

Chong Chen

Lead Product Architect, IBM Platform LSF Family

chong@ca.ibm.com

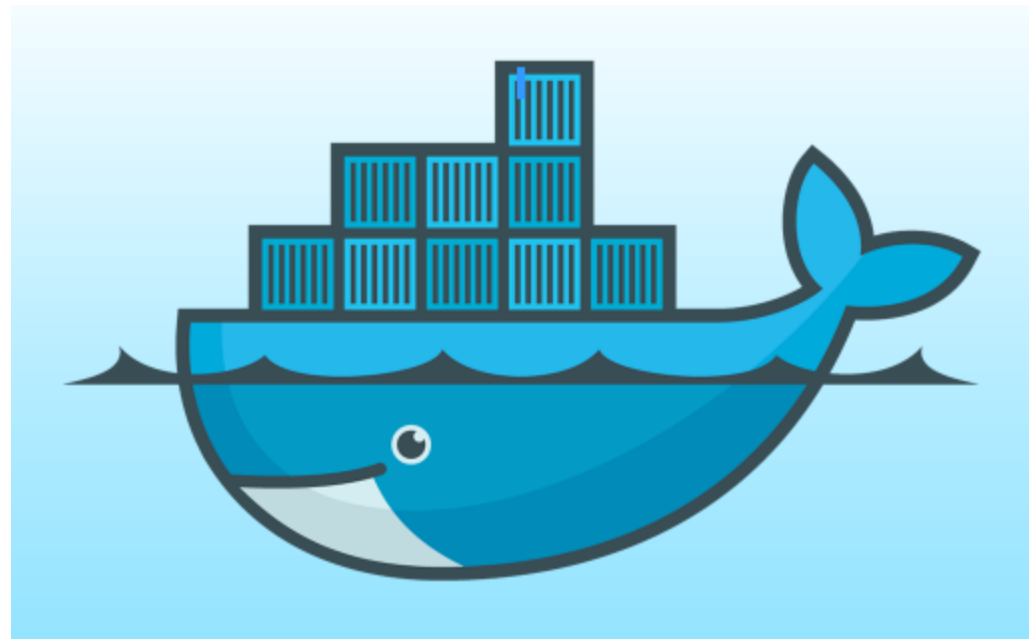
Using Docker in High Performance Computing

November 2014



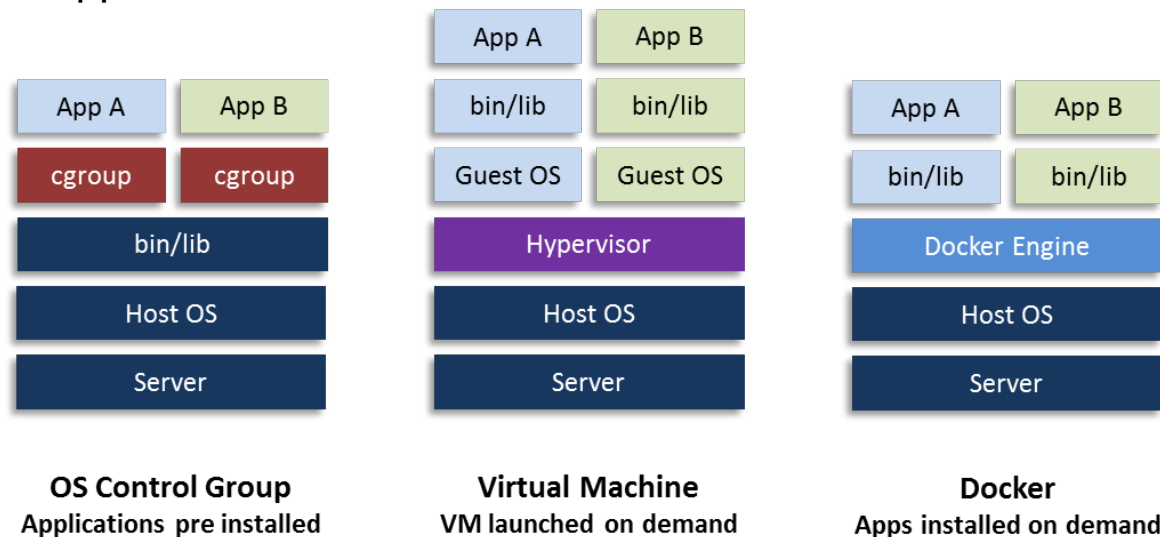
Agenda

- **What is Docker?**
- **LSF & Docker**
- **Next step**
- **Q & A**





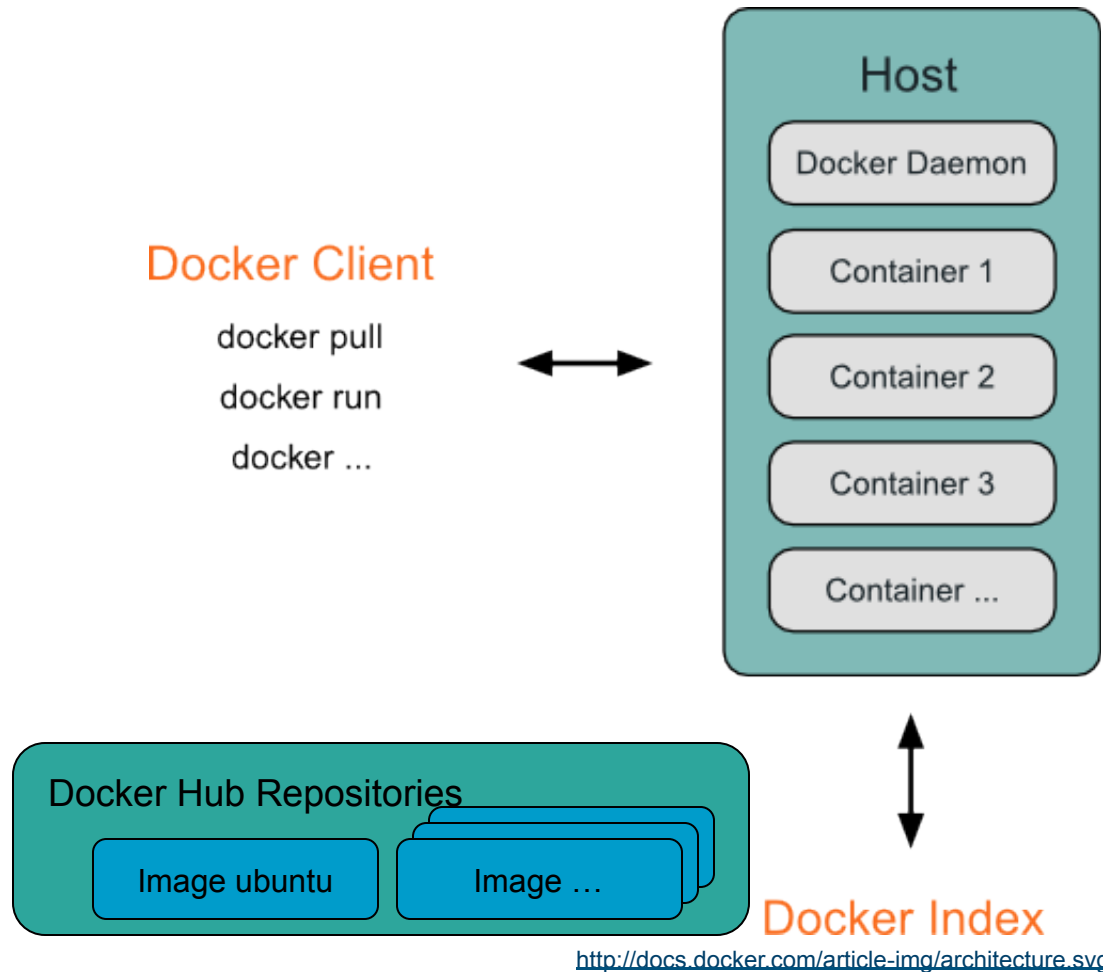
- In short, Docker is a lightweight container technology built on top of linux container, cgroup and AUFS (Another UnionFS)
 - Think of it as “a Virtual Machine, without the overheads or size of a VM, that installs a complete application stack on demand.”



- Hot topic in “cloud & big data” to develop, ship and run applications anywhere and solve “dependency hell” challenges
 - All environment/library/distribution dependencies can be encapsulated within the container, so two applications that would have conflicting libraries requirement can now share the same host.

Docker Architecture

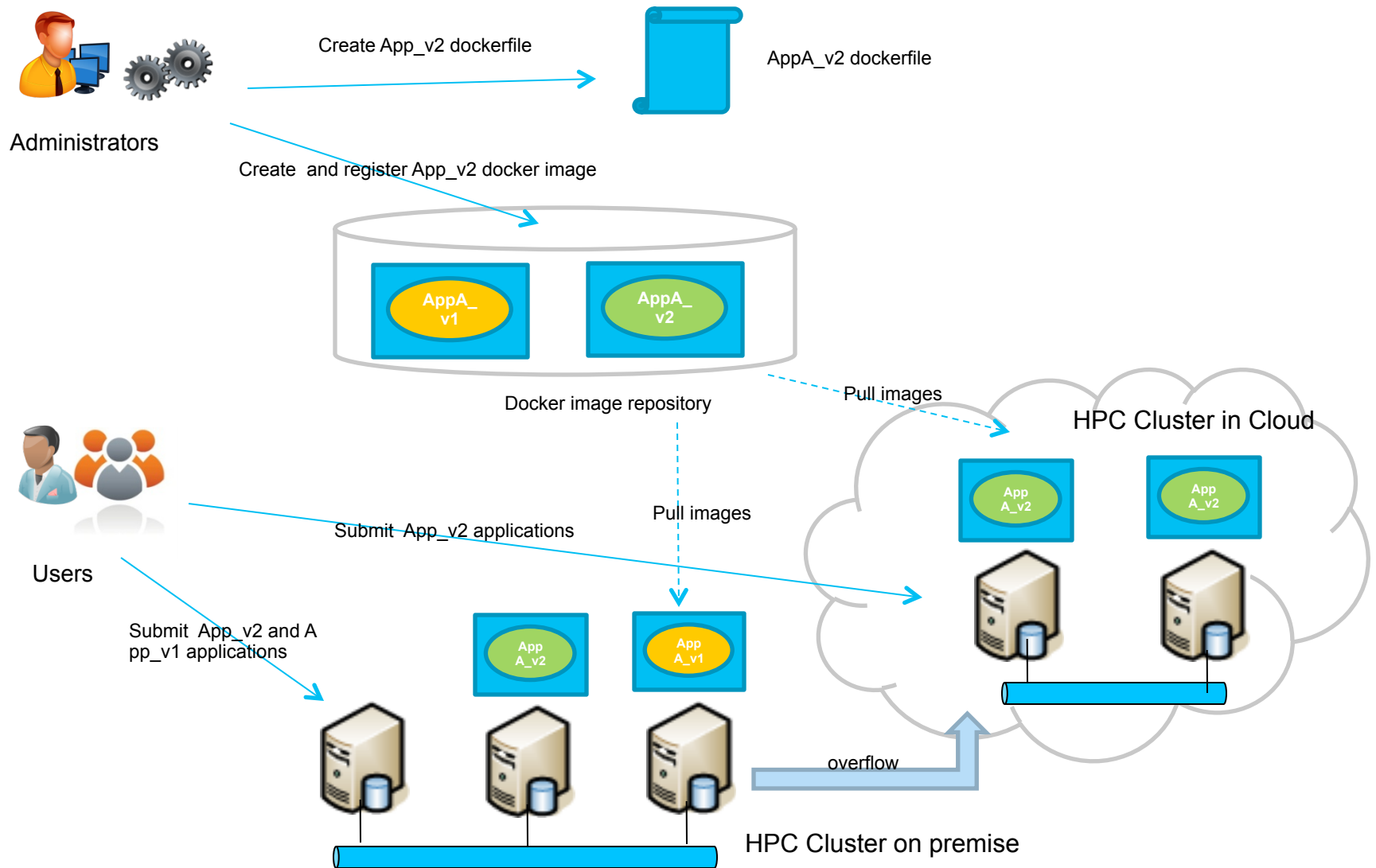
- The Docker Daemon
- The Docker Client
- Inside Docker
 - Docker Images
 - Docker Repositories
 - Docker Containers



Containers in High Performance Computing

- Not a new idea ...
 - Workload resource isolation
 - Process tracking
 - Job controlling
 - Checkpoint/restart, live migration
 - ...
- Examples:
 - IBM AIX WLM
 - Linux control group
 - Virtual machine
 - ...

Docker in High Performance Computing



Potential Benefits

- Resource guarantee and performance isolation
 - Docker leverages Linux control group, a nature extension to cgroup
- Application encapsulation and cloud mobility
 - Application typically has lots of dependency on numerous libraries for correct execution. Challenge to move application from one system to another or out on to cloud.
 - Docker makes it easy
- Application lifecycle management
 - Allows different version of applications easily coexisting in the same environment.
- Consistency, repeatability and compliance
 - Docker provides a way to run application in a known pre-defined environment. Each time, application runs in the exactly the same environment
- Lightweight, fast and transparent
 - Docker offers identical application performance on bare metal system and fast management operations.

Using Docker with IBM Platform LSF



Open Beta Download:
<http://smconnect.net/tc/plsf/downloads.html>

IBM Platform MapReduce Accelerator for LSF

The MapReduce programming model represents a potential new methodology in HPC, and many people are interested in exploring the applicability of the techniques. IBM Platform MapReduce Accelerator for LSF enables MapReduce workloads to be submitted to LSF, and scheduled or executed in a similar manner to other multi-node/parallel jobs. This allows users to benefit from the enhanced MapReduce performance, without the need to invest in a dedicated MapReduce environment.

Simply click the beta build below, and ask questions in the [Forum](#).

Date	Type	Description	Download
22 Aug 2014	Beta	IBM Platform MapReduce Accelerator for LSF	Beta 3

Collaborate

WIKI
Read and contribute best practices

Forum
Ask and respond to technical questions

Support Portal

Easy, Fast, Smart. Your customized support experience.

Follow Platform Computing

White paper:
<https://ibm.biz/BdFzPY>

IBM Platform LSF Integration with Docker

Docker is an emerging container technology on the Linux Platform that allows users to develop, ship, and run applications almost everywhere. This open beta integrates Platform LSF with Docker to automatically start the Docker container on the execution host of an HPC cluster after users have submitted jobs with the Docker image. The integration honours LSF resource allocation (CPU, memory and affinity) and supports full job control, including job suspension, resumption and termination, and it can also collect run time container resource usage.

Try the beta build below, and ask questions in the [Forum](#).

Date	Type	Description	Download
1 Aug 2014	Beta	IBM Platform LSF Integration with Docker	Beta 2

How to use it

```
# [lsfadmin@gecko1]$ lshosts
HOST_NAME      type      model    cpuf  ncpus  maxmem  maxswp  server  RESOURCES
gecko1         X86_64   Intel_EM 60.0   16    63.9G   1.9G    Yes    (mg docker)
gecko2         X86_64   Intel_EM 60.0   16    63.9G   1.9G    Yes    (mg docker)
gecko3         X86_64   Intel_EM 60.0   16    63.9G   1.9G    Yes    (mg docker)
```

Docker image name

```
[lsfadmin@gecko1]$ bsub -Is -a "docker (panxun/bwa)" /bin/sh
Job <402> is submitted to default queue <interactive>.
<<Waiting for dispatch ...>>
<<Starting on gecko2>>
$ echo $LSB_JOBID
$ 402
$ hostname
Docker_SL-402
$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04 LTS"
```

Cluster name and LSF JobID

```
[lsfadmin@gecko1]$ lsrunc -m gecko2 docker ps
CONTAINER ID  IMAGE                COMMAND             CREATED          STATUS          PORTS          NAMES
df3840bflab1 panxun/bwa:latest    /bin/sh             8 minutes ago   Up 8 minutes   Docker_SL-402
```

A real example

Docker image name

```
[lsfadmin@gecko1]$ bsub -n 4 -cwd "/gpfs/fpo/scratch/%J_%I" -a "docker (panxun/bwa) "  
"bwaaln -t 4 -l 40 -n 3 -k 2 /gpfs/fpo/bwadata/UCSC_HG19/ucsc.hg19.fasta /gpfs/fpo/  
bwadata/GZIP_FASTQ_3PAIR/SRR034939_1.fastq.gz > SRR034939_1.sai"  
Job <404> is submitted to default queue <normal>.
```

```
[lsfadmin@gecko1]$ bjobs  
JOBID    USER      STAT  QUEUE          FROM_HOST    EXEC_HOST    JOB_NAME     SUBMIT_TIME  
404      lsfadmi  RUN   normal         gecko1       gecko2       *939_1.sai  Jul 22 23:59
```

```
[lsfadmin@gecko1]$ bpeek 404
```

```
<<output from stdout>>  
Unable to find image 'panxun/bwa' locally  
Pulling repository panxun/bwa
```

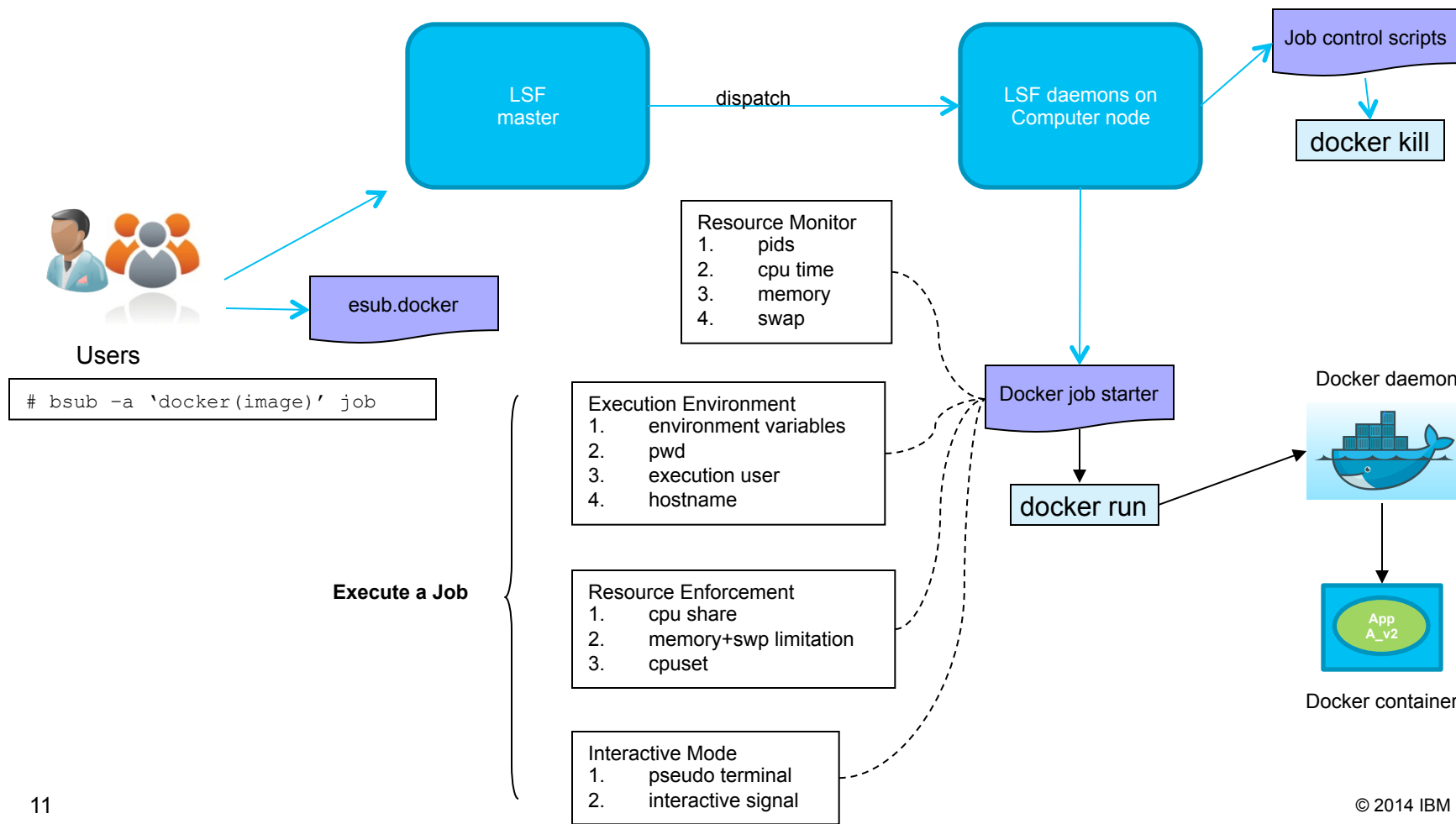
Downloading and installing image

```
[bwa_aln_core] calculate SA coordinate... 75.86 sec  
[bwa_aln_core] write to the disk... 0.04 sec  
[bwa_aln_core] 262144 sequences have been processed.  
[bwa_aln_core] calculate SA coordinate... 74.39 sec  
[bwa_aln_core] write to the disk... 0.04 sec  
[bwa_aln_core] 524288 sequences have been processed.  
[bwa_aln_core] calculate SA coordinate... 73.04 sec  
...
```

```
[lsfadmin@gecko1 tmp]$ ls -l /gpfs/fpo/scratch/404_0/SRR034939_1.sai  
-rw-rw-r-- 1 lsfadminlsfadmin 67334144 Jul 23 00:05 /gpfs/fpo/scratch/404_0/  
SRR034939_1.sai
```

How it works

- Leverage existing IBM Platform LSF plugins mechanism (esub, job starter, job control etc)



Resource Requirement and Enforcement

- CPU Shares

- Translate IBM Platform LSF total number of slots to relative cpu share on docker.
bsub -n <num> -> docker run -c <num>

- Memory Fencing

- Enable IBM Platform LSF cgroup feature and translate *-M/-v* to docker memory option
bsub -M/-v -> docker run -m

- Affinity & NUMA aware scheduling

- IBM Platform LSF selects which hosts and which cores/sockets and pass information to docker to enforce it.
-n 4 R "span[hosts=1] affinity[core]" -> docker run -cpuset

Next Step and Interest Areas

- Tighter integration with IBM Platform LSF job resource collection framework
 - Leverage built-in runtime IBM Platform LSF resource collection and reporting mechanism instead of using bpost
- Docker image aware scheduling
 - Docker image can be big and it takes time to download and install it.
 - Prefer to go to hosts with image pre-installed
- Local image management
 - Image should be cleaned up periodically when not required to save disk space

Please download and try it out



Open Beta Download:
<http://smconnect.net/tc/plsf/downloads.html>

IBM Platform MapReduce Accelerator for LSF

The MapReduce programming model represents a potential new methodology in HPC, and many people are interested in exploring the applicability of the techniques. IBM Platform MapReduce Accelerator for LSF enables MapReduce workloads to be submitted to LSF, and scheduled or executed in a similar manner to other multi-node/parallel jobs. This allows users to benefit from the enhanced MapReduce performance, without the need to invest in a dedicated MapReduce environment.

Simply click the beta build below, and ask questions in the [Forum](#).

Date	Type	Description	Download
22 Aug 2014	Beta	IBM Platform MapReduce Accelerator for LSF	Beta 3

IBM Platform LSF Integration with Docker

Docker is an emerging container technology on the Linux Platform that allows users to develop, ship, and run applications almost everywhere. This open beta integrates Platform LSF with Docker to automatically start the Docker container on the execution host of an HPC cluster after users have submitted jobs with the Docker image. The integration honours LSF resource allocation (CPU, memory and affinity) and supports full job control, including job suspension, resumption and termination, and it can also collect run time container resource usage.

Try the beta build below, and ask questions in the [Forum](#).

Date	Type	Description	Download
1 Aug 2014	Beta	IBM Platform LSF Integration with Docker	Beta 2

Collaborate

WIKI
Read and contribute best practices

Forum
Ask and respond to technical questions

Support Portal

Easy, Fast, Smart. Your customized support experience.

Follow Platform Computing

Appendix

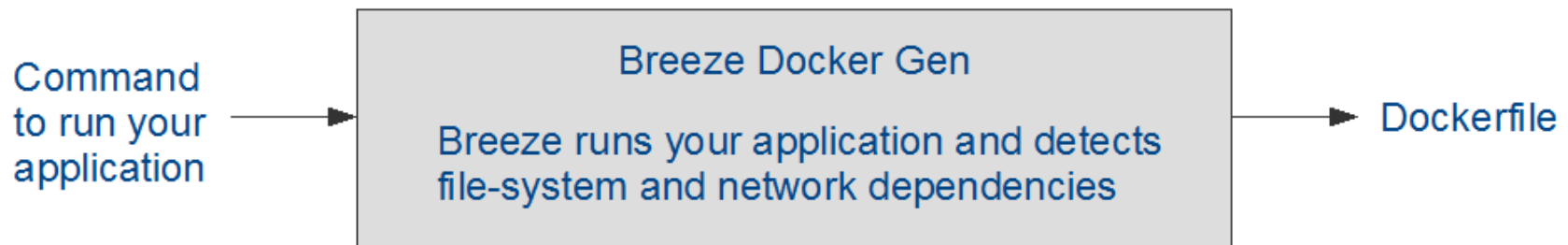


Dockerize application



Ellexus make tools to help you solve the problems that arise when deploying and tuning complex Linux applications

Ellexus have a new product: **Breeze-Docker-Gen**, a tool for automatically generating docker files



Breeze-Docker-Gen uses run-time dependencies and system settings to generate the install layers, addition files and network settings needed by your application so you include only the files you need.

Breeze-Docker-Gen is designed to save time when packaging third party tools or complex applications with unknown dependencies.

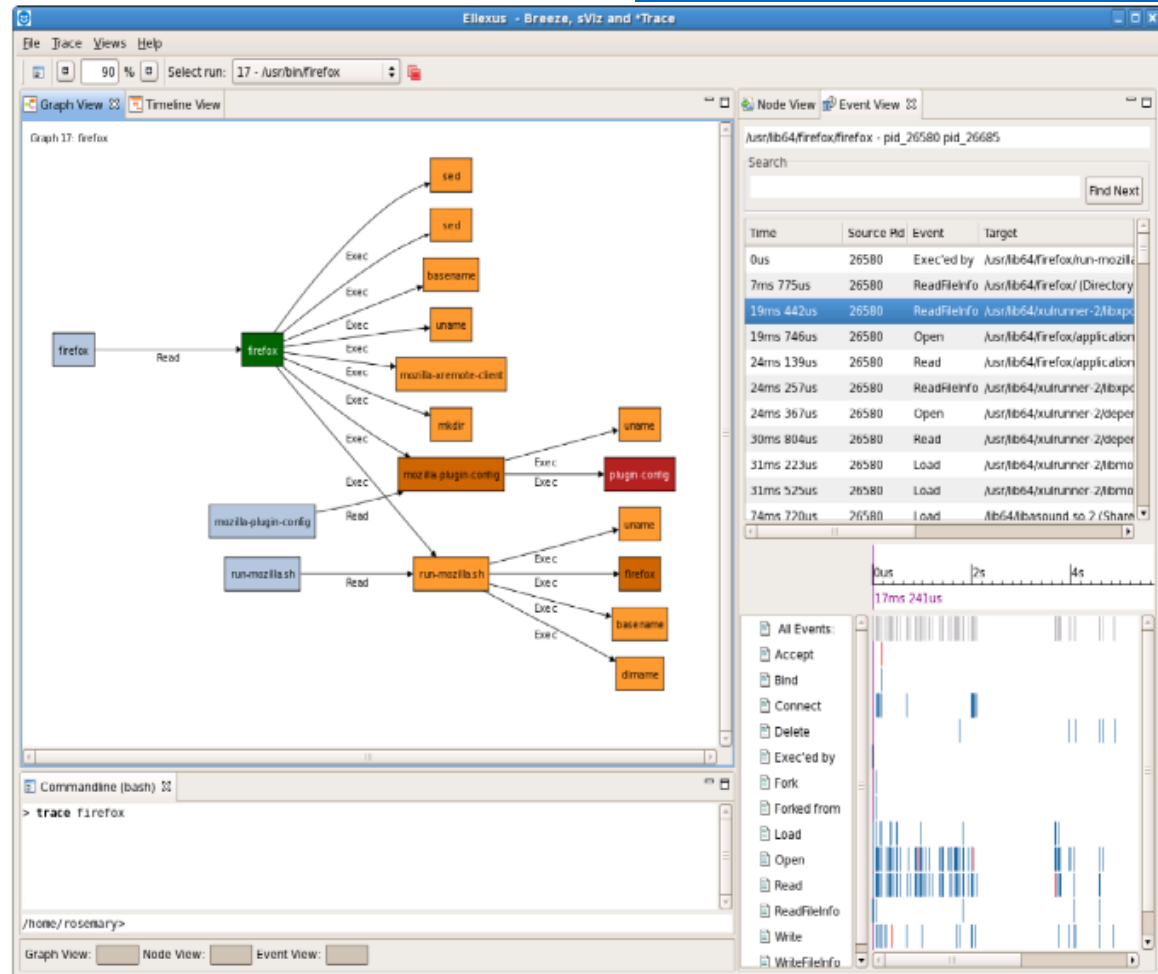
Dockerize application



Breeze-Docker-Gen uses the Ellexus Breeze tracing technology to monitor your application as it runs. Breeze sees every program call, every file open and every network access.

It can then generate a list of dependencies and the Docker file from this information.

The screen shot (right) shows the Firefox start up and config sequence in Breeze.



Thank You For Your Time

