SAS® GLOBAL FORUM 2020 #SASGF

Paper 6007-2020

Expanding SAS® Grid Manager for Platform: Lessons from the Field

Bill McMillan, IBM United Kingdom; Qingda Wang, Kinson Chik, IBM Canada

ABSTRACT

SAS Grid Manager provides a powerful multi-tenant computing environment that enables high availability and accelerates processing for analytical workloads.

"SAS Grid Manager for Platform" and "Platform Suite for SAS" are built upon IBM's LSF family of products. In addition to powering the vast majority of SAS Grid deployments, LSF manages the Summit (#1), Sierra (#2), Lassen (#10) and Pangea3 (#11) systems on the Top500[1] list, and the compute grids of many of the world's largest semiconductor, automotive, aerospace and financial institutions.

These large-scale deployments may seem worlds away from your typical SAS Grid, but they use the same underlying LSF technologies and capabilities which can be leveraged in your SAS Grid environment today to increase user productivity and deliver better business outcomes.

This paper will examine four key areas where we have helped SAS Grid users and others to derive increased business value by leveraging additional LSF functionality. Namely:

- Enhanced Scheduling and tools for User Productivity
- Accelerated Computing with GPUs
- Leveraging Containers and Kubernetes
- Hybrid Cloud for on-demand burst capacity with data management.

These will be illustrated using client examples.

INTRODUCTION

Analytics is no longer the remit of a handful of data analysts providing mystical insight into an organizations data. Analytics is everywhere, empowering the whole enterprise. Everyone wants access, and many are trying to use new methods, such as AI/ML/DL, to derive greater value and provide greater insight from our oceans of data.

SAS provides many excellent applications and solutions to help on this endeavor, but as with the volumes of data rapidly growing, so are the number of different tools and potential applications. In a recent survey, it was found that the average data analyst uses 7 [KDNuggets2018] different tools.

[1] https://www.top500.org/list/2019/11/

From an IT perspective this brings many new challenges – not only in having to support these applications, but in delivering suitable infrastructure in a timely manner and handling the additional data growth and governance with supplying this data to the infrastructure where these applications reside.

While SAS Grid Manager for Platform is restricted to running just SAS applications, LSF, itself, typically manages very diverse and heterogeneous environments for many of the world's largest semiconductor, health care and life sciences, automotive, aerospace and financial institutions. These large-scale deployments may seem worlds away from your typical SAS Grid, but they use the same underlying LSF technologies and capabilities which can be leveraged in your SAS Grid Manager for Platform environment today.

The one attribute that all these environments have (apart from running LSF) is that they are not dedicated to a single application. They are multi-tenant supporting a wide range of applications, application frameworks and users.

To summarize, our work with SAS over the years has led to many SAS Grids being deployed. As new tools and workloads appear, new ways to manage and consume core IT emerge, and organizations desire to leverage existing investments to handle change and growth, our commitment to evolve alongside should be apparent in this paper through what clients ask us to help with and how we provide capability in these areas to drive efficiency – of use, of management, and [of course] performance.

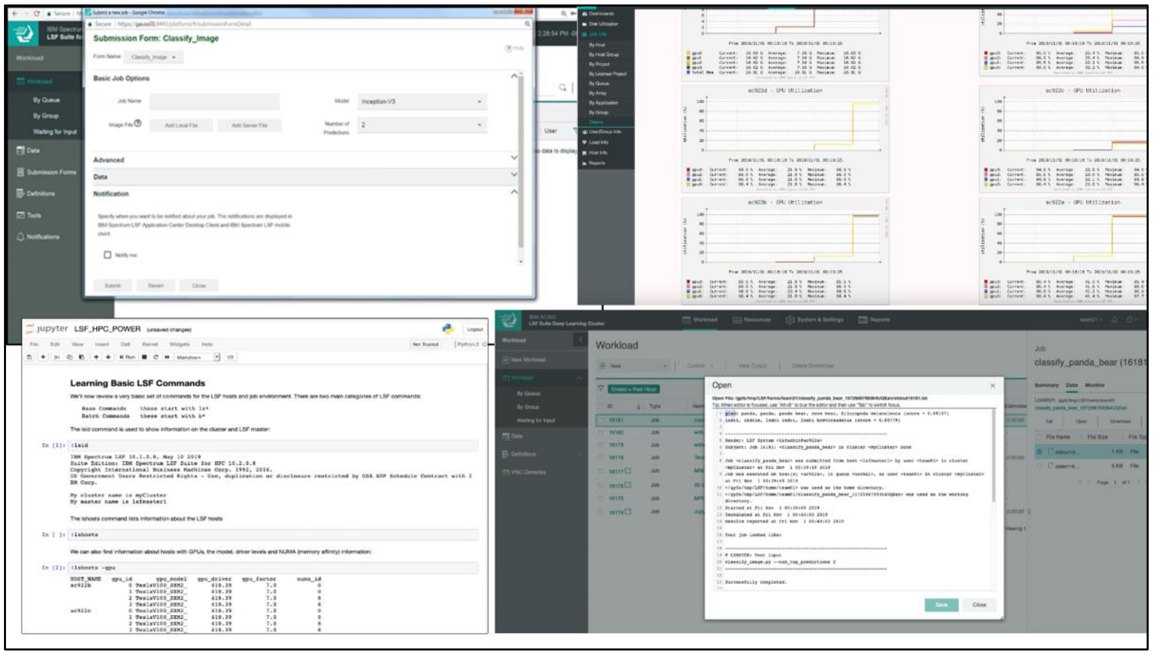
Thus, the aim of this paper is to provide you with insight into how other organizations are leveraging LSF technologies for scheduling, GPUs, Cloud, Containers and Kubernetes for both SAS and non-SAS workloads, thereby delivering additional IT agility and business value.

SAS WORKLOAD ORCHESTRATOR

SAS 9.4M6 introduced SAS Workload Orchestrator as an alternative solution to what was renamed SAS Grid Manager for Platform. LSF[2], like SAS, consists of multiple components. Haig2019 presents a comparison of SAS Workload Orchestrator and LSF. However, it only compares the LSF components that SAS includes to be used, and not the whole LSF Suite. For example:

- REST-API: The mobile application, for example, communicates with LSF via a RESTful API which is generally available. There is also a version of the traditional LSF command line that uses the RESTful API for communication.
- Pluggable Scheduling Logic: The core design of the LSF scheduler is based on pluggable scheduling modules. This translates to not only can you write your own scheduling logic; you can extend or over-ride parts of existing scheduling modules.
- Type of Jobs that can be run: There are no inherent restrictions on the workload LSF can run – if you can launch if from a command line you can run it in LSF. In addition to the command line, there is a full Python API.
- Embedded GUI: Multiple GUI's are available tailored to user personae. For example, Welch2019 illustrates LSF's Application Center GUI running containerized Jupyter Notebooks, MPI, TensorFlow, TensorBoard, Horovod, PyTorch with and without GPUs as shown in Figure 1.

[2] The IBM Spectrum LSF family is available in multiple editions ranging from LSF Standard Edition, which is just the core scheduler, through to LSF Suite for Workgroups, Suite for HPC, Suite for Enterprise and the Suite for High Performance Analytics. Unlike LSF Standard Edition, the LSF Suites use Ansible for installation.



Learning Basic LSF Commands

We'll now review a very basic set of commands for the LSF hosts and job environment. There are two main categories of LSF commands:

```
Base Commands    these start with ls*
Batch Commands   these start with b*
```

The lsid command is used to show information on the cluster and LSF master:

```
In [1]: !lsid
IBM Spectrum LSF 10.1.0.8, May 10 2019
Suite Edition: IBM Spectrum LSF Suite for HPC 10.2.0.8
Copyright International Business Machines Corp. 1992, 2016.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with I
BM Corp.

My cluster name is myCluster
My master name is lsfmaster1
```

The lshosts command lists information about the LSF hosts

```
In [ ]: !lshosts
```

We can also find information about hosts with GPUs, the model, driver levels and NUMA (memory affinity) information:

```
In [2]: !lshosts -gpu
HOST_NAME   gpu_id       gpu_model   gpu_driver   gpu_factor      numa_id
ac922b           0 TeslaV100_SXM2_       418.39          7.0            0
                 1 TeslaV100_SXM2_       418.39          7.0            0
                 2 TeslaV100_SXM2_       418.39          7.0            8
                 3 TeslaV100_SXM2_       418.39          7.0            8
ac922c           0 TeslaV100_SXM2_       418.39          7.0            0
                 1 TeslaV100_SXM2_       418.39          7.0            0
                 2 TeslaV100_SXM2_       418.39          7.0            8
                 3 TeslaV100_SXM2_       418.39          7.0            8
```

Figure 1 Application Center with TensorFlow and Jupyter Notebooks

There are many additional capabilities, including dynamic reconfiguration, in the LSF family that are beyond the scope of this paper. For those who are interested, the LSF Suite for High Performance Analytics provides all the capabilities described in this paper.

WORKLOAD POLICIES

LSF has been used across a broad range of industries for over 25 years. As such, and based upon various Client needs, many different policies and tuning options have been developed. It is sometimes referred to as the "swiss army knife" of scheduling – you're not quite sure what all the blades are for, but they are there for a reason.

SAS Environment Manager exposes many of LSF's settings and policies, but not everything that would benefit managing additional analytics workloads is surfaced. SAS has made readily available what was felt needed to manage their SAS environment workloads. It is not practical to discuss every LSF option available, but we'll highlight a number of common scenarios that have come up repeatedly within the context of the SAS audience.

INTERACTIVE WORKLOADS

Let's start with some basics – like prioritization. LSF provides numerous scheduling policies to allow workloads to be prioritized with multiple service levels. The key question is ultimately "how long is the user willing to wait for an answer". With batch-orientated work it may be minutes, or it could be hours or even days – but when someone is running interactively, the answer is "now" or "ASAP" – and any perceived delay (whether real or imagined) often results in the belief that IT is not providing good enough service.

A Banking client reported user dissatisfaction due to exactly this reason. They had attempted to address it by adjusting the queue priority, but the users were still unsatisfied.

At a very high level, LSF performs a scheduling cycle every MBD_SLEEP_TIME seconds, and the default value of this parameter varies depending on the version and edition of LSF that you are using. There are multiple parameters that impact the overall latency between a job being submitted and when it starts; therefore, changing the scheduling interval or the queue priority will not address the client requirement.

The key to addressing this is setting:

```
NEW_JOB_SCHED_DELAY=0 in lsb.queues
```

This forces LSF to immediately evaluate any new job submitted into that queue, thus minimizing any perceived scheduling delay.

AUTO-SELECTION OF QUEUES

The above setting met the client's initial requirement; however, they didn't want to expose queue selection to their users. Could they have the queue auto-selected?

They were considering LSF's "esub" feature to create a submission filter which would re-direct the incoming workloads. But LSF already has a simple method for doing this:

```
DEFAULT_QUEUE=normal interactive in lsb.params
INTERACTIVE=ONLY in the Interactive Queue definition in lsb.queues
INTERACTIVE=NO in the Normal Queue definition in lsb.queues
```

This simple change means that when an interactive job is submitted (bsub -I) it will automatically get routed to the interactive queue and be dispatched with the minimal latency. There are additional policies that can be used to automate queue selection.

FAIRSHARE

A government client wanted to share the cluster among several departments, with a different priority for each department. However, they did not want any department to be starved of resources when the highest priority department had a large amount of work. LSF fairshare scheduling can be used to address this type of requirement.

Fairshare scheduling, as the name implies, attempts to address the issue of sharing resources through assigning shares to users. It divides the processing power of the LSF cluster among users and queues to provide fair access to resources, so that no user or queue can monopolize the resources of the cluster and no queue will be starved.

This type of scheduling calculates a dynamic priority for each user by analyzing not only how many shares a user has but also determining the current and historical workload on a grid. This includes the number of job slots reserved and in use by the user, the amount of time jobs have been running, and the cumulative run time of finished jobs (adjusted so that recently used CPU time is weighted more heavily than CPU time used in the distant past). There are specific types of fairshare that handle resource contention across groups of user, queues and hosts.

```
FAIRSHARE=USER_SHARES[[dev, 10] [test, 10] [default, 3]] in lsb.queues
```

There are numerous other factors that can be included in the fairshare calculation and there are different ways it can be applied to achieve different business goals. Fairshare is just one of the many policies available and can be combined with SLA (service level agreement) scheduling to guarantee a share of the resource, or a specific throughput to meet our objectives.

CONTROL GROUPS

In a multi-user environment, resource contention is often identified as a problem. In many cases it is not due to users being malicious, it's often down to the users not really knowing what resources their jobs are taking, or coding errors where a query returns significantly more data than expected.

In this case, the same client had some unstable (i.e. buggy) home grown application that was running on their grid servers. At random times the application would create many threads which ultimately consumed all processing resources on the server and crippled their SAS workload.

While you can set memory, runtime, process limits, etc for the job, hitting these limits will usually result in the workload being terminated. Sometimes this may be ok, but in other cases it may result in knock on issues within the business process. Sometimes you just want to ensure the job is confined to a set of resources that it cannot exceed.

Linux provides a capability known as a control group (cgroup) which allows a set of processes to be bound to a set of cores, limited in memory consumption, and limit the workload to which devices (such as GPUs) that it can use[3]. You can enable CGROUP enforcement in LSF by:

```
LSF_RESOURCE_ENFORCE="cpu mem" in lsf.conf
```

This setting will place each LSF job in its own cgroup, virtually walled off from every other job, and ensuring the job cannot exceed its allocated resources.

By enabling LSF's cgroup enforcement, this workload was bound to just the cores allocated to it by LSF. Thus, when it spawned many threads, these were also automatically bound to the same cores, and had no impact on the other SAS workloads.

Cgroup's can also be used to control access to physical devices such as GPUs. In most environments GPU's are scarce resources, and while there is huge focus on developing GPU enabled AI applications, many organizations struggle to justify dedicated GPU servers when they may often lie idle, especially outside of office hours. [See below how GPUs can schedule and be allocated to workloads via LSF.]

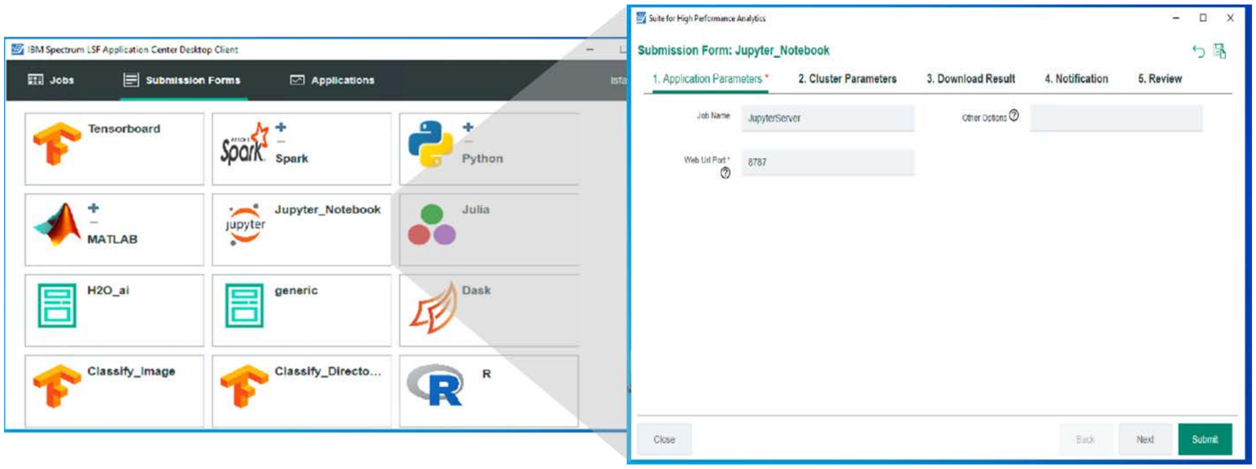
LSF DESKTOP CLIENT

SAS Studio provides clients convenience and mobility for SAS applications. The web browser-based programming environment lets you access your files and work on SAS coding from anywhere using your desktop or laptop. Given the growth of general analytics usage and the rise of open source tools, SAS users often use applications like Jupyter Notebooks,

[3] Control Groups offer many more capabilities which are also supported in LSF but are beyond the scope of this paper.

TensorFlow, Dask, PyTorch and Spark as well. For these non-SAS workloads, IBM Spectrum LSF Application Center provides similar convenience and mobility.

Using Application Center, you can define a web-based submission form for any application. The submission form hides all the details of the remote application and LSF cluster settings. With the submission form, a new user requires very little training to become productive, with clients citing significant time savings [RedBull2019].



Job Name: JupyterServer

Other Options:

Web Url Port *: 8787

Figure 2 Desktop Client

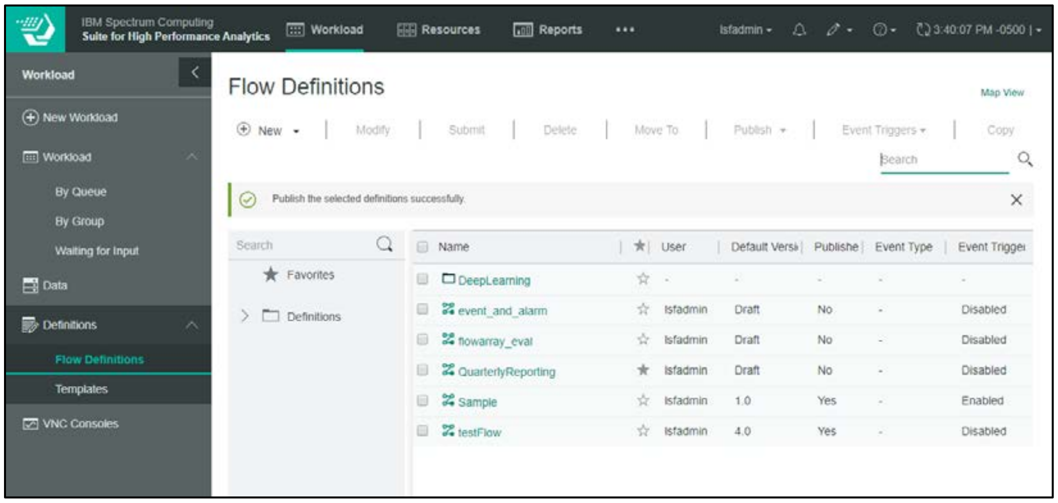
In addition to the browser-based Application Center, LSF also provides a mobile client (for Android and iOS) and desktop client. Both the mobile client and desktop clients are built upon the REST API. The light-weight desktop client allows you to submit your applications from your Windows desktop to run in a remote cluster and easily check the results.

The desktop client for Microsoft Windows greatly simplifies the management of jobs by enabling users to submit by right clicking on application input files. Output files can be automatically written back to the desktop when the job completes, and users can additionally receive job status notifications on the desktop.

WORKFLOWS

SAS users using SAS Scheduler with SAS Grid Manager for Platform will be familiar with the java client for Flow Manager. We were approached by a client in the government sector who needed to tighten their security and didn't want users physically logging into the SAS servers to design, submit and manage workflows.

Tricky with the Java client, but the LSF Suite does include a web-based version of Flow Manager which is integrated with Application Center. This provides the freedom to manage the workflows from any browser. Furthermore, the underlying scheduler server logic does not change keeping the flow behavior the same.



Flow Definitions

Publish the selected definitions successfully.

Name	★	User	Default Versi	Publishe	Event Type	Event Trigger
DeepLearning	☆	-	-	-	-	-
event_and_alarm	☆	lsfadmin	Draft	No	-	Disabled
flowarray_eval	☆	lsfadmin	Draft	No	-	Disabled
QuarterlyReporting	★	lsfadmin	Draft	No	-	Disabled
Sample	☆	lsfadmin	1.0	Yes	-	Enabled
testFlow	☆	lsfadmin	4.0	Yes	-	Disabled

Figure 3 Flow Manager

For the users who need to create new flows, Application Center has a built-in flow editor [Figure 4]. In addition, Application Center provides fine grained role-based access control (RBAC) which allowed the client to define exactly who could create flows along with who could manage specific flows.

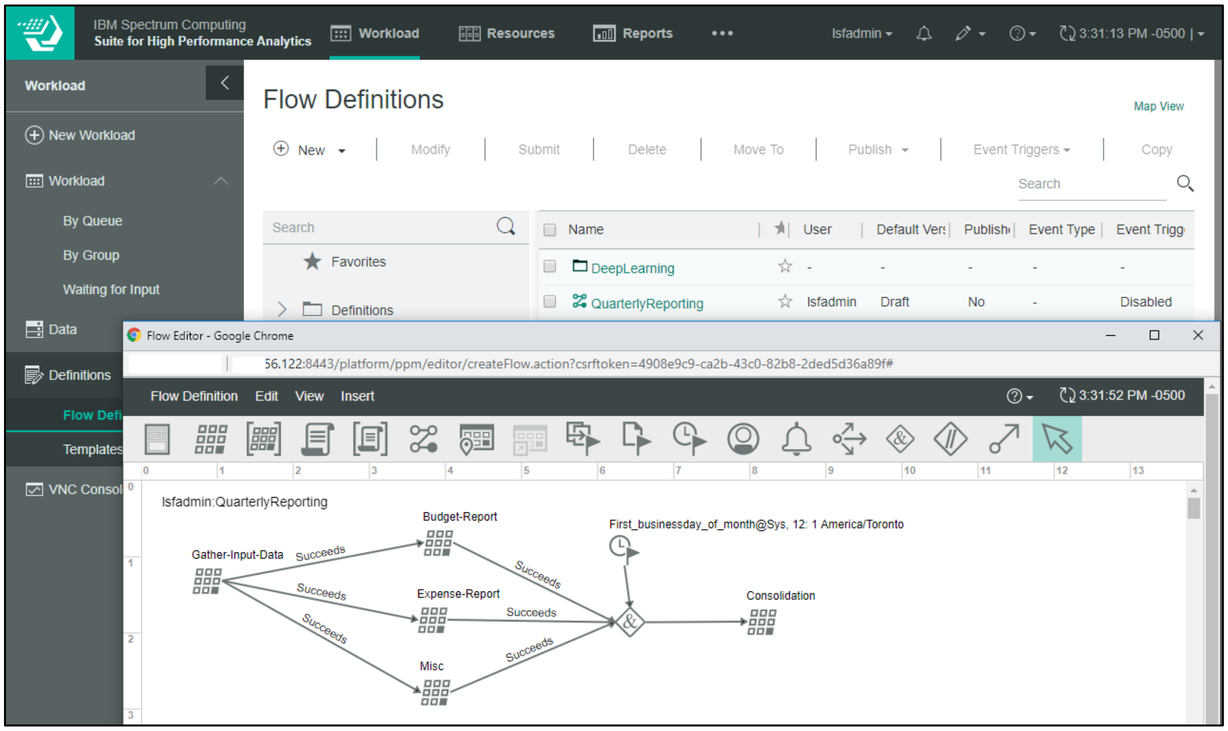


Figure 4 Flow Editor

By having both the flow editor and flow manager integrated into Application Center it provides a centralized interface to design and manage flows all together. This adds freedom to create and submit any type of workflow. All types of workflows can be managed together with no conflict. This is helpful to the clients which not only have interest in running SAS workflows but also running TensorFlow, Horovo, PyTorch, etc. LSF Application Center has a separate RBAC for better access control over the workflows. For example, many of our clients in the Life Science industry are using this capability along with open source tools like CWL and Toil to create complex workflows for Genomics sequencing [Wang2017, He2018].

GPU SUPPORT

SAS programs are increasingly leveraging GPU's to get order of magnitude improvements in performance [Bequet2017, Thompson2018]. In many cases, clients view GPU's as a simple "yes or no" device but in reality, many difference attributes need to be considered when scheduling workloads to them such as driver and CUDA versions, memory, topology, the mode the GPU is running in etc.

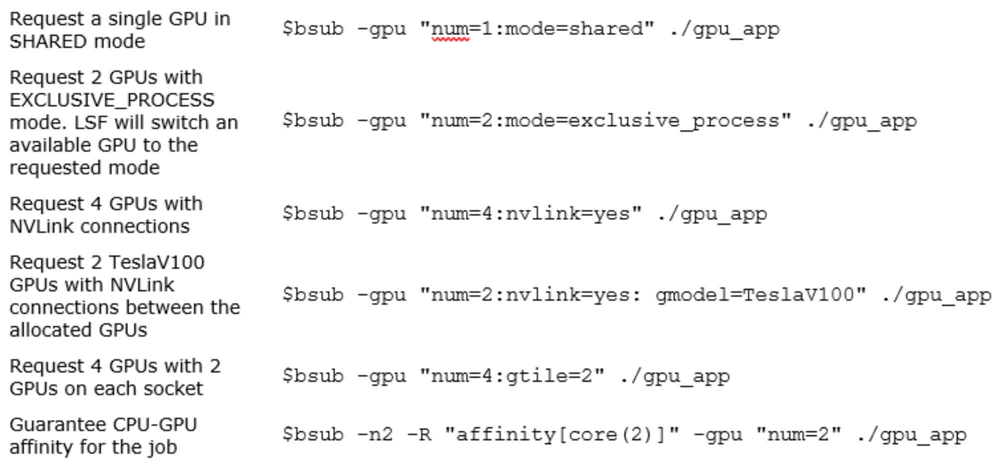
A client in health care and life sciences was developing an AI model in Python and wanted to ensure that users who were running these GPU workloads automatically got access to the GPU's, and more importantly, those who were not supposed to be using them did not. Enabling:

```
        LSF_RESOURCE_ENFORCE="gpu" in lsf.conf
```

ensures that only jobs that explicitly request GPU resources can access them. Any job that does not explicitly request GPU resources are blocked from accessing the GPU's. This provides IT with clear visibility of who and which workloads are using the GPU's.

In 2008, we released our first GPU integration kit for LSF. There have been many advances since then, and GPUs have come to the forefront in supporting ML/DL AI workloads such as Tensorflow and PyTorch. LSF's support for GPU's has continued to evolve and most recently[4] this has been extended to support autodetection and autoconfiguration of GPUs.

LSF supports a broad range of GPU capabilities including topology, NVLink, and accounting on x86 (including DGX/DGX2), Power and ARM servers. Some examples are shown in Figure 5.



Request a single GPU in SHARED mode	`$bsub -gpu "num=1:mode=shared" ./gpu_app`
Request 2 GPUs with EXCLUSIVE_PROCESS mode. LSF will switch an available GPU to the requested mode	`$bsub -gpu "num=2:mode=exclusive_process" ./gpu_app`
Request 4 GPUs with NVLink connections	`$bsub -gpu "num=4:nvlink=yes" ./gpu_app`
Request 2 TeslaV100 GPUs with NVLink connections between the allocated GPUs	`$bsub -gpu "num=2:nvlink=yes: gmodel=TeslaV100" ./gpu_app`
Request 4 GPUs with 2 GPUs on each socket	`$bsub -gpu "num=4:gtile=2" ./gpu_app`
Guarantee CPU-GPU affinity for the job	`$bsub -n2 -R "affinity[core(2)]" -gpu "num=2" ./gpu_app`

Figure 5 Examples of GPU Options

While the majority of GPU configuration is automatic, there are some additional optional features that you can enable:

- CGROUP access control.
- Support for NVIDIA's Data Center GPU Manager (DCGM). This provides additional GPU health metrics and access to ECC error information.

[4] Available in LSF 10.1.0.6

- Power Management – GPU’s can be powered down if not in use which can provide significant power savings.
- Enable GPU usage to be considered in LSF’s fairshare scheduling policy.

CONTAINER SUPPORT

The basic concept of containers has been around on UNIX operating systems for many years, but they only really took off on Linux with the introduction of Docker.

Containers were intended to be lightweight and portable. For those writing microservices and web services, they typically are lightweight. For many others adopting containers, lightweight becomes a relative term.

In many industries the principal benefit of containerization is simplifying application deployment – i.e. portability. And while they may not contain an OS instance, it is not atypical to have the whole legacy application and all its dependencies in a single container which is Gigabytes or even tens of Gigabytes in size [SAS2018a].

One benefit of this approach is that there is very little difference between scheduling the application and scheduling the containerized application:

```
$bsub my_app
$bsub -app docker my_containerized_app
```

The fact that the application is now a single file addresses many of the traditional objections of running different applications in the same OS instance such as what if installing Application X leads to issues or library incompatibilities with Application Y, which is the primary application on the server? With the “guest” application in a container it does not change anything on the host and allows other applications to share the instance, and it is scheduled and managed as any other job – if there is contention for resources it can be throttled or terminated. We are seeing many clients adopt using this approach to drive up utilization of existing environments.

Now Gigabyte sized containers are not exactly lightweight, and whether they are truly portable becomes a function of storage space and network speed. LSF does provide several additional capabilities that help with running “full app” containers:

- Security: LSF takes responsibility for the container lifecycle, which means the user does not need to be in the DOCKER_USERS group, and thus the user never gains or has the potential to gain elevated privileges.
- Control: A user could potentially install a container from any 3rd party repository – something that makes IT Security feel very uneasy – how do we secure and audit the environment when we don’t know what is installed or the provenance of it? Thankfully LSF allows you to strictly control which repositories can be used, and indeed which containers and versions of containers can be used.
- Auditability: The administrator has a central view of which containers are installed where, and when they were last used. This allows storage management policies to decide when to delete large unused containers.
- Affinity: If a container is gigabytes in size it may take tens of minutes or longer to download and install. This is not really an issue if it is then going to run a service that is going to exist for hours or days. But if it is to run a job and the job is only a few

minutes in duration, that's very inefficient. LSF will try to re-use container images rather than download them again, reducing the startup overhead.

- Flexibility: LSF supports multiple container technologies including Docker, Nvidia Docker, Singularity, and Shifter.

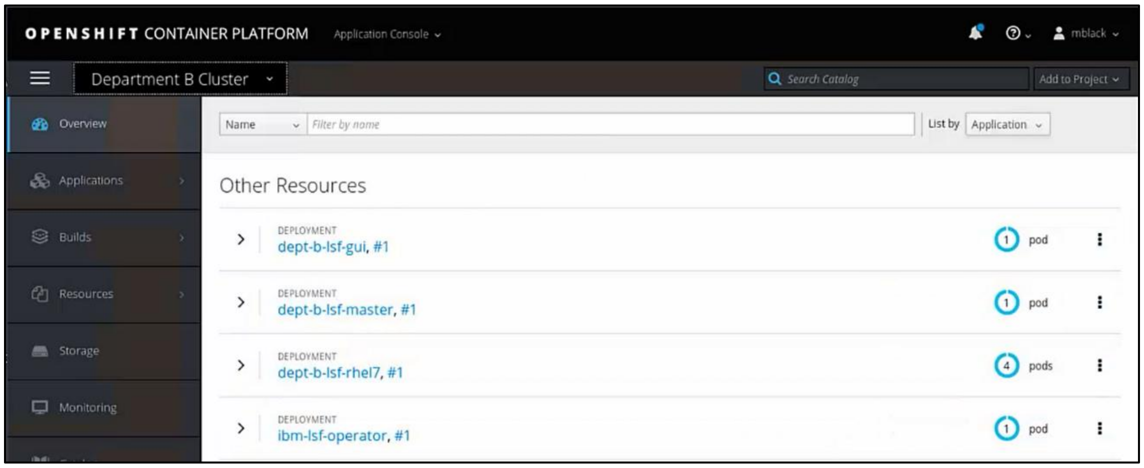
The alternative approach involves re-architecting the application to consist of multiple loosely connected micro-services each in their own container. Such applications, for example SAS Viya [SAS2020] require the orchestration of containers through frameworks such as Docker Swarm or Kubernetes.

THE RISE OF KUBERNETES

If you work in IT, it would be virtually impossible to miss the rise of Kubernetes. We are seeing a dramatic shift in the market where our clients are looking to take advantage of Kubernetes for both existing and new workloads. How to achieve this without extensive application rework or standing up multiple environments does present IT with interesting challenges. In this section we'll discuss a possible approach that has been used at several clients.

While SAS has announced a partnership [SAS2019] with Red Hat around SAS Viya and OpenShift, the Red Hat flavor of Kubernetes, as well as SAS Viya on IBM Power [IBM2019], it is beyond the scope of this paper to discuss containerizing SAS. For those interested in a deeper discussion, we would recommend [Furbee2019a, Furbee2019b, Zennick2019].

For those who are interest, the LSF Suite can be run in Kubernetes. We provide a Kubernetes Operator to provide one click deployment of the cluster Figure 6.



Other Resources

DEPLOYMENT	dept-b-lsf-gui, #1	1 pod
DEPLOYMENT	dept-b-lsf-master, #1	1 pod
DEPLOYMENT	dept-b-lsf-rhel7, #1	4 pods
DEPLOYMENT	ibm-lsf-operator, #1	1 pod

Figure 6 LSF running on OpenShift

WORKLOAD (POD) ORCHESTRATION IN KUBERNETES

Kubernetes is an excellent container orchestration platform and excels at managing stateless and stateful services. All workloads in Kubernetes run in a pod, which is a collection of one or more containers and associated resources required to run an instance of that workload.

Decisions about scheduling and placement of pods onto the underlying infrastructure is made by the Kubernetes Scheduler. This pod placement is completely independent of any application level workload scheduling that occurs within the pods. For example, SAS Workload Orchestrator can be managing the work within the pods, while the Kubernetes scheduler is controlling the placement of the actual pods onto the infrastructure.

The default scheduler in Kubernetes is relatively simplistic since it was designed to handle services along with the assumption that the cluster could grow and shrink as needed. This means that there are challenges and shortcomings when more dynamic or ad-hoc workloads need to be run. Table 1 illustrates some of the differences between traditional schedulers and the Kubernetes scheduler.

	Workload Schedulers	Kubernetes
Focus	Highly Scalable schedulers with rich scheduling policies that have developed over many years.	Cloud Native container orchestration platform designed for managing services / microservices.
Workloads	Typically, ad-hoc, either user or calendar driven with workloads typically running as the submission user or as a service user.	Typically, managed services running as service users.
Environment	Mix of bare metal, virtual machines, containers and cloud.	Everything containerized.
Container Usage	Primarily used as a deployment mechanism. (Very) large containers containing the whole application and expected to run with the submitting users' credentials.	Orchestrated Services composed of multiple containers. Typically running as service users.
Resource Model	Assumes resources (time, space, money) are finite which requires sharing & prioritization.	"Cloud Native" – assumes resources are infinite and the environment can always be automatically enlarged.

Table 1 Workload Attributes

Kubernetes as a cloud native orchestrator has three core autoscaling functions that help arbitrate between different workload (pod) demands:

- The horizontal autoscaler allows a service or workload to horizontally scale – e.g. creating more instances (pods) of a web server to handle an increase in load or launching more instances of the CAS server in Viya.
- The vertical autoscaler allows a single instance to grow vertically – i.e. to consume more system resources than originally defined. As with a VM, today if you vertically scale a pod it needs to be restarted, unlike an LSF job where the resource limits on a job can be dynamically changed without restarting.
- If there is more work than there is space available, the cluster autoscaler then kicks in to enlarge the cluster. If you are running on the cloud then enlarging the cluster is usually possible, subject to budget. But if you are running on premise or are budget constrained, then enlarging is likely not possible - you need to arbitrate between competing workloads with a finite amount of space meaning you need a workload orchestrator.

This leads to an interesting contrast. On one hand we have the Cloud providers talking about running very large Kubernetes environments supporting multiple lines of business and applications, while many on premise organizations are often running multiple Kubernetes silos, unable to leverage the benefits of a more consolidated Kubernetes environment due to the deficiencies in the core scheduler.

The Kubernetes community has recognized these deficiencies and the Scheduling Working Group does plan to incrementally enhance the scheduler. But that does not help us today.

ENHANCED SCHEDULING KUBERNETES PODS

Kubernetes is however designed to be extendable through plug-ins and custom resource definitions (CRD), and we do have an orchestrator available: LSF.

We have created a new plugin scheduler for Kubernetes based on LSF called HPAC (High-performance Pod Allocation for Containers). In simple terms, this means that the HPAC scheduler is taking responsibility for orchestrating and prioritizing pod placement requests and doing so in a manner that is completely transparent to the workload in the containers.

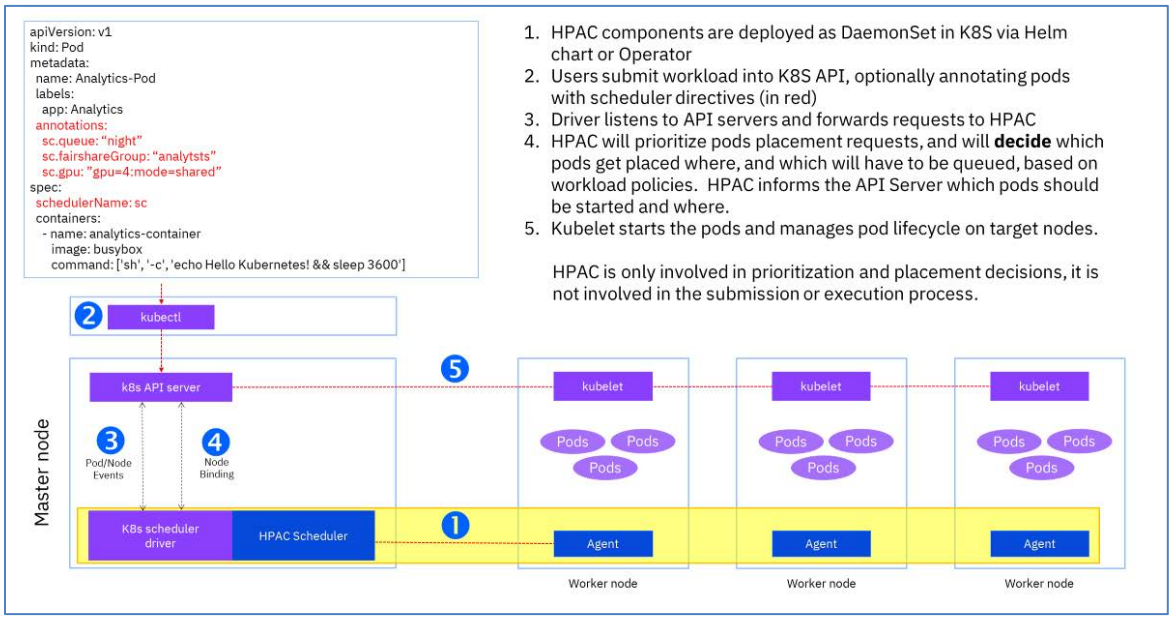
For example, to scale the number of CAS workers in Viya you would use the Kubernetes command [Furbee2019b]:

```
$kubectl scale deployment sas-viya-cas-worker --replicas=6 -n myviya run
```

The Kubernetes scheduler then decides where to place the additional replicas in the cluster.

If you enabled HPAC as the Kubernetes Scheduler, then the command would be the same – HPAC does not change the application logic. The only difference would be that the decision on where to place the new pod replicas within the Kubernetes environment would be made by HPAC. Decisions on where workloads run within the pods is still made by the application. A Kubernetes Administrator can use HPAC and the associated workload policies to orchestrate which pods get started and which pods will queue based on fairshare and SLA's.

Figure 7 illustrates how this HPAC scheduler plugin works. It can be configured as the default scheduler, or explicitly called just for some workloads. For those wanting to fine tune workloads, HPAC scheduler specific annotations can be specified in the application's yaml.



```
apiVersion: v1
kind: Pod
metadata:
 name: Analytics-Pod
 labels:
  app: Analytics
 annotations:
  sc.queue: "night"
  sc.fairshareGroup: "analytsts"
  sc.gpu: "gpu=4:mode=shared"
spec:
 schedulerName: sc
 containers:
  - name: analytics-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

1. HPAC components are deployed as DaemonSet in K8S via Helm chart or Operator
2. Users submit workload into K8S API, optionally annotating pods with scheduler directives (in red)
3. Driver listens to API servers and forwards requests to HPAC
4. HPAC will prioritize pods placement requests, and will **decide** which pods get placed where, and which will have to be queued, based on workload policies. HPAC informs the API Server which pods should be started and where.
5. Kubelet starts the pods and manages pod lifecycle on target nodes.

HPAC is only involved in prioritization and placement decisions, it is not involved in the submission or execution process.

Figure 7 Enhanced Kubernetes Pod Scheduler

ELASTIC WORKLOADS

In AI, there are frameworks that support Elastic Distributed Training and Elastic Distributed Inference[5] which will grow and shrink the required number of pods allocated to a user or group of users depending on who else is using the environment. For example, if there are no other users, the first user could be allocated all CPUs and GPUs allowing their AI training job to progress as quickly as possible. When a second user wants to use the system, the first user is automatically shrunk back to half the GPUs. When a third user wants resources, it is reallocated again and so forth. Such behavior requires coordination between the application framework and the underlying scheduler.

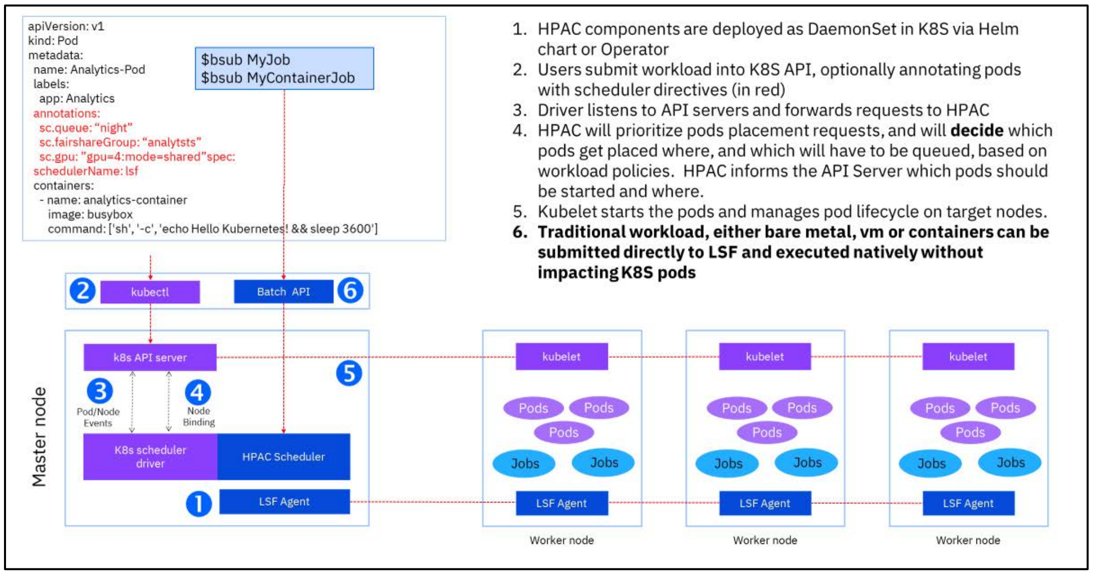
A HYBRID LSF-KUBERNETES ENVIRONMENT

It is rarely practical for an organization to containerize all workloads overnight. One solution would be to stand up two environments – one for legacy workloads and the other for Kubernetes workloads. While this is the simplest approach, it is frequently viewed as the most expensive.

An alternative approach would be to create a hybrid environment that supported both containerized and non-containerized workloads. We previously discussed how LSF supports this for Docker, Singularity and Shifter containers, but what about Kubernetes?

As HPAC is based on LSF, we can also deploy the HPAC-Kubernetes integration in a hybrid mode with LSF execution servers, and have HPAC act as a single brain orchestrating Kubernetes and non-Kubernetes workloads in the same hardware cluster, and even within the same OS instance as shown in Figure 8. Kubernetes can be deployed on all or just a subset of the servers that LSF is installed on. With HPAC acting as a single brain, it will prioritize both LSF and Kubernetes workloads and decide on placement and allocation of resources.

Most importantly, this means that both the traditional IT environment, and the Kubernetes environment can shrink and grow dependent upon business workload priorities.



```
apiVersion: v1
kind: Pod
metadata:
 name: Analytics-Pod
 labels:
  app: Analytics
 annotations:
  sc.queue: "night"
  sc.fairshareGroup: "analytsts"
  sc.gpu: "gpu=4:mode=shared"spec:
 schedulerName: lsf
 containers:
  - name: analytics-container
   image: busybox
   command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

```
$bsub MyJob
$bsub MyContainerJob
```

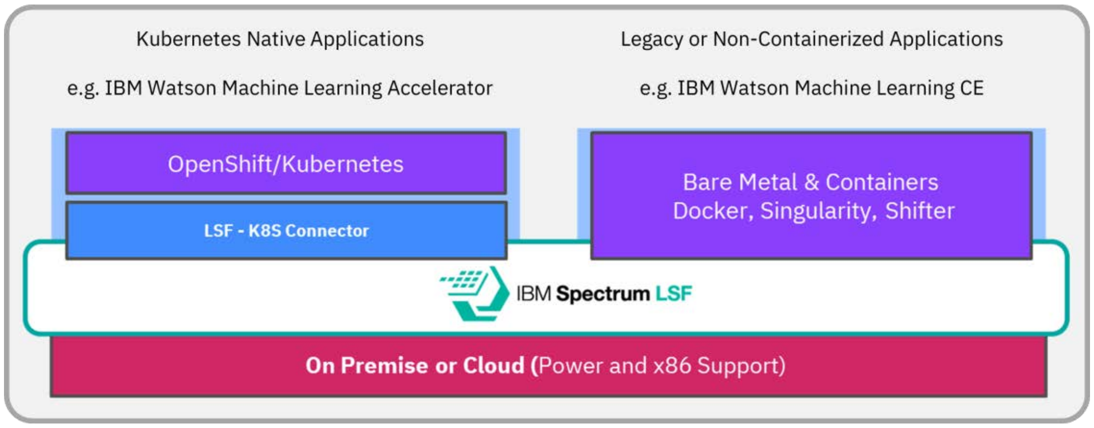
1. HPAC components are deployed as DaemonSet in K8S via Helm chart or Operator
2. Users submit workload into K8S API, optionally annotating pods with scheduler directives (in red)
3. Driver listens to API servers and forwards requests to HPAC
4. HPAC will prioritize pods placement requests, and will **decide** which pods get placed where, and which will have to be queued, based on workload policies. HPAC informs the API Server which pods should be started and where.
5. Kubelet starts the pods and manages pod lifecycle on target nodes.
6. **Traditional workload, either bare metal, vm or containers can be submitted directly to LSF and executed natively without impacting K8S pods**

Figure 8 Mixed Kubernetes and non-Kubernetes Workloads

[5] For example, IBM Watson Machine Learning Accelerator.

This hybrid approach can be used to mix legacy and cloud native applications, and to support application modernization. For example, IBM Watson Machine Learning Community Edition provides containerized versions of popular AI frameworks compiled for IBM Power Systems. It is typically used directly by LSF as shown on the right-hand side of Figure 9.

On the left-hand side is IBM Watson Machine Learning Accelerator which is a Kubernetes native application – using this hybrid mode we can run both in the same environment and avoid the costs of setting up two separate environments.



Kubernetes Native Applications
e.g. IBM Watson Machine Learning Accelerator
Legacy or Non-Containerized Applications
e.g. IBM Watson Machine Learning CE
OpenShift/Kubernetes
LSF - K8S Connector
Bare Metal & Containers
Docker, Singularity, Shifter
IBM Spectrum LSF
On Premise or Cloud (Power and x86 Support)

Figure 9 Supporting Application Modernization

CLOUD & HYBRID CLOUD

Cloud provides the opportunity to tackle new problems, introduce new processes, or reduce costs, but it is not a panacea for everything. We've heard many tales of significant challenges from clients where they have been told to "just use the cloud" without any real appreciation of what is required. Moving to a cloud-based mail service is straightforward, but moving an end to end business process, and all the associated data takes a lot more awareness and thought. SAS have done a lot of that thought and provided guides to running SAS 9.4 and SAS Viya on the Cloud [SAS2018b, SAS2018b].

If you are running SAS Grid Manager for Platform as part of your cloud deployment, you can leverage its Resource Connector capability to have the size of the overall cluster flex in response to workload demands [SAS licensing considerations are not to be ignored] – adding or removing VM instances depending business goals. This LSF capability is supported on AWS, Azure, GCE, IBM Cloud and with OpenStack.

HYBRID CLOUD

If you have a significant investment in your on-premise infrastructure or wholesale movement of your data to the cloud is impractical for volume, commercial or legal reasons then the Cloud doesn't look all that enticing.

We are seeing many, many clients leveraging cloud and hybrid cloud for different classes of workload, especially for R&D or Dev/Test. These typically use reference [or cleansed] data sets which do not pose commercial or legal issues in them being used off prem.

Even if you do not have commercial or legal inhibitors to leveraging hybrid cloud, there are other issues to consider:

- Application Licensing – can you run off premise, and what is the licensing model for as a service use?
- Latency – there are startup costs involved with spinning up new instances in the cloud, and users will certainly notice these running interactive applications. Horton2020 proposes a novel method leveraging functionality in the Resource Connector to alleviate this.
- Data – this is the key issue for many who try to leverage hybrid cloud - getting the right data to the cloud in a timely manner. Luckily, the combination of LSF's MultiCluster, Resource Connector, and Data Manager can help.

A HYBRID ARCHITECTURE

Figure 10 illustrates a typical hybrid cloud environment.

- MultiCluster decides what and when to forward work to the Cloud, based on workload and business rules.
- Resource Connector decides when to scale cloud resources, and thus when to incur costs, also based on business rules. Most cloud providers have their own "autoscalers" for LSF which are typically just based on the number of pending jobs – so scale up as quickly and to as many as possible to reduce the pending jobs. This also typically means incurring the maximum cost. One client switched from the Cloud Provider's own autoscaler to our resource connector and immediately saw a 30% saving in cloud costs, while maintaining the same service levels. As previously mentioned, the Resource Connector also supports OpenStack, which is very important as many organizations have internal OpenStack based environments. Which means the Grid can burst into the internal OpenStack environment where data movement costs are much lower or may not exist at all.
- Data Manager extends scheduling decisions to include data requirements. You want to ensure that the required data is available in the cloud _before_ any instances are spun up and costs incurred. If the required data for a given job is not available, Data Manager will invoke a transfer job to make it available – how data is moved is site configurable. Data Manager will also try and deduplicate transfers – so if you are running a 1000 step parametric sweep over the input data, it will only move it once, and not 1000 times.

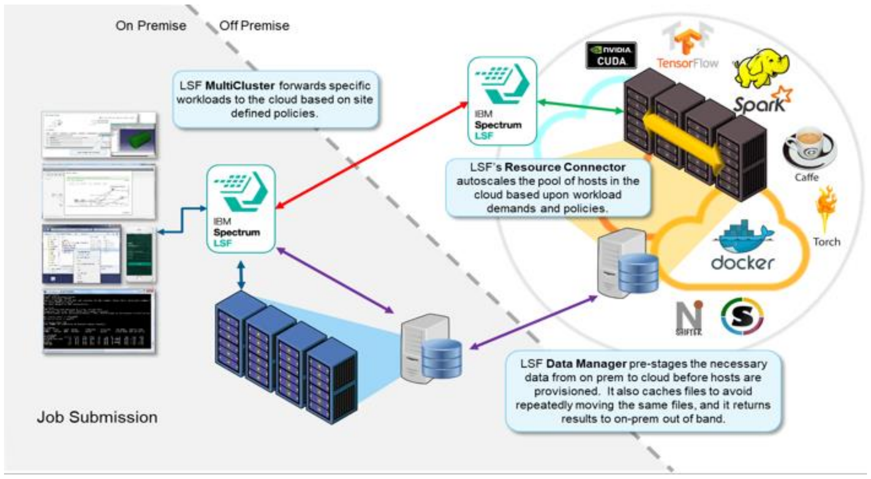

On Premise
Off Premise
LSF MultiCluster forwards specific workloads to the cloud based on site defined policies.
IBM Spectrum LSF
NVIDIA CUDA
TensorFlow
Spark
LSF's Resource Connector autoscales the pool of hosts in the cloud based upon workload demands and policies.
Caffe
Torch
docker
LSF Data Manager pre-stages the necessary data from on prem to cloud before hosts are provisioned. It also caches files to avoid repeatedly moving the same files, and it returns results to on-prem out of band.
Job Submission

Figure 10 - Hybrid LSF Architecture

SERVERLESS OR FUNCTION AS A SERVICE

For those who remember RPCs (remote procedure calls), Serverless or Function as a Service (FaaS) doesn't seem all that different – it allows a function to be offloaded somewhere else. In the context of Cloud and Containers it does offer some interesting advantages in that you can define your own function that will be run, and it will typically execute in a container on the cloud – and you only pay for time consumed, with no need to worry about setting up resources, queues or anything else.

And this is great, if where the function is executing is where your data is…i.e., on the cloud. If you must export all the necessary data to the cloud, then pull all the results back again, you may incur more time and costs in data movement than in computation.

FAAS & GRID

We recently had a client approach us with an interesting problem. They had python users wanting to use the simplicity of a FaaS model, but all their data was already in their grid environment and they didn't want to replicate it all to the cloud. Given everything else we were already supporting in their grid environment, could we support the FaaS users?

After a bit of thought, well quite a lot of thought, we put together the pieces we've already discussed in this paper to build a prototype solution using iPython and a Jupyter Notebook as the front end:

```
[1] import pandas as pd
    import numpy as np
    from sklearn import linear_model

[2] def regression_func(file)
       df = pd.read_csv(file)
       cdf =df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION','CO2EMISSIONS’]]
       msk = np.random.rand(len(df)) < 0.8
       train = cdf[msk]
       regr = linear_model.LinearRegression()
       train_x = np.asanyarray(train[['ENGINESIZE']])
       train_y = np.asanyarray(train[['CO2EMISSIONS']])
       regr.fit (train_x, train_y)
       return regr

[3] id = lsf.sub(regression_func, array_of_input_files)

[4] regression = lsf.get(id)
```

The user defined function, regression_func() in the example above, is sent to the LSF cluster via the RESTful API and executed as an array on the LSF cluster, each element in its own container.

While this as present is just a prototype, it illustrates how the different capabilities already within the LSF Suite can be combined to address new workloads.

CONCLUSION

What Lessons have we learned from the field?

Firstly, the workload management requirements for those running SAS and other analytical workloads are not all that different from those in other industries. Existing LSF capabilities can be readily applied to address these requirements especially as we see the merge of high-performance computing and high-performance analytics uses.

Secondly, as with other industries, Cloud is a hot topic, but there are significant challenges in bursting on premise workloads to the cloud, particularly in relation to data – but there are workable solutions.

Thirdly, Containers and Kubernetes are viewed as key technologies for the modernization of most enterprise's application infrastructure. While Kubernetes is an excellent orchestrator there are challenges in managing certain classes of workload, particularly in AI.

And finally, there are a lot of capabilities "under the hood" in SAS Grid Manager for Platform that many SAS Grid clients are unaware of. We would be delighted to discuss how to get the most out of your investment.

REFERENCES

Bequet2017: Bequet, H & Chen, H. *Accelerate your SAS Programs with GPUs*,

https://support.sas.com/resources/papers/proceedings17/SASSD0706-2017.pdf

Furbee2019a: Furbeee, J: *Getting started with SAS Containers*

https://blogs.sas.com/content/sgf/2019/03/06/getting-started-sas-containers/

Furbee2019b: *Deploying the Full SAS Viya Stack in Kubernetes*

https://blogs.sas.com/content/sgf/2019/06/10/deploying-the-full-sas-viya-stack-in-kubernetes/

Haig2019: Haig, D. Introducing SAS®Workload Orchestrator, the New SAS®Grid Manager Workload Manager

https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3430-2019.pdf

He2018: He, YY. *Go With the Flow – Accelerating Complex Computational Workflows*

https://www.hpcwire.com/solution_content/ibm/cross-industry/go-with-the-flow-accelerating-complex-computational-workflows/

Horton2020: Horton, G, Paper # SASGF2020

IBM2019: Accelerate insights with SAS® Analytics on IBM® POWER9™

https://www.ibm.com/blogs/systems/accelerate-insights-with-sas-analytics-on-ibm-power9/

KDNuggets2018: 2018 KDnuggets annual software survey.

https://www.kdnuggets.com/polls/

Redbull2019: https://www.ibm.com/it-infrastructure/spectrum-computing

SAS2018a: *Understanding Containers and SAS®9.4 Container Deployment.*

http://support.sas.com/resources/papers/understanding-containers-sas-9-4-container-deployment.pdf

SAS2018b: *SAS Grid on the AWS Cloud*

https://s3.amazonaws.com/aws-quickstart/quickstart-sas-grid/doc/sas-grid-on-the-aws-cloud.pdf

SAS2018c: *SAS Viya on the AWS Cloud*

https://aws-quickstart.s3.amazonaws.com/quickstart-sas-viya/doc/sas-viya-on-the-aws-cloud.pdf

SAS2019: *SAS and Red Hat collaborate to optimize analytical capabilities across the hybrid cloud.* https://www.sas.com/en_is/news/press-releases/2019/october/sas-red-hat-openshift-hybrid-cloud.html

SAS2020: *SAS® for Containers: Bringing speed, agility and scale to cloud deployments.*

https://www.sas.com/en_gb/solutions/cloud-computing/on-providers/deployment-patterns/for-containers.html

Thompson2018: Thompson, W. *Why you need GPUs for your deep learning platform*

https://blogs.sas.com/content/subconsciousmusings/2018/10/16/why-you-need-gpus-for-your-deep-learning-platform/

Welch2019: Welch, J. *Tensorflow & Pytorch Examples running in IBM Spectrum LSF Suite v10.2 cluster using IBM Power server*

https://www.youtube.com/watch?v=zJkjuwFHD3M

Wang2017: Wang, Q. *Running CWL Workflows with LSF through Toi*

https://developer.ibm.com/storage/2017/05/04/cwl-workflow-lsf-toil/

Zenick2019: Zenick, B, Gomez, I & Koob, M. *Modernizing Your SAS Analytics Platform with Containers.*

https://www.zencos.com/blog/flexible-low-cost-sas-analytics-containerized-solution/

All URL's last accessed February 17, 2020

ACKNOWLEDGMENTS

The authors would like to especially thank Cheryl Doninger for her many years of leadership with and around SAS Grid Manager and Platform Suite for SAS along with her tireless effort to enable the teams to provide a high quality solution enabling SAS customers to get even more insight out of their businesses and value out of their analytics software investment. The SAS research and development team of Scott Parrish, Doug Haigh, Randy Williams, Paula Kavanagh amongst others over time have assisted in the same regard. Ken Gahagan, Tony Brown, Margaret Crevar, and Glenn Horton at SAS have supported the focus on performance and deployment in the field with deep architecture and real-world expertise. Finally, Kevin Go has been our Alliances support and has helped manage the SAS-IBM relationship to many great successes. To all of you thank you!

Additionally, the authors would like to thank IBM development leaders Gang Pu, and Jin Chi He. Finally, we wish to thank Gregg Rohaly for Alliances support and helping manage the SAS-IBM relationship to many great successes.

RECOMMENDED READING

- *https://www.ibm.com/us-en/marketplace/spectrum-computing-for-hpa*
- *https://www.ibm.com/support/knowledgecenter/SSGFRP_10.2.0/welcome/suite_hpa_kc_overview.html*
- *LSF User Community http://ibm.biz/LSFCommunity*
- *SAS® GRID Computing For Dummies® https://www.sas.com/en/whitepapers/sas-grid-computing-for-dummies-108762.html Most of the examples in this book are using SAS Grid Manager for Platform.*
- *IBM Spectrum LSF Suites Best Practices – IBM RedBooks, March 2020.*
- *IBM Spectrum Scale Best Practices for Genomics Medicine – IBM Redbooks, April 2018.*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Biill.McMillan@uk.ibm.com
Qingda Wang qwang@ca.ibm.com
Kinson Chik kchik@ca.ibm.com