Information Technology
Infrastructure Branch

WAIC workshop

Presented by: Vadim Malkin

June 25, 2019
Agenda

• Brief Introduction
• Monitoring
• Accessing the System
• Submitting Jobs
• Introduction to Docker
• Use Cases
• Question
Back to the Future: From WEIZAC to WEXAC
Current WEXAC Status

- Computing power: more than 300 servers, above 7000 cores – the biggest Weizmann Institute of Science campus cluster
- Storage capacity: 2 PByte total capacity, about 1 PByte in use
  - 1.6 Pbyte of additional storage for StorWIS, about 0.9 PByte in use
- Modules: more than 600 scientific applications with different versions installed in WEXAC.
- Nvidia NGC repository integration.
- Containerized ready environment
Best practices to compile serial and parallel applications.

- Use Latest Compilers and frameworks
- Use the relevant servers for compilation
- Use a fast interconnect (Infiniband)
- In parallel application cases use a MPI library with good collectiveness (MVAPICH2 or Intel MPI)
- Don’t re-invent the wheel – use optimized libraries, like NetCDF, HDF5, Intel MKL, Nvidia NGC
- Use a last FFT library (FFTW or Intel FFTW wrappers)
- Use productivity tools like DDT, mpiP, OpenMPI, STAT, Nvidia profilers for debugging and application profiling
Ganglia Monitoring and Graphics Tool

• Ganglia monitoring tool is primary built for monitoring clusters of servers, and it does its job at the best.

• It represents the overall performance of a cluster of servers in an overview

• Suppose you have multiple clusters in a datacenter, then in that case you can arrange them and call it as a grid and have an overall performance overview of that grid.

• Performance monitoring to the base machine of a collection of virtual machines, can be done using ganglia monitoring tool.
WEXAC real-time monitor: Ganglia and RTM
Ganglia Grid Overview
Platform RTM

- Can be Used to monitor and manage LSF jobs in Real Time
- [https://www.youtube.com/watch?v=if7R_lld4Wg](https://www.youtube.com/watch?v=if7R_lld4Wg) – learning Video
- [http://rtm.wexac.weizmann.ac.il](http://rtm.wexac.weizmann.ac.il) – WEXAC’s RTM

hpc@weizmann.ac.il
IBM.

RTM is monitoring 190 hosts on 1 cluster.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Queue</th>
<th>Project</th>
<th>ruleSet</th>
<th>4 CPUs</th>
<th>Jobs</th>
<th>4</th>
<th>Run Time (Avg)</th>
<th>Run Time (Max)</th>
<th>Run Time (Min)</th>
<th>Run Time (Median)</th>
<th>Run Time (Throughput)</th>
<th>Run Time (Throughput)</th>
<th>Run Time (Throughput)</th>
<th>Run Time (Throughput)</th>
<th>Job Throughput (Hourly Avg)</th>
<th>Job Throughput (Hourly Std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>4</td>
<td>1617</td>
<td>6444</td>
<td>2.67m</td>
<td>14.18m</td>
<td>36s</td>
<td>54.46s</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>22.437</td>
<td>11.116</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>1</td>
<td>967</td>
<td>975</td>
<td>12.04m</td>
<td>2.80m</td>
<td>3s</td>
<td>18.03m</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4.985</td>
<td>0.995</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>1</td>
<td>863</td>
<td>868</td>
<td>34.64m</td>
<td>3.70m</td>
<td>22s</td>
<td>56.14m</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1.722</td>
<td>0.296</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>1</td>
<td>852</td>
<td>855</td>
<td>11.64m</td>
<td>3.07m</td>
<td>1.57m</td>
<td>8.73m</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>5.198</td>
<td>1.587</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>1</td>
<td>844</td>
<td>844</td>
<td>15.62m</td>
<td>2.09m</td>
<td>1s</td>
<td>26.03m</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>3.841</td>
<td>0.566</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>1</td>
<td>724</td>
<td>741</td>
<td>19.82m</td>
<td>1.99m</td>
<td>1s</td>
<td>22s</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1.219</td>
<td>0.682</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>2</td>
<td>723</td>
<td>1464</td>
<td>27.20m</td>
<td>2.1f</td>
<td>5.12m</td>
<td>17.28m</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>2.199</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>IBM-RTM</td>
<td>new-all-q</td>
<td>default</td>
<td></td>
<td>1</td>
<td>561</td>
<td>560</td>
<td>7.5t</td>
<td>21.95t</td>
<td>3.47s</td>
<td>3.14t</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.077</td>
<td>0.004</td>
<td></td>
</tr>
</tbody>
</table>
RTM Batch Users
GPU monitoring tools

- [http://waicmgmt01.wexac.weizmann.ac.il:30200/](http://waicmgmt01.wexac.weizmann.ac.il:30200/)
- Grafana monitoring tool allow to visualize DGX machines utilization

hpc@weizmann.ac.il
GUI portal for GPU status
MobaXterm for Windows users

• Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more

• Download: http://mobaxterm.mobatek.net/download.html
MobaXterm User Interface

- MobaXterm Professional v8.6
  (X server, SSH client and network tools)

- Your computer drives are accessible through the /drives path
- Your DISPLAY is set to 132.77.134.71:0.0
- When using SSH, your remote DISPLAY is automatically forwarded
- Your HOME folder is not persistent: it will be erased on restart
- Each command status is specified by a special symbol (✓ or ✗)

Registered to Weizmann Institute of Science (6 users)

[2017-02-12 17:48.47] ~
[vadim.vadim-laptop] ➤
MobaXterm – Create SSH session

Secure Shell (SSH) session
MobaXterm – WEXAC prompt

[vadim.vadim-laptop] $ ssh vadimm@access.wexac.weizmann.ac.il

Welcome to access.
This server has been upgraded to RedHat 7.3.
If you have any problems write an email to hpc@weizmann.ac.il.
Enjoy.

Last login: Sun Feb 12 17:48:00 2017 from v-lap.weizmann.ac.il
[vadimm@access ~]$  

Map Network Drive – Windows workstation \data.wexac.weizmann.ac.il
Universal WIS credentials

[Image of Windows Security dialog box with the following details:]
- Enter Network Password
- Enter your password to connect to: data.wexac.weizmann.ac.il
- Username: wismain\vitalyg
- Password: ********
- Domain: wismain
- Remember my credentials

[Options: OK, Cancel]
Access WEXAC from Ubuntu \data.wexac.weizmann.ac.il
Access WEXAC from Ubuntu

Connect to Server

Server Address

smb:// data.wexac.weizmann.ac.il

For example, smb://foo.example.org

Recent Servers

Browse  Cancel  Connect
Universal DIS credentials used to access WEXAC from Ubuntu

Password required for share labs on data.wexac.weizmann.ac.il

Connect As
- Anonymous
- Registered User

Username: vitalyg
Domain: wismain
Password: **********

- Forget password immediately
- Remember password until you logout
- Remember forever

Cancel | Connect
Access WEXAC from MAC
Access WEXAC from MAC using universal DIS credentials

Enter your name and password for the server
data.wexac.weizmann.ac.il

Connect As:
- Guest
- Registered User

Name: wismain\vitalyg
Password: **************

Remember this password in my keychain

Cancel  Connect
WEXAC Modules

```
[vlataly@access -]# module avail
```

```
annocar/2016/eb01
artemis/1.47
aspara/3.5.4
assemblytics/2016Dec11
backcomb/1
blast/2.2.26
caps/9.1
cellrarger/1.2.1
chimer/1.12.2
circles/0.69-3
cuda/7.5
cuda/8.0.44(default)
cytosc/3.4.0
eigen/3.3.0
fastx/0.11.5
fastx_screen/0.9.5
GATK/3.7
```

```
/home/sa/HDF5/Modules/default/modfiles/files/general
/home/sa/HDF5/Modules/default/modfiles/files/intel
/home/sa/HDF5/Modules/default/modfiles/files/gnu
```

```
annocar/2016/eb01
artemis/1.47
aspara/3.5.4
assemblytics/2016Dec11
backcomb/1
blast/2.2.26
caps/9.1
circle/1.2.1
chimer/1.12.2
circles/0.69-3
cuda/7.5
cuda/8.0.44(default)
cytosc/3.4.0
eigen/3.3.0
fastx/0.11.5
fastx_screen/0.9.5
GATK/3.7
```

```
/home/sa/HDF5/Modules/default/modfiles/files/general
/home/sa/HDF5/Modules/default/modfiles/files/intel
/home/sa/HDF5/Modules/default/modfiles/files/gnu
```

```
/home/sa/HDF5/Modules/default/modfiles/files/general
/home/sa/HDF5/Modules/default/modfiles/files/intel
/home/sa/HDF5/Modules/default/modfiles/files/gnu
```
Modules (contd.)

[vitalyg@access ~]$ module --help

Modules Release 3.2.10 2012-12-21 (Copyright GNU GPL v2 1991):

Usage: module [ switches ] [ subcommand ] [subcommand-args ]

Switches:
- H|--help     this usage info
- V|--version  modules version & configuration options
- f|--force    force active dependency resolution
- t|-- terse    terse format avail and list format
- l|--long     list format avail and list format
- h|--human    readable format avail and list format
- v|--verbose  enable verbose messages
- s|--silent   disable verbose messages
- c|--create   create caches for avail and apropos
- i|--icase    case insensitive
- u|--userv[ice,expert,advanced]  set user level to (nov[ice],exp[ert],adv[anced])

Available SubCommands and Args:
  + add|load modulefile [modulefile ...]
  + rm|unload modulefile [modulefile ...]
  + switch|swap [modulefile] modulefile2
  + display|show modulefile [modulefile ...]
  + avail modulefile [modulefile ...]
  + use [-a|--append] dir [dir ...]
  + unuse dir [dir ...]
  + update
  + refresh
  + purge
  + list
  + clear
  + help modulefile [modulefile ...]
  + whatis [modulefile [modulefile ...]
  + apropos[keyword string
  + apropos.add modulefile [modulefile ...]
  + initprepend modulefile [modulefile ...]
  + initrm modulefile [modulefile ...]
  + initswitch modulefile1 modulefile2
  + initlist
  + initclear
Modules (contd.)

[vitalyg@access ~]$ module list
Currently Loaded Modulefiles:
  1) matlab/R2016b  2) mathematica/11.0  3) R/3.3.2  4) python/2.7
[vitalyg@access ~]$ module load bowtie2/2.2.9
[vitalyg@access ~]$ module list
Currently Loaded Modulefiles:
  1) matlab/R2016b  3) R/3.3.2  5) bowtie2/2.2.9
  2) mathematica/11.0  4) python/2.7
[vitalyg@access ~]$ module load samtools/1.3.1
[vitalyg@access ~]$ module list
Currently Loaded Modulefiles:
  1) matlab/R2016b  3) R/3.3.2  5) bowtie2/2.2.9  7) samtools/1.3.1
  2) mathematica/11.0  4) python/2.7  6) icompiler/2017
[vitalyg@access ~]$
WEXAC Job Submission & Control commands

bsub - submits a batch job to LSF
bjobs - displays information about LSF jobs
bhist - displays historical information about jobs
bkill - sends signals to kill, suspend, or resume unfinished jobs
bmod - modifies job submission options of a job
bpeek - displays the stdout and stderr output of an unfinished job
bstop - suspends unfinished jobs
bresume - resumes unfinished jobs
bswitch - switches jobs from one queue to another
### bsub – Commonly Used Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q qname</td>
<td>submits the job to the specified queue</td>
</tr>
<tr>
<td>-o file</td>
<td>redirect stdout, stderr and resource usage information of the job to the specified output file</td>
</tr>
<tr>
<td>-e file</td>
<td>redirect stderr to the specified error file</td>
</tr>
<tr>
<td>-oo/-eo file</td>
<td>same as -o/-e, but overwrite file if it exists</td>
</tr>
<tr>
<td>-i filename</td>
<td>use the specified file as standard input for the job</td>
</tr>
<tr>
<td>-n number</td>
<td>specify number of job slots</td>
</tr>
<tr>
<td>-g jobgroup</td>
<td>submit job to specified group</td>
</tr>
<tr>
<td>-J jobname</td>
<td>assigns the specified name to the job</td>
</tr>
<tr>
<td>-R res_req resource</td>
<td>runs job on a host that meets the specified resource requirements</td>
</tr>
<tr>
<td>-L shell</td>
<td>Initializes the execution environment using the specified login shell</td>
</tr>
</tbody>
</table>
bsub – Methods for Submitting Jobs

• By script or command
  $ bsub -q new-new-all.q -J example -o example-%J.o -e example-%J.e date

• By job spooling
  $ bsub < job_file

  job_file example:
  #BSUB -q new-all.q
  #BSUB -J example
  #BSUB -o example-%J.o
  #BSUB -e example-%J.e
  date
bsub – Methods for Submitting Jobs

Interactively

$ bsub
bsub> #BSUB -q new-all.q
bsub> #BSUB -J example
bsub> #BSUB -o example-%J.o
bsub> #BSUB -e example-%J.e
bsub> date
bsub> ^D
Job <2387> is submitted to queue <new-all.q>.
Resource Requirements (-R)

Resource requirement string is divided into following sections:

- **Selection** - `select[selection_string]`
- **Usage** - `rusage[rusage_string]`
- **Ordering** - `order[order_string]`
- **Locality** - `span[span_string]`
- **Same** - `same[same_string]`
- **CU** - `cu[cu_string]`

Span and same sections are used for parallel jobs
Example 1. Select execution host candidates that have at least 2GB free RAM
$ bsub -R "select[mem>2GB]" myJob

Example 2. Select execution host candidates with Infiniband interconnect
$ bsub -R "defined(ib)" myJob

Example 3. Select candidate hosts that have 1000MB free swap and order by amount of available memory
$ bsub -R "select[swp>1000] order[mem]" myJob
Resource Requirements Examples – Parallel jobs example of usage

**Example 4.** Candidate hosts should have min 500MB free RAM, job will reserve 400MB RAM.

$ bsub -n 4 -R "select[mem>500] rusage[mem=400]" myJob

**Example 5.** All slots required for a parallel job should reside on the same host

$ bsub -n 4 -R "span[hosts=1]" parallelJob

**Example 6.** 12-CPU parallel job should run on up to 4 CPUs per host, all candidate hosts must have the same CPU model

$ bsub -n 12 -R "span[ptile=4] same[model]" parallelJob
Array Jobs – example of usage

• A job array is submitted using the syntax:

```
job_name_spec[index | start_index-end_index:step]
```

```
[vitaly@access mathematica]$ cat array_math.job
#$SUB -q new-short
#$SUB -J mathematica[1-10]
#$SUB -o results.log
#$SUB -e results.log
math -script FullCluster.n $[LSB_JOBINDEX]
[vitaly@access mathematica]$ bjobs
```

```
Job <618114> is submitted to queue <new-short>.
[vitaly@access mathematica]$ bjobs
```

```
<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
</table>
```

[vitaly@access mathematica]$
Array Jobs

[vitalyg@access mathematica]$ cat array_math.job
#BSUB -q new-short
#BSUB -J mathematica[1-10:2]
#BSUB -o results.%J
#BSUB -e results.%J
math -script fullCluster.m ${LSB_JOBINDEX}
[vitalyg@access mathematica]$ bsub < array_math.job
Memory reservation is (MB): 2048
Memory Limit is (MB): 2048
Job <618115> is submitted to queue <new-short>.
[vitalyg@access mathematica]$ bjobs

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
</table>

[vitalyg@access mathematica]$
bjobs – View Jobs

- a  Display information about jobs in all states, including recently finished jobs
- A  Displays summarized information about job arrays
- d  Display information about jobs that finished recently
- l| -w  Display information in long or wide format
- p  Display information about pending jobs
- r  Display information about running jobs
- g  job_group  Display information about jobs in specified group
- J  job_name  Display information about specified job or array
- q  queue  Display information about jobs in specified queue
- u  user  Display information about jobs for specified users/groups
<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>478504</td>
<td>vitalyg</td>
<td>RUN</td>
<td>short</td>
<td>access.wexa</td>
<td>cn165.wexac</td>
<td>example3</td>
<td>Jan 17 10:47</td>
</tr>
<tr>
<td>478502</td>
<td>vitalyg</td>
<td>RUN</td>
<td>new-all.q</td>
<td>access.wexa</td>
<td>cn165.wexac</td>
<td>example1</td>
<td>Jan 17 10:47</td>
</tr>
<tr>
<td>478503</td>
<td>vitalyg</td>
<td>USUSP</td>
<td>new-all.q</td>
<td>access.wexa</td>
<td>cn165.wexac</td>
<td>example2</td>
<td>Jan 17 10:47</td>
</tr>
<tr>
<td>478505</td>
<td>vitalyg</td>
<td>RUN</td>
<td>short</td>
<td>access.wexa</td>
<td>cn160.wexac</td>
<td>example4</td>
<td>Jan 17 10:48</td>
</tr>
<tr>
<td>478506</td>
<td>vitalyg</td>
<td>RUN</td>
<td>short</td>
<td>access.wexa</td>
<td>cn160.wexac</td>
<td>example5</td>
<td>Jan 17 10:48</td>
</tr>
<tr>
<td>478507</td>
<td>vitalyg</td>
<td>RUN</td>
<td>short</td>
<td>access.wexa</td>
<td>cn160.wexac</td>
<td>example6</td>
<td>Jan 17 10:48</td>
</tr>
<tr>
<td>478509</td>
<td>vitalyg</td>
<td>PEND</td>
<td>short</td>
<td>access.wexa</td>
<td></td>
<td>example7</td>
<td>Jan 17 10:50</td>
</tr>
</tbody>
</table>
bjobs – Example 2: detailed job view

[vitalyg@access ~]$ bjobs -l 478509

Job <478509>, Job Name <example7>, User <vitalyg>, Project <default>, Status <PEND>, Queue <short>, Command <sleep 900>
Tue Jan 17 10:50:42: Submitted from host <access.wexac.weizmann.ac.il>, CWD <$HOME>, Output File <dev/null>, Re-runnable, Requested Resources <rusage[mem=2048]>, Specified Hosts <cn149>;

MEMLIMIT
2 G
PENDING REASONS:
There are no suitable hosts for the job;

SCHEDULING PARAMETERS:

loadSched r15s r1m r15m ut pg io ls it tmp swp mem
loadStop

ngpus ngpus_shared ngpus_excl_t ngpus_excl_p ngpus_prohibited
loadSched
loadStop

gpu_temp0 gpu_ecc0 gpu_temp1 gpu_ecc1 gpu_temp2 gpu_ecc2 nmics
loadSched
loadStop

mic_temp0 mic_temp1 mic_freq0 mic_freq1 mic_power0 mic_power1
loadSched
loadStop

mic_freeemem0 mic_freeemem1 mic_util0 mic_util1 mic_ncores0 mic_ncores1
loadSched
loadStop

RESOURCE REQUIREMENT DETAILS:
Combined: select[(type = any) && (type == any)] order[r15s:pg] rusage[mem=2048.00]
Effective: -
bhist – Jobs History options

- Display information about all jobs
- Display information in brief, long, or wide format
- Display information about finished jobs
- Display information about pending jobs
- Display information about suspended jobs
- Display job events chronologically
- Display information about completed, dispatched, submitted, or all jobs in specified time window
- Display information about jobs submitted to specified queue
- Display information about jobs submitted by user or all users
<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>JOB_NAME</th>
<th>PEND</th>
<th>PSUSP</th>
<th>RUN</th>
<th>USUSP</th>
<th>SSUSP</th>
<th>UNKW</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>478502</td>
<td>vitalyg</td>
<td>example1</td>
<td>0</td>
<td>0</td>
<td>539</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>539</td>
</tr>
<tr>
<td>478503</td>
<td>vitalyg</td>
<td>example2</td>
<td>1</td>
<td>0</td>
<td>46</td>
<td>486</td>
<td>0</td>
<td>0</td>
<td>533</td>
</tr>
<tr>
<td>478504</td>
<td>vitalyg</td>
<td>example3</td>
<td>1</td>
<td>0</td>
<td>520</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>521</td>
</tr>
<tr>
<td>478505</td>
<td>vitalyg</td>
<td>example4</td>
<td>0</td>
<td>0</td>
<td>471</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>471</td>
</tr>
<tr>
<td>478506</td>
<td>vitalyg</td>
<td>example5</td>
<td>1</td>
<td>0</td>
<td>467</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>468</td>
</tr>
<tr>
<td>478507</td>
<td>vitalyg</td>
<td>example6</td>
<td>0</td>
<td>0</td>
<td>464</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>464</td>
</tr>
<tr>
<td>478509</td>
<td>vitalyg</td>
<td>example7</td>
<td>342</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>342</td>
</tr>
</tbody>
</table>
bhist – Example 2: detailed job information

[vitalyg@access ~]$ bhist -l 478502

Job <478502>, Job Name <example1>, User <vitalyg>, Project <default>, Command <sleep 999>
Tue Jan 17 10:47:25: Submitted from host <access.wexac.weizmann.ac.il>, to Queue <new-all.q>, CWD <$HOME>, Output File </dev/null>, Re-runnable, Requested Resources <rusage[mem=2048]>

MEMLIMIT
2 G

Tue Jan 17 10:47:25: Dispatched 1 Task(s) on Host(s) <cn165.wexac.weizmann.ac.il>, Allocated 1 Slot(s) on Host(s) <cn165.wexac.weizmann.ac.il>, Effective RES_REQ <select[((type = any ) && (type = any))] order[r15s:pg] rusage[mem=2048.00] >;

Tue Jan 17 10:47:25: Starting (Pid 13646);
Tue Jan 17 10:47:25: Running with execution home </home/labs/training/vitalyg>, Execution CWD </home/labs/training/vitalyg>, Execution Pid <13646>;

Summary of time in seconds spent in various states by Tue Jan 17 10:58:15

<table>
<thead>
<tr>
<th>State</th>
<th>Pend</th>
<th>Psusp</th>
<th>Run</th>
<th>Ususp</th>
<th>Ssus</th>
<th>Uknw</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
</tr>
</tbody>
</table>
Manipulating Jobs

**bkill** - send signals to kill, suspend, or resume unfinished jobs
   Tip: use JobID 0 to kill all your jobs

**bmod** - modify job submission options of a job

**bpeek** - display the stdout and stderr output of an unfinished job

**bstop** - suspend unfinished jobs

**bresume** - resume unfinished jobs

**bswitch** - switch jobs from one queue to another
Manipulating Jobs

[vitalyg@access ~]$ bsub -q new-all.q -oo output -eo error -J example sleep 999
Memory reservation is (MB): 2048
Memory Limit is (MB): 2048
Job <478532> is submitted to queue <new-all.q>.

[vitalyg@access ~]$ bjobs 478532

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>478532</td>
<td>vitalyg</td>
<td>RUN</td>
<td>new-all.q</td>
<td>access.wexa</td>
<td>cn169.wexac</td>
<td>example</td>
<td>Jan 17 12:54</td>
</tr>
</tbody>
</table>

[vitalyg@access ~]$ bstop 478532
Job <478532> is being stopped

[vitalyg@access ~]$ bjobs 478532

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>478532</td>
<td>vitalyg</td>
<td>SUSP</td>
<td>new-all.q</td>
<td>access.wexa</td>
<td>cn169.wexac</td>
<td>example</td>
<td>Jan 17 12:54</td>
</tr>
</tbody>
</table>

[vitalyg@access ~]$ bresume 478532
Job <478532> is being resumed

[vitalyg@access ~]$ bjobs 478532

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>478532</td>
<td>vitalyg</td>
<td>SUSP</td>
<td>new-all.q</td>
<td>access.wexa</td>
<td>cn169.wexac</td>
<td>example</td>
<td>Jan 17 12:54</td>
</tr>
</tbody>
</table>

[vitalyg@access ~]$ bjobs 478532

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>478532</td>
<td>vitalyg</td>
<td>RUN</td>
<td>new-all.q</td>
<td>access.wexa</td>
<td>cn169.wexac</td>
<td>example</td>
<td>Jan 17 12:54</td>
</tr>
</tbody>
</table>
Manipulating Jobs

Example 2

```
[vitalyg@access ~]$ bjobs
JOBID USER STAT QUEUE FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
478529 vitalyg RUN new-all.q access.wexa cn146.wexac example Jan 17 12:41
478530 vitalyg PEND new-all.q access.wexa example Jan 17 12:43
```

Parameters of job <478530> are being changed

```
[vitalyg@access ~]$ bmod -q new-short 478530
[vitalyg@access ~]$ bjobs
JOBID USER STAT QUEUE FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
478530 vitalyg RUN new-short access.wexa cn146.wexac example Jan 17 12:43
478529 vitalyg RUN new-all.q access.wexa cn146.wexac example Jan 17 12:41
```

Information Systems
Cluster Query Commands

lsinfo  - displays load sharing configuration information
lshosts - displays hosts and their static resource information
lsload  - displays load information for hosts
bhosts  - displays hosts and their static and dynamic resources
bqueues - displays information about queues
bmgroup - displays information about host groups and compute units
busers  - displays information about users and user groups
bugroup - displays information about user groups
Query Commands

Example 1. Show resource information for compute nodes cn141 and cn142
[vitalyg@access ~]$ lshosts cn141 cn142
HOST_NAME type model cpuf ncpus maxmem maxswp server RESOURCES
.cn141.wexac X86_64 Intel_EM 60.0 16 63.8G 23.4G Yes (mpichp4 intelmpi openmpi)
.cn142.wexac X86_64 Intel_EM 60.0 16 63.8G 23.4G Yes (mpichp4 intelmpi openmpi)

Example 2. Show load information for compute nodes cn141 and cn142
[vitalyg@access ~]$ lsload cn141 cn142
HOST_NAME status r15s r1m r15m ut pg ls it tmp swp mem
.cn142.wexac.wei ok 1.0 1.0 1.0 3% 0.0 0 3834 18G 23.4G 54.3G
.cn141.wexac.wei ok 2.0 20.0 14.0 9% 0.0 0 3862 18G 23.4G 58.1G

Example 3. Show statistics for hosts cn141 and cn142
[vitalyg@access ~]$ bhosts cn141 cn142
HOST_NAME STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV
.cn141.wexac.weizma ok - 32 20 20 0 0 0 0
.cn142.wexac.weizma ok - 32 10 10 0 0 0 0
Example 4. Show queues

```
[vitaly@access ~]$ bqueues
QUEUE_NAME PRIO STATUS MAX J1/U J1/P J1/M NJOBS PEND RUN SUSP
bio-nem 85 Open:Active - - - 0 0 0 0 0
sorek-high-prio 80 Open:Active - 6 - - 0 0 0 0
bio 80 Open:Active - - - - 0 0 0 0
schwartz 80 Open:Active - - - 0 0 0 0 0
feul 80 Open:Active - - - 1 1 0 0
bio-bart 75 Open:Active - - - - 0 0 0 0
bio-guest 70 Open:Active - - - - 0 0 0 0
wicc-priority 65 Open:Active - - - - 0 0 0 0
fleishman-prior 60 Open:Active - 40 - - 0 0 0 0
hame-priority 60 Open:Active - 6 - - 0 0 0 0
koren-priority 60 Open:Active - - - - 0 0 0 0
lancet-priority 60 Open:Active - 6 - - 0 0 0 0
schneidman-prio 60 Open:Active - 60 - - 0 0 0 0
all-high 60 Open:Active - 1000 - - 0 0 0 0
new-short 0 Open:Active - 1024 - - 0 0 0 0
```

Example 5. Display information about all hosts

```
[vitaly@access ~]$ bhosts
HOST_NAME STATUS J1/U MAX NJOBS RUN SUSP USUSP RSV
cn888.wexac.weizm ok - - 24 20 20 0 0 0
cn981.wexac.weizm ok - - 24 20 20 0 0 0
cn983.wexac.weizm unavail - - 32 0 0 0 0 0
cn884.wexac.weizm ok - - 32 20 20 0 0 0
cn101.wexac.weizm ok - - 24 20 20 0 0 0
cn104.wexac.weizm ok - - 24 20 20 0 0 0
cn105.wexac.weizm ok - - 24 20 20 0 0 0
cn106.wexac.weizm ok - - 24 20 20 0 0 0
cn107.wexac.weizm ok - - 24 20 20 0 0 0
cn108.wexac.weizm closed - - 24 20 20 0 0 0
```
What Is A Docker Container?

• A Docker container is a mechanism for bundling a Linux application with all of its libraries, data files, and environment variables so that the execution environment is always the same, on whatever Linux system it runs and between instances on the same host.

• Unlike a VM which has its own isolated kernel, containers use the host system kernel. Therefore, all kernel calls from the container are handled by the host system kernel. DGX systems uses Docker containers as the mechanism for deploying deep learning frameworks.
Why Use A Container?

- One of the many benefits to using containers is that you can install your application, dependencies and environment variables one time into the container image; rather than on each system you run on. In addition, the key benefits to using containers also include:
  - Install your application, dependencies and environment variables one time into the container image; rather than on each system you run on.
  - There is no risk of conflict with libraries that are installed by others.
  - Containers allow use of multiple different deep learning frameworks, which may have conflicting software dependencies, on the same server.
  - After you build your application into a container, you can run it on lots of other places, especially servers, without having to install any software.
  - Legacy accelerated compute applications can be containerized and deployed on newer systems, on premise, or in the cloud.
  - Specific GPU resources can be allocated to a container for isolation and better performance.
  - You can easily share, collaborate, and test applications across different environments.
  - Multiple instances of a given deep learning framework can be run concurrently with each having one or more specific GPUs assigned.
  - Containers can be used to resolve network-port conflicts between applications by mapping container-ports to specific externally-visible ports when launching the container.
Virtual Machines vs. Containers

Virtual Machines

- App 1
- App 2
- App 3
- Bins/Lib
- Guest OS
- Hypervisor
- Infrastructure

Containers

- App 1
- App 2
- App 3
- Bins/Lib
- Guest OS
- Container Engine
- Operating System
- Infrastructure

Information Systems
Docker Lifecycle

• DGXWS01 is used to build custom Docker images
• An image can be pulled and manipulated as required or built from a Dockerfile created by a user
• The image can be manipulated and then pushed to a private repository
Docker Cheat Sheet

**ORCHESTRATE**

- Initialize swarm mode and listen on a specific interface:
  docker swarm init --advertise-addr 20.1.0.2

- Join an existing swarm as a manager node:
  docker swarm join --token <manager-token> 20.1.0.2 22377

- Join an existing swarm as a worker node:
  docker swarm join --token <worker-token> 20.1.0.2 22377

- List the nodes participating in a swarm:
  docker node ls

- Create a service from an image exposed on a specific port and deploy 3 instances:
  docker service create --replicas 3 --name web app

- List the services running in a swarm:
  docker service ls

- Scale a service:
  docker service scale web=5

- List the tasks of a service:
  docker service ps web

**RUN**

- Stop a running container through SIGTERM:
  docker stop web

- Stop a running container through SIGKILL:
  docker kill web

- Create an overlay network and specify a subnet:
  docker network create --subnet 16.1.0.0/24 --gateway 16.1.0.1 --d overlay mynet

- List the networks:
  docker network ls

- List the running containers:
  docker ps

- Delete all running and stopped containers:
  docker rm --force (docker ps -aq)

- Create a new bash process inside the container and connect it to the terminal:
  docker exec -it web bash

- Print the last 100 lines of a container's logs:
  docker logs --tail 100

**BUILD**

- Build an image from the Dockerfile in the current directory and tag the image:
  docker build -t myapp:3.0

- List all images that are locally stored with the Docker engine:
  docker images

- Delete an image from the local image store:
  docker rmi alpine:3.4

**SHIP**

- Pull an image from a registry:
  docker pull alpine:3.4

- Retag a local image with a new image name and tag:
  docker tag alpine:3.4 myregistry/myalpine:3.4

- Log into a registry (the Docker Hub by default):
  docker login my.registry.com:8080

- Push an image to a registry:
  docker push myregistry/myalpine:3.4
How to pull a prebuilt docker image from a repository

- [https://ngc.nvidia.com/](https://ngc.nvidia.com/)
- Pull the Docker image from NVIDIA NGC repository:

```bash
docker pull nvcr.io/nvidia/tensorflow:19.05-py3
```

07/11/2020 23:27:40  15.8MB  18.5MB  [Etchys] [d5d1]  docker pull nvcr.io/nvidia/tensorflow:19.05-py3
```bash
root@d5d1:~ # docker pull nvcr.io/nvidia/tensorflow:19.05-py3
7e6591305420: Already exists
8090d5b84ea8: Already exists
3c461606e089: Already exists
95b085e43511a: Already exists
6ca460804a8f: Already exists
2e31f8e6e0f4: Already exists
6b756ee3ae971: Already exists
2a464620168: Already exists
7f2717c07ed0: Already exists
69eaeb99332: Already exists
bc90ca27b13c: Already exists
1878768be77f: Already exists
8030f58947b: Already exists
717c5c6f979c: Already exists
1835f16138c7: Already exists
55f759f3ad8: Already exists
82c2576870f8: Already exists
5524d5f6001a: Already exists
f3f14c44df0: Already exists
5052b6e5ac0c: Already exists
4c425ec37810: Already exists
4e3b68d646c2: Already exists
4e218705b47c: Already exists
9d526d4f4c3: Already exists
2e5e3be48901: Already exists
7f7f65c52bd: Downloading [=================] 26.9MB/76.22MB
a2901b85d20: Download complete
47f11a9e74f8: Downloading [=================] 1.182MB/5.352MB
7b0b3f5a380: Downloading [==] 1.617MB/52.75MB
1f77f7f477: Waiting
```
Build an image from a dockerfile

A dockerfile is a script that contains instructions to customize and configure a container from a base image.

Here are some common commands:

- **FROM** is Mandatory as the first instruction. It denotes the base image to be built from. Use a tag to specify the image.
- **RUN** = runs commands after Docker image has been created.
- **CMD** = Run any command when Docker image is started. Use only one CMD instruction in a Dockerfile.
- **ENV** = Set container environment variables.
- **WORKDIR** = Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions.

Instructions can be found here: [https://docs.nvidia.com/ngc/ngc-user-guide/custcontfrm.html#custcontfrm](https://docs.nvidia.com/ngc/ngc-user-guide/custcontfrm.html#custcontfrm)

Example using horovod and creating an image from a dockerfile (file is located in slide notes):

```bash
mkdir horovod-docker
cp /shareDB/wexac_workshop/Dockerfiles/Horovod/Dockerfile horovod-docker/Dockerfile
docker build -t horovod:latest horovod-docker
```
How to modify a docker image

There are two main ways to edit a docker image.

1. The best practice is to modify a dockerfile to include any changes you would require. This approach is preferred as one can easily document changes and better maintain the image.

2. Alternatively, one can use `docker commit` to create a new image from a currently open container.
How to push an image to the local repository

• Tag the docker image:
  `docker tag pytorch:19.05-py3 ibdgx001:5000/tensorflow-test`

• Push the tagged docker image to the local repository:
  `docker push ibdgx001:5000/tensorflow-test`

• Check:
  Docker images

• You should see both the original image as well as the tagged image
How to run an interactive job

Run an interactive job by using the -Is option:
```
bsub -q waic-interactive -gpu num=1:j_exclusive=yes -Is /bin/bash
```

Opening an IDE:
```
bsub -q waic-interactive -gpu num=1:j_exclusive=yes -Is /bin/bash
source /etc/profile
module load anaconda
export DISPLAY=<your local computer ip here>:0
spyder
```
How to run a job using a pulled image

Create a file to run the job:

```bash
cat useCase1
#BSUB -J useCase1
#BSUB -env LSB_CONTAINER_IMAGE=ibdgx001:5000/tensorflow-test
#BSUB -app nvidia-gpu
#BSUB -gpu num=2:j_exclusive=yes
#BSUB -q waic-short
#BSUB -oo test_useCase_out.log
#BSUB -eo test_useCase_err.log
mpiexec -np 2 python /workspace/nvidia-examples/cnn/resnet.py --layers=50 --precision=fp16 --log_dir=output/resnet50_useCase1
```
How to run a job using a pulled image

Run the job:

```
cat useCase1 | bsub
```

You can check the job using `bjobs/bpeek`:

```
bjobs -q waic-short -J useCase1
bpeek -f -J test_useCase
```
Submitting an array job

Array Script:

```
./seq_arr.sh -f c.sh -e 10
```

The `-e` must match the elements in the array. In this case (c.sh) there are ten, “#BSUB -J test[1-10]”.

bsub file(c.sh):

```
#BSUB -q new-short
#BSUB -J test[1-10]
#BSUB -o out/out.%J
#BSUB -e err/err.%J
#BSUB -H
sleep 5
```

The `-H` option is very important! (Holds the job in the PSUSP state when the job is submitted).
cat << EOF_useCase
#BSUB -J useCase1
#BSUB -env LSB_CONTAINER_IMAGE=nvcr.io/nvidia/tensorflow:19.03-py3
#BSUB -app nvidia-gpu
#BSUB -gpu num=2:j_exclusive=yes
#BSUB -q waic-short
#BSUB -oo useCase1_out.log
#BSUB -eo useCase1_err.log
mpiexec -np 2 python \\n/workspace/nvidia-examples/cnn/resnet.py \\--layers=50 \\--precision=fp16 \\--log_dir=output/resnet50_useCase1
EOF_useCase
# UseCase1 + Grafana

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Commands</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit a job</td>
<td>bsub &lt; ~/useCase1</td>
<td>Job is submitted to queue waic-short</td>
</tr>
<tr>
<td>Verify the job is enqueued and starts running</td>
<td>bjobs -q waic-short -J useCase1</td>
<td>Job useCase1 appears in the output, the job state is RUN</td>
</tr>
<tr>
<td>Verify the job is running correctly</td>
<td>bpeek -f -J useCase1</td>
<td>Job progresses as expected</td>
</tr>
<tr>
<td>Verify the job executed correctly</td>
<td>Inspect file useCase1_out.log</td>
<td>File contains lines:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Step Epoch Img/sec Loss LR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1.0 156.8 7.726 8.697 2.00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 10.0 1289.8 5.758 6.729 1.62000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 20.0 4417.6 2.035 3.010 1.24469</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30 30.0 6297.6 0.010 0.991 0.91877</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40 40.0 6303.1 0.000 0.976 0.64222</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50 50.0 6339.3 0.000 0.968 0.41506</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Average Img/Sec value for steps 30+ should be above 6200</strong></td>
</tr>
<tr>
<td>Navigate to Grafana UI</td>
<td>Navigate to <a href="http://waicmgmt01.wexac.wi">http://waicmgmt01.wexac.wi</a></td>
<td>Successful navigation to Grafana</td>
</tr>
<tr>
<td></td>
<td>izmann.ac.il:30200</td>
<td>Home Dashboard</td>
</tr>
<tr>
<td>Navigate to GPU Nodes dashboard and select the</td>
<td>Per-GPU utilization and Total GPU utilization is</td>
<td></td>
</tr>
<tr>
<td>node running. Check node utilization</td>
<td>above 90%</td>
<td></td>
</tr>
</tbody>
</table>

![Grafana Dashboard Screenshot](image)
cat << EOF_useCase2 > ~/useCase2
#BSUB -J useCase2
#BSUB -env "LSB_CONTAINER_IMAGE=ibdgx001:5000/usecase2_1"
#BSUB -app nvidia-gpu
#BSUB -gpu num=1:j_exclusive=yes
#BSUB -q gpu-short
#BSUB -o useCase2_out.log
#BSUB -e useCase2_err.log
export MLM_LICENSE_FILE=1700@license.weizmann.ac.il
cd /shareDB/Wexac_workshop/useCase2_code/usecase2/Matlab/exp_code
echo; echo; echo "Running MATLAB"
echo "Test 2.1.1."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('0', 0, 1); exit"
echo; echo; echo "Test 2.1.2."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('0', 16, 1); exit"
echo; echo; echo "Test 2.1.3."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('16', 0, 1); exit"
echo; echo; echo "Test 2.1.4."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('16', 16, 1); exit"
echo; echo; echo "Test 2.1.5."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('64_32_16_8_4_0', 0, 1); exit"
echo; echo; echo "Running PYTHON CONVERT"
echo "Test 2.2.1."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('64_32_16_8_4_0', 16, 1); exit"
echo; echo; echo "Test 2.2.2."
/usr/local/bin/matlab -nodisplay -nosplash -nodesktop -r
"eval_validation_set_exact('64_32_16_8_4_0', 0, 1); exit"
echo; echo; echo "Test 2.3.1."
python3 imagenet_resnet50_readout.py --train 'hi' --test-blur 0 --cond 'eyes' --layer 3 --input-type 'face_wo_chinrest' --gpu 1 --batch-size 10
echo "Test 2.3.2."
python3 imagenet_resnet50_readout.py --train 'hi' --test-blur 0 --cond 'head' --layer 3 --input-type 'face_wo_eyes' --gpu 1 --batch-size 10
echo "Test 2.3.3."
python3 imagenet_resnet50_readout.py --train 'low' --test-blur 0 --cond 'eyes' --layer 3 --input-type 'face_wo_eyes' --gpu 1 --batch-size 10
echo "Test 2.3.4."
python3 imagenet_resnet50_readout.py --train 'low' --test-blur 16 --cond 'head' --layer 3 --input-type 'eyes' --gpu 1 --batch-size 10
EOF_useCase2
## UseCase2 - Matlab and PyTorch under a common environment

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Commands</th>
<th>Expected results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify custom image exists</td>
<td><code>docker image ls testing:usecase2</code></td>
<td>Image testing:usecase2 exists</td>
</tr>
<tr>
<td>Tag image for upload to repository</td>
<td><code>docker tag testing:usecase2 ibdgx001:5000/usecase2_1</code></td>
<td>Successful execution</td>
</tr>
<tr>
<td>Upload image to cloud repository</td>
<td><code>docker push ibdgx001:5000/usecase2_1</code></td>
<td>All layers are pushed successfully</td>
</tr>
</tbody>
</table>
# UseCase2 - Matlab and PyTorch under a common environment

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Commands</th>
<th>Expected results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit a job</td>
<td>bsub &lt; ~/useCase2</td>
<td>Job is submitted to queue waic-short</td>
</tr>
<tr>
<td>Verify the job is enqueued and starts running</td>
<td>bjobs -q waic-short -J useCase2</td>
<td>Job useCase2 appears in the output, the job state is RUN</td>
</tr>
<tr>
<td>Verify the job is running correctly</td>
<td>bpeek -f -J useCase2</td>
<td>Job progresses as expected</td>
</tr>
<tr>
<td>Verify Matlab part of the job executed correctly</td>
<td>Inspect useCase2_out.log file</td>
<td>Matlab runs with outputs similar to the following:</td>
</tr>
<tr>
<td>Verify &quot;python convert&quot; part of the job executed successfully</td>
<td>Inspect useCase2_out.log file</td>
<td>Log contains following output:</td>
</tr>
<tr>
<td>Verify &quot;python imagenet resnet50&quot; part of the job executed successfully</td>
<td></td>
<td>Running PYTHON CONVERT</td>
</tr>
</tbody>
</table>

*Test 2.2.1: convert_columbiagaze_imdb_mat_to_pkl SUCCESSFUL!*
UseCase3 - Intensive I/O – video dataset

cat << EOF_useCase3 > ~/useCase3
#BSUB -J useCase3
#BSUB -env LSB_CONTAINER_IMAGE=nvcr.io/weizmann1/waic:usecase3
#BSUB -app nvidia-gpu
#BSUB -gpu num=2:j_exclusive=yes
#BSUB -q gpu-short
#BSUB -oo useCase3_out.log
#BSUB -eo useCase3_err.log
export
PATH=/opt/conda/bin:/usr/local/mpi/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:${PATH}
cd useCase3_code
/opt/conda/bin/python train.py --t_win 31 --epochs 2 --tag useCase3 -b 8
EOF_useCase3
## UseCase3 - Intensive I/O – video dataset

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Commands</th>
<th>Expected results</th>
</tr>
</thead>
<tbody>
<tr>
<td>First copy code to home directory:</td>
<td><code>cp -R /shareDB/wexam_workshop/useCase3_code/ ~/</code></td>
<td></td>
</tr>
<tr>
<td>Submit a job</td>
<td><code>bsub &lt; ~/useCase3</code></td>
<td>Job is submitted to queue waic-short</td>
</tr>
<tr>
<td>Verify the job is enqueued and starts running</td>
<td><code>bjobs -q waic-short -J useCase3</code></td>
<td>Job <code>useCase3</code> appears in the output, the job state is RUN</td>
</tr>
<tr>
<td>Verify the job is running correctly</td>
<td><code>bpeek -f -J useCase3</code></td>
<td>Job progresses as expected. After running for about 5 minutes <code>useCase3_err.log</code> file's last line looks like: train 0: 1%</td>
</tr>
<tr>
<td>Verify the job executed correctly</td>
<td>Wait till the job finishes running (should take about 6-7 hours). Inspect <code>useCase3_*</code>.log files</td>
<td><code>useCase3_err.log</code> file doesn't contain any significant errors. Any errors related to <code>qsub</code> can be ignored. <code>useCase2_out.log</code> file contains expected job execution logs</td>
</tr>
</tbody>
</table>
Troubleshooting for usecases

For the use cases we need to copy the code to our home directory (run from your specific home directory):

```
cp -R /shareDB/wexac_workshop/useCase3_code/ ~/
```

If output/ exists after running useCase1 remove it and try running useCase1 again

```
rm -R output/
```
WEXAC Web Resources

http://www.weizmann.ac.il/hpc
https://insightiq.weizmann.ac.il
http://master-ops.wexac.weizmann.ac.il/ganglia/
http://lsfutil.wexac.weizmann.ac.il
https://www.facebook.com/HpcAtWeizmann/
http://wiki.weizmann.ac.il/ai_wiki/index.php/WAIC_cluster

Join the WhatsApp group!
https://chat.whatsapp.com/05qVVCCcR8v9234vit28gc
Questions?

hpc@weizmann.ac.il
Thank you!

hpc@weizmann.ac.il