Provision and use of GPU Resources for Distributed Workloads via the Grid

Daniel Traynor, Queen Mary University of London, CHEP2019
Overview

• Motivation

• Integrating GPUs into your cluster. (It’s really easy now)

• Using GPUs - use cases at QM.

• Observations and discussion points.
Motivation

• GPUs are a commodity, programmable parallel architecture, ubiquitous as CPUs but offer significantly more parallel “streams”.

• GPUs are significantly faster than CPU for appropriate problems and GPU optimised workflows often scale better when adding additional GPUs.

• GPU Performance (FLOPS) per watt is better than CPUs.

• GPUs Performance (FLOPS) per $ is better than CPUs.
Deployment and Grid enabling GPUs

- Deployment is really easy now, see backup for details. We use SLURM where jobs get exclusive use of a GPU.

- Previously we enabled GPUs via a CreamCE with requirement that user had to request a GPU in the jdl (GPU=1). CreamCE is being decommissioned.

- arcCE the is simplified by adding in arcce.conf to the subsection for the GPU queue

```
[queue:centos7_gpu]
...

slurm_requirements= -gres=gpu:1 -n4
...
```

- Now all you have to do is submit a job to the centos7_gpu queue and you will get one GPU+4cores+12GB RAM.
Use Cases
IceCube

• Initial GPU deployment was driven by desire to support icecube.

• Why GPUs: Modelling photon propagation through ice. Light propagation is pretty much what is done in video games, so they started using GPUs. GPUs are doing a good job in photon simulation — up to 300 times faster than (single core ?) CPUs (https://sciencenode.org/feature/simulating-icecube-data-using-gpus.php)

• Have had both individual users (PhD students) and official grid production.

• Have yet to repeat for CentOS7.
Cern@school

- Cern@school Significant use of GPUs at QMUL for about a year.


- Using https://github.com/willfurnell/lucid-grid/ (Python 3 (Anaconda) with Tensorflow) for the actual particle detection.

- Using the GPUs significantly sped up the workflow compared to using the CPU, and really is needed for Tensorflow use. Work done by pre university students.

- Deployment of AMBER and GROMACS on a GPGPU testbed of EGI by the MoBrain Competence Centre. Main use is for structural biology.

- Uses docker containers in udocker. udocker is a basic user tool to execute simple docker containers in user space without requiring root privileges.

- The only issue is that we regularly upgrade the kernel and NVIDIA drivers, so e..g DisVis and PowerFit application containers must be re-built with the corresponding NVIDIA driver in order to work in that site.

- NO Significant use of GPUs at QMUL. I think They really wanted GPUs in the cloud.
LHC

- Atlas have active group looking at GPU usage. Other LHC experiments have plans.

- Issue with using QMUL GPUs (CREAM CE need to set GPU=1 in the JDL). Should be simplified with ARC CE.

- Pre GDB on the subject https://indico.cern.ch/event/689511/

- Note even though deployment via containers using singularity we still need drivers and local libraries installed locally to access and use GPUs (CUDA + libcudnn).
Other Users

• Lots of individual HEP researchers using GPUs at home institutes. Some of this will end up needing to scale up on the grid.

• MoEDAL developing new methods using Machine learning (Tensorflow) to ID magnetic monopoles signatures in Nuclear track detectors (https://indico.cern.ch/event/559774/contributions/2669803/attachments/1509702/2354134/MachineLearningMonopolesAndMoedal.pdf).
Observations and Discussion

Points
Anti Motivation

• Costs 5K per server + 6K for high end GPU.

• One high end GPU uses the same power as mid range duel socket server (300W).

• No point in buying GPU to speed up you work x10 if you only use it 1% of the time.

• “The GPU code gets a 200x speed improvement over a single CPU core”. However my server has 64 Cores and costs half as much!

• Utilising the power of GPUs is hard – parallel algorithms – CPU Workflows to GPU workflows – performance pitfalls- even if using frameworks like tenserflow.

• Broad overview of GPU vs CPU impact: https://www.anandtech.com/show/14466/intel-xeon-cascade-lake-vs-nvidia-turing
## Not all GPUs are the same

<table>
<thead>
<tr>
<th></th>
<th>NVIDIA K40</th>
<th>NVIDIA V100</th>
<th>NVIDIA RTX 2080 SUPER</th>
<th>AMD MI60</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RAM</strong></td>
<td>12GB ECC</td>
<td>32GB ECC</td>
<td>8GB</td>
<td>32GB ECC</td>
</tr>
<tr>
<td><strong>Memory bandwidth</strong></td>
<td>288GB/s</td>
<td>900GB/s</td>
<td>496GB/s</td>
<td>1024GB/s</td>
</tr>
<tr>
<td><strong>32bit (TFLOPS)</strong></td>
<td>5</td>
<td>14</td>
<td>11.15</td>
<td>14.7</td>
</tr>
<tr>
<td><strong>64bit (TFLOPS)</strong></td>
<td>1.68</td>
<td>7</td>
<td>0.349</td>
<td>7.4</td>
</tr>
<tr>
<td><strong>16bit (TFLOPS)</strong></td>
<td>N/A</td>
<td>28</td>
<td>22.3</td>
<td>29.5</td>
</tr>
<tr>
<td><strong>8bit (TFLOPS)</strong></td>
<td>N/A</td>
<td>112</td>
<td>89.2</td>
<td>59</td>
</tr>
</tbody>
</table>

Just asking for a “GPU” might not performance increase you expect.

- **16/8 bit support (older GPUs)**
- **64 bit support (consumer grade GPUs)**
- **8 bit performance (v100 vs MI60)**
Informed Choice

• Deep Learning, GPUs not always the right choice!

• Convolutional networks and Transformers: Tensor Cores > FLOPs > Memory Bandwidth > 16-bit capability
  Recurrent networks: Memory Bandwidth > 16-bit capability > Tensor Cores > FLOPs

• Rule of thumb. Here are some prioritisation guidelines for different deep learning architectures (https://timdettmers.com/2019/04/03/which-gpu-for-deep-learning/):
  
  • 64bit (HPC/ simulation)
  
  • Enterprise GPUs (little choice here).

• Notes consumer grade GPUs may not have the build quality to last intensive use.
Alternative Hardware

- Intel CPU chips - AVX512, Deep Learning (DL)Boost (+Vector Neural Network Instruction (VNNI) set), bfloat16. Developing new discrete competitive GPUs.

- AMD GPUs/CPUs - significant effort to develop ecosystem to enable and optimise AMD hardware (Radeon Instinct) in HPC (ROCm Platform) and open source projects (e.g. Tensorflow, Caffe).

- FPGAs (intel, Xilinx), Dedicated ASICs (Google TPUs, Intel Nervana NNP).

- But buying Nvidia is a safe bet but doesn’t help develop competition.
Observations

• Further development of GPU resources will probably need dedicated funds. £/$/€15K for a cheap GPU server (~3 compute servers / >200TiB usable storage).

• New Nvidia EULA of driver software prevents use of non enterprise GPUs (e.g. 1080Ti) in a “data centre”. Can get research exception.

• Often see that CPU usage on our nodes are 100%. Possible limiting factor in making full use of GPU resources.

• We have significant performance difference not just between generations of GPUs (K40,P100,V100) but also for different types of calculations (8/16/32/64bit). Difficult for blind usage via pilot jobs.
Conclusions

• Deployment of GPUs on the grid is not hard. Getting them used is harder.

• Not all GPUs are the same. Different software require different balance of hardware features.

• Not yet clear what the workload/workflow will be. Will impact hardware choices (esp HL-LHC).

• Hardware development and related software support is in flux.

• HEP needs to move from “what can we do” on GPUs to “what should we do”.

• No accounting in APEL for GPUs.

• What do grid sites do? Chicken and egg problem, sources of funding for GPUs.
Obtaining 1st GPU

- Recycled desktop workstation capable of powering a GPU (top end GPU requires ~300W on top of existing CPU… + right connector 2*PCIe 8-pin)

- Obtained Free GPU from NVIDIA GPU Grant Program (e.g. https://developer.nvidia.com/academic_gpu_seeding).

Dell T5500 Workstation + Nvidia K40c GPU
Obtaining Cheap GPUs

• Buy a gaming GPU.

• 1* NVIDIA 1080 Ti founders edition. Dell Alienware Aura 2017

• Brought “off the shelf” Dell Alienware PC. Able to buy via a framework agreement and get delivery in <10days (good for end of financial year).

• Be careful Nvidia has restrictions on use of consumer GPUs in data centres. Research exceptions available.
Obtaining more GPUs

• Brought HPE server + Enterprise GPUs (K80s).

• Funded through money down “the back of the sofa”

• Nvidia+CUDA dominate the market so little point buying others YET.

HPE DL380 + 2* Nvidia K80s (~4* K40)
System Deployment


- Physically install a GPU and test that the kernel see it
  - `lspci | grep -i nvidia`
  - `01:00.0 3D controller: NVIDIA Corporation GK110BGL [Tesla K40c] (rev a1)`

- Install the CUDA repo, install CUDA and reboot (this should install drivers).
  - `yum install cuda gcc kernel-devel-$(uname -r) kernel-headers-$(uname -r)`

- Run "nvidia-smi" to check GPU is available.

- Enable persistence of driver state across CUDA job runs (driver stay loaded).
  - `systemctl [start|enable] nvidia-persistenced`

- Install the CUDA Deep Neural Network library for tensor support
  - `yum install libcudnn7-7.4.1.5-1.cuda10.0.x86_64`

- Compile and run the test program
  - `cp -r /usr/src/cudnn_samples_v7/ $HOME`
  - `cd $HOME/cudnn_samples_v7/mnistCUDNN`
  - `make clean && make`
  - `./mnistCUDNN`

Note: may need to blacklist native Nvidia kernel driver
SLURM Batch System Deployment

- Modify slurm to enable support for Generic resource (GRES), e.g. in /etc/slurm/slurm.conf
  
  - GresTypes=gpu
  
  - ...

  - NodeName=cn291 CPUs=8 Gres=gpu:teslaK40:1 RealMemory=31911 Sockets=1 CoresPerSocket=4 ThreadsPerCore=2 State=UNKNOWN

  - ...

  - PartitionName=centos7_gpu Nodes=cn291 MaxMemPerCPU=20480 DefMemPerCPU=12288 Default=NO MaxTime=99:00:00 State=UP

- In /etc/slurm/gres.conf

  - NodeName=cn291 Name=gpu Type=teslaK40 File=/dev/nvidia0

- cgroups for slurm should be enabled and in /etc/slurm/cgroup.conf (set exclusive use of a GPU for a user)

  - ConstrainDevices=yes

- Submit jobs: sbatch --gres=gpu:1 -n1 test_gpu.sh
Integrate with SGE

- **Create host complex**: `qconf -mc`

  #name               shortcut   type        relop requestable consumable default
  urgency
  
  -----
  ...
  gpu     gpu     INT     <=     YES     YES     0     0
  ...

- **Add complex attribute to host**: `qconf -me cn291`

  hostname          cn291.htc.esc.qmul
  load_scaling      NONE
  complex_values    gpu=1
  user_lists        NONE
  xuser_lists       NONE
  projects          NONE
  xprojects         NONE
  usage_scaling     NONE
  report_variables  NONE

- **Submit job**: `qsub -l gpu=1 testgpu.job`