KFUPM HPC Workshop

April 20-30 2015

Mohamed Mekias

HPC Solutions Consultant

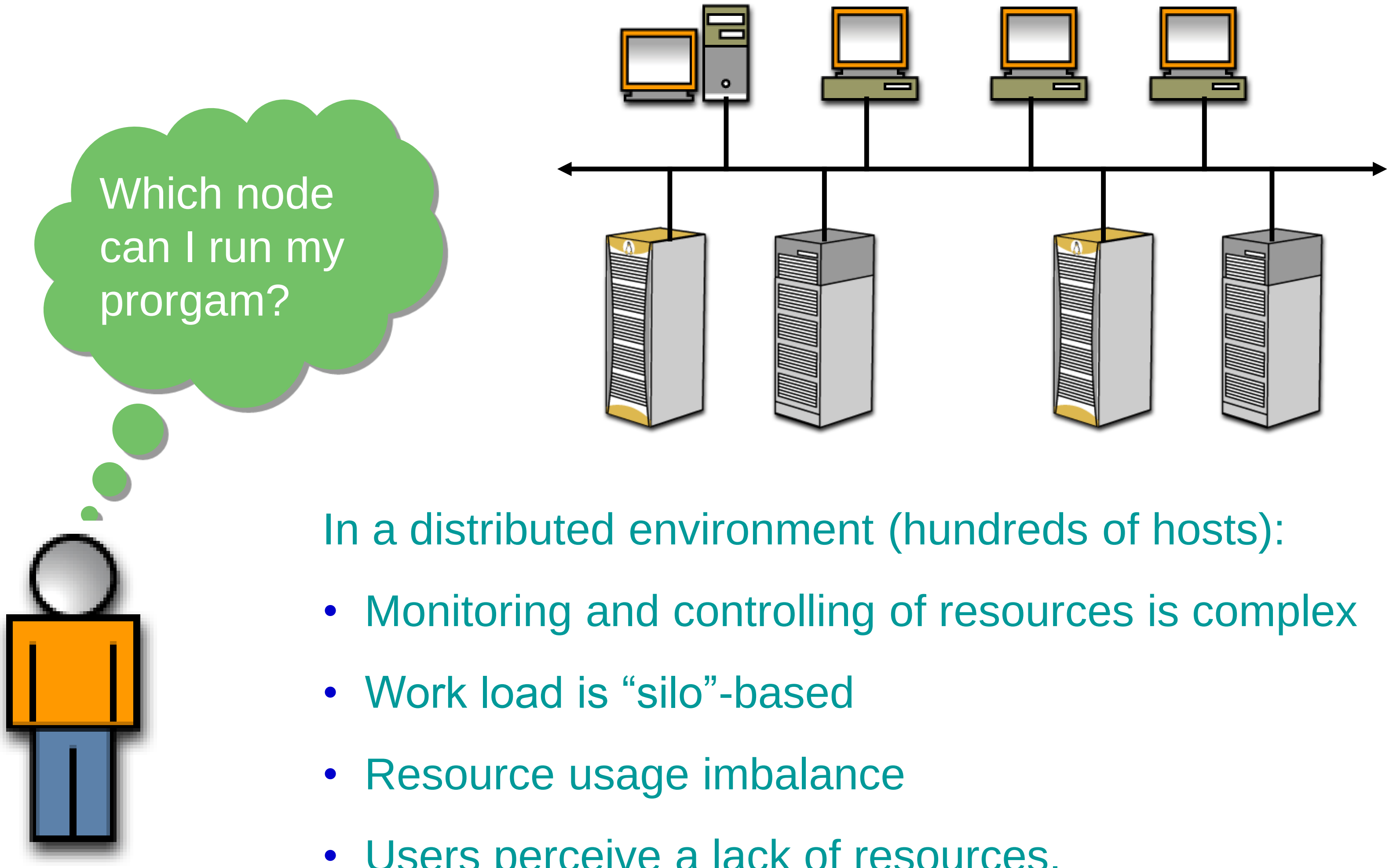
A Quick Tour of IBM Platform LSF

Quick introduction to LSF for end users

- IBM Platform LSF (load sharing facility) is a suite of distributed resource management products that:
  - Connects computers into a cluster (or grid)
  - Monitors loads of systems
  - Distributes, schedules, and balances workload
  - Controls access and load by policies
  - Analyzes the workload

- Provides transparent access to all available resources

Without Platform LSF

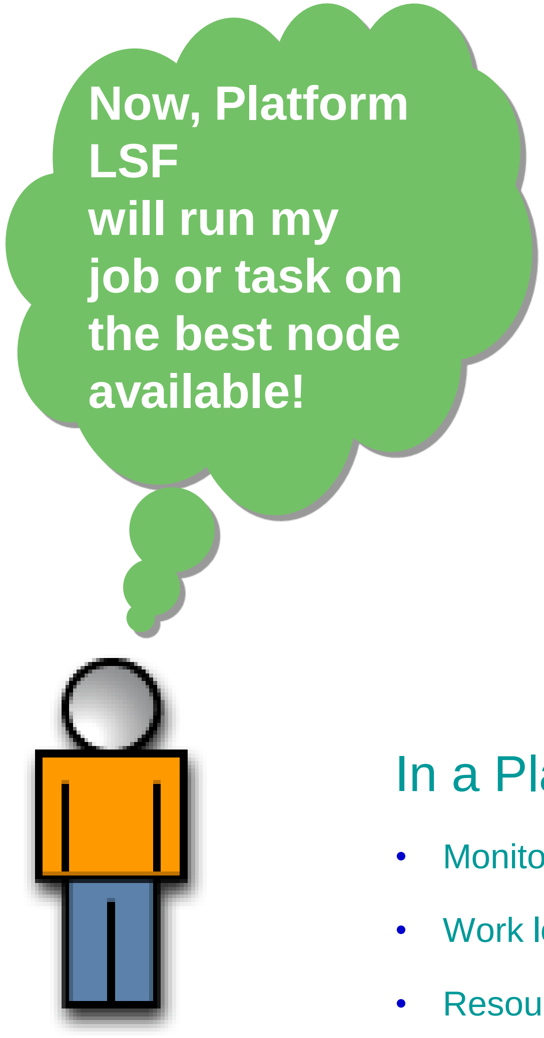


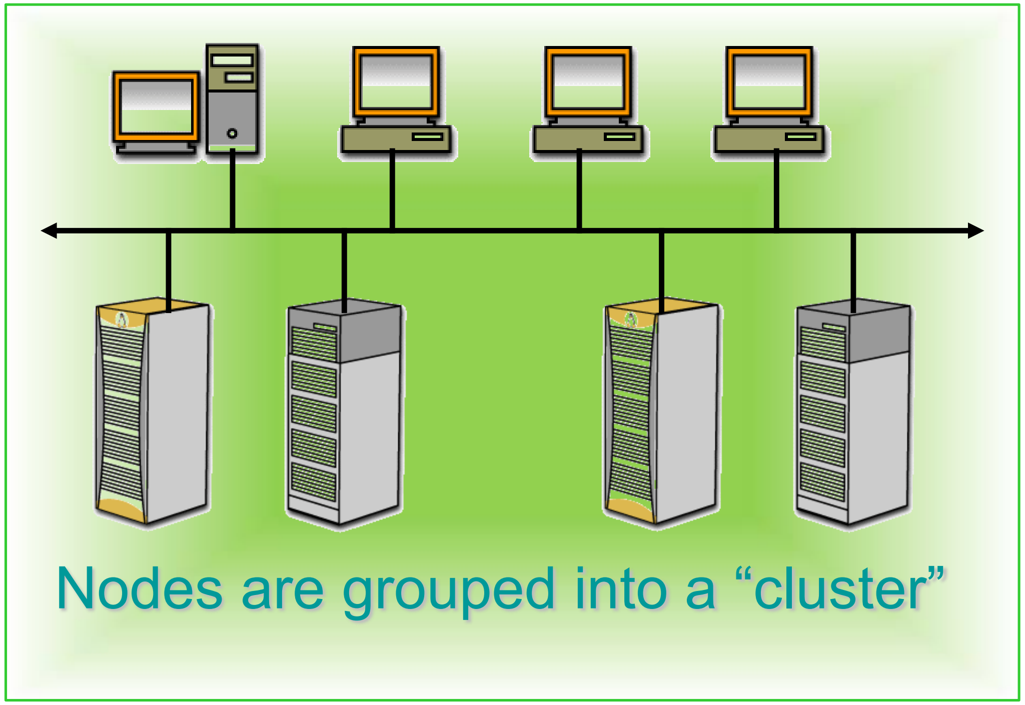
In a distributed environment (hundreds of hosts):

- Monitoring and controlling of resources is complex
- Work load is “silo”-based
- Resource usage imbalance
- Users perceive a lack of resources.

- Without LSF (scheduler/workload manager) many Issues come to the surface almost immediately
    - Over utilization of certain hosts, known to users
        - If I login to node000 once, next time I will likely login to the same host
        - Host could already be very busy by another user's job
    - Under utilization of other hosts
    - Poor utilization of resources
    - Poor satisfaction
    - Admin get frustrated
    - Etc...

With Platform LSF





In a Platform LSF environment (hundreds of hosts):

- Monitoring and controlling of resources is simple
- Work load is balanced
- Resource usage is balanced
- Users jobs are spread out across cluster nodes resulting in resources not being over utilized
- New hosts become utilized as soon as LSF is installed and started on the host.

Benefits

- **Cost management**
  - Spend money where it counts - increase profits.
- **Time management**
  - Reduce resource idle time.
- **Productivity management**
  - Improve personnel productivity, design, and product quality
  - Increase job throughput
  - Reduce time-to-market and costs.
- **Resource management**
  - Maximize license utilization
  - Maximize resource usage
  - Maximize sharing of resources.

LSF Key Terminology

Cluster (At least one machine)

A collection of networked hosts running Platform LSF.

Master host (required)

A cluster requires a master host. This is the first host installed. The master host controls the rest of the hosts in the grid.

Master candidates (optional)

Master failover host.

Server host (optional)

A host within the cluster that submits and executes jobs and tasks.

Client host (optional)

A host within the cluster that only submits jobs and tasks

Execution host (at least one. Any host above can be also execution host)

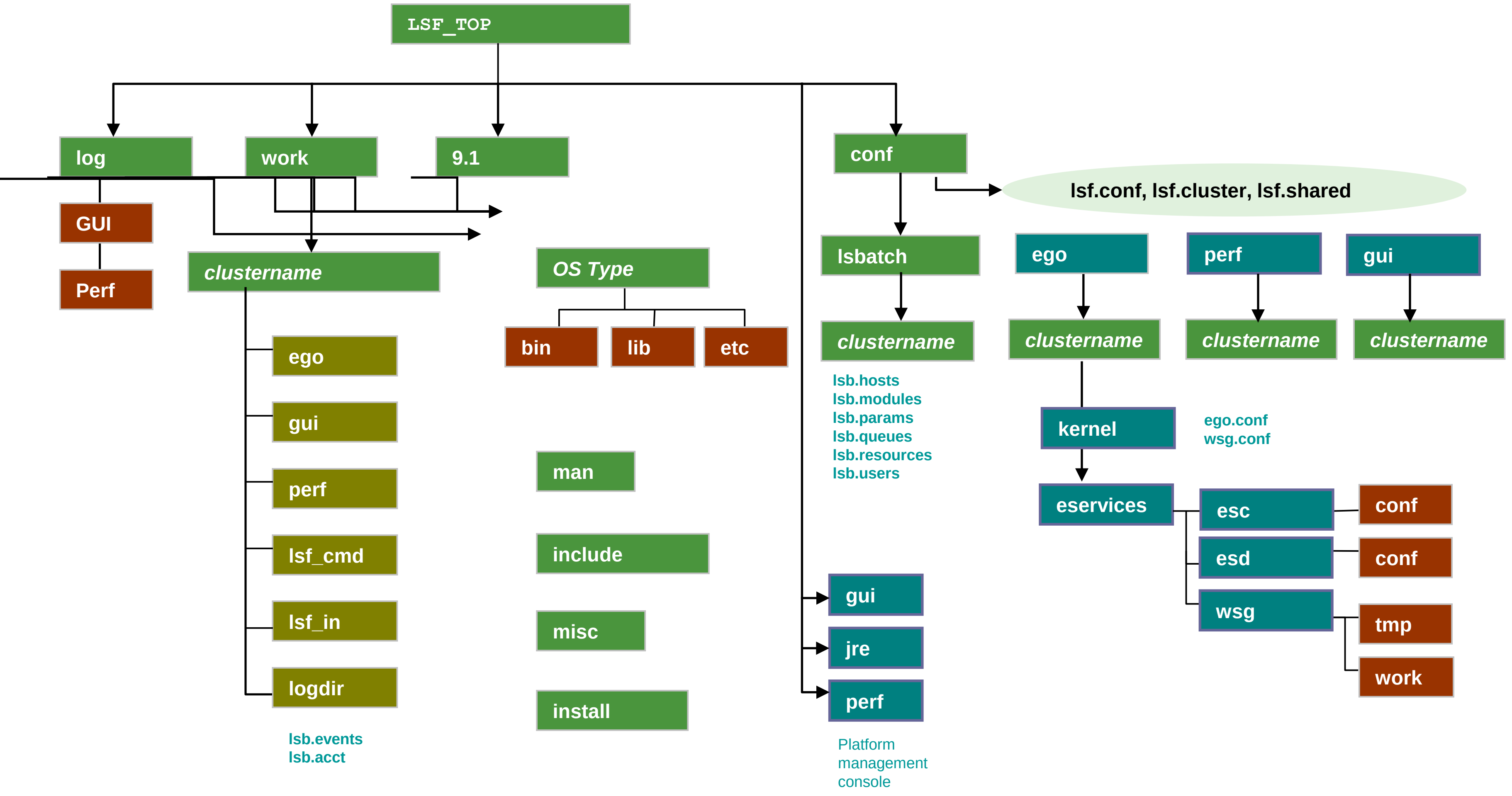
The host that executes the job or task.

Submission host (optional, can be master host)

The host from which a job or task is submitted.

Job

LSF Directory Structure



Essential LSF Commands

LSF provides scripts to setup your user shell environment

```
LSF_TOP/conf/profile.lsf
LSF_TOP/conf/cshrc.lsf
```

- `lsid`: displays the current LSF version number and master host name

```
[root@hpcmaster ~]# lsid
IBM Platform LSF Express 9.1.3.0 for IBM Platform HPC, Jul 04 2014
Copyright IBM Corp. 1992, 2014. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

My cluster name is phpc_cluster
My master name is hpcmaster.kfupm.edu.sa
```

LSF Commands

- `bhosts`: displays hosts and their static and dynamic resources

```
[root@hpcmaster ~]# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP
RSV
gpu00              closed         -     20     20     20      0      0      0
gpu01              closed         -     20     20     20      0      0      0
gpu02              closed         -     20     20     20      0      0      0
gpu03              closed         -     20     20     20      0      0      0
gpu04              ok             -     20     16     16      0      0      0
gpu05              closed         -     20     20     20      0      0      0
gpu06              closed         -     20     20     20      0      0      0
gpu07              ok             -     20      2      2      0      0      0
gpu08              closed         -     20     20     20      0      0      0
gpu09              closed         -     20     20     20      0      0      0
gpu10              closed         -     20     20     20      0      0      0
gpu11              ok             -     20     19     19      0      0      0
hpcmaster.kfupm.ed closed         -      0      0      0      0      0      0
node000            closed         -     20     20     20      0      0      0
node001            ok             -     20     13     13      0      0      0
node002            closed         -     20     20     20      0      0      0
node003            closed         -     20     20     20      0      0      0
node004            closed         -     20     22     20      1      1      0
node005            closed         -     20     20     20      0      0      0
node006            ok             -     20     19     19      0      0      0
node007            ok             -     20      2      2      0      0      0
node008            closed         -     20     20     20      0      0      0
node009            ok             -     20      2      2      0      0      0
node010            closed         -     20     20     20      0      0      0
node011            ok             -     20      2      2      0      0      0
node012            ok             -     20      2      2      0      0      0
node013            closed         -     20     20     20      0      0      0
node014            closed         -     20     20     20      0      0      0
node015            ok             -     20      2      2      0      0      0
node016            closed         -     20     20     20      0      0      0
node017            closed         -     20     20     20      0      0      0
node018            closed         -     20     20     20      0      0      0
node019            closed         -     20     20     20      0      0      0
```

Status closed: not accepting jobs

Status ok: accepting jobs

LSF Commands

`lsload`: displays the load information for hosts

```
[root@hpcmaster ~]# lsload
HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp
mem
gpu01               ok   0.0   0.0   0.2   0%   0.0   0   335  398G 31.2G 62.1G
node014             ok   0.0   0.0   0.0   0%   0.0   0 49152  402G 31.2G 62.2G
node019             ok   0.3   0.0   0.0   0%   0.0   0   362  401G 31.2G 62.2G
hpcmaster.kfupm     ok   1.9   1.5   1.5   9%   0.0   2     3   89G 62.4G  118G
gpu07               ok   9.0   9.0   9.0  45%   0.0   0  2232  399G 31.2G 59.9G
node012             ok   9.0   9.0   9.0  45%   0.0   0  9040  401G 31.2G 59.3G
```

LSF Job Submission & Control

- All LSF commands are located in:

  /shared/ibm/platform_lsf/9.1/linux2.6-glibc2.3-x86_64/bin/

- Should already be in your environment
- If in doubt execute the command 'env | grep -i lsf'
- Only a small subset of LSF commands is relevant to end users
  - `bsub [options] command [cmdargs]`
  - `bjobs [-a][-J jobname][-u usergroup|-u all][…] jobID`
  - `bhist [-a][-J jobname][-u usergroup|-u all][…] jobID`
  - `bbot/btop [jobID | "jobID[index_list]"] [position]`
  - `bkill [-J jobname] [-m] [-u ] [-q] [-s signalvalue]`
  - `bmod [bsub_options] jobID`
  - `bpeek [-f] jobID`
  - `bstop/bresume jobID`
  - `bswitch destination_queue jobID`

bsub—main command

bsub [commonly used options]

`-n #`	- number of CPUs required for the job
`-o filename`	- redirect stdout, stderr, and resource usage information of the job to the specified output file
`-oo filename`	- same as -o, but overwrite file if it exists
`-e filename`	- redirect stderr to the specified error file
`-eo filename`	- same as -e, but overwrite file if it exists
`-i filename`	- use the specified file as standard input for the job
`-q qname`	- submits the job to the specified queue
`-m hname`	- select host(s) or host group; keywords all and others can be used
`-J jobname`	- assigns the specified name to the job

bsub—options

bsub [options]

`-Q “[exit_code] [EXCLUDE(exit_code)]”` - Success exit code

`-L login_shell` - Initializes the execution environment using the specified login shell

`-n number` - if `PARALLEL_SCHED_BY_SLOT=Y` in `lsb.params`, then specify number of job slots, not processors

`-g jobgroup` - submit job to specified group

`-sla serviceclass` - submit job to specified service class

`-W runlimit` - if `ABS_RUNLIMIT=Y` uses wall clock time

`-app` - application profiling

Condensed Notation

```
bsub -m “node[000-011,019]+1 gpu[02-11]+2” my_job
```

This means node000 to node011 are my choices but if not possible use gpu02-gpu11 as second choice

Commands that can use this notation:

bsub brun bmod brestart brsvadd brsvmod bswitch bjobs bhist bacct brsvs bmig bpeek

bsub—limit options

`bsub` [setting limits]

`-C core_limit` - set the per-process core file size limit (KB)

`-c cpu_time` - limit the total CPU time for the job ([HH:]mm)

`-cwd dir_path` - specify the current working directory for the job

`-W runlimit` - set the run time limit for the job ([HH:]MM)

`-We run_time` - specifies an estimation run time for the job ([HH:]MM)

bsub—more options

`bsub` [options]

`-B`	- send email when the job is dispatched
`-H`	- hold the job in the PSUSP state after submission
`-N`	- email job header only when job completes
`-b begin_time`	- dispatch the job after the specified time with year and time
`-G user_group`	- associate job with specified user group (fairshare)
`-t term_time`	- specify the job termination deadline with year and time
`-u mail_user`	- send email to the specified address

Interactive Jobs

bsub

```
-I     – submit an interactive batch job
-Ip     – submit an interactive batch job with pseudo-tty support
-Is     – submit an interactive batch job and create a pseudo-tty with
  shell      mode support
```

```
[root@hpcmaster ~]# bsub -I sleep 10
Job <2190> is submitted to default queue
<medium_priority>.
<<Waiting for dispatch ...>>
<<Starting on node007>>
```

```
[root@hpcmaster ~]# bsub -I hostname
Job <2191> is submitted to default queue
<medium_priority>.
<<Waiting for dispatch ...>>
<<Starting on node007>>
node007
```

Alternative job submission--LSF scripts

- All LSF commands can be encapsulated in:
  - Script file like bsub <<script_file>>
  - For example:

```
bsub -q high_priority -a fluent -n 4 ./my_fluent_launcher.sh
```

  - bsub < spool_file
  - Or Interactively,
  - Just type bsub
  - bsub> “type bsub instructions”

  - Platform HPC Web portal
    - We will do a live demo
    - Use this method whenever possible

Can we run a job bypassing a queue? lsrun and lsgrun

- Yes
  - But admin will be mad at you
  - Admin might disable this feature
  - You should always submit using the queuing system
- lsrun&lsgrun
  - Submits a task to Platform LSF for execution
  - Uses the Platform LSF base system only (no queues policies)
  - Task will run even if hosts are `closed`
  - Tasks will run immediately (no scheduling delay)
  - Tasks will not run when a host is `busy`
  - Can be disabled by admin with configuration change

Terminating your job—bkill

`bkill`

Bulk job termination--New

`bkill -b` option for terminating a bulk set of jobs from the system.

This eases administration and decreases the time to delete a large number of jobs.

Example: To terminate all jobs of the current user

```
% bkill -b 0
```

Viewing submitted job information--bjobs

bjobs

Can display parallel jobs and condensed host groups in an aggregate format

`-a`	Display information about jobs in all states (including finished jobs)
`-A`	Display 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
`-m host_list`	Display information about jobs on specified hosts or groups

Job name: Wild card "*"

-J Option

Supports multiple wildcards in any position (beginning, middle, and end) of a job and job array names

Examples:

```
-J “jobA*”

-J “job*A”

-J "*A”

-J “*”   - To match all job

-J "myAr*ayA[1]“
```

View submitted job information example

```
% bjobs -u all -a

JOBID USER  STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME  SUBMIT_TIME
1233  user1 DONE  normal    training8 training1 *sortName Nov 21 10:00
1234  user1 RUN   priority  training8 training1 *verilog  Nov 21 10:00
1235  user2 PEND  night     training9           *sortFile Nov 21 10:03
1236  user2 PEND  normal    training9           *sortName Nov 21 10:04


% bjobs -u all -J "*sort*:

JOBID USER  STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME  SUBMIT_TIME
1233  user1 DONE  normal    training8 training1 *sortName Nov 21 10:00
1235  user2 PEND  night     training9           *sortFile Nov 21 10:03
1236  user2 PEND  normal    training9           *sortName Nov 21 10:04
```

Viewing historical job information

bhist

`-a`	Display information about all jobs (overrides `-d`, `-p`, `-r`, `and` `-s`)
`-b|-l|-w`	Display information in brief, long, or wide format
`-d`	Display information about finished jobs
`-p`	Display information about pending jobs
`-s`	Display information about suspended jobs
`-t`	Display job events chronologically
`-C|-D|-S|-T start_time end_time`	Display information about completed, dispatched, submitted, or all jobs in specified time window
`-P project`	Display information about jobs belonging to specified project
`-q queue`	Display information about jobs submitted to

View historical job information examples

```
% bhist
```

Summary of time in seconds spent in various states:

```
JOBID   USER    JOB_NAME  PEND  PSUSP   RUN   USUSP   SSUSP   UNKWN   TOTAL
2299    alfred   *eep 100  5      0      2       0       0       0       7
2300    alfred   *eep 100  4      0      2       0       0       0       6
2301    alfred   *eep 100  4      0      2       0       0       0       6
2302    alfred   *eep 100  3      0      2       0       0       0       5
```

```
% bhist -l 2302


Job <2302>, User <lsfuser>, Project <default>, Command <sleep 100>

Wed Mar 30 13:53:44: Submitted from host <host1>, to Queue
   <normal>,CWD<$HOME>;

Mon Mar 18 13:30:37: Dispatched to <host3>, Effective RES_REQ <select[type
   ==local] order[r15s:pg] >;

Wed Mar 30 13:53:47: Starting (Pid 24595);

Wed Mar 30 13:53:52: Running with execution home </home/lsfuser>,
   Execution CWD </home/lsfuser>, Execution Pid <24595>;

Wed Mar 30 13:55:32: Done successfully. The CPU time used is  0.0 seconds;

Wed Mar 30 13:55:32: Post job process done successfully;


MEMORY USAGE:

MAX MEM: 3 Mbytes;  AVG MEM: 3 Mbytes
```

Summary of time in seconds spent in various states by Wed Mar 30 13:55:32

Manipulating jobs options

`bbot`	– moves a pending job to the bottom of the queue.
`btop`	– moves a pending job to the top of the queue.
`bkill`	– sends a signal to kill, suspend or resume unfinished jobs (use a job ID of "0" to kill all your jobs). New scalability improvements resulting in improved performance and user experience.
`bmod`	– modifies job submission options of a job.
`bpeek`	– displays the stdout and stderr of an unfinished job.
`bstop`	– suspend unfinished jobs.
`bresume`	– resumes one or more suspended jobs.
`bswitch`	– switches unfinished jobs to another queue.

Job Arrays

- LSF supports job arrays
  - Array size by default is 1000
  - Admin can change this value to any integer
- Usage model:
  - Same executable but different input files
  - Great for large number of short jobs
  - Monitored as a single job unit
  - Once submitted the individual jobs/tasks will be launched by LSF as independent jobs
  - Used for regression testing (EDA)
  - Rendering
  - File conversion etc...

Creating job arrays

Syntax:

```
bsub -J "JobArrayName[index, …]" -i in.%I YourApp
```

`-J "JobArrayName[…]"`

names and creates the job array index can be start[-end[:step]] where:

```
start - staring index
end   - last index
step  - increment
```

`-i in.%I`

input files to be used and %I is the array index reference

YourApp is the app you want to launch

Examples of Job Arrays

Example #1

```
% bsub -J "test[1-1000]" -i "in.%I" -o "out.%J.%I" appA
  Job <104> submitted to default queue <normal>
```

The `%I` is the index value of the job array and `%J` is the `jobid` assigned for the submitted job array.

Example output files:

```
out.104.1, out.104.2, out.104.3, out.104.4,
```

Example #2

```
% bsub -J "test[1-100:3]" -i "in.%I" -o "out.%J.%I‟ appA
  Job <105> submitted to default queue <normal>
```

Step through the index values by iteration of three.

Example output files:

Example Job Arrays

Example #3

```
% bsub -J "test[1-10,45,90]" -i "in.%I“ -o "out.%J.%I" appA
  Job <104> submitted to default queue <normal>
```

Specific index values can be specified.

Example output files:

```
out.104.1, out.104.2, out.104.3, … ,out.104.10, out.104.45,
   out.104.90
```

Another Example of Job Arrays

Example #4

```
% bsub -J "test[1-200]%2" -i "input.%I" appA
  Job <104> submitted to default queue <normal>
```

Only 2 elements from this job array may run concurrently in the cluster at one time.

```
[lsfuser@host1 ~]$ bjobs -A 104
JOBID    ARRAY_SPEC  OWNER   NJOBS PEND DONE  RUN EXIT SSUSP USUSP PSUSP
104      test[1-2  lsfuser     200  194    4    2    0     0     0     0
```

Example output files:

```
out.104.1, out.104.2, out.104.3, … , out.104.45, out.104.90
```

Monitoring job arrays

- Can monitor the whole array as in:

```
% bjobs 104
JOBID USER  STAT QUEUE  FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
104   user1 RUN  normal training8 training7 test[1]  Jun 6 15:40
104   user1 RUN  normal training8 training5 test[2]  Jun 6 15:40
104   user1 PEND normal training8           test[3]  Jun 6 15:40
```

- Or selected array elements as in:

```
% bjobs "104[90]"
JOBID USER  STAT QUEUE  FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
104   user1 PEND normal training8           test[90] Jun 6 15:40
```

- View the summary status of the job array

```
% bjobs -A 104
JOBID ARRAY_SPEC OWNER NJOBS PEND DONE  RUN EXIT SSUSP USUSP PSUSP
104    test       user1  1000  221  549  230   0     0     0     0
```

Terminating Job Arrays

To terminate all elements of an array:

```
% bkill 104
Job <104>: Operation is in progress
```

To terminate an individual element:

```
% bkill "104[90]"
Job <104[90]> is being terminated
```

To terminate a group of elements:

```
% bkill "104[1-10,75,90]"
Job <104[1]> is being terminated
…
Job <104[10]> is being terminated
Job <104[75]> is being terminated
```

File transfer option (-f)

- LSF can both copy input files/output files from/to the remote execution host
  - Less an issue at KFUPM since the submission host is the same as master node
- Syntax: bsub –f “local_file operator [remote_file]”
- Where the operator can be any of the following:

> Copies local file to remote file before jobs start

< Copies remote file to local file after job completes

<< Appends the remote file to the local file after job completes

>< or

<> Copies local file to remote file before job starts, and remote file to local file after job completes.

Examples of File transfer

```
bsub -f "/tools/scripts/arrays/in.1 > /tmp/in.1" \
     -f "~/out.1 < /tmp/out.1" \
       /tools/scripts/array_app2 /tmp/in.1 /tmp/out.1
```

Submit `array_app2`

copy input file:

from `"/tools/scripts/arrays/in.1`

to `/tmp/in.1`

After job has completed,
copy the output file from `/tmp/out.1` to `~/out.1`

```
Example:

ls -al ~/out.*

-rw-rw-r--  1 lsfuser lsfuser     2 Mar 19 14:07 out.1
```

Some Advanced LSF features

- Users can submit jobs and also express how their jobs should run and what resources they require
- Each job or task can have its own resource requirements
- Server hosts that match the resource requirements of the job are considered as candidates for execution
- Can be set in queues and/or for individual jobs.
- Resource requirement string
  - Platform LSF *Job level* resource requirement: `bsub -R`
  - *Queue level* and *Application profile* resource requirement: `RES_REQ`
  - Describes the resources required by a job or task.
  - Used for mapping tasks and jobs onto execution hosts.

Resource Strings

- A resource requirement string is divided into the following sections:

Selection	- select[*selection_string*]
Ordering	- order[*order_string*]
Usage	- rusage[*rusage_string*]
Locality	- span[*span_string*]
Same	- same[*same_string*]
Compute unit	- *cu[cu_string]*
Affinity	- *affinity[affinity_string]*

- Complex logical expressions can be formed with resource strings that LSF evaluates and matches it against available resources/hosts
- If a host matches it becomes a candidate for execution
- We only cover relevant ones here by examples
  - Locality, Selection & Usage

Selection—select string

```
bsub –R "select[type==any && swp>=300 && mem>500]" job1
```

Select a candidate execution host of any type which has at least 300 MB of available swap and more than 500 MB of available memory.

The select keyword can be omitted if the selection section is first in the resource requirement string as in this example:

```
bsub –R“(ut<0.50 && ncpus==2) || (ut<0.75 && ncpus==4)“ job2
```

Select a candidate execution host the CPU utilization is less than 0.50 and the number of CPUs is two, or the CPU utilization is less than 0.75 and the number of CPUS is four.

Locality-- The span string

- Locality of processes ca be expressed by the -R “span[]”
- Specifies the locality of a **parallel** job.
- Supported options:
  - `span[hosts=1]` which indicates that all processors allocated to this job must be on the same execution host.
  - `span[ptile=n]` which indicates that up to n processors on each execution host should be allocated to the job.
  - `span[ptile=![,HOSTTYPE:n]` uses the predefined maximum job slot limit in `lsb.hosts` (MXJ per host type/model) as the value for other host model or type, other then those host type is specified.
  - When defined at both job-level and queue-level, the job-level definition takes precedence

Span string examples

```
$ bsub -n 16 -R "select[ut<0.15] order[ut] span[hosts=1]"
  parallel_job1
```

Meaning:

All processors required to complete this job must reside on the same execution host with CPU utilization <=15/%

```
$ bsub -n 16 -R "select[ut<0.15] order[ut] span[ptile=2]"
  parallel_job2
```

Meaning:

Up to two CPUs per execution host can be used to execute this job therefore at least eight execution hosts are required to complete this job. Hosts must has <=15% CPU utilization

Resource reservation: rusage string

```
$ bsub -R "select[type==any && swap>=300 && mem>500] order[swap:mem]
  rusage[swap=300,mem=500]" job1
```

On the selected execution host, reserve 300 MB of swap space and 500 MB of memory for the duration of the job.

```
$ bsub -R rusage[mem=500:app_lic_v2=1 || mem=400:app_lic_v1.5=1]"
  job1
```

Job will use 500 MB with app_lic_v2, or 400 MB with app_lic_v1.5.

Resource reservation is ignored for

More rusage string

```
$ bsub -R “select[ut<0.50 && ncpus==2]
   rusage[ut=0.50:duration=20:decay=1]“ job2
```

On the selected execution host, reserve 50% of cpu utilization and linearly decay the amount of cpu utilization reserved over the duration of the period.

```
$  bsub -R "select[type == SUNSOL && mem > 300]
   rusage[mem=300:duration=1h]" job3
```

On the selected execution host, reserve 300 MB of memory for one hour.

The order string

- The order string keyword allows for the ordering of resource the most relevant for a job
- Hosts candidates are ordered from best to worst
- Example:

```
$ bsub -R "select[type==any && swp>=300 && mem>500] order[mem]" job1
```

Order the candidate execution hosts from the highest to lowest amount of available memory.

Some Useful LSF Environment Variables

- **When writing LSF spool script files the following variables are available at submission**
- **LSB_DJOB_HOSTFILE Path to the hostfile**
- **LSB_DJOB_NUMPROC The number of slots allocated to the job**
- **LSB_HOSTS The list of hosts selected by LSF to run the job**
- **LSB_JOBID The job ID assigned by LSF**
- **LSB_JOBINDEX The job array index**
- **LSB_JOBINDEX_END Contains the maximum value of the job array index**
- **LSB_JOBINDEX_STEP Step at which single elements of the job array are defined**
- **LSB_JOBNAME The name of the job**
- **LSB_MCPU_HOSTS The list of the hosts and the number of CPUs us**

Example of a bsub spool file

- This script was used to run HPL on KFUPM cluster

```
#BSUB -J HPL
#BSUB -oo out.%J
#BSUB -eo err.%J
#BSUB -n 640
#BSUB -R "span[ptile=20]"
cd /home/mmekias
#Start the mpd daemons
mpdboot -r ssh -n 33 -f hosts
# Run MPI program
mpiexec -nolocal -ppn 20 -n 640 ./xhpl_intel64
mpdallexit
```

Job Name

Output file. -o0 means override if file exists

error file. -e0 means override if file exists

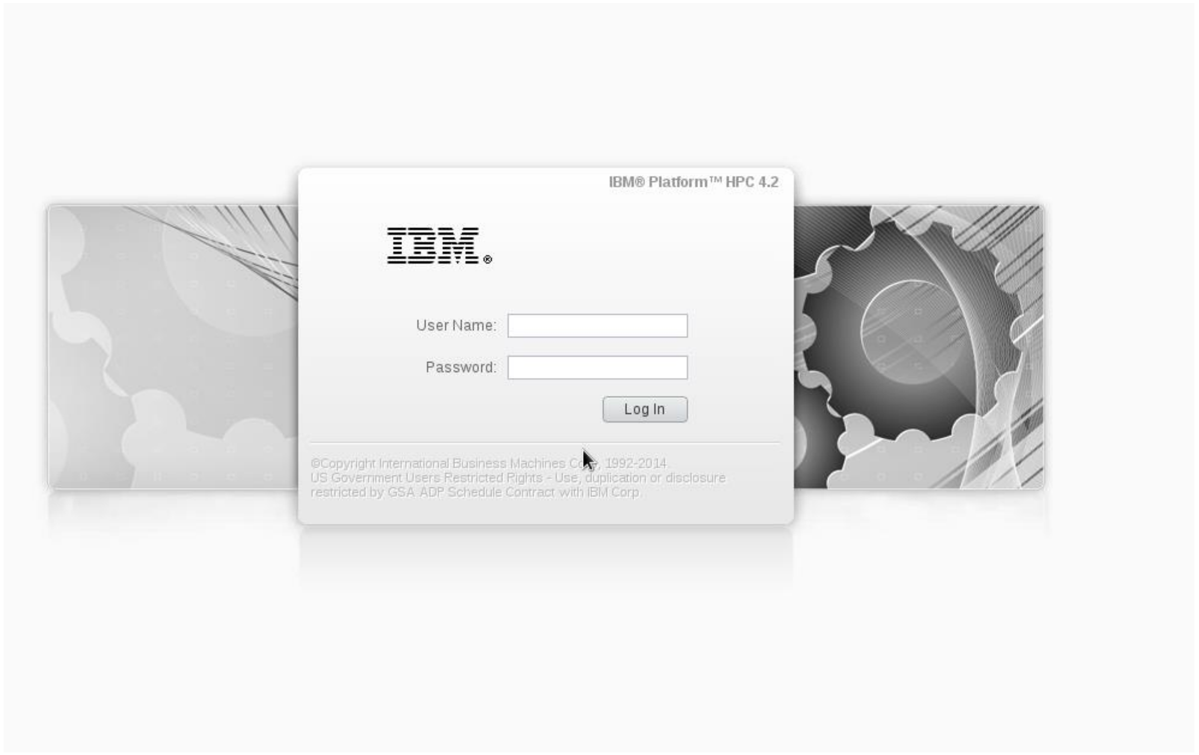
Number of processes. Here the total #

How many processes per node

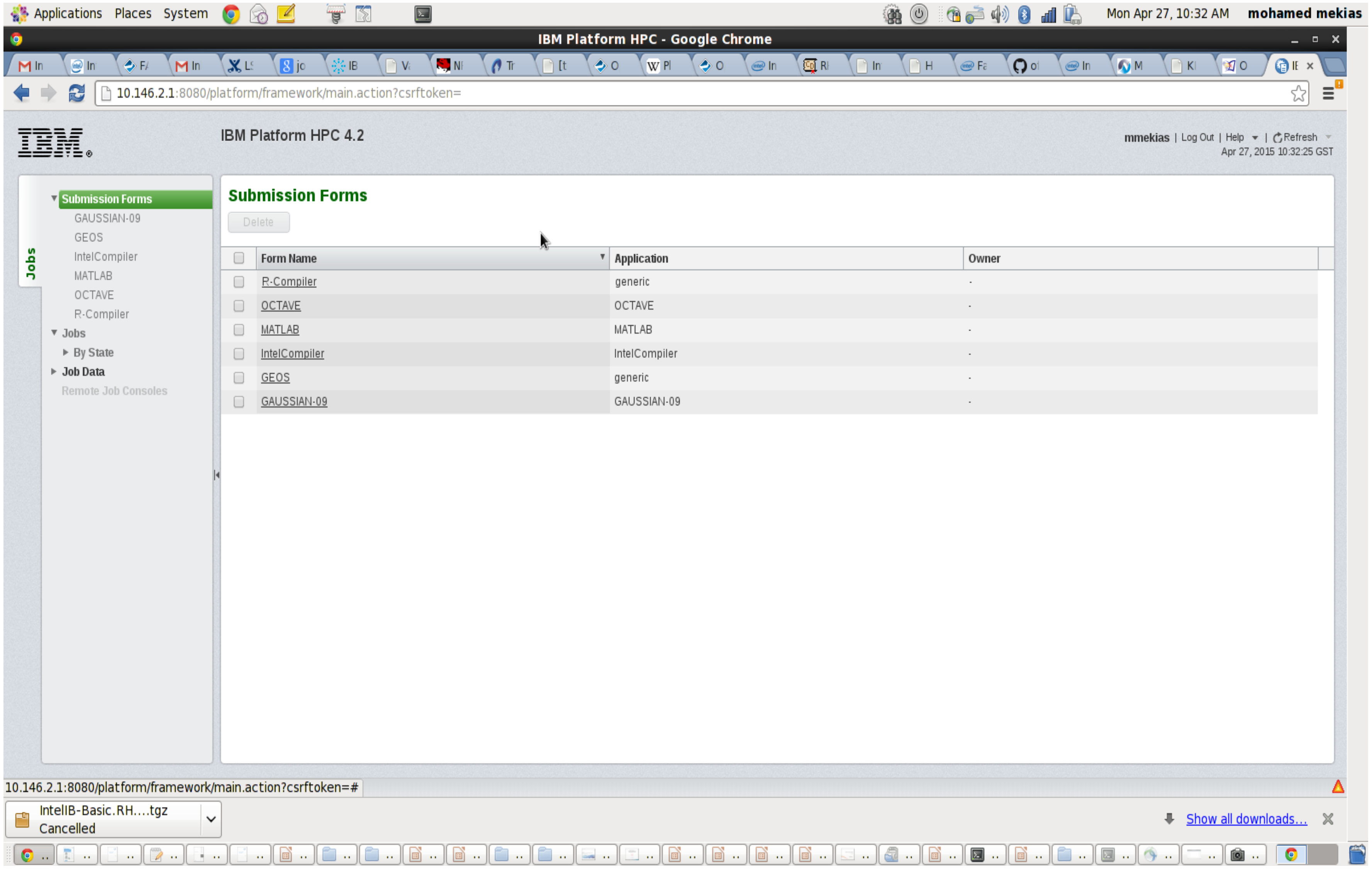
Job specific details

Web portal demonstration

- Please bring the webgui:
  - http://10.146.2.1:8080
  - Login with your cluster credentials

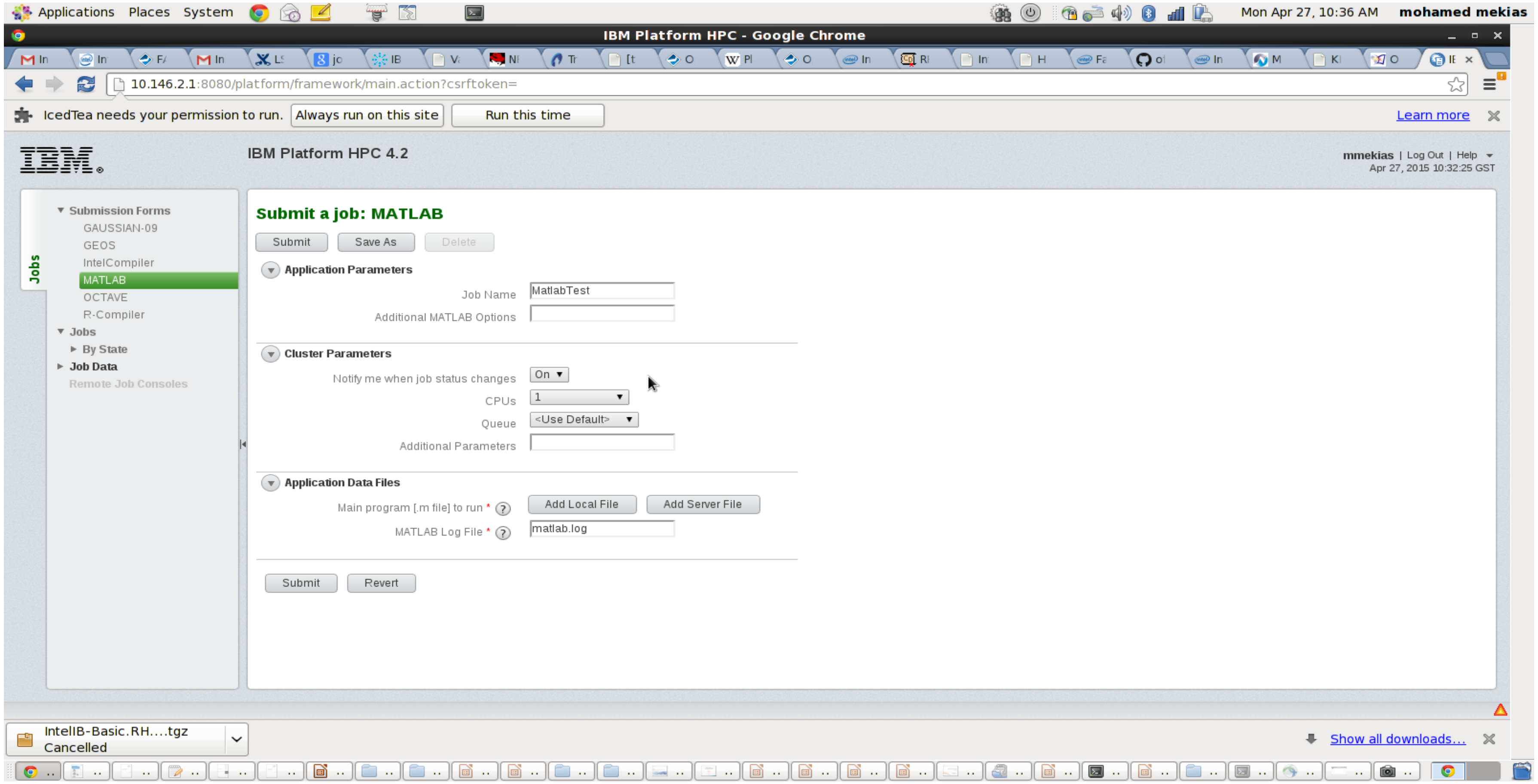


©Copyright International Business Machines Corp, 1992-2014
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



Submission Forms

Form Name	Application	Owner
R-Compiler	generic	-
OCTAVE	OCTAVE	-
MATLAB	MATLAB	-
IntelCompiler	IntelCompiler	-
GEOS	generic	-
GAUSSIAN-09	GAUSSIAN-09	-



Applications Places System Mon Apr 27, 10:36 AM mohamed mekias

IBM Platform HPC - Google Chrome

10.146.2.1:8080/platform/framework/main.action?csrftoken=

IcedTea needs your permission to run. Always run on this site | Run this time | Learn more

IBM Platform HPC 4.2

mmekias | Log Out | Help
Apr 27, 2015 10:32:25 GST

- Submission Forms
  - GAUSSIAN-09
  - GEOS
  - IntelCompiler
  - MATLAB
  - OCTAVE
  - R-Compiler
- Jobs
  - By State
- Job Data
- Remote Job Consoles

Submit a job: MATLAB

Submit | Save As | Delete

Application Parameters

Job Name	MatlabTest
Additional MATLAB Options	

Cluster Parameters

Notify me when job status changes	On
CPUs	1
Queue	<Use Default>
Additional Parameters	

Application Data Files

Main program [.m file] to run *	Add Local File / Add Server File
MATLAB Log File *	matlab.log

Submit | Revert

IntelB-Basic.RH....tgz Cancelled | Show all downloads...