
  - Become system independent
Grazie per la vostra attenzione!
Thank-You for your attention!